How to share cyber intelligence automatically in real-time by using system architecture process

Abdiweli Mohamed Halane
School of Computing and Digital Media
London Metropolitan University

This thesis is submitted for the degree of
Doctor of Philosophy

May, 2025

Dedicated to my loving wife and sons…

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this Thesis are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other University. This Thesis is the result of my own diligent work and includes nothing that is the outcome of any collaborative work. This thesis contains over 65,000 words including appendices, references, footnotes, tables and equations and has 192 references which are duly cited.

Abdiweli Mohammed Halane

May, 2025

Acknowledgements

Writing my PhD thesis has been a considerable and transformational experience. The proverb, "Life is what occurs while you are formulating alternative plans," undoubtedly pertains to the thesis writing process. Life progresses continuously, irrespective of deadlines or academic milestones, and throughout this period, numerous aspects have transformed and developed. This effort, jokingly referred to as “The Paper” by most of my closest family members, has been shaped by more than mere study and writing. Many extraordinary individuals have persistently supported and helped me to push it. I am profoundly grateful to all of them, particularly:

My supervisors, Dr Alexandros Chrysikos and Professor Karim Ouazzane; I greatly appreciate your guidance and persistence on my behalf. Your unending support and eagerness have enabled me to engage in research on subjects that truly interest me. I see the same motivation and fervour in your scholarly endeavours, and I appreciate that you allow me to engage accordingly. Thank you for every meeting and discussion throughout the years. You acknowledged my occasional need for solitary work while ensuring that you monitored my progress. Your guidance has shaped my academic journey and instilled confidence in my abilities.

I would not have finished this thesis without the support and love of my wife, Siciido Hassan Mohamed; my sons, Mohamed and Suleyman; and my sister Fariya and brother Abdifitaax. You all have encouraged my academic interests from day one. Thank you.

Abstract

Cyber dangers have grown, as cyber-attackers exploit unknown but extant software and system vulnerabilities through phishing, ransomware, and malware attacks. Even though cyber intelligence (CI) sharing is important as a defence mechanism, its implementation is hindered by issues such as lack of standardisation, lack of automation, lack of trust between businesses, need for timely and efficient processes, legal barriers, and privacy concerns. Current CI sharing platforms rely heavily on informal community-based platforms or semi-automated methods which includes emails communications, phone calls, voucher systems, or in-person meetings, and struggle to distribute timely, relevant, and actionable intelligence. This delay in distribution makes shared intelligence less effective against malware attacks especially, and makes cyber defence not proactive. This study tackles these malware detection and defence limitations by recommending the Proactive Self Defence Cyber Intelligence Sharing (PSDCIS) Model; a novel model that automatically integrates security systems in real-time CI exchange, that is fully automated without human intervention. The model is akin to the philosophy of Sun Tzu, by allowing systems to engage with threat situations, get fresh malware intelligence, and share it in real time to initiate proactive defence. To develop the PSDCIS model, this study investigated three CI sharing models which are Structured Intelligence Sharing Model, Detection Maturity Level (DML) Model, and the Cyber Threat Intelligence (CTI) Model. In addition, five CI sharing systems were assessed which are Collective Intelligence Framework (CIF), Malware Information Sharing Platform (MISP), Webroot Intelligence Network (WIN), Darknet and Deepnet Cyber Security Threat Intelligence, and the Proactive Threat Analysis of Cyber Threat Intelligence. The problem discovered was that even though these models gather malware intelligence and centralise them effectively, the delay in processing and the lack of an all-in-one system integration reduce the operational value of real-time defence. In response, this research study created the PSDCIS model; an automatic State Machine (SM) model that can gather

intelligence, analyse datasets using Machine Learning classification algorithms and instantly share malware intelligence with managers, system administrators and security systems, and also mitigate broader cyber risks.

Table of contents

List of figures

List of tables

List of abbreviations

CI	Cyber Intelligence
CT	Cyber Terrorism
CTI	Cyber Threat Intelligence
ML	Machine Learning
CS	Cyber Security
ILC	Intelligence Life Cycle
CSS	Company security systems
CIF	Collective Intelligence Framework
PSDCIS	Proactive Self-Defence Cyber Intelligence Sharing
DML	Detection Maturity Level
WIN	Webroot Intelligence Network
SVM	Support Vector Machine
MED	Maximum Entropy Discrimination
APT	Advanced Persistent Treat
SIEM	Security Information and Event Management
TISPs	Threat Intelligence Sharing Platforms
TISP	Threat Intelligence Sharing Platform
SBHG	Stability-Based Hashed Gibbs Sampler Model
CybOX	Cyber Observable Expression
STIX	Structured Threat Information Expression
TAXII	Trusted Automated Exchange of Indicator Information
NATO	North Atlantic Treaty Organisation
MISP	Malware Information Sharing Platform
OpenIOC	Open Incident of Compromise
IODEF	Incident Object Description Exchange Format
CRITs	Collaborative Research into Threats
API	Application Programming Interface
CIRCL	Computer Incident Response Centre Luxembourg
HUMINT	Human Intelligence

IMINT	Imagery Intelligence
OSINT	OpenSource Intelligence
MASINT	Measurement and Signature intelligence
CYBINT	Cyber Intelligence
SOCMINT	Social Media Intelligence
SNSs	Social Media Networking Sites
K-NN	K-Nearest Neighbours
ANN	Artificial Neural Networks
GBM	Gradient Boosting Machines
XGBoost	Extreme Boosting
IoT	Internet of Things
DDoS	Distributed Denial of Services
SQL	Structured Query Language
URL	Uniform Resource Locator
ATM	Adobe Type Manager
IRS	Incident Response Team
SGD	Stochastic Gradient Decent
CSV	Comma Separated Values
ROC-AUC	Area Under the Receiver Operating Characteristic Curve
REST	Representational State Transfer
SMTP	Simple Mail Transfer Protocol
JSON	JavaScript Object Notation
XML	Extensible Marked Language
SM	State Machine

CHAPTER ONE: INTRODUCTION

Introduction

The connectivity of businesses has become increasingly important to the smooth operation of commerce, which depends on information exchange and computer systems, as global information technology networks continue to expand. Internet offers a lot of advantages in one aspect, but also a means exploited for cyber-espionage and cyber-hacking. Cybercriminal and cyber terrorist groups use the internet for a wide range of activities, including delivering malicious assaults, learning more about a target company, planning attacks, and inciting cyber terrorism (UNODC, 2010). The success of cybercriminals and the rising number of cyber-attacks on information assets, expose the shortcomings of the information security and Cyber Intelligence (CI) systems now in place.

Amidst various cyber threats, malware has become the most pressing, most damaging and most flexible. This includes viruses, Trojans, ransomware, worms, spyware and all forms of malicious software which attackers use as weapons to disrupt operations, corrupt systems, steal sensitive data and make illegal profit. Malware, due to its ability to escalate rapidly, spread vulnerabilities across computer networks before any cyber defence mechanism swings into action, are very challenging. Against this backdrop, malware is a threat, and a very important focus for this research in contribute to securing Cyber Intelligence (CI) systems. With the existence of a wide array of CI operations against cyber threats, this study focused on malware detection and how to mitigate it, which mirrors the weakness of existing CI models and the solution to develop for this gap.

Malware attacks by computer criminals, as a challenge became prominent in the early 1940s as technology was advancing, yet security experts and system engineers were unaware of the software's and systems' security flaws at the time. Due to this, hackers were able to identify system vulnerabilities earlier and used them to attack the systems. Hackers discovered vulnerabilities in systems and

software, and used them as unauthorised access to networks, hardware, and computer systems over the internet (Weimann, 2004).

As personal computers and the internet grew in the early 2000s, businesses formed local networks and threat diversity multiplied. In addition, cybercrimes also got new opportunities, when brand new virus types that did not involve file downloads were created at the early stages of this period (Bhadwal, 2015). High profile threat of cyber-attacks happened in 2011 as when Sony's PlayStation Network was breached compromising 77 million PSN user accounts and made services and PlayStation portable consoles unavailable. Sony had to shut down systems due to the attack (Olaniran et al., 2014).

In 2012, an unknown pattern of fraud was spotted by Union Savings Bank in Danbury, Connecticut, on roughly twelve of the debit cards issued to customers. These cards were discovered to have been used near a private school's cafeteria. This demonstrated that malware when deployed can undermine institutional trust and consumer confidence in corporate circles. A hacktivist group called Anonymous, launched a cyber-attack on the Singaporean government in 2013, compromising the protest censorship laws (Bhadwal, 2015).

The Stuxnet worm attack against Iran's nuclear enrichment process was another incident allegedly planned by the American and Israeli secret services, according to the Iranian government (MacAskil and Greenwald , 2013) that showed up as a new level to cyber operations that damaged physical infrastructure through digital infiltration. Another emergence was when a group from a neighbouring nation began disseminating Syrian Electronic Army, targeting websites run by Syrian opposition groups and worked to remove, deface, or destroy them (Noman, 2013). SecureWorks also recording 7.7 million attempted cyber-attacks coming from China, which was a toolset called "Leopard in a Hole" sold on underground markets, and this broadened the pool of malware attackers. The program assisted penetration testers and hackers in identifying and exploiting SQL injection flows (Dwyer, 2009). Malware is therefore a persistent and growing element among other cyber threats. Hackers had found previously unknown weaknesses in the systems

that were processing information at the time, which they exploited to compromise the systems. Growing complexity of malware compromises, has given rise to a decrease in Cybersecurity (CS) making it tasking for security experts to safeguard organisations (Tounsi, 2019).

Studies about malware attacks have shown that Phishing which deceives victims to divulge confidential information, have become mainstream for cybercriminals. Cybercriminals seem to be targeting information systems and applications more frequently including launching ransomware (Nobanee et al., 2023), malware, viruses, spyware, adware, and other malicious software (Thomasian and Adashi, 2021). Every computer system is vulnerable to cyber-attacks due to different viruses and malware that are available in the cyberspace.

Nevertheless, computer users can evade malicious data alteration to their computer systems by being cyber intelligent, knowing how the malware attacks systems and the method it uses to spread in the computer. The phrase malware is a general term that explains threats including Trojan horse virus, worms, and other system viruses. Every time a person launches an internet connection for activities not limited to reading emails or sharing files over the web, the personal computer systems become vulnerable to malware attacks (Lipson, 2002).

Once a malware attacks a computer system, it causes malicious damages to the system's boot files, data folders, software and also the System BIOS boot sector. Consequently, it corrupts essential data, and the computer system is forced to shut down. The core issue is that these malicious software programs are programmed to spread instantly into different computers in a few seconds (Hart et al., 2020). With the number of malware attacks, human intervention is not enough for timely attack analysis and proper response.

In the early 1960s was when businesses and organisations started proactively protecting their information systems by using passwords and several levels of security (Murphey, 2019). Ray Tomlinson, an American software engineer, created "Reaper," in 1971 as the first antivirus program, to fight the creeper virus (Terekhov, 2017). After his invention, two German companies - G Data and Ultimate Virus

Killer (UVK), followed suit and produced their own anti-virus software. Then the 1980s came and Network firewalls were created to protect networks against cyber-attacks and improper network configuration. The primary purpose of firewalls was to protect internal networks from external and online threats (Ingham and Forrest, 2006). Even though information security has advanced, attackers continue to infiltrate systems by finding new flaws and deploying highly advanced tricks to evade detection by the security measures already in place.

Researchers from government, academia, and industry began putting together new strategies in early 2004 to explore how they can use threat intelligence to identify, classify and respond in defence to attacks, using techniques like tracking vulnerabilities and Internet Protocols' (IP) reputation, even prior to the use of the term, as cybercriminals became more skilled (Cook, 2004). Over the past century, intelligence has given rise to Cyber Intelligence (CI), Warner (2017) opines that intelligence according to the US military doctrine, is information that a commander deems essential for decision-making along with the sources, techniques, and procedures employed to obtain such information. Whereas, CI is defined as the knowledge and techniques developed to guard against unauthorised access, system flaws, and hacker attacks from cybercriminal organisations, cyber terrorist groups, and other online threats (Conti, Dargahi, and Dehghantanha, 2022). CI is related to protecting firms' network infrastructure, internet connections, and data from interference and unauthorised access.

CI obtains intelligence in different ways including Open-Source Intelligence (OSINT), CI (CYBINT), and Human Intelligence (HUMINT) (Aditya and Enbody, 2014). Intelligence gathered from these sources provide insight into financial frauds, application vulnerabilities, compromised or weak credentials, insider threats, social engineering of malware attackers (Wang et al., 2023). How much CI is important is now acknowledged, and also the fact that cyber-attacks persist because cyber criminals leverage on the reality, that organisations and government practise outdated CS and are now concerned about it. CI became acknowledged as a significant concept in Cybersecurity in 2014 (Poputa-Clean, 2015) and CI collection

and analysis platforms became mainstream, using system logs, threat databases and various repositories to provide intelligence (Aditya and Enbody, 2014). These platforms however have limited capabilities that limit their ability to deal with malware.

When considering the advantages of CI, the factor of real-time delays in sharing is critical as it is caused by the reliance on third-party CI firms, the monopoly of CS providers, and the community-based manner of sharing intelligence in bits. Most cyber defenders rely on reactive strategies, and responds only after a Cyber-attack has occurred, thereby losing the battle at the initial stage of defence. Even though every source of cyber-intelligence available today claims to offer real-time knowledge sharing, the problem of doing so is widespread. Information is shared manually via email, conferences, phone conversations, web portals, data streams, and shared databases (Wagner et al., 2019). These issues mean that CI platforms are not equipped to counter the most urgent challenge of malware, in real time.

This research addresses the limitations of the current CI systems and models, utilizing existing literature explicit about the challenges of CI sharing. By doing so, its rationale is focused on identifying that malware is a prevalent form of cyber-attack challenging organisations, governments and individuals. Malware also spreads more rapidly than most cyber threats, and so a great case study for developing effective CI system or model. Even though malware is focal to this study, the solution developed is relevant across broader application of CI.

The solution is to develop a model to gather, process, analyse, and share malware-related intelligence (data) in real-time, thereby enhancing preparation for defence against potential cyber-attacks. This research underscores the significance of CI as a subject matter for professionals globally. To achieve its objectives, this study developed an automated malware detection model with machine learning techniques and integrated it into the Intelligence Life Cycle (ILC) framework. This framework was used to illustrate the elements of the new proactive CI model and to assess the extent to which it contributed to CI or cyber threat intelligence (CTI) by defending against emerging cyber-attacks.

1.1 Research question

What are the main problems that make it hard for businesses to share CI effectively? How can a real-time automated CI sharing model that uses ML algorithms help to solve these problems and make businesses operate safely?

1.2 Problem statement

Businesses today face a rising number of common malware-attacks, and the damage they do is enormous. Malware-attacks and other cyber-crimes have been in existence since early 20th century, and in recent times, cybercriminals have presented huge risks through malware-attacks targeted at critical system infrastructure (Ericsson, 2010). According to Leyden (2008), RBS Worldpay acknowledged that cybercriminals broke into their web systems, and tampered with electronic payment services, which include payroll cards and PINS for All PIN-enabled cards. The damages of the electronic payment services stretched to about 2.6 million US dollar (Leyden, 2008). Sega gaming company announced in 2011 that 1.3 million customers' records had been hacked by cybercriminals called bizarre twist and LulzSec group, which were the same groups that claimed responsibility for hacking Sega and Sony before (Cohen, 2011).

2.4 million emails were sent from the Syrian administration officials and companies, and a new release which the publisher titled Syria Files was announced during London press conference. The file contained information about Syrian dictator Al-Assad and the opposition forces, and several of them were the information about how many people that died in the civil war in Syria (Greedberg, 2012). The number of cyber-attacks tracked in that year totalled 781, according to a report available by the Identity Theft Resource Centre (ITRC) and sponsored by IDT911. The report also stated that this data breach became the second-highest data breach on record since they started tracking breaches in 2015 (Idtheftcenter, 2015).

51.1% of organisations worldwide were targeted by ransomware in 2018, 56.1% in 2019, 62.4% in 2020, 68.5% in 2021, 71% in 2022, and 72.7% in 2023 (Statista, 2023). Nevertheless, it will be fair to say that the advantage of having the

internet could be overshadowed by the damages caused by cyber attackers. (Arenas, 2017). Businesses and governments need to consider the above phenomena from the perspective of an attacker and apply the same knowledge that hackers do when they leverage systems for malicious purposes. Applying that knowledge is now crucial for corporate and government organisations in order to protect themselves against impending Cyber-attacks. Proliferation of Cyber Intelligence (CI) has grown as an entirely new instrument for cyber defenders to act against increasing attacks from hackers (Wagner et al., 2019).

The automation, and standardization of cyber information exchange is fundamental in improving efficiency but has introduced new issues (Vázquez et al., 2012). Furthermore, it is regarded as essential to stop present and forthcoming attacks by adopting a proactive rather than solely reactive approach (Sigholm and Bang, 2013). According to Vázquez et al., (2012) it is challenging to establish a CI system that efficiently employs and shares knowledge timely. In addition, companies encounter difficulties in establishing a mechanism that effectively utilizes competitive intelligence and that makes the information useful. The principal barrier that many industries encounter prior to disseminating their own CI is effectively using information, specifically understanding it and applying its solutions. Researchers found that businesses desire to be involved in an efficient and automated sharing process, but inadequate models and instruments complicate this endeavour.

Although CI is available for both commercial and free open source use and is crucial for business and government, it can be difficult to share intelligence from companies to businesses and from government to government. Currently, there are diverse systems of CI sharing available for free or commercial based, however academic researchers have noted the challenges and requirements for CI sharing platforms (Sauerwein et al., 2017). While, the concept of Threat Intelligence Platforms (TIPs) was launched with complete requirements to facilitate information sharing, enable automation, and facilitate the generation, refinement and vetting of data (Dandurand and Serrano, 2013), current CI sharing solutions are biased, and

different software developers are presenting loopholes while prioritizing their products (Brown et al., 2015).

Sharing CI among enterprises and governmental organizations could be difficult, even though it is established to defend systems against Cyber-attacks. Unfortunately, there are a lot of psychological and technical barriers to sharing intelligence, which results in a general lack of interest (Brilingaitė et al., 2022**).** First, there are many group strategies of the accidents such as ATT&CK (Corporation, 2020; Brilingaitė et al., 2022). Secondly, every system that shares information regarding threats is unique and different, and for consumers to safely make use of the information, they must comprehend even the unclear requirements. In addition, for this data to be used at all, both the sender and the recipient ought to have the same interpretations because by implication, it is expected that Law and privacy be respected, and knowledge be trusted. It is therefore pertinent for experts in computer science to discover methods of getting around the challenges. However, the largest obstacles that suppliers of CI sharing are encountering include automation, standardisation, real-time CI sharing, and connection with infrastructure security systems.

The Directive (EU) 2016/1148 report indicates that sharing new CI is recognised as the key issue when preventing cyber-attacks. In Great Europe, the European Union Agency for Network and Information Security (ENISA) began a progress collaboration between Governmental and National Computer Security Incident Response Teams (CSIRTs) (Official Journal of the European Union, 2016). The Executive Order No. 13691 produced by American president also provides for cooperation between the Department of the Homeland Security and the private information security companies (Administration of Barack Obama, 2015).

The fact is that conventional procedures of protection are no longer adequate to respond to the emergence of new cyber threats. It is essential to research more knowledge strategies that will automatically empower CS experts and those interested in the new CS adventure, on the levels of CI that can make for stronger and effective layers of security prevention (Navetta and Mathur, 2015).

Furthermore, strategies require universal collaboration tactics on how to standardise intelligence sharing and trust to exchange cyber threat information. The CS community has developed a novel approach of a promising CI-based Cybersecurity Co-operative Research that specifically cover information gathering, analyses, and sharing of CI. Regrettably, these plans have been disorganised, and delayed by lack of an internationally coordinated front that includes businesses, institutions, governments, and the public towards eliminating the cause of the problem. From the viewpoint of CS, it seems to be interoperability issues between the research and development framework, as business institutions point towards the CI sharing and research method put in place (Dwyer, 2009). It is important that the implementation takes a standard holistic method in its action while analysing and sharing CI data more efficiently and in real time (Johnson et al., 2016).

Despite the widespread use of CTI, businesses around the globe still have difficulty properly making sense of the vast amount of raw data they gather and using same to create pertinent cyber threat information (Sillaber et al., 2016). The absence of standardization, and automation for evaluating CTI programs and the sharing of information in a timely, effective, legally-binding, and privacy-enabled manner are other challenges.

1.2.1 Challenges Identified

The study has highlighted several factors that make it challenging to share intelligence information (Johnson et al., 2016). According to the National Institute of Standards and Technology (NIST) report, part of challenges that exist between consumers and the producers of CI information sharing platforms include:

Building Trust: Trust connection between different institutions based on intelligence information sharing needs to be created and maintained by determination. Current communication methods between the companies are not helping cyber information sharing (Lipson, 2002).

Accessing external CI information: Businesses require infrastructure to be able to obtain external intelligence sources, but they might not have the capability

to access external threat networks. Retrieving CI externally and leveraging the local decision-making processes will prevent the organizations from being attacked according to the NIST report (Johnson et al., 2016).

Legal: According to the Information System Security Association (ISSA), exchanging specified private CI information, or even unspecified ones, can pose a legal and operational risk. For example, an announcement made on an intelligence information exchange, even when unclassified, can be traced back to the original corporate personnel and used legally against the firm in question (Navetta and Mathur, 2015).

Quality: Most the threat intelligence suppliers concentrate on just gathering and sharing more threat data, but the risk of duplicated information is high as well as spending valuable time and energy without result (Sillaber et al., 2016). The issue with data quality is a drawback for open-source intelligence (Amoroso, 2011). While most commercial or private sources of threat intelligence are only available for a fee.

Speed: CI came in to help companies to prevent the cyber-attacks late. Report indicates that sharing intelligence data in real-time could very much help organisations match or exceed the speed of cyber-attacks (Johnson et al., 2016).

Identity: Cyber-attackers can provide false threats to deceive CI devices, and data from well-known sources can be changed if poorly handled. Not identifying the appropriate patterns and critical information points in the threat data, creates the possibility of that data not turning into intelligence and then into valuable knowledge that would provide valuable real-time intelligence for security professions.

Sharing intelligence through a community: In a community-based sharing technique, CI is gathered, processed, and evaluated but stored in one place for all. The disadvantage of this community sharing, that is centred on pushing information between community members is that, it might no longer be accurate or usable intelligence once it has been shared.

Within these circumstances, if one considers the current trends of cyber-attack as essential and the cyber warfare emerging, then one cannot abandon the significance of implementing a winning real-time cyber response attitude. It is required for security professionals to remain continuously ahead of the cybercriminals in relation to military plans and implementation. From the standpoint of CI, real-time attitude can be called real-time CI defence and sharing by which defenders would engage the cybercriminals by implementing real-time intelligence gathering, analyses and sharing techniques. They could continuously generate new knowledge to safeguard bulletproof protection of cybercriminals' infrastructures. Nevertheless, this plan needs validation over consistent measurement and serious assessment.

1.2.2 Novelty and Comparative Advantage of PSDCIS Model

Unlike TAXII/STIX-enabled MISP platforms that focus on regulating the set-up of CTI and powering community sharing structures among organisations, the PSDCIS model brings a different approach. This novel model has no use for community-shared data, or already processed data rather it fully automates the process using finite state machine-driven system. This system in real time, scrapes, classifies and validates malware signatures with the help of supervised machine learning models and this makes it non-dependent on external validation or other threat repositories. Therefore, intelligence is generated, shared and consumed accurately and at the same time. Accuracy is enhanced hugely by the continuous retraining of models. Due to its custom features and capability for of real-time processing, automation and supervised learning, the PSDCIS model solves the problem of duplication, dormancy and limited useability that TAXII/STIX-enabled platforms are known to have. The PSDCIS model brings with it, practical novelty that makes sharing of CI proactively effective.

Table 1: Comparative Analysis of PSDCIS and TAXII/STIX-enabled MISP Platforms

Dimension	TAXII/STIX-enabled MISP Platforms	Proposed PSDCIS Model
Design	Community-driven threat data sharing with standardised setups (STIX) and transport protocols (TAXII).	Automated finite state machine (FSM) system integrating supervised ML models (LightGBM, Decision Tree, XGBoost).
Automation	Requires human oversight for collection, classification, and sharing/distribution.	Fully automated intelligence collection, classification, and sharing/distribution without human intervention.
Data Source	Primarily external data and submissions by members to community repositories.	Direct web scraping from Kaggle, VirusShare, and other sources; real-time malware signature validation.
Speed	Sharing may be delayed by community validation, duplication, or policy restrictions.	Intelligence is processed and disseminated in real-time, matching or exceeding the speed of cyber-attack.
Trust Dependency	Relies on inter-organisational trust and governance frameworks for participation.	Removes external trust dependency by internally generating, verifying, and

Dimension	TAXII/STIX-enabled MISP Platforms	Proposed PSDCIS Model
		sharing/distributing intelligence.
Accuracy	Risk of redundancy, low-quality, or outdated threat data.	High accuracy through continuous ML-based classification and VirusTotal confirmation.
Scalability	Limited by governance, subscription models, and centralised repositories.	Scalable through automation and modular FSM states, adaptable to organisational infrastructure.
Cost Model	Often commercial, subscription-based, or restricted access.	Designed to operate in-house with minimal recurring cost, leveraging open datasets and APIs.

Source: Researcher's Review 2025

Research significance

This research focuses on the development of the novel model – PSDCIS, which mitigates malware attacks by seamlessly providing real-time, automated sharing of CI, using ML algorithms for processing and analysing data, integrating intelligence into company security systems, and sharing same with system administrators through email.

1.2.3 Enhanced detection and response of the cyber-attacks

Organisations can improve their capacity for identifying and mitigating cyber risks by adopting real-time automated solutions combined with ML algorithms. They could immediately identify emerging risks and execute strategies to mitigate

their effects by consistently gathering and analysing intelligence from external data sources (Rookard and Khojandi, 2024).

1.2.4 Better Decision-Making

ML algorithms have the capability to manage substantial amounts of data efficiently and specifically, enabling, managers and system administrators to learn helpful insights into emerging cyber threats. This enables the capability to make more well-informed conclusions and have active stance towards CS in their information systems (Sarker et al., 2020).

1.2.5 Tailored cyber-Intelligence

ML algorithms can be trained to categorise and rank risks according to their seriousness and pertinence to the organisation. This guarantees that system administrators have intelligence that is customised to their particular security requirements and preferences (Martínez Torres et al., 2019).

1.2.6 Real-Time Sharing and Integration

With integration of automated solutions into company security systems, the sharing of threat intelligence can occur instantaneously, allowing for prompt measures to be implemented to reduce the impact of potential threats. This seamless connection guarantees that threat intelligence is efficiently employed to improve the organisation's overall security stance (Haque and Krishnan, 2021).

1.2.7 Efficient Communication

Sharing intelligence to system administrators using email guarantees that crucial cyber threat information is conveyed directly to the relevant staff promptly. This direct communication line enables prompt response and implementation of mitigation measures (Rafaeli and Ravid, 2003).

1.2.8 Scalability and Flexibility

Real-time automated solutions, powered by ML algorithms, are highly scalable and adaptable to the evolving cyber threat phenomena. They can handle

large volumes of data and quickly adapt to new threats and attack vectors, ensuring that CSS remain effective over time (Harish Padmanaban, 2024).

1.2.9 Reduction of False Positives

ML algorithms can help reduce the number of false positives by accurately identifying and classifying genuine threats. This ensures that system administrators can focus their attention and resources on addressing the most critical security issues (Watson and Holmes, 2020).

1.2.10 Continuous Improvement

Automated solutions powered by ML algorithms can continuously learn from new data and feedback, allowing them to improve their threat detection capabilities over time. This ensures that organisations remain ahead of cyber threats (Chen et al., 2024). This study is significant because when all these challenges are addressed through research and the ML algorithms develop automated real-time solutions, organisations can significantly enhance their CS strategies, better protect their assets, and mitigate the risks associated with cyber threats.

Reviews of extant literature recorded a dearth of evidence on models, literature, or systems that embrace proactive defence system integration and real-time CS intelligence sharing. No true solution has been put in place to address this issue, even though most studies available identified the difficulties, advantages, and necessity of real-time cyber information sharing. Meanwhile, this gap is an overarching problem that enterprises and government agencies are facing. Furthermore, existing open-source and commercial CI sharing services focus on collecting, processing, analysing, and storing intelligence without integrating information systems and sharing in real-time. The issue of integration led to a lack of uniformity between information security systems and intelligence providers. Thus, this research is noteworthy from this perspective. Likewise, it can be argued that the effective deployment of the Proactive Self-Defence Cyber Intelligence Sharing (PSDCIS) model-based on state machine automation which was developed during this research, will provide businesses and government organisations with the

ability to identify, counter, and react promptly to potential Cyber-attacks. This will protect them against potential harm or loss of data resulting from Cyber-attacks.

Similar research

Comparable scholarly publications, papers, and subjects related to this research have been examined as seen in Table 1. Additionally, the distinction between this research and summaries of other related studies have been tabulated.

Table 2: Similar Research Papers

S/N	Research topic	Authors	Research conclusion
1	CTI Issue and Challenges.	Abu et al., (2018) research article published by Indonesian Journal of Electrical Engineering and Computer Science.	This research focused on the present status of development for the commonly used language and technologies in the field of CTI. Furthermore, there are obstacles that have been discovered in the field of CI, such as issues with data quality and the need for a CTI sharing platform. Additionally, there is a call for a community-based CTI exchange programme, which may be facilitated through research-based questionnaires. This identified the challenges and concluded that further research is required.
2	A Theoretical review on the importance of Threat Intelligence Sharing and the challenges.	Sukhabogia et al., (2021) Research Article published by Turkish Journal of Computer and Mathematics Education.	This study found that there are multiple sharing platforms now in operation. However, regrettably, these platforms mostly prioritise the gathering of data rather than its analyses. This study recognised the current obstacles within the CI domains but did not provide any solutions to address these difficulties.
3	Cybersecurity and Information Sharing: Legal Challenges and Solutions	Nolan (2015) report published by Congressional Research Service prepared member of committee and congress	This study investigated the many legal concerns that occur when sharing CS intelligence between the government and private sector. This paper suggests an analysis of the key legislative proposals, which include the Cyber Intelligence Sharing and Protection Act (CISPA), Cybersecurity Information Sharing Act (CISA), and the Cyber Threat Sharing Act (CTSA).
4	From Cybersecurity Information	Brown et al., (2015)	Researchers have once again discovered ongoing challenges in the CI exchange, mainly pertaining to basic issues around data sharing. These challenges are mostly related to concerns regarding privacy

	Sharing to Threat Management		and legal considerations. However, they did not offer a solution for those issues; instead, they acknowledged that further deliberations and investigations were necessary.
5	Overcoming Information - Sharing Challenges in Cyber Defence Exercises	Brilingait et al., (2022) Published by Journal of Cybersecurity	The study identified nine key variables that are detrimental to the prioritisation and execution of risk tasks. These issues include an excessive concentration on technical tasks, inadequate understanding of the necessity of information-sharing, lack of knowledge about relevant standards and tools, and poor awareness of the benefits associated with these abilities. In addition, the study proposed that research could be devoted to the development of automatic analysis of risk artefacts to evaluate the quality of the shared information.
6	Sharing Cyber-Threat Information: An Outcomes-based Approach	Willis (2012) published by Intel Corporation	This study examined the advantages of exchanging CI among corporate and public partnerships and highlights the necessity of sharing CTI. Furthermore, the study identifies significant challenges. Issues of sovereignty and the existence of national and regional laws can impose restrictions on the exchange of information between nations. This study suggests that there are key areas of opportunity that should be focused on. These areas include automating the process of collecting and acting on shared information, creating secure collaboration platforms, and improving analytical capabilities.
7	How to exchange security events? Overview and evaluation of formats and protocols.	Sauerwein et al., (2017) published 13th International Conference on Wirtschaftsinformatik	This study applied a cross-validation technique to validate the barriers that exist in CTI platform. The research studied 22 different software platforms and produced an exploratory study. Findings indicate a growing interest in the sharing of threat information. However, the study and practice in this area lack a uniform definition of what exactly constitutes threat intelligence sharing. Furthermore, the current state of threat intelligence sharing is like data warehousing and does not offer genuine intelligence.
8	Tools and Standards for CTI Projects	Farnham and Leune, (2013)	This paper evaluated the existing CI solutions available in the market for managing and sharing intelligence, as well as the issues associated with them. The management of continuous integration

	GIAC (GCPM) Gold Certification		(CI) is widely acknowledged as a problem, and there are numerous efforts underway to address it, it states. There are numerous standards now established or being developed for CI. This study suggests that businesses should select tools based on the specifications of their systems, as there is no universal standard that encompasses all available tools.
9	Cybersecurity Intelligence Gathering: Issues with knowledge Management	Gyamfi and Williams, (2018). Published IGI Global and researchgate.	This study applied the agility of the Scrum methodology to collect cyber-intelligence, which was then leveraged to develop the KM-CIG model for decision-making purposes. Intelligence, as a process, usually entails gathering data and then refining it into information that can be used and acted upon. The study also identified "data inconsistencies, security and privacy, communications with end-users and stakeholders, organizational culture, Governance-Risk-and-Compliance GRC), competitive advantage, business continuity plan". Study recommends: "It is essential to guarantee that the intelligence data is contextualised, meaning that it is relevant and directly applicable to the decision-making process in CI. The information must accurately determine how well the data satisfies the specific needs and requirements of the decision-making process".
10	Artificial Intelligence in Cyber Defence	Tyugu and Branch, (2011)	This study focused on how artificial intelligence (AI) is used in cyber defence domains. In general, AI can be considered in two ways a) Science aimed to discover essence of CI, b) Science providing methods for solving complex problems such classification and recognition of malware. The study indicates that the application of ML will significantly improve the cyber defensive capabilities of systems, particularly in the areas of data processing, analysis, and classification.
11	Malware Analysis and Detection Using ML Algorithms	Akhtar and Feng, (2022). Published by School of Computer and	This article addressed the pivotal significance of ML algorithms in analysing malware signatures and accurately identifying dangerous software with minimal false positives, hence enhancing accuracy.

		Communication, Lanzhou University of Technology	
12	Performance Evaluation of ML Classifiers in Malware Detection	Nikam and Deshmuh, (2022). Published by 2022 IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE)	This paper analysed the performance of different ML algorithms by utilising a dataset obtained from an external source. The objective is to determine which methods are most effective. The performance of these algorithms is assessed by measuring accuracy, AUC (Area Under the Curve), false positive rates, and false negative rate.
13	Model for Sharing the Information of CS Situation Awareness between Organizations	Kokkonen et al., (2016). Published International Conference on Telecommunications (ICT)	This model aimed to address the issue of information sharing between business and governmental organisations by utilising the STIX and TAXII standards. The developers of the model believed that these standards are commonly used in the field of CI sharing. Consequently, they created a model named the topology of information sharing community with the lowest risk level has been calculated with Dijkstra's algorithm. Furthermore, they have recognised that one of the most difficult challenges in information sharing is the real-time of intelligence. They determined that future implementation of the model with various case scenarios is necessary, and the ongoing development of the model is progressing.

Source: Adapted from (Kokkonen et al 2016; Nikam & Deshmuh, 2022; Akhtar & Feng, 2022; Tyugu & Branch, 2011; Gyamfi & Williams, 2018; Farnham & Leune, 2013; Sauerwein et al., 2017; Willis 2012; Brilingait et al., 2022; Brown et al., 2015; Nolan 2015; Sukhabogia et al., 2021; Abu et al., 2018.)

Research Aim and Objectives

The study investigates the current barriers that prevent the efficient exchange of CI across businesses, with a goal to enhance CI sharing by creating and deploying a real-time automated CI sharing model that will be integrated into companies' security systems and enable swift response to potential cyber threats. The study encompasses five objectives which are to:

1. Establish the current challenges in the exchange of CI in organisations through review of extant literature;
2. Assess the existing models, systems and processes for sharing and exchange of CI across industries.
3. Develop an automated model for sharing CI in real-time for data processing and analyses.
4. Integrate company security systems (CSS) by implementing the newly created CI sharing model towards guaranteeing a prompt response to possible cyber threats.
5. Develop recommendations for enhancing real-time CI sharing capabilities and overcoming existing challenges, while improving CS and enabling organisations to leverage collective intelligence.

Layout of Thesis

This thesis is structured and presented in seven chapters, ranging from chapter one to seven. Chapter one contains the introduction of the study, research aim and objectives (RA and RO), research questions (RQ), problem statement, research significance and identification of similar research. Chapter two provides background information on the numerous types of Cyber-attacks and malware attacks that occur in the global cyberspace, as well as an overview of the models and systems currently in use for CI sharing as well as the challenges it faces. This chapter also covers information collection, intelligence sources, malware detection techniques, and machine algorithms used to classify malware and benign files, and intelligence sharing mechanisms. In chapter three, the first sections cover research methodology overview, provides an overview of the various research theories, and identifies the research design that is most relevant

to this research. It also provides an explanation for the pursuit of a quantitative research approach. The research technique is also explained in detail in this chapter, including how data was collected, handled, examined, and shared. Chapter four covers the theory behind creating the Proactive Self-Defence Cyber Intelligence Sharing (PSDCIS) model. The model includes sub-models for gathering, processing, and analysing cyber intelligence as well as sharing intelligence with managers, system administrators, and security systems.

Chapter five presents the real-world implementation of the research deployment model and converts theory into practical application by using state machine automation process. In addition, this chapter explains deployment processes and tools required to build the deploy model as well as the automation process of the proposed model in a live environment. Chapter six explains the problems that come up when sharing CI and how the proposed model will solve them. It also dwells on the results of training and testing the model, how sharing intelligence in real time solves problems, the results of deploying the model, the research's limitations and what needs to be done next. It ends with some suggestions. Chapter seven addresses the study's conclusions, suggestions for more study, and future directions.

CHAPTER 2: LITERATURE REVIEW

Overview

When malware kits became increasingly common in late 2006, the malware threat landscape changed. Early kits, like MPack, were victims of their success since they were unable to keep up with the rapidly expanding demand. Nonetheless, services were able to address these growing issues quickly enough, and people became aware that there are several exploit kits available on the dark web.

However, attacks and exploit kits, which target businesses and individuals in America, Europe, and Asia, continue to rise in popularity as cyber threats. Not only were they perceived, but in 2015 there were signs that they significantly grew their volumes and varied assault vendors (Trabelsi et al., 2016).

Although the Downadup worm (also known as Conficker) is getting outdated, its position as one of the most reliable indicators for detecting system vulnerabilities has kept it more prevalent than it would have been without CI. The CS sector is currently more aware of malware threats due to the emergence of numerous new worms that have successfully breached networks in various parts of the world. From the first time viruses were discovered until the present day, the scope of cyber-attacks has developed rapidly, and every year defences against cyber-attacks have improved, but have been unable to stop the most recent attacks.

The widespread CS issues that businesses encounter today are a result of the haphazard and inexperienced choices made in the early days of the World Wide Web. The opportunity to take advantage of such a new technological advancement increased the information that significantly altered human civilization on an economic, political, and social level. Nonetheless, it is accurate to claim that the benefits of having access to the internet may outweigh the harm caused by cybercriminals (Arenas, 2017). CS professionals have developed affordable solutions in recent years to achieve a trustworthy and secure internet, driven by a spirit of teamwork and shared obligations. Such outcomes are putting the challenges out there today. Currently, there is a huge variety of applications being used on the

internet due to human beings' unbridled desire to push the web to its limits. Internet of Things (IoT), cyber-physical systems (CPS), and enterprise infrastructure, are the latest development of the internet (Dwyer, 2009). These technologies introduced new weaknesses that fraudsters, cyber attackers, and professionals in cyber-warfare are yet to exploit, making them far from immediate vulnerability to Cyber-attacks. It is very important to look into more advanced knowledge techniques that will automatically empower CS specialists, people looking for new CS adventures, levels of CI that can make users feel safe, and provide an active layer of security (David Navetta and Utsav Mathur, 2015).

In addition, to standardize intelligence sharing and build confidence in exchanging information about cyber threats, strategies call for an all-encompassing technique of collaboration. The cyber-intelligence community, has created a revolutionary method, from cyber-intelligence-based CS cooperative research that focuses on information gathering, analysis, and sharing. Unfortunately, the absence of a globally unified front that involves industries, institutions, governments, and the public to address the root of the issue, has resulted in these initiatives being disorganised, proprietary, and delayed. From the point of view of CS, there seem to be issues with how the research and development frameworks work together. Commercial institutions point to the need to set up a way of sharing CI and conducting research (Dwyer, 2009). It is imperative that a uniform, comprehensive approach be put into practice to analyse and distribute CI data more quickly and effectively (Chris Johnson; Lee Badger; David Waltermire; Julie Snyder; Clem Skorupka;, 2016).

This chapter explores current literature in the area of Cyber-attacks and malware attacks that occur in the global cyberspace. It also provides an overview of the existing models and systems currently in use for CI sharing and attendant challenges being faced. This chapter also covers information collection, intelligence sources, malware detection techniques and intelligence sharing mechanisms.

Current Challenges and the Intelligence Needed to Prevent Cyber attacks

In order to prevent cyber-attacks, a variety of actions must be taken. They include understanding behaviour and tactics of threat actors and their strategies such as ransomware, social engineering, phishing, password cracking, man-in-the-middle interceptions, SQL injections and zero-day exploits. Also identifying areas that can become easily vulnerable in applications, IoT devices, operating systems, network protocols, etc. Understanding in details the signatures of malware and their classifications eg Trojans, viruses, worms, spyware, and rootkits, which will help fast detection, and defence. Another aspect is knowing the patterns of incidents and the propagation techniques of attack vectors such as remote desktop exploitation, website infiltration, email attachments and botnets. Knowing the assets, and infrastructure that attackers target, as well as tracking their tools and technologies like uncovering malicious codes they have in their hidden networks is also important. Another aspect is having insight into the financial motivation that drive attacks, so as to stop credit card theft, or even breaches that are motivated politically. From the foregoing, prevention ultimately depends on automated, real-time CI gathering, analysis, and sharing that tackles feeds from all possible threat repositories, and defending systems against threats before attackers succeed.

Review of Exploratory Survey Study

CI gathering and dissemination is divided in three main parts spanning model concepts and taxonomies that make up its use, automated methods that create, filter and transform data into structured intelligence and thirdly standards that analyse data using machine exchange. There are operational models that have established intelligence to cyber defence activities. One of them is the cyber kill chain that modifies incidences of cyber intrusion as campaign stages and advocates for it as an intelligence-driven disruption of campaigns by cyber threat actors (Hutchins, Cloppert and Amin, 2011). The cyber kill chain later moved CTI to become less static as an indicator and become a behavioural and campaign artefacts in its support

mechanism (Hutchins et al., 2011). MITRE ATT&CK and heuristics such as Pyramid of Pain which are frameworks for cyber attackers' behaviour are used as priority scales for CTI generation and sharing as well as for analyses because it makes it difficult for adversaries to penetrate (Strom et al., 2020; Bianco, 2013). Efforts at standardising operations has made CTI exchange easy to manage. According to Barnum, (2014), STIX established a structure for relationships, units and observables, and so became the main language for most CI sharing platforms. Other studies and surveys highlighted technical, social and political barriers to CI sharing especially how fragmented incentives, access control and trust are. Several research have focused on automation of CI generation and analysis, just like several works that propose systemic frameworks that spot incidents, map raw data to STIX, analyse sensitive datasets and distribute STIK through TAXII endpoints (Sadique et al., 2018, Haque & Krishnan, 2021).

An exploratory survey by Alaeifar et al. (2024) identified issues regarding CI sharing stating that there is no universal definition of CI and this made CI-sharing practices by some organisations not scientifically standardised. They use common sense instead and that makes knowledge transfer complicated and sharing intelligence difficult.

2.1.1 Research Finding 1: No universal definition of cyber intelligence sharing

Despite the existence of models for representation such as Structured Threat Information eXpression (STIX) and exchange standards, for example Trusted Automated eXchange of Indicator Information (TAXII) of CI, inquiry and study have not yet produced a wide-ranging interpretation and shared an understanding of what constitutes a cyber-intelligence-sharing program, Sauerwein et al. (2017) identified distinct types of CI-sharing platforms and eight recognised cyber companies emphasized sharing CI between businesses. On the other hand, seven other instances show that only data was shared not CI in its strictest analytical form. The data exchanges automatically combined different available paid and open-

source information security data foundations. One of the identified sharing platforms offered a mixture from a cyber-intelligence sharing method, where data from the different data sources added together with CI were presented by the contributor company. Furthermore, the contributors identified four tools which contain a primary repository that offered beneficial intelligence information that is exact-in-context such as information about any discovered malware behaviour, virus patterns etc. Two other platforms focused on exchanging technical information (Sauerwein, et al., 2017).

2.1.2 Research Finding 2: STIX is the de-facto standard CI structure

Existing studies show that the landscape of standards available and used to describe and share threat data is limited in comparison with the number of vendors sharing CI (Barnum, 2014). The research shows that most CI interactions rely on measures such as OpenIOC, STIX, and IODEF. STIX was identified as the de-facto standard for relating threat- related information and CI. More than two-thirds of the surveyed platforms have import and export data skills and support the criteria. In detail, ten platforms depend on STIX, two on Open Indicators of Compromise (OpenIOC), two on both, and one platform on Incident Object Description Exchange Format (IODEF). For example, the Open Threat Exchange (OTX) platform provides STIX as well as OpenIOC import and export functionalities. The STIX platforms contain eight cores of cyber-attackers' ideas as autonomous with reusable ideas, taking their interrelationship into account. The eight constructs include cyber observables (e.g., IP addresses, file names, hashes), indicators, incidents, adversary strategies and methods, and courses of actions (including attack patterns, kill chains, etc.), exploit targets (e.g., vulnerabilities, weaknesses), sequences of achievement (e.g., incident response, mitigation strategies), cyber-attack campaigns, and cyber threat actors. In confirmation of STIX's abilities, all these constructs can be found in all analysed platforms.

2.1.3 Research Finding 3: Platforms Focused on Sharing Compromise Indicators

The CI-sharing platforms give attention to exchanging indicators of compromise (IoCs). IoCs include intelligence that allows the discovery of malicious actions, for example, the names of exploited companies, suspected IP addresses, abnormal user behaviours, files that are malware infected, etc. Open Threat Exchange (OTX) is an example that focuses on sharing the threat indicators in support of detecting malware activities. While the OpenIOC standard is intended to share threats, the analysed platforms use the STIX's Observable and Indicator concepts to define and disseminate them (Barnum, 2014).

2.1.4 Research Finding 4: Closed Cyber Intelligence Platforms.

Researchers have identified twenty-two cyber-intelligence exchange platforms available is on the marketplace. Six among the platforms are free to use, whereof four are open-source tools created under the GNU General Public License. They are the Malware Information Sharing Platform (MISP), Collective Intelligence Framework (CIF), Collaborative Research into Threats (CRITs) and MANTIS Cyber-Intelligence Management Framework. The Open Threat Exchange (OTX) and Soltra Edge platform in contrast are free-to-use but were not released under an open-source license. The outstanding 16 platforms are closed-source, making CI infrastructures more proprietary (Barnum, 2014).

2.1.5 Research Finding 5: Most CI sharing platforms focus on data collection instead of analysis

CI proposed by most CI-sharing companies does not use scientific know-how to process outcomes in a structured way as in the context of data security intelligence. CI is supposed to be the product of an intelligence lifecycle model that includes various actions like planning, data collection, analysis, and dissemination Siri Bromander et al. (2020). Instead, most tools fundamentally concentrate on the data collection (information gathering) stage, and ignore the other essential activities of the CI procedure – transforming data into actionable intelligence.

Consequently, most CI platforms are mere data warehouses than real CI-sharing programs.

Furthermore, researchers have observed that programs available have inadequate cyber-attack analysis and data visualisation abilities which are necessary for knowledge sharing platforms and data mining solutions from other specialities. This is so far exceptional as the value of these platforms are constrained by the user's inability to interpret and act on the threat intelligence (Sander and Hailpern, 2015). Besides, only a few of these platforms offer easy connection to third-party tools that would allow additional analysis of the collected CI information. Current CI sharing programs provide very limited analysis capabilities, such as browsing, attribute-based filtering and searching of information. Only a few fractions of platforms implement pivot functionalities, which allow the visualisation of the relationships between the CI constructs.

2.1.6 Research Finding 6: Trust issues between users and cyber-Intelligence sharing providers

For CI sharing platforms to be effective, research findings indicate that it depends on the presence or absence of trust between the organisations contributing to the program, and the providers of the intelligence exchange. Organisations may share private or sensitive information, so there needs to be trust between the information processing industries and the company that offers CI sharing, to prevent mishandling. This is indicative that consumers of CI-sharing companies require caution when dealing with intelligence presented.

Additionally, restricting data access to these intelligence companies is necessary to prevent any unauthorised usage. For example, exchanged CI might be of huge benefit to cybercriminals. To succeed in addressing trust issues and controlling access, CI sharing platforms must implement control mechanisms to define what knowledge to exchange, how to share, and with whom. Additionally, controlling access to information is necessary since these platforms might be of potential interest to cyber-attackers. The research noted that six platforms, employed

functionalities and features that will enhance trust like developing closed and trusted group access, peer-to-peer connections, anonymisation of shared threat intelligence, and policies for maximum control of privacy and security. However, these platform functionalities remain limited in access control and ranking mechanisms. Researchers adopt privacy and trust models for exchanging cyber-intelligence information, in order to reduce the risk of data exposure and help manage trust (Steinberger et al., 2020).

2.1.7 Research Finding 7: Academic perspectives of cyber sharing

In November 2011, the OpenIOC standard was published and set the foundation for CI sharing and by that time there was insufficient consideration towards CI sharing in academics and practice. Dandurand and Serrano (2013) were the first advocates for a cyber-intelligence management company that had requirements to accomplishing it. By 2014, the comprehensive STIX and TAXII standards were published, and afterwards the number of articles and entrepreneurs contributing to CI sharing programmes had developed fundamentally. For example, the volume of printed papers in 2015 tripled that of 2014, due to the fact that the demand for CI sharing companies was still in nascent stage and growing. According to Poputa-Clean (2015), it can be assumed that the number of programmes and scientific papers will continue to develop dramatically.

2.1.8 Research Finding 8: Lack of automated Cyber sharing system.

CI sharing platforms currently provide less sophisticated automated intelligence-sharing capabilities. Hence, most organisations still use manual means to share and collect valuable intelligence despite the demand for real-time cyber threat detection. Also, the success of CI programs depends on how willing stakeholders are to share information. This is controlled by access to shared resources offered by companies to encourage information security professionals to take part. A lack of CI in businesses means less intelligence gathering and, more importantly, a shortage of automatic sharing of sensitive information. These activities still require manual effort alongside classical file importing functionalities.

Optimal CI sharing systems require real-time threat intelligence across board (Hamza, 2012).

The user interfaces of CI platforms play a huge role in the quick addition of new intelligence data and need robust collaboration to reach the desired defence goal. The overall exploratory research presented an explanation of the increase in demand of CI sharing in study and practice, i.e., the number of publications that refer to this subject and the quantity of organisations that multiplied over the last few years. Such scholarly contributions include Johnson et al., (2016) on cyber threat information sharing, Brown et al., (2015) on information sharing to threat management CS, Nolan (2015) on the legal dimensions of CI sharing and Abu et al., (2018) on CTI issues and challenges, amongst others. Furthermore, there is increased focus in knowledge and practice since various scholars review the principles of CI sharing, although there is already a hybrid of answers in the market. This traces back to a lack of a systematic understanding of what CI sharing is about, due to the diversity of CI sharing principles. Barnum (2014) argues that one of the most critical gaps is the lack of description and characterisation of CI sharing principles, especially towards automation support etc.

Efforts beyond technology should include a process for how a prospective CI sharing policy that presents "real" intelligence, should be conceptualised, shared and operated, so as not to remain somewhat of data warehousing, with insufficient cyber data analysis abilities (Barnum, 2014).

Existing Models for Cyber Intelligence Exchange

The long-lasting quote by Sun Tzu – “Know the enemy and know yourself” – has consistently influenced various aspects of life, including CTI. To recognise how a cybercriminal is likely to conduct an attack, we must acquire knowledge of their motives and what they are aiming to achieve. Once we know their aim and which resources they are likely to attack, we can better concentrate on effectively protecting assets (Darling, 1996). If you do not know yourself and the enemy, you will either die or surrender in every battle (Luzwick, 2000).

Hence, CI is knowledge about dangerous motives, objectives and strategies of cybercriminals that helps organisations safeguard their assets from cyber-crimes. Notably, there is no extant research that has addressed the implementation of a Proactive Self-Defence Cyber Intelligence Sharing (PSDCIS) model in real-time at an organisational level. This informs the gap between CTI models in theory and in practice, which will be covered in more detail within the methodology framework in Chapter Four. In essence, intelligence is the tacit understanding necessary to prevail during a conflict.

2.1.9 Classification of Cyber Intelligence (CI) sharing tools

CI sharing is contingent upon its model or environment. It is important to understand the different technical details of the models of CI sharing tools and the patterns used to understand several issues in CI, such as privacy, trust, and practical problems. Mohaisen et al., (2017) identifies the classification of CI sharing tools based on various classification principles that includes:

1. Structure: In accordance with the format and structure of the intelligence supplied, the CI sharing tools can be divided into structured (standard) and unstructured models.
2. Centralisation: Depending on whether a centralised intelligence-sharing unit (source) is present or not.
3. Function: CI sharing tools are also classified based on their technical functionality. Structured intelligence systems that use standards can be differentiated into enumeration, scores, and application of language and transport mechanism for data encoding and sharing.

2.1.10 Unstructured Intelligence Sharing

Sharing operational security information about malicious assaults among an information-sharing community is the core goal of CI sharing, in a bid to understand and secure their cyberspace. The core goal of intelligence sharing does not depend on whether the intelligence shared is unstructured or structured. Custom acquisition, sharing, and exchange of unstructured data about cyber threats and attacks occurs

through regular communication channels like email and file-sharing websites (Mohaisen, et al., 2017). Even though the difficulties of privacy, trust, and the law are not considered when sharing information, the information shared may nevertheless include information about the data originator. Despite the usage of proprietary formats by suppliers of information security institutions, unstructured intelligence exchange nevertheless takes place (Mohaisen, et al., 2017).

2.1.11 Structured Intelligence Sharing

In contemporary information systems that utilize frameworks and standardized formats, automatic sharing of CI is essential. Structured intelligence sharing refers to the systematic and standardized exchange of information, facilitating straightforward processing by various systems and tools. A substantial amount of work has been undertaken to comprehend the application of scenarios. Numerous platforms employ regulated and defined frameworks for the transfer of CI, including National Institute of Standards and Technology Cybersecurity Framework (NIST CS), the Intrusion Kill Chain model, and STIX CTI (Arenas, 2017).

This section analyses a case study involving the model for sharing CI as well as other models and systems utilized by one of the top threat intelligence platforms, STIX as developed by the (Barnum, 2014).

STIX is a protocol that is being developed in collaboration with all Cybersecurity stakeholders for the standard capture, classification, and transfer of standardized CI information.

The STIX approach for managing and sharing CI is illustrated in Figure 1. The model consists primarily of four parts: UC1 evaluating cyber danger, UC2 defining indications patterns for cyber risk, UC3 organizing cyber threat response operations, and UC4 sharing cyber threat information.

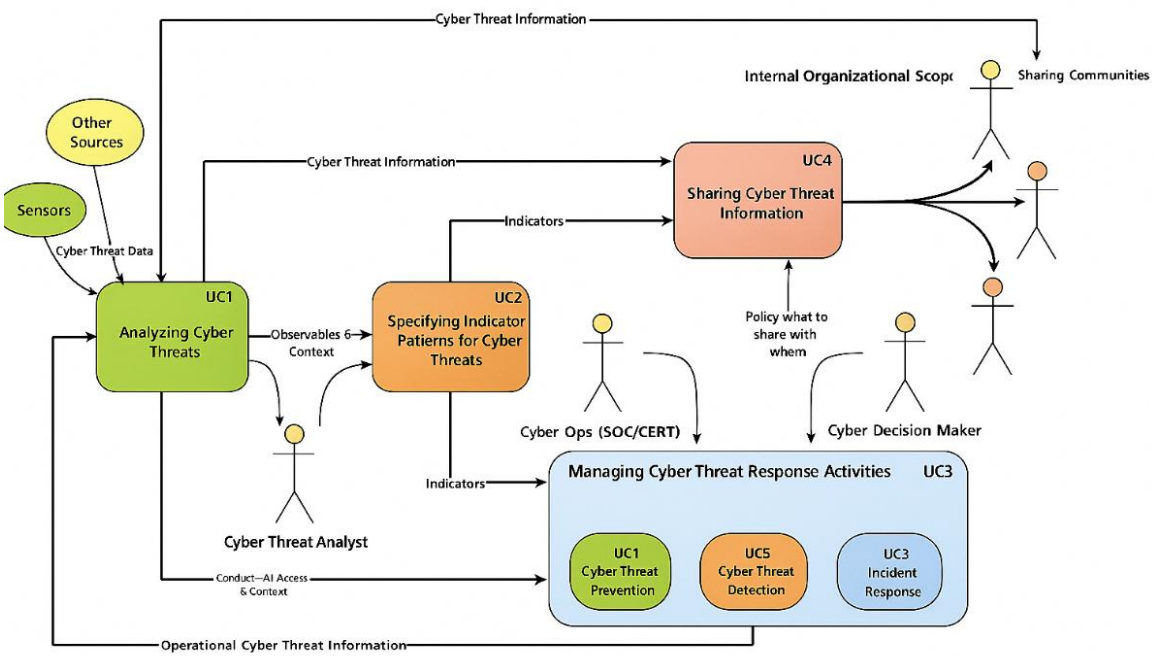


Figure 1: U*se case model targeted STIX*

Source*:* Adapted from Barnum, (2014)

1. UC1 Analysing Cyber Threat: Analysing data on cyber threat incidents that has been structured or unstructured from various automated or manual feed sources is the first line of action in CTI. In this stage, the model aims at understanding the relevant cyber threats, identifying them, and completely characterising them so that the attack's intricate details can be fully articulated and developed. The knowledge gained from understanding threat behaviours includes information about threat actors, cyber-related acts, reactions, and capacities. The purpose of UC1 is to convert threat information into useful intelligence indicators.
2. UC2 Specifying Indicators Cyber-Threats: The second use case in CTI as a process of analysing a cyber-threat, identifies patterns that represent observable characteristics of a particular cyber-attack, along with their cyber context and associated information for managing, interpreting, and applying the patterns and resulting outcomes. This procedure may be carried out manually or with the aid of

automated tools and organised data on CS threats. For instance, the actual location of the threat, patterns of known phishing attacks, and the development of a strategy to counteract detected acts.

3. UC3 Managing Cyber Threat Response: The third use case within cyber decision-makers and CTI operations involves experienced Cybersecurity administrators and decision-makers who together strive to decipher, react to and mitigate cyber-attack activities as well as investigate any suspected attack. The goal is to install precautionary measures in mitigating vulnerabilities, flaws, or human errors that could be targeted by cyber-attackers to exploit the systems. Upon the detection of malware or a phishing attempt, appropriate courses of action may be taken, such as blocking suspect traffic or implementing blocking rules at email connectors. The three sub-modules that make up this section of the module are incident response, cyber threat detection, and cyber threat prevention.

   a) Cyber threat prevention: Preventive measures are implemented by decision-makers who assess the vulnerabilities that cyber-attackers may exploit, and in response to the identified cyber threats, select the most appropriate activity to be carried out.

   b) Cyber threat detection: Detection entails the relevant courses of action carried out either manually or automatically by cyber operation personnel to monitor for signs of any infiltration. Cyber-attacks are also tracked to find out how often they occur.

   c) Incident response: When cyber threats are noticed, cyber operation personnel will start the incident response and investigative process. They will try to pinpoint the attack's primary cause and implement a specific mitigation strategy. Examples include if malware was installed on affected computers, which systems were affected, and what damage there is in terms of data loss (Barnum, 2014).

4. UC4 Sharing Cyber Threat Information: Based on the established framework of trust, the fourth use case in CTI operations has to do with the degree of consistency, context, and control, that the decision-makers develop as a policy for transmission of threat information to other sharing communities and how it should be processed. According to the rules established by the threat decision-makers, the policy

constrains associated shared community to uphold confidentiality, integrity and usability of the data which might be manually or automatically exchanged with communities or trusted associates, so that they would learn from the information amassed by the sharing group and a more resilient cyberspace will emerge (Barnum, 2014).

The purpose of creating the earlier CI-sharing model was to contribute to and address the present issues facing CI, but even after this model was made public, problems with trust, privacy, and legal issues have not been resolved. The report that emanated from the CSIS Strategic Technology Program stated that sharing CI is a crucial first step in creating a formidable cyber-defence (Ertan et al., 2020, pp.1–267). The same report also demonstrates how information-sharing societies have developed to establish the best models for collective defence mechanisms. The most viable of those societies are the Financial Services Information Sharing and Analysis Centre (FS-ISAC), and the Electricity Sector Information Sharing and Analysis Centre (ES-ISAC) as they operate under the Information Sharing and Analysis Centre (ISAC) providing an important vector for sharing CI between the private-sector communities (Zheng and Lewis, 2015). Despite this, the effectiveness of programs for exchanging CI is given credibility due to existing trust maintained inside shared networks and built strategically and collaboratively for common interest and objectives. To make CI sharing stronger, the US congress has granted responsibility protection and proposed access to government-provided CI and recently introduced laws intended to improve the sharing of CI. Due to concerns over privacy, trust, and legal issues, none of the incentives has progressed (Zheng and Lewis, 2015). Even though the model employs innovative detection, prevention, and response techniques, it nevertheless provides CI via community share points by utilizing passive intelligence sharing.

The disadvantage is that cyber attacker can join the sharing community and pose as an authorized group member with the intention of collecting information from the community members. Another problem with this model is that CI may not actually exist because it gathers information from both internal and external

community members, analyses it, and then stores it in a database. The community then extracts intelligence from the database, which means the intelligence process does not provide real-time security. Additionally, when raw intelligence is gathered from community members' internal systems, the raw data often includes username and password in addition to information on the data source's internal IP address and security settings. The company's fiscal interest may be harmed by disclosing that information to community members. While the present CS prevention strategy relies heavily on sharing CI, a new method of sharing CI is required that does not jeopardize privacy, trust, or legal considerations.

In response to this need, this thesis proposes a new PSDCIS model that can gather and analyse data from multiple intelligence sources, categorize them using a highly supervised ML classification algorithm, and incorporate new insights into the company's information security system in real time.

2.1.12 The Detection Maturity Level model (DML)

The Detection Maturity Level (DM) model was developed by Ryan Stillions and launched through a blog post in 2014 (Stillions, 2014). The concept was initially used to categorize corporate maturity in terms of their capacity to detect, understand and respond to specific threat information. Cyber threat information might include indicators of broken systems, tactics, techniques, and procedures (TTPs) of the attacker initiator details, cyber threat intelligence, and other pertinent information. In 2016 the model was expanded to include a ninth level called "attacker identity," and it was made available as seen in Figure 2 for use in the semantic representation of cyber threats.

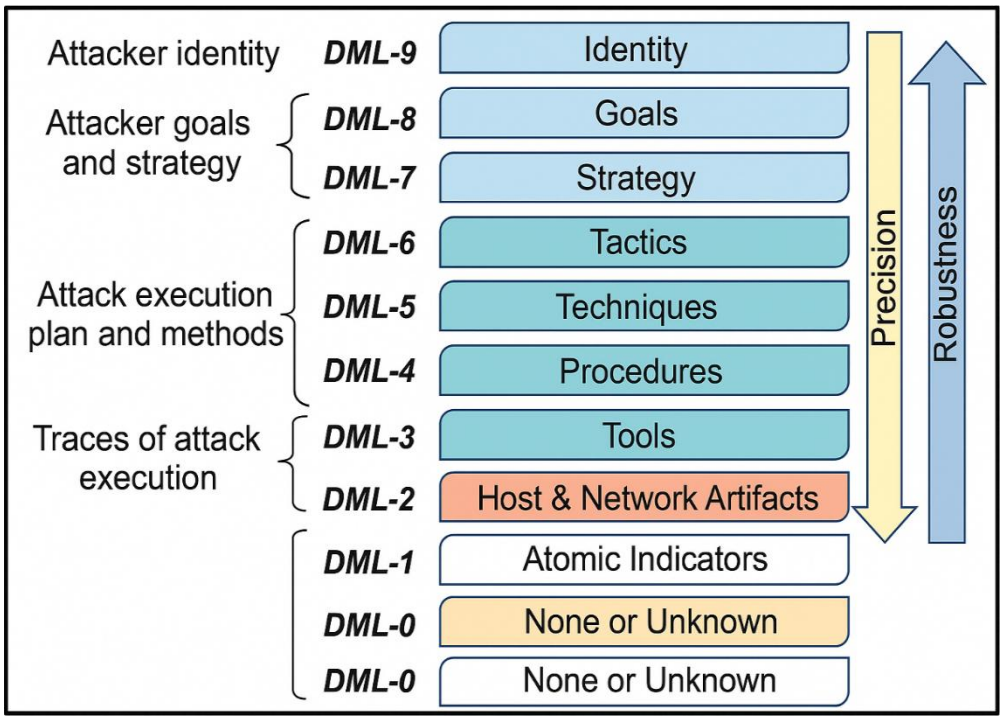


Figure 2: *Detection Maturity Level model DM*

Source: Adapted from Stillions (2014)

The above model demonstrates the growing importance given to the detection of cybercriminal attacks, where it is anticipated that a threat actor will employ many techniques and tricks, and therefore mechanisms must also be capable of detecting threats in a sophisticated way with precision and strength. An idea of what inspires cyber-attackers is important knowledge to track if the aim of cyber threat detection and prevention is to be achieved. Inspiration could be defined as the force behind the attitude that motivates actions toward the accomplishment of the stated objective. The benefit of achieving a goal may act as an incentive. The motivation behind cyber-attackers often remains the same, despite their changing intentions. Understanding a cyber-attacker's motivation helps CS professionals limit attacks and anticipate scenarios that attackers can exploit to compromise systems by narrowing the attack path that attackers may focus on.

Even though a new module such as "attacker identity" was added to the DML model, the direction of sharing CI for prevention, detection, and reaction is still far behind when compared to the increasingly sophistication of most recent cyber-

attacks. The DML model uses what Stillions called the "precision and robustness tactics" to track attack execution while identifying attackers' objectives, strategies, and tools employed during the attack process. A major weakness that the DML model has is the lack of facilitation of intelligence sharing among community members or organisations.

CI collection, processing, and analysis without sharing is equivalent to a person with a loaded gun who is unable to fire it when the enemy approaches. Sharing CI has the advantage of preventing the loss of information assets. In light of this, an actionable intelligence must always be the end goal in threat intelligence lifecycle to improve CS (Abu, et al., 2018). Time is vital when it concerns response in CS, as battling cyber-attack proactively and not just reactively is so crucial (BlueFort Security, 2023). A study by BlueFort Security (2023) concluded that proactive mechanisms under the General Data Protection Regulation (GDPR) helps with cyber threat protection and to detect in real-time, external cyber malicious attacks.

2.1.13 The Cyber Threat Intelligence (CTI) Model

The CTI Model illustrated in Figure 3 was developed to assess and classify taxonomies, information sharing standards and ontological frameworks related to CI (Mavroeidis and Bromander, 2017).

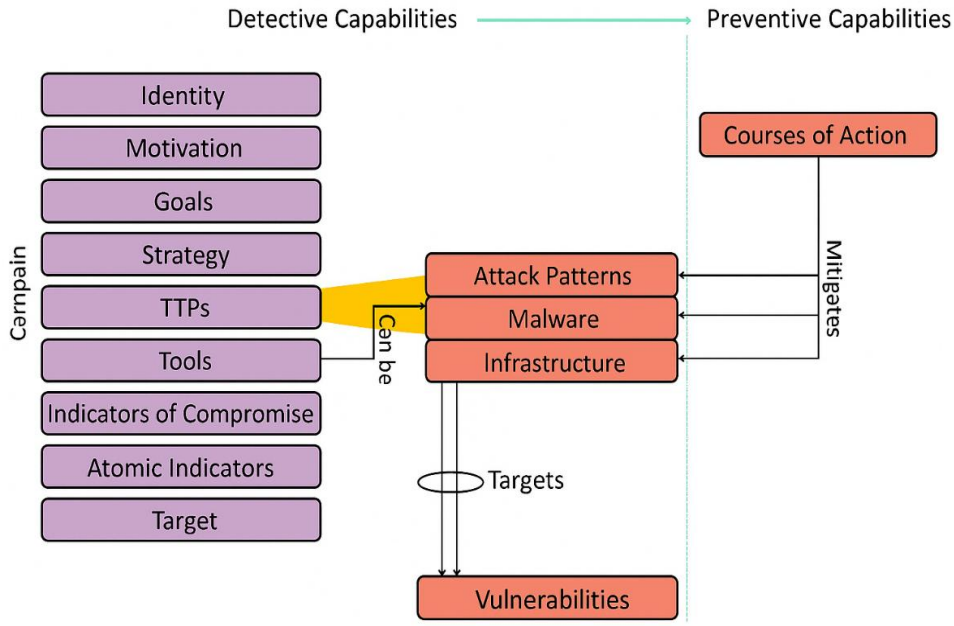


Figure 3: *CTI model*

Source: Adapted from Mavroeidis and Bromander (2017)

According to Mavroeidis and Bromander (2017) there remains a need for a new model to be developed and it should embody threat intelligence comprehensively. Additionally, they assert that the CTI model is not organised and tiered in the same way as the DML model, even though the model can be used to illustrate the type of data needed for advanced CI and potential assault attribution Mavroeidis and Bromander (2017). Additionally, the CTI model is made to facilitate sophisticated forensic processes and defensive capabilities, strengthening the security management of the cyber intelligence organisations themselves. The strategic management is hinged on three distinct strengths: detection, prevention and course-of-action execution capabilities.

1. Detective skills include the ability to identify and define harmful or suspicious behaviours, including tactics, techniques and procedures (TTPs), attack patterns, malware types, and the equipment that cyber attackers employ to target actors. Atomic indicator is a sub-module under detection whose purpose is to identify information with a brief life span, such as IP addresses, domain names, and file

hashes. Most of the time, CI would be useless without this asset, making it crucial for the exchange of CI.

2. Preventative measures lower the possibility that a company may be disrupted by cyber-attacks. According to a model, this module works very closely with the detective module and intercepts cyber threat before they happen.
3. The course-of-action aspect describes operational counter measurements that can be implemented to prevent cyber-attacks.

Despite the CTI model's attempts to supply the two crucially important capabilities—detective and preventive—it does not specify or explain how to act or share intelligence within communities. To effectively share CI, malicious entities and malware codes must be mitigated. Aside from that, automatically exchanging intelligence is a step in the right direction to stop today's sophisticated cyber-attacks. According to Darley and Schreck (2018), to achieve this, technologies and solutions capable of starting automated defensive actions by defining common interfaces must be created to respond to incoming CTI in real time. Info-security magazine notes that the existing best practices of CI sharing contains the transmission of indicators of compromise (IOCs) (Darley and Schreck, 2018). As the threat-sharing communities gets more mature, the next steps are to share more intelligence to inform clearer decision-making along with advice on detective or preventive directions of action (Darley and Schreck, 2018).

A study published by the Echo Network of the European Union identifies four different models for exchanging CI: Hub-and-Spoke, Peer-to-Peer, Source-Subscribe and Hybrid models (Gomez, Vanfretti and Olsen, 2018). However, each of these mentioned models has the same set of challenges such as:

i. Hub-and-Spoke: In this hub-and-spoke design, there features a central hub that receives data from the participating members (the spokes). This model is sustained by continuous operation of the hub as any off time or delay can be detrimental to the entire sharing process in (Gomez, Vanfretti and Olsen, 2018). Peer-to-peer as a model is often described by the freedom it provides from being in a central hub, as it allows any member of a community to communicate and

share with any other member directly which presents a security risk. Due to the absence of a central authority, managing numerous trust relationships becomes harder as community participation increases. See Figure 4 for different models of CI sharing.

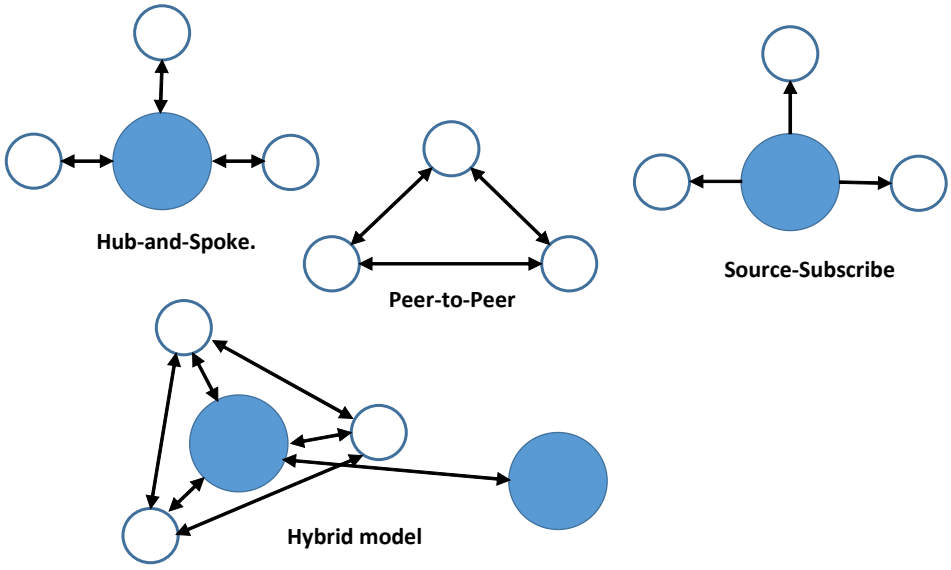


Figure 4: *Different models of CI sharing*

Source: Adapted from Gomez, Vanfretti and Olsen (2018)

ii. Source-Subscribe: In this model, cyber threat information is distributed to a group of subscribers by a single entity. This is a typical commercial model that most private intelligence service providers use. This model ensures that only the commercial service providers alert the users.

iii. Hybrid Models: In this model, elements of the above models are found, as for instance, for the exchange of threat indications, a peer-to-peer paradigm might be used, while an incident-response data would be sent to a central hub. This hub would analyse the data arriving from many organisations and provide analytical reports that could be used by everyone. This hybrid model is challenging to set up and maintain. The governance of this model is difficult due to the mechanics of sharing information between two separate architectures.

Figure 4 outlines these models and their distinct features of operation.

Looking at the analysed models, reveals that current CI-sharing models are struggling to counteract recent advanced attacks since they are not sharing information in real-time. Several variables account for this: a) challenges from legal, privacy and trust; b) the structural inability of models to automate information exchange; c) financial issues faced by customers who must subscribe to the intelligence from commercial providers.

Broggi (2013), made a strong case for the real-time CTI-sharing which includes three critical categories of intelligence that when exchanged will enhance cyber-defence. They are; sharing of threat data (the kinds of malware circulating the Internet and the nature of the threats a given entity has faced), vulnerability data (the flaws an intruder might exploit to access a computer system), and countermeasure data (the actions an entity has taken to prevent or mitigate the effects of a cyber-attack. Nonetheless, as stated by Nolan (2015), there is broad consensus regarding the need for improved cyber-information exchange. Cyber experts also agree that present governmental and private sector information sharing efforts are completely inadequate**.** This study proposes a new PSDCIS model that automatically acquires, processes, analyses, and distributes intelligence to organisational information security devices in real-time.

Existing Cyber Intelligence (CI) systems

Currently, there are diverse platforms for CI sharing, available for free or commercial based. They support the Cybersecurity ecosystem with information exchange. Sauerwein et al. (2017) noted that literature has identified challenges and requirements for CI sharing platforms (Sauerwein et al., 2017). Dandurand and Serrano (2013), launched a theory for threat intelligence sharing platforms, highlighting that such systems must have these three major requirements: (a) facilitate information sharing in a structured way (b) enable automation, and (c) facilitate the generation, refinement and authentication of data. Current CI sharing solutions are biased, and different software developers are presenting products (Brown, Gommers and Serrano, 2015).

The main concern in the minds of CS professionals is, to what extent these CI tools contribute to the required CI sharing delivery benchmark. This remains unclear. Only a limited number of studies have tackled how existing CI platforms can be categorised. Different standardisation attempts have been conducted to facilitate CI data sharing in a standardised approach. A good instance will include formats and protocols that will improve operations, and they include; i) Open Incident of Compromise (OpenIOC), ii) Cyber Observable eXpression (CybOX), iii) Structural Threat Information eXpression (STIX), and iv) Incident Object Description Exchange Format (IODEF). They ensure how cyber threat information will be exchanged in a structured way (Kampanakis, 2014; Steinberger et al., 2020; Martin, 2008).

The SANS Institute provides a summary of a small collection of cyber threat intelligence platforms, and it includes; i) Collective Intelligence Framework (CIF), ii) Collaborative Research into Threat (CRITs), iii) MANTIS Cyber Intelligence Framework, iv) Malware Information Sharing Platform (MISP), and v) Soltra edge. Irrespective of the varieties in the operations of these platforms, Sauerwein, et al., (2017) conclude that the market of CI sharing is still developing. With the different implementation strategies, all the above-mentioned CI platforms or threat intelligence have the same limitations in their ways of exchanging threat information. They include implementation strategies contained in community memberships, commercial subscription services and open-to-source platforms which still provides limited resources to contribute proactively to current emerging cyber-attacks.

2.1.14 Collective Intelligence Framework (CIF)

The CIF is a client-server-based system for cyber intelligence management designed to process, and deliver information on cyber vulnerabilities. Developed by the Research and Education Networking Information Sharing and Analysis Centre, popularly known as REN-ISAC, CIF is responsible for the creation of collective threat intelligence. This system has a server component that gathers and stores data

on CTI. The compiled threat intelligence may include the attackers' IP addresses, Autonomous System Numbers (ASNs), email addresses, domain names, and particular internet URLs. Multiple client programmes have the capability to access the stored threat intelligence (Korte, 2021). Figure 5 shows collective intelligence sharing frameworks to aid understanding.

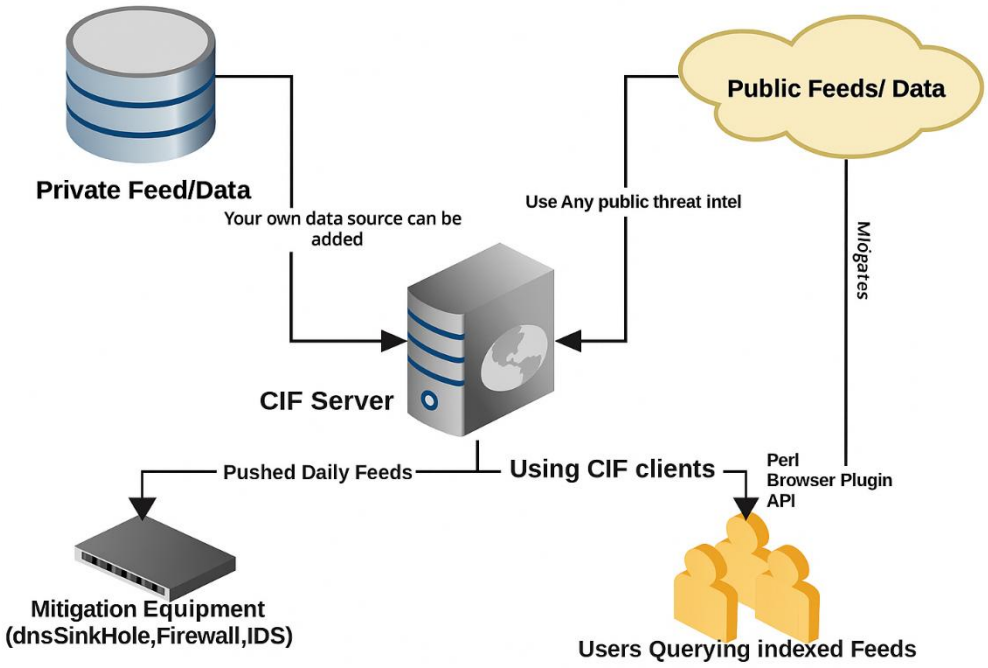


Figure 5: *Collective Intelligence sharing*

Source: Adapted from Korte (2021)

CIF is a cyber-intelligence management system that receives and stores CTI from internal and external sources, and can decipher malicious activities. The stored information could be used for actions like threat identification, incident response, detection, and mitigation.

According to (Reneke et al., 2015), CIF is an example of CI warehousing. It collects threat information under scrutiny from the diverse sources and stores that intelligence in a storage, supporting CS operations. The data is used to choose incidence response, detection and mitigation. The model pulls in several data-observations from many fronts, and initiates a sequence of messages and over time will spell out a reputation score that will aid decision making. The problem with CIF

is in its utilisation of data warehousing, which may result in the intelligence not truly being transformed into information. In isolation, information possesses limited intrinsic value. The ability to convert raw data into actionable intelligence is critical and such must possess the qualities of being timely, accurate, and relevant (Friedman and Bouchard, 2015).

2.1.15 Malware Information Sharing Platform (MISP)

An alliance of software engineers from Computer Incident Response Centre Luxembourg (CIRCL) with many other sponsors established the open-source Malware Information Sharing Platform (MISP) also known as the Cyber Intelligence Sharing Platform as illustrated in Figure 6 (CIRCL, 2012).

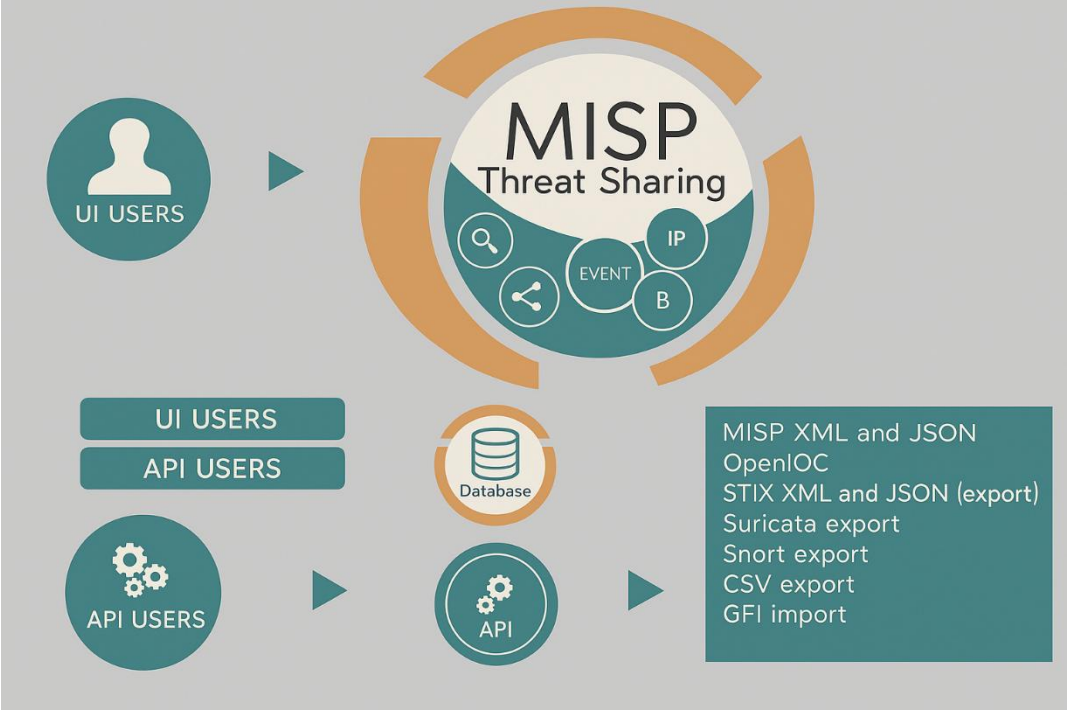


Figure 6: *Malware Information Sharing Platform*

Source: Adapted from CIRCL (2012).

The development of this methodology was intended to improve threat detection and responsiveness to cyber-attacks in Luxembourg and abroad (CIRCL, 2012). The MISP-open-source CI sharing platform enables the exchange of information between businesses, including indicators of compromise (IOCs), threat

intelligence data, adversary TTPs, or any other information regarding threat. The creators also contend that consumers benefit from community-based intelligence exchange in a collaborative manner regarding current malware or threat (CIRCL, 2012). Although the module is useful for giving organisations threat intelligence, its mode of exchange operations cannot proactively defend against today's sophisticated cyber threats. Fully automated real-time intelligence sharing is necessary to match the onslaught of ongoing cyber-attacks, which begins with gathering intelligence, processing, analysing, and sharing in real-time. Automation of intelligence exchange will be useless without real-time sharing.

2.1.16 Intelligence Sharing in the Malware Information Sharing Platform

The MISP is facilitated using a Traffic Light Protocol (TLP) system as its dissemination mechanism. In this TLP system, three colour codes are used to distinguish between levels of privileges shared, and they are:

a) Grey colour, which represents the sharing community consisting of outgoing North Atlantic Treaty Organisation (NATO)/NCIRC members.
b) Red colour, which represents the sharing communities in the private sector and CERT, and the
c) Yellow colour, which represents other communities, as illustrated in Figure 7. The commercial/private sector and CERT communities establish a connection with each other through an intelligence clearance store. However, NATO/NCIRC adopts a synchronisation mechanism to exchange intelligence (CIRCL, 2012).

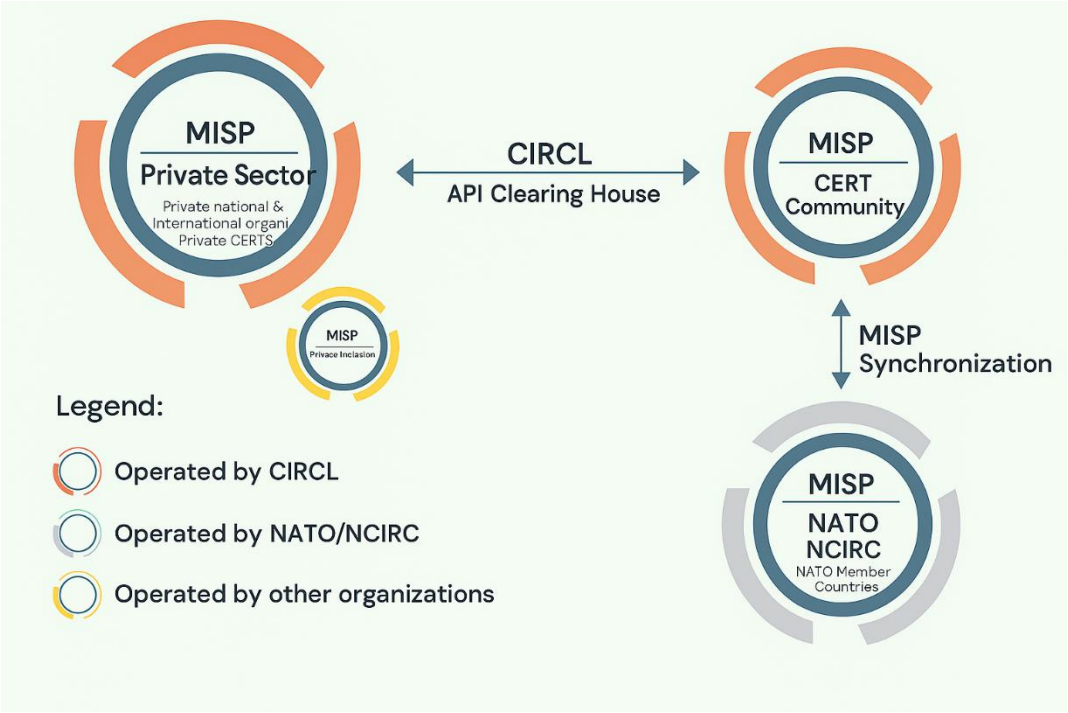


Figure 7: *How MISP shares intelligence in the community*

Source: Adapted from CIRCL (2012).

While knowledge exchange with MISP is facilitated by a community-based membership system, one drawback is that anybody can become a member, posing a potential danger of exposing personally identifiable information (PII). Upon joining this community, malicious cyber actors, including hackers can collect information about its members, which they could subsequently use to launch attacks on them. Although this model receives support from NATO, it still has challenges due to lack of trust from certain community members who are sceptical of government-funded CI platforms.

2.1.17 Webroot Intelligence Network (WIN)

WIN was developed to be an advanced and sophisticated CTI system that uses analytics from big data to address cyber-attacks. WIN as developed by Webroot collaborates with billions of bits of data from millions of detectors to generate a big detection network. Furthermore, it integrates intelligence from sources which include honeypots, client feedback, global sensors, spam traps and other open-

source intelligence. Figure 8 shows how every Webroot act as a detection vector that autonomously feeds data into the Webroot Intelligence Network. Also, over 27 million data sources provide data back into the Webroot system (Möller, 2020).

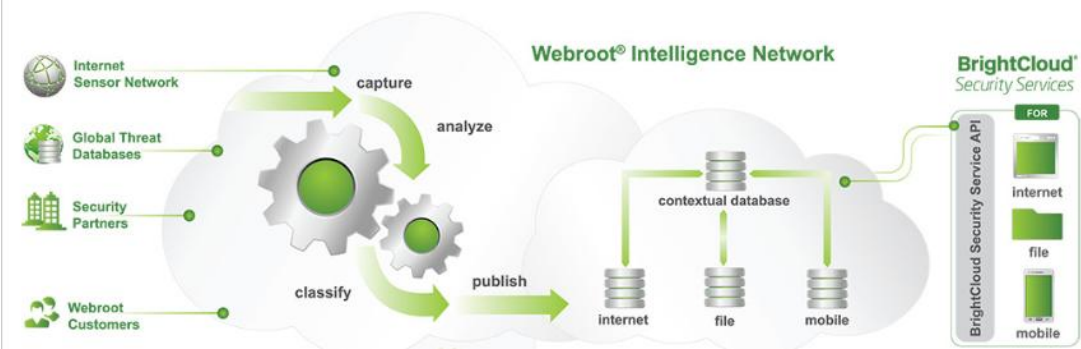


Figure 8: *Webroot Intelligence Network*

Source: Adapted from Williams and Shaw (2016)

Webroot uses data from its clients, external security partners, the global threat database, and internet sensors as a source for threat information, as seen by the WIN. Webroot collects data, analyses it, and stores it in a contextual threat intelligence database.

A progressive ML technology approach of Webroot lies at the heart of the Webroot Intelligence Network. To fill work queues for human analysts, many security firms rely on first-and second-generation ML models like either Bayesian Networks or Support Vector Machine (SVM) models. WIN uses a third-generation ML algorithm called Maximum Entropy Discrimination (MED), to analyse web threats in an incredibly accurate and scalable manner. The WIN can simply categorise 2500 URLS per second at the minimum error rate of about 1%. To deliver reliable intelligence, the WIN has; 13 billion URLs, 4 billion file behaviours, 460 million domains, 900 million IP addresses and 10 million mobile Apps.

This large data infrastructure makes CTI accessible to Webroot's technology associates and clients and supports them to stay ahead of the increasing cyber-attacks (Möller, 2020). Although Webroot gathers, processes, and analyses the intelligence, there is no real-time sharing of CI. More so, according to Barnum

(2014), standardization is necessary, according to the Webroot developers' CI. Consequently, there is a need for a collaboration that will enable intelligence to be shared between security systems and businesses in an organised and real-time manner. Their study suggests a proactive cyber intelligence sharing framework for self-defence that actively operates within the Cyber Integrated Leadership Command (Cyber-ILC) model, while utilizing effective cyber deception-based methods. Once the winning strategy has been put into practice, the outcome will be autonomously shared with network administrators and allied security systems. The trust that will grow between the security systems and real-time information sharing system will be strengthened to help CI platforms respond to threats dynamically.

2.1.18 Darknet and Deepnet Cybersecurity Threat Intelligence

The Darknet and Deepnet Cybersecurity Threat Intelligence as a system is made up of three unique units that were developed independently before being integrated into systems. The system is in operation and actively gathering new CI. The components are:

i. Crawler: The crawler is a software package made to relate with websites and capture HTML documents. It is also made to gather forum discussions from the Darknet due to the operational variations and access restriction measures for each platform (Bergman and Popov, 2023). The researchers additionally created isolated crawlers for various market and forum platforms to make for easy recognition by specialists. The crawler gathers information about the aftermath of discussions from online forums and markets.

ii. Parser: This was developed to gather information about sales of fresh and new malware, exploitative sales, and discussions of attack tools in cybercriminals' forums. This comprehensive database of cyber threat information is maintained. The system is designed to keep two distinct cyber threat databases, one for forums and the other for marketplaces, and keeps them up to date (Bergman and Popov, 2023). Each crawler has its own parser, and they work together to update the system and obtain new CI data, as the parser shares new information with

the crawler. To obtain additional time-varying threat information, the crawler re-crawls a list of suitable links that the parser has provided for it (Bergman and Popov, 2023).

iii. Classifier: To find linked CI information and themes from marketplaces and forums, researchers developed a ML algorithm using a tagged dataset. The classifier filters out information and subjects related to drugs, weapons, etc. Figure 9 explains this process graphically.

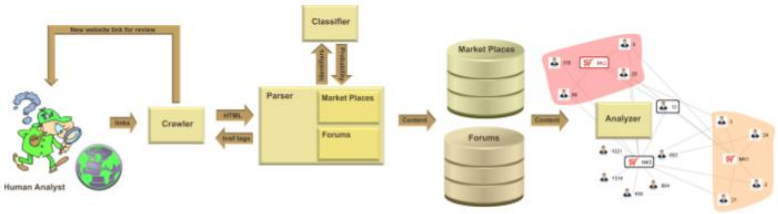


Figure 9: *Darknet and Deepnet for Proactive CS Threat Intelligence*

Source: Adapted from Bergman and Popov (2023)

According to Bergman and Popov (2023), using the Darknet and Deepnet model to improve CTI focuses on acquiring information from malicious website hacking and then analysing the information obtained without actively sharing the results. This passive process allows for a large degree of information collection without real-time sharing of information with security systems. This paradigm relies solely on the signal from source information collection because most of the key intelligence-gathering sources are not actively interacted with. The ability to actively defeat the opponent will expand because of gathering CI from many sources.

2.1.19 Proactive Protection and Cyber Threat Analysis of CTI

In order to tackle Advanced Persistent Threat (APT) assaults, Fujitsu developed a proactive detection and defence model from the CTI framework. The automatic saving of attack artefacts discovered by diverse types of sensors and cataloguing systems is the initial stage in this model. The following steps entail conducting a cyber-threat analysis against the archived artefacts, extracting CTI and

sharing that information to various sensors and Security Information and Event Management (SIEM) systems, and then taking appropriate action, such as cutting off the infiltrated links that connect to the terminal where the incident occurred (Osako et al., 2016). Figure 10 identifies components in this model.

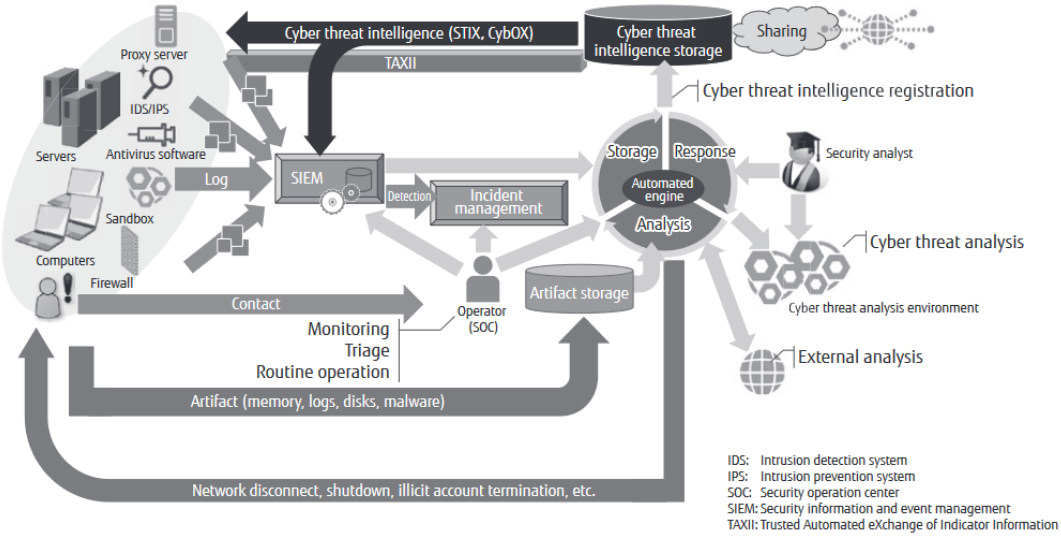


Figure 10: *Proactive defence model based on cyber threat analysis of CTI. Source: Adapted from Kabay (2005)*

i. Security Information and Event Management (SIEM): This is the part of the model that collects all the logs from internal systems and stores the data in a central database.
ii. Artefacts Storage: This contains memory dumps, disk errors, system logs, and malware samples.
iii. Automated Intelligence Engine: This ensures a structured assessment of cyber threats by analysing all gathered threat intelligence from internal network devices and external sources.
iv. Threat Intelligence Storage: This stores all the intelligence produced and keeps them for future proactive use.

According to Kabay (2005), CI is enormously valued as valuable information for discovering and protecting against cyber-attacks. Howbeit, there is necessity for

common grounds of operation to enable intelligence to be effectively shared between proactive defence systems and organisational security systems in a structured and real-time manner. The Proactive defence model that is based on cyber threat analysis of CTI, combines events from a diverse range of security sensors and logs from various information systems and evaluates them to decipher whether an incident has occurred based on precise rules and CTI-derived evidence. The model used passive cyber-intelligence defence practises, which proves the need for real-time cyber-intelligence defence systems. The CrowdStrike report of 2019 highlighted why this is necessary by publishing that there is need for real-time CI, by stating that interactions are always ongoing in the world of Cybersecurity which involves both adversaries and defenders. Therefore, businesses must know the cyber-attackers' next moves, so as to proactively modify their defence systems in anticipation of upcoming attacks (Kurtz, 2019).

More so, Geers (2011) emphasizes the need for social, economic, and political variables to be involved in cyber-attacks, even in cases where intelligence is not shared. The severity of a cyber-attack might arise from an undetermined equation and for an undisclosed motive, just as strategic issues necessitate well devised solutions. This underscores the importance of integrating strategic discipline into the practice of CI sharing. An in-depth review of existing literature on these concerns reveals the absence of viable solutions or models that can effectively tackle the issues of real-time cyber intelligence sharing and defence.

Challenges in Cyber Intelligence (CI) sharing

While CI sharing is advantageous for protecting against cyber threats, sharing intelligence between corporate and government organisations poses challenges. Sukhabogia and Anushab (2021); Martins and Medeiros (2022) agree that collecting information about potential threats, examining and exchanging it across security teams is very challenging and demanding due to the different factors involved. While there are benefits to sharing intelligence among businesses, there are also

barriers that hinder cooperation between them. Sharing CI information is problematic due to numerous factors such as:

i. The handling of CI knowledge requires utmost care and precision, since any mishandling may result in detrimental harm to an organisation's reputation. Leaked CI information can potentially disclose the weaknesses within the organisation to both organisations and hackers.
ii. Trust is a crucial concern when it comes to exchanging CI between organisations and the Threat Intelligence Sharing Platforms (TISPs).
iii. The shared CTI is typically substantial in terms of volume. The abundance of information gathered from diverse sources, including open source and commercial channels, poses a challenge in identifying the specific intelligence that is truly valuable for an organisation.
iv. The costs associated with implementing CI is substantial, which poses a challenge for small and medium-scale enterprises (SMEs) with limited budgets, to consider it as an option.
v. A lack of standardization between different CTI providers also causes difficulties for consumers who use the information.
vi. Data warehousing or community sharing delays the usability of intelligence.
vii. Lack of automated systems for CTI sharing and integration of the information system.

Several vendors and systems in the market offer threat intelligence, however most of them concentrate on data collection rather than data analysis (Sukhabogia and Anushab, 2021). Additionally, Abu et al., (2018) identified four distinct challenges that are similar to the ones mentioned above. When examining the sharing of CTI and considering the challenges and barriers faced by consumers and producers of TISPs, some difficulties and hurdles include:

i. Threat Data Overload: A significant number of companies continue to have challenges in dealing with a substantial volume of threat data and a deficiency in staff proficiency to effectively utilise their threat intelligence programmes (Abu, et al., 2018).

ii. Threat Data Quality: This deals with how accurate, timely and relevant, CTI information collected is.
iii. Privacy and Legal Issue: Data protection regulatory infringement or even confidential agreements may be breached while sharing intelligence, which is risky.
iv. Interoperability Issue in TISPs: Diverse data formats of the intelligence across TISPs can hinder effective integrated communication. This theory is supported by Martins and Medeiros (2022) who identified same challenges.

Researchers like Wagner et al., (2019) argued that while it is acknowledged that CI sharing has emerged as a great strategy for cyber defenders to actively reduce the impact of rising cyber-attacks, the automation of CTI sharing, as well as its implementation has presented new practical difficulties for researchers and practitioners, and they must be addressed cautiously.

2.1.20 Risks of sharing

Sharing CI to unauthorised parties, including within an organisational structure, can provide a risk that may discourage stakeholders from using automation (Vázquez et al., 2012). Tosh et al., (2015) identified and described three potential consequences that stakeholders may encounter while exchanging CI. The first is that, sharing CI with competitors may lead to free riding behaviour, whereby some organisations could benefit from collective intelligence shared without contributing by sharing their own data in return. Secondly, this one-sided phenomenon would result in a violation of trust among stakeholders. Thirdly, sharing sensitive cyber information especially unauthorised can bring about detrimental publicity, which may consequently, have a negative corporate impact on market value and stock prices.

In addition, Haustein et al., (2013) have identified the risks related to sharing information inside community forums. They feared that internal facts and credentials belonging to organisations such as email addresses, personal names, and other identifiers could be exploited by cyber attackers, leading to reputational harm.

Further to this (Al-Ibrahim, et al., 2017) stated that cyber criminals could use the intercepted CI to launch attacks against community members who have shared intelligence. This calls for stringent measures to facilitate CI sharing.

2.1.21 Timeliness and Trust Issues in Cyber Intelligence Sharing

Globally, cyber-attacks happen every second and in strategic ways. Due to this dynamism, CI sharing must be swift and effective in its operations (Pawliński, 2014). Timeliness makes the value of any intelligence robust, and delays diminish this value fast, sometimes in just an hour and damages would have occurred (Farnham and Leune, 2013). CI sharing that is real-time, automated, and reliable is necessary for proactive defence against cyber threats of any kind (Al-Shamisi, 2014; Aidman et al., 2015).

Timeliness is also a factor that can increase trust among CI-sharing partners, because it is currently a challenge for partners to trust the process of intelligence exchange, until after several engagements and assessment of reputation of the vendors. (Kokkonen et al., 2016; Meng et al., 2015). Since threat intelligence is very sensitive and can contain people's private information, it is expected that only trusted community members should access it, so as to safeguard against it leaking to cyber attackers who will use it to cause mayhem. Even though third-party interventions have been available to create trust, the fact that government is involved in sharing frameworks like STIX and TAXII, makes certain partners concerned that government may influence the output (Burger et al., 2014). Therefore, timeliness and trust have been foundational challenges of CI sharing.

2.1.22 Cyber Intelligence Sharing Privacy

Even though advanced technologies have been implemented to mitigate privacy concerns associated with the sharing of CI, members of the CI sharing community continue to be hesitant to reveal information due to persistent risk of violating privacy laws. Several solutions have been created to boost anonymity of

the content of CI sharing such as k -Anonymity which was introduced to remove the risk of de-anonymisation (Sweeney, 2002). This however is not a guaranteed proof against attackers with prior knowledge. Although, Papaspiliopoulos and Roberts (2007) developed the Stability-Based Hashed Gibbs Sampler (SBHG) model to address the issue at hand, it is unfortunate that their solution did not completely eradicate the privacy concerns associated with information sharing.

In cross-border settings, CI sharing has to comply with legal requirements such as the European Union's GDPR which requires that personal data processing is lawful, follows strict rules (eg minimisation of data, and pseudonymisation) and is proportionate, for the safety of consumers (Voigt and Von dem Bussche, 2017). In addition, the Directive on Security of Network and Information Systems (NIS Directive) which helps providers of essential services and digital services to share relevant intelligence is needed to ensure data protection compliance (European Parliament and Council, 2016).

These various frameworks are lacking thus neglecting human rights protection. Their lack of adaptability, compliance that is enforced automatically and in real-time as well as inability to scale makes the mechanisms of the PSDCIS model a legal, technical and sustainability breakthrough in the Cybersecurity sphere, as it can operate within the various international legal frameworks. The PSDCIS model trustworthy frameworks, including features for sharing costs between members, and continuously updates the system to ensure scalability and sustainability.

2.1.23 Cooperation Issue in Threat Intelligence Sharing Platforms

Vázquez et al., (2012) critically examined the current issue of the lack of cooperation faced by CI sharing platforms. Primarily, lack of standardisation in CI sharing platforms was identified as an impeding factor between the intelligence producers and customers. Numerous standards and structures on the CI sharing platforms create exchange issues that causes delays and even total communication breach between stakeholders. Furthermore, Barnum (2014) posits that MITRE group tried as an initiative to produce three different standards specifically designed

to promote a better information exchange, and they are; Cyber Observable Expression (CybOX), Structured Threat Information Expression (STIX) and Trusted Automated Exchange of Indicator Information (TAXII). In addition, Abu et al., (2018) noted that, even though MITRE model provides a forward response, issues of cooperation could linger if the data is not standardised among platforms based on interoperability, legal and or technical restrictions.

2.1.24 Data Accuracy

Accuracy of CI data begins with the cyber defence mechanism. According to EY (2014), CI effectiveness happens when time factor is given due consideration and assessment. The report presented that the accuracy of CI should either give a close representation or deliver actual activities that occurred in the cyberspace. The main concern posed, is the presence of several incomplete data within the cyberspace which could give rise to threats and uncertainties. In other words, such inadequacies would render CS professionals inefficient in countering cyber-attacks because of dearth of intelligence regarding the threat initiator. SailPoint (2023) and Al-Shamisi, (2014) indicated the need for the accuracy of the data which is crucial for all affected parties. Data accuracy is evident in its crucial role of securing firm assets, identities, and other critical information. CS systems rely on available information as a fundamental source. Therefore, without proper specification of CI, organisations may end up investing significant amounts of money in the wrong direction and still be vulnerable to cyber-attacks. It is undeniable that the correctness of CI data is crucial for firms utilising CI systems such as PSDCIS, as it requires highly reliable intelligence data.

Moreover, knowledge should also be used accurately and in real-time when it is newly produced. Sanka et al. (2024) stated that CI must be reliable and accurate to defend against looming cyber-attacks. The significance of timeliness in PSDCIS is fundamental in delivering the kind of information that can be used in real-time self-defence.

Researchers have underscored the difficulties that are inherent in the exchange of CI. The absence of a consensus among current CI providers regarding the definition of CI is glaring. While certain individuals possess knowledge regarding the collection and processing of cyber information within the internal networks of an organisation, others are knowledgeable about the entirety of the procedure, including the analysis, but refrain from disclosing the outcome to other enterprises. According to a study, most CI technologies are solely concerned with information gathering, ignoring the other critical phases of the process. Consequently, many publicly accessible CI systems are more closely associated with data storage than authentic CI exchange services. Furthermore, it is evident that most of the enterprises under investigation do not have adequate confidence in one another to facilitate the exchange of information.

A proactive approach to self-defence CI sharing has been developed in response to the escalating frequency of active cyber-attacks in recent times and the inadequate real-time cyber defences that can effectively repel them. Businesses desire an imminent CI system that will offer real-time protection against the growth of cyber threats.

2.1.25 Community Sharing Approach Models and Systems

The practice of sharing information within a community focused on CI, as a response to the increasing complexity of cyber-attacks, can be considered an ineffective defence mechanism in a contemporary setting. The problems of community sharing include:

i. Free-riders: These are members of a community that do not contribute to intelligence creation that they enjoy.
ii. Restricted access: Some businesses may not get access to the intelligence shared within a community.
iii. Competitors: Other companies that join the community might compete with other members who are in the community, and they might collect information

from competitors. Additionally, cyber attackers may join the community and use the information they gather to target businesses within it (Koepke, 2017).

The growing advancement of cyber-attacks in the cyberspace, makes implementing a community intelligence sharing approach challenging, as it involves storing sensitive intelligence that can be mishandled. Everyone who participates in recent developments in CI sharing can comprehend the ineffectiveness of such community sharing approach.

In January 2000, the White House released its *National Plan for Systems Protection* as a foundational and complete effort towards defending the country's information systems and essential resources against cyber-attacks (GOA, 2000). The plan recommends achieving it as twin goals of: (a) making the U.S. government a standard for Cybersecurity and (b) developing a partnership between government and private sector to protect critical national infrastructure. The report mentions many essential factors to be implemented such as:

i. Detecting attacks and unauthorised intrusions
ii. Developing intelligence and law enforcement capabilities to protect information systems essentially.
iii. Sharing attack warning and information on time.
iv. Creating capabilities for rapid response.
v. Protecting privacy, civil liberties, and proprietary interests.

While cyber-attacks are advancing with new tactics to evade security systems, the U.S. government programme included strengthening cyber-attack information sharing efforts with businesses. In specific strides, the programme created a joint force between the Critical Infrastructure Security and National Infrastructure Assurance Council to expand commercial and government communications (GOA, 2000). It also supports the establishment of Information Sharing and Analysis Centres (ISACs) to enable private and public sectors to share actual data of cyber-attacks and vulnerability. The report supports the importance of intelligence sharing, in relation to protecting privacy, civil liberties, and proprietary interests.

The GOA report suggests two cardinal challenges: the psychological mind-power needed to challenge the will and ruthless aggression of cyber attackers, belligerent groups, and other enemies. The second element of the GOA report implies developing strategic intelligence actions to protect critical systems such as electricity grids, banking systems and government computer networks by using the winning attitude of intelligence. It also highlighted that sharing cyber information must be done timely, indicating the importance of sharing knowledge in real-time as a crucial defence against cyber-attacks. On the same front, the GOA report stressed on why exchanging intelligence is vital, upholding privacy, trust, and legal compliance as drivers of intelligence sharing (Kim and Kim 2018).

Even though there are different implementations of CTI sharing throughout diverse sectors, the co-operation established among CS vendors came up with a new phenomenon called "coopetition" which is when collaboration exists with competition (Zrahia, 2018). Since the intelligence sharing is directly related to the core interest of the businesses, community challenges are more significant when the businesses are direct rivals or have other disputes of importance. By sharing intelligence, businesses demand to preserve their competitive advantage, in order to safeguard their commercial interest, intellectual property, and compliance to legal and regulatory systems (Zrahia, 2018).

When considering models like STIX, DML, CTI, as well as proactive detection systems like Darknet, CIF and Webroot, it is important to recognise that the existing CI models/systems in the market, function as data warehouses. They do not offer the essential variety of information that would stop cybercriminals from achieving their objectives. This limitation arises from the fact that all the mentioned systems/models operate using a data warehousing approach. Iovino (2015), featured the CIF as a CI warehousing system instead of a progressive intelligence exchange service. Additionally, the GOA (2000) report emphasises the need for the protection of privacy, civil rights, and business activities/efforts alongside information sharing.

2.1.26 North Atlantic Treaty Organisation MISP Community sharing

In 2017, NATO proposed a MISP – an open-source intelligence sharing platform where community members and partners exchange and contribute threat intelligence. The ideology behind this was “share to win." NATO members and partners sent the threat information they encountered to the central location, where other bonafide members of the community collected the information. Although this method of information sharing is more advanced, it did not resolve the existing challenges in intelligence sharing, such as legal issues, automation integration, and standardisation protocols. Private sector entities still view data protection and privacy rules as their main concern, which discourages their participation in CTI sharing (Nweke and Wolthusen, 2020). The illustration in Figure 11 explains several facts such as:

i. How vital CI sharing is for national, international, financial organisations and the military.
ii. That current intelligence is shared through a community share point.
iii. Community members contribute to the information feed.
iv. The prevalent method of intelligence sharing is data warehousing, and there is no real-time knowledge sharing in place.

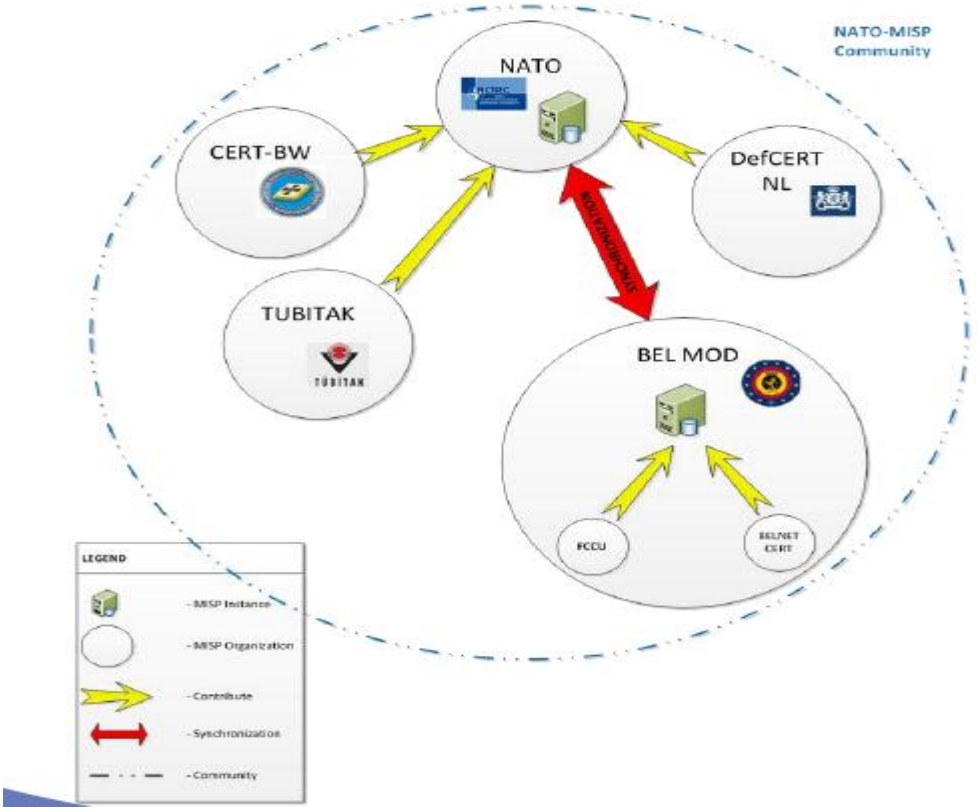


Figure 11: *NATO MISP Community intelligence sharing*

Source: (Vandure, 2017, p.7)

The diagram in Figure 12 demonstrates the transfer of DHL data across members of the NATO community. Furthermore, it indicates the dissemination of this data and the corresponding occurrences.

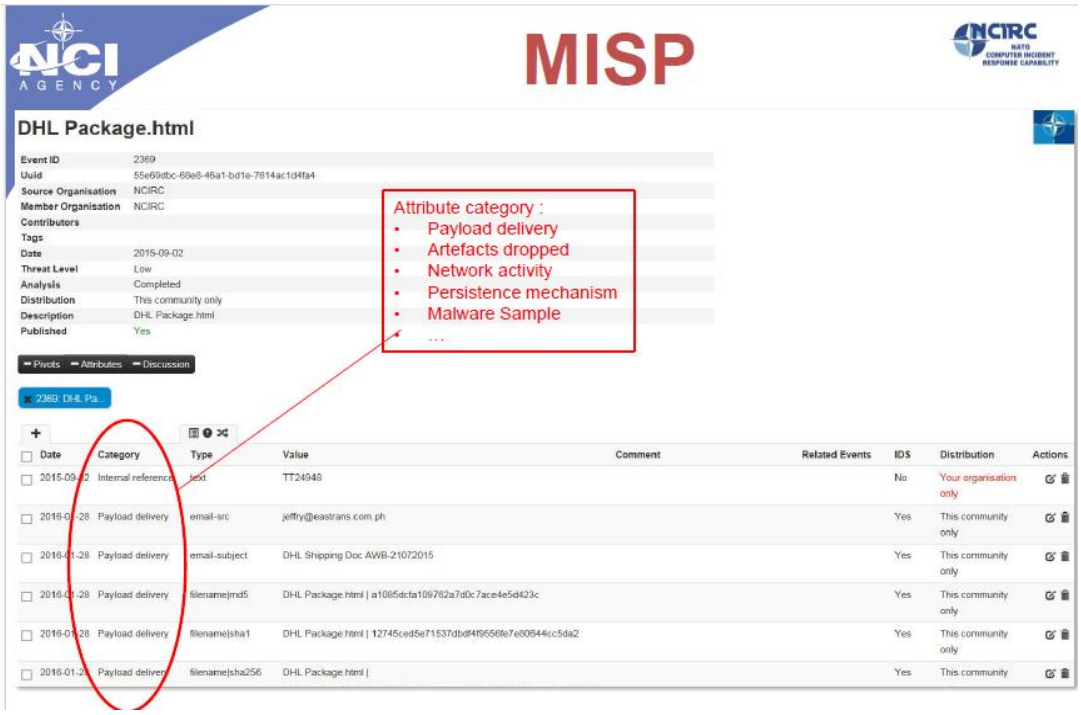


Figure 12: *How information is shared in MISP*

Source: Adapted from Vandure (2017)

Our objective is to illustrate the importance of sharing knowledge in real-time and the figure shows how information is disseminated within communities. Community members exchange information they have discovered after a system breach, and when they collaborate in sharing information, it might be either timely or untimely for implementation. Gaining intelligence before cyber-attackers achieve their objectives is the most efficient approach.

Considering, this fact, it is evident that the current models and platforms for exchanging CI, which are motivated by the business objective of preventing intrusions, have certain limitations. In the future, small and medium-sized organisations need to adopt a real-time CI sharing system. At this juncture, it is imperative to have immediate and ongoing exchange of CI, especially for medium-sized and smaller enterprises who lack the financial means to purchase information from conventional intelligence sources. Limitation in current models and systems

This research has argued that the practice of intelligence warehousing with deficiency of not sharing information in real time would certainly not be sufficient

to compete against advanced Cyber-attacks. Instead, the cybercriminals can easily manipulate the weakness of CI systems, and get inspired and enabled to initiate attacks before intelligence information is shared. Identifying the stage at which current models begin to concentrate on data collection, processing, and storage is very viable. Comparing the concept of developing new models with current models and systems, as well as the direction of the proactive defence strategy implemented in the PSDCIS clearly demonstrates that current models are not actively responding to emerging threats. Figure 13 illustrates how current reviewed models/systems share intelligence.

Figure 13: *How the current CI sharing functions*

Source: Researcher 2023

The critical question that has emerged is - what are the specific limitations of the existing CI models and technologies in relation to their main objective of preventing, detecting, and responding to Cyber-attacks. Figure 13 depicts the process by which current intelligence systems and models collect, process, analyse,

and store new knowledge in a centralized repository. Community members then access this repository through an authentication system. Centralizing intelligence in a repository makes it intangible; information needs to be pertinent, useful, and valuable (Dalziel, 2015). The basis of the heatmap of CI models, as shown in Figure 14, highlights the limitations of current models.

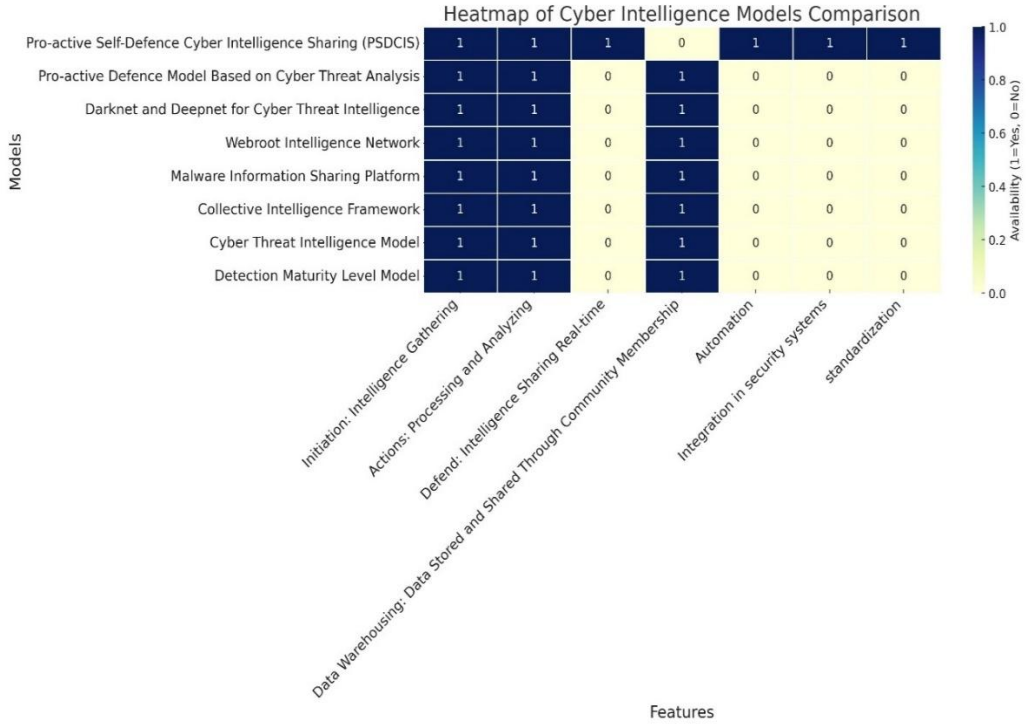


Figure 14: Difference between current models/system and proposed model
Source: Researcher (2024)

Based on the common method of the ILC, the best criteria to compare models on intelligence gathering, has to be concerned with processing, analysing, and sharing intelligence.

Comparing the above-named models is based on binary plotting, where (0 = No), meaning that the model does not have a particular feature, and (1 = Yes) means it has the features. Examining each model's actions uses the following points:

a) Gathering intelligence;
b) Processing and analysing it;

c) Sharing it in real time;
d) Storing it in data warehouses and community memberships;
e) Automating intelligence;
f) Integration of intelligence into information security systems; and
g) Standardisation.

All the models gather, process, and analyse data, but the methods of data exchange, intelligence warehousing, standardisation, and automation in the ILC are the main differences. The current review models utilize the data warehousing method for disseminating intelligence, whereas the proposed model employs real-time intelligence sharing and integrates all business security systems.

2.1.27 Summary of Intelligence sharing challenges

Previously, organisations shared Cybersecurity information using informal means such as email, discussions, phone calls, voucher systems, or face-to-face meetings. Lately, there has been a growing inclination towards establishing sharing organisations that utilise networked technology to generate semi-automated exchange of critical cyber threat information (Brown, Gommers, and Serrano, 2015). Sharing CI among specialists across businesses offers a potential strategy to counter today's sophisticated cyber-attacks. Researchers determined that not all organisations possess the financial resources to independently set up viable CS systems. Consequently, companies will perpetually rely on the expertise and knowledge of other institutions (Fenz, et al., 2014).

According to Dandurand and Serrano (2013), more than a few companies identified difficulties and obligations as regards principles for CTI exchange. Besides, quite a few uniformity efforts have been made to help standardise CS intelligence distribution such as the creation of OpenIOC, CyBOX, STIX, IODEF or TAXII (Kampanakis, 2014). These standards form the basis for today’s CI sharing protocols. Meanwhile, only partial information and reports of how current threat intelligence sharing platforms function can be accessed (Poputa-Clean, 2015). The SANS Institute brought forward a summarised list of open-source threat

intelligence platforms, and it includes CIF, CRITs, MANTIS Cyber-Intelligence Management Framework, MISP, and Soltra Edge, and concludes that the market for threat intelligence sharing is still emerging. Brown et al (2015), made a case for the expectations that Cybersecurity communities have concerning an all-encompassing threat intelligence authority based on a few CI sharing opportunities. The supplementary platforms available, only focus on a subdivision of available CI sharing companies without offering an in-depth analysis of the state-of-the-art. Various tools and vendor studies roughly connected with CI sharing have been noted especially in governance, risk and compliance (GRC) management software (Racz, et al., 2011), cloud computing vendors (Repschlaeger, 2013) and the Cybersecurity software market (Dey, et al., n.d.).

Despite the obvious advantages of information sharing, there are still huge challenges. For example, official intelligence sharing is a challenging topic due to legal, reputation and operational challenges. Lawyers are cautious, and very disturbed when organisations and the general public openly disclose that they have experienced cyber-attacks or breaches (Schreck, 2018). The primary debates revolve around the questions; what is being shared, how it is being shared, and with whom it is being shared with. The core of the problem lies in the concerns regarding the integrity of defence mechanisms in place and the potential violation of intelligence privacy and confidentiality. The "how" question refers to the level of technical expertise required for effective CI sharing.

Among stakeholders in organisations, some of them support the more relaxed but practical approach to defining the de facto information exchange formats based on specialist software tools. Others argue that the validated formats based on open standards will gain broader acceptance due to increased interoperability among various business-related and open-source network defence tools (Schreck, 2018). At the end, the “with whom” problem addresses the issue of trust because as the intelligence community grows, the old, reliable, interpersonal model of trust is no longer able to sustain direct relationships with every member of the CI sharing community. The above fact highlights the need to create the cyber-intelligence

community that has an efficient workable CI exchange system. To achieve intelligence sharing, there is need to develop tools that adapt standardized interfaces for automated defensive actions based on new knowledge.

Community versus Real-time Sharing

The essence of CI sharing is to proactively detect malicious actions and stop them before they strike and disrupt business activities. For that reason, after detecting them, sharing the intelligence is next in importance. However, how this intelligence is shared makes the whole difference between community sharing and real-time sharing. Also using ML algorithms to identify such malicious activities during the processing and analyses of data adds to the difference between these two methods of intelligence sharing or exchange.

Community sharing: The NATO Community is a good example of a community-based intelligence exchange made possible by placing trust in one another. A clearing house receives the fresh incident data and acts as a data repository for all community members connected through an Application Programming Interface (API). Members use APIs to transmit and retrieve data from the CIRCL community (NATO, 2013). Although those kinds of community facilitate the exchange of information, they have negatives, such as lack of trust, sensitivity of data, concerns with legal compliance, misuse of data, and competitive disadvantages from organisational perceptions (Wickr, 2021).

Real-time sharing: The real-time intelligence sharing process distinguishes itself from the community-based one by its ability to effectively handle the collection of intelligence from external sources, subsequent processing, analysis, and ultimately sharing intelligence with both human handlers and automated systems. The entire procedure is fully automated, and integrates intelligence seamlessly into business operations before attackers can capitalise on vulnerabilities to gain access. In addition, artificial intelligence algorithms are used to categorise behaviours as either malicious or non-malicious. This idea of automating CTI and using ML, is supported by lots of researchers in literature (Czosseck et al., 2011;

Poputa-Clean, 2015). Figure 15 illustrates the process of sharing intelligence among NATO member countries, the private sector, and the CERT community.

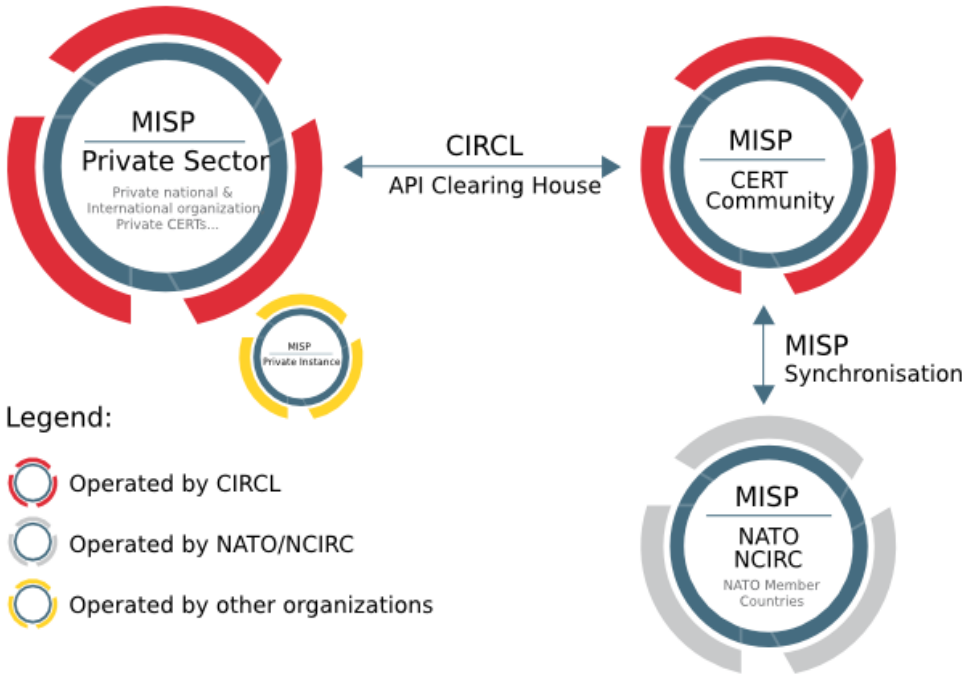


Figure 15: *How Intelligence is distributed between NATO Members*

Source: Adapted from NCIAgency (2017)

The private sector and the CERT community exchange intelligence through an API clearing house, while NATO member countries and CERT, harmonise their processes. Despite MISP being more advanced than traditional sharing methods, the model still lacks privacy safeguards, legal compliance, automation, and ability to integrate into business defence systems.

We have observed the distinction between community-based intelligence sharing and real-time intelligence sharing, and it has become evident that real-time information offers numerous advantages in guarding against cyber-attacks. Furthermore, a review and analysis of literature reveal that modern cyber threats and how they manifest makes automation of CI sharing of paramount importance. Researchers like Motlhabi et al (2022) also recognised the importance of addressing

current intelligence sharing challenges, which encompass automation, legal factors, policies, cultural differences, and the human role in threat intelligence sharing.

The United States Department of Homeland Security acknowledged in its National Cyber Security Programme report that information sharing is nowhere sufficient or comprehensive (Administration of Barack Obama, 2015). In the same vein, the Bureau of National Affairs (BNA), released a static report that identified how intelligence is shared. In it, 58 percent of businesses rely on peer-to-peer systems for information exchange and done through informal platforms (Sullivan, 2016). The organisations also used "post to all" method to share the information (Peretti, 2014).

The reason the PSDCIS model is proposed becomes obvious when analysing traditional process of community information sharing. These points confirm the fact that the current challenges of CI sharing are real and demands urgency. One must therefore acknowledge that the concept of intelligence warehousing and creating intelligence to fight cyber attackers, is not enough to counter threats when examined through the lens of CI sharing guideline, and Cyber Security concepts. Hence this strongly supports the reasoning for implementing Direction 3, which means the nurture of real-time intelligence sharing attitude.

The Correlation between Intelligence Sharing and Defence in Legal Contexts

The goal of self-defence in law is to protect businesses and individuals from harm. However, the evolving nature of information technology development and especially cyber threats has created fresh challenges in the applicability of these laws. Frequently, they have been forced to adapt to traditional criminal combat, and some countries are still grappling with how to incorporate cyber laws into their constitutions without compromising other legal provisions. Although cyber-attacks may have the same attributes as weapons of attack – intent to harm, damage severity and premeditated action – it is hard for businesses to implement self-defence laws. Internationally, self-defence as a legal instrument is documented in Article 51 of the

Charter of the United Nations. The implementation of self-defence laws in CI presents several challenges, including the complex task of identifying the cyber attacker and the global nature of the attacks. Regardless of the international and national legal issues, these challenges persist. To prevent upcoming cyber-attacks, PSDCIS requires legal cognisance, integration and compliance with international law, and much more to with effectively tackle these challenges.

Significance of data in intelligence

According to Conti et al., (2018), gathering threat intelligence, exploiting vulnerabilities, malware, and other potential CS threats as well as understanding the operational methods of threat actors, is a critical function in identifying them. Cyber threat is explained as an instant ability of adversaries to create specific attacks and direct into specific target assets, either systems or information. It is intelligence about the cyber threat that triggers businesses to prepare to protect themselves. When organisations answer to the critical issues that cyber risks face, like "who is then possible to target", "what resources", "where, when, how and why", then they stand a greater chance of protecting their resources (CREST, 2019)

Generally, the words; data, information and the phrase Cyber Intelligence are frequently used wrongly and interchangeably (CREST, 2019). Figure 16 illustrates this process clearly.

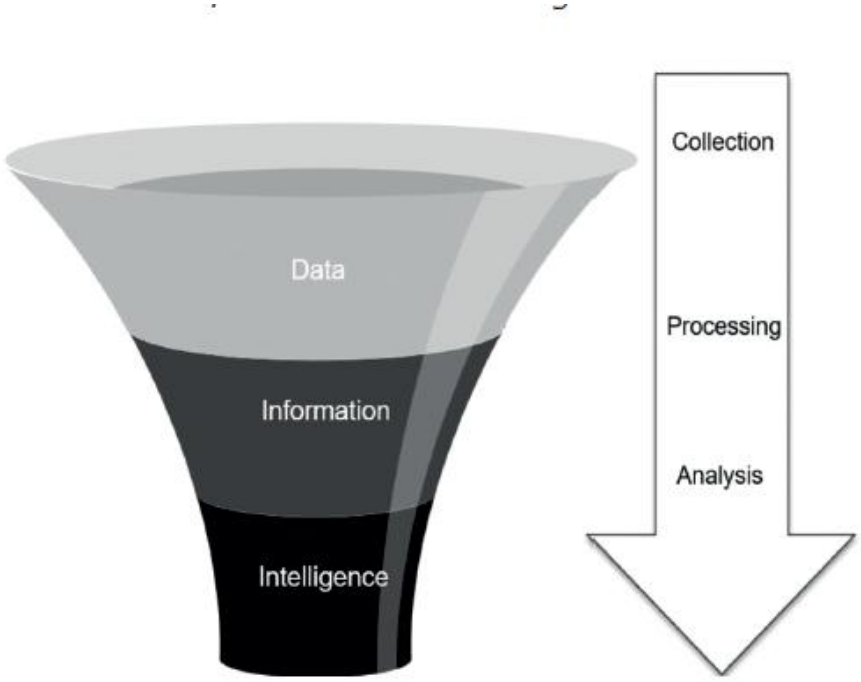


Figure 16: *Producing CI from raw data*

Source: Adapted from CREST (2019)

i. Data: This is the simple reality that tends to be offered in massive amounts in relation to data logs, dark web data, social media data, honeypot retrieved data, server logs, IP addresses, as examples. By itself, only raw data is of incomplete value.

ii. Information: This is generated when data is gathered to deliver a valuable output. For example, the collective number of firewall logs which shows suspicious activities of attack.

iii. Intelligence: This is the output of processing and analysing information, and can be used for decision-making on how to protect assets. For instance, the gathered firewall logs data is reviewed with the timeline of the incident report concerning similar events. Intelligence allows CS developers to learn the characteristics of the attack and produce new defence mechanisms to combat threats accurately. Data, therefore plays a fundamental and crucial role in the knowledge-generating process, as intelligence cannot be generated in the absence of data.

Intelligence Life Cycle Framework

To effectively implement any CI model, there must be adherence to a threat intelligence lifecycle structure. This is the Intelligence Life Cycle and has five steps of threat intelligence processing; a) planning b) collection c) processing d) analysis, and E) sharing. This lifecycle is a circular process, making for continuous refinement of CI. Panhalkar (2024) posits that is such frameworks and models, help guide CI teams towards creating TIPs that are effective and relevant. Figure 17 shows the process in the ILC.

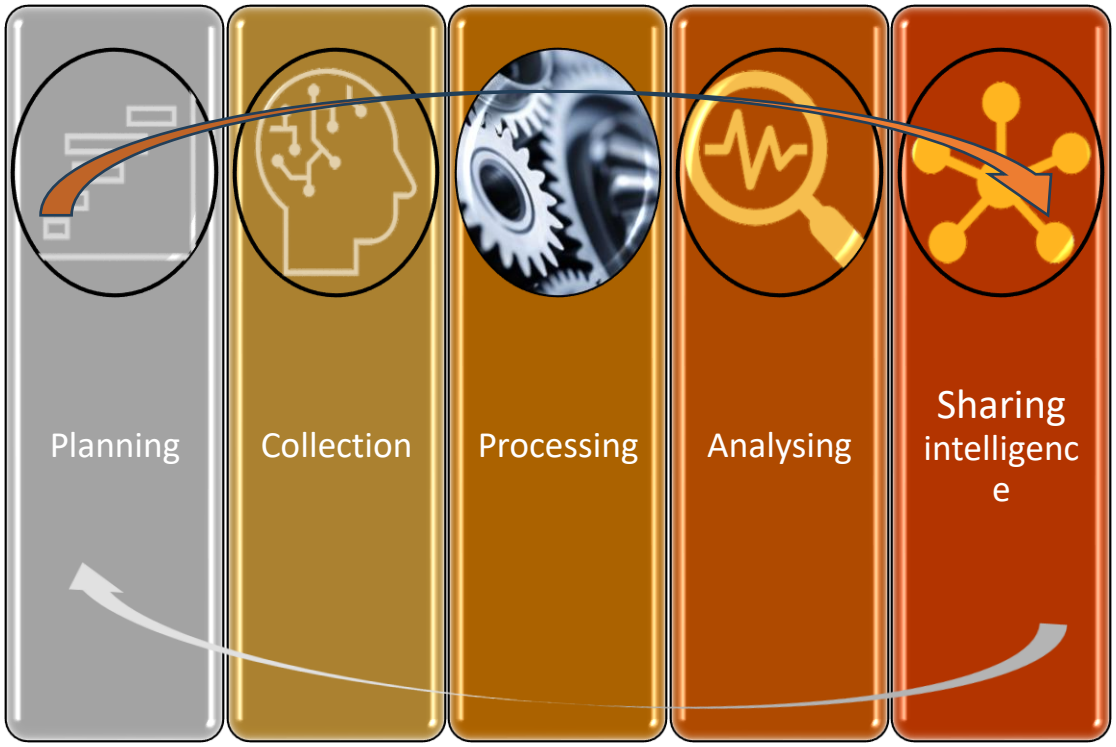


Figure 17: *ILC process*

Source: Researcher's review (2023)

i. Planning: In this step, a proper plan is made based on determined objectives i.e. the reason for developing the CI model, and the information that should be a priority during collection. In addition, identifying the sources of data, the methods of data collection and establishing a collection plan is what gives direction and scope.

ii. Collection: In this step, we need to concentrate on more on collecting desired data that is defined in the first process. Intelligence can be gathered in difference ways such as: Human Intelligence (HUMINT), and Imagery Intelligence (IMINT), OpenSource Intelligence (OSINT) and Measurement and Signature intelligence (MASINT) (Panhalkar, 2024).
iii. Processing: Up to this step, the data in not in proper formats; it is raw data. The raw data will be transformed into meaningful information that is easily understood by humans. In this step, classification algorithms and data transformation machines are employed to transform these raw data into human readable or usable formats.
iv. Analysis: After the processing step, the proper format data will be analysed. The analysis comprises of information, discoveries, and predictions that facilitate the assessment and anticipation of attacks and outcomes. The analysis needs to be objective, timely, accurate, and relevant for use (Panhalkar, 2024).
v. Sharing: After analysing the data, an information product that includes adversary threat indicator profiles, security alerts, and malicious file signatures is produced, which is then distributed to stakeholders and consumers as intelligence used to prevent future attacks.

Benefits of Automated Real-Time Cyber Intelligence Sharing

A key benefit of automated real-time CI sharing within the business, is that intelligence is accessible in real-time and enables threat defence in a timely manner. Based on Sun Tzu's active defence philosophy in The Art of War, which states that: "*We will not attack unless we are attacked, but we will surely counterattack if attacked"* (Project 2049 Institute, 2015). Moreover, live CI integration that allows for collection of new knowledge and alerts, and immediately exchanging it with the business security systems will prevent issues with privacy, and trust.

Methods of Cyber Intelligence Collection/Gathering

CS has benefited from the use of CI for a long time. Despite being intangible, the actionable information gathered is used to enhance knowledge sharing in businesses. Prevention, detection, and timely response to threats are the main goals of CS. Information sharing is used to reduce cyber risks by publicly disclosing vulnerabilities and exploits. CI is seen as both a process and a resource, with forewarnings that prompt proactive actions, resulting in the mitigation of potential dangers (Yeboah-Boateng, 2018).

The most effective forms of intelligence gathering for targeted attacks are OSINT, Cyber Intelligence (CYBINT), and HUMINT and Social Media Intelligence (SOCMINT) (Sood, 2014).

i. Cyber Intelligence: CYBINT offers business and government with a powerful, highly automated platform for the safe collection, processing, analysis, storage, and distribution, of data pertaining to criminality, fraud, and espionage actions to support intelligence-driven investigations (Sood, 2014).
ii. Open-source intelligence: OSINT is a method of using open-source tools to collect information from publicly available sources especially from the internet and then analysing it to decide or take some action (Sood, 2014).
iii. Social Media Intelligence: SOCMINT refers to the techniques and technologies that allow companies or governments to monitor social media networking sites (SNSs), such as Facebook, Instagram, Snapchat, Twitter etc. SOCMINT which is a subcategory of OSINT includes monitoring of content, such as messages or images posted, by social media users on a social media networking site. This information involves person-to-person, person-to-group, group-to-group, and includes interactions that are private and public.
iv. Human Intelligence: HUMINT is intelligence derived overtly or covertly from human sources based on a relationship between an intelligence source and a human handler. HUMINT can provide value understanding that tools may not be able to detect about threat actor behaviours. Before launching a targeted attack, the attacker wants to do reconnaissance using the power of openly

available information collected using various intelligence-gathering techniques. The most popular sources of knowledge are the Internet, traditional mass media like magazines and newspapers, publications like journals and conference proceedings, business documents, and exposed networks. Many providers publish their content online so that search engines like Google and Bing can find the required information (Yeboah-Boateng, 2018)

2.1.28 Intelligence Gathering Process

To maximize the benefits of CS, the stages of the CI lifecycle are a key factor for every CI gathering process. As data is acquired, the intelligence process is used as a guide to make the most of the intelligence (FlashPoint Team, 2024). There are various components to the intelligence collection process, such as: a) selection and discovery of intelligence sources, b) extraction of intelligence from the selected sources c) processing and analysing data d) identifying of attack indicators or threats. Figure 18 illustrates the procedures for obtaining CI, including the various stages that need to be finished before, during, and after a cyber-attack.

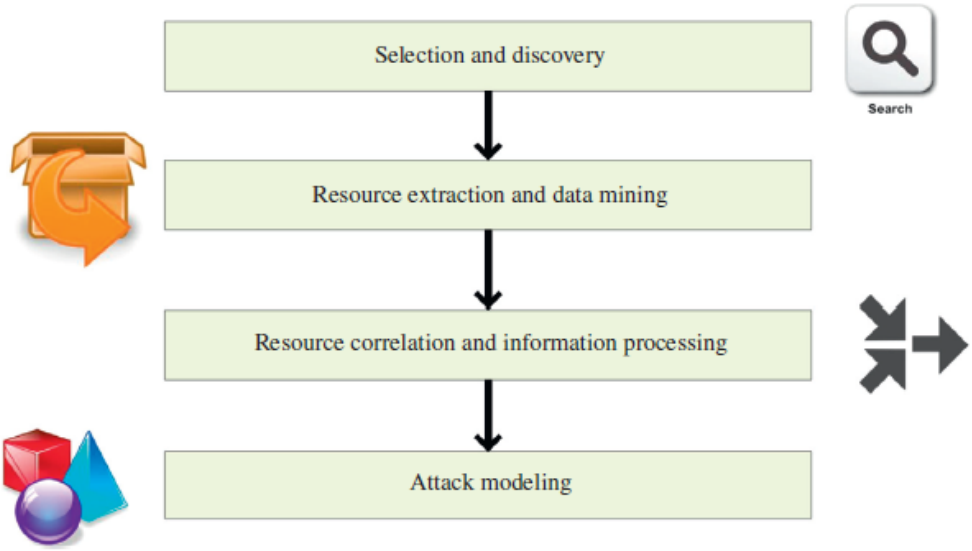


Figure 18: *Generic intelligence gathering model*
Source: Adapted from Sood (2014)

Figure 18 shows strategies used by attackers and it includes:

i. Selecting the source of the intelligence: In this stage both attacker and CI institutions collect public information available via online social networks (OSNs), websites that holds intelligence, reports and white papers from the government, and historical records on businesses and their employees (Sood, 2014).
ii. Resources extraction and Data Mining: The stage of extracting publicly accessible information from both businesses and attackers. This information includes geographic location data, historical employer records, relationship networks, academic contribution, malware signatures, IP addresses, and other personal identifiers (Sood, 2014).
iii. Resource Correlation and Information Processing and Analysing: Is the process of converting raw collected data into useable information. It has to do with filtering data, and analysing it until it is changed into a strategic intelligence product. The analysis methods and clarification depend on the ML analytic algorithms used. ML procedures such as supervised classification, anomaly detection and clustering algorithms have shown to be useful to various CS intelligence processing and analyses.
iv. Attack Modelling: Once the information has been analysed, both the attacker and the intelligence agencies can ascertain its relevance according to their objectives. The military and intelligence services are the agencies that use this approach of obtaining intelligence, therefore it is not exclusive to CS specialists. This is comparable to the military's intelligence gathering without the enemy's knowledge as the commanders only conduct this after being convinced of the enemy's malicious intentions (Geers, 2011). The attack modelling emphasises the significance of the intelligence collection phase throughout the development of any threat intelligence system as the key to providing valuable intelligence.

2.1.29 Sources of Intelligence

CI systems depend on data collected from many sources. This data must first be gathered before it can be processed and analysed for potential information. As a result, gathering intelligence is a vital and crucial step in the intelligence analytical process. Identifying the source of any potential intelligence should be done earlier, because it is a more crucial stage before information gathering. Researchers use the intelligence gathered from sources listed in table 3 to develop and create new cyber defence systems.

Table 3 Intelligence Sources

Intelligence Type	Description of data Source	Intelligence Source
Intelligence gathering database for malware samples	A database-source of malware samples accessible for use by security researchers, incident responders, forensic analysts, and the morbidly inquisitive	https://virusshare.com/
Malware collection for research	The largest indexed online collection of articles, malware samples, and source codes used by researchers.	https://www.vx-underground.org/
Collaborative Malware Analysis Platform	A web-based exchange medium that gives access to collaboration between malware analysts.	https://beta.virusbay.io/
Phishing Kit Tracker	Database for phishing sites source codes and kits used by attackers	https://github.com/marcoramilli/PhishingKitTracker
Machine Learning Malware database	Online community for data scientists with datasets for ML applications etc.	https://www.kaggle.com/datasets

Malware Data Centre	Offers a huge collection of high-quality malware samples for research	http://samples.virussign.com /samples/

Source: Researcher's review (2023)

Another way to acquire intelligence from source, outside databases and online collaborative platforms is through capturing network activities and local logs through security systems and monitoring applications.

As noted in the previous literature reviewed above, gathering intelligence is a critical step in developing any model or system for defending against cyber-attacks. Data can be gathered using a variety of techniques, including CYBINT, OSINT, SOCMINT, and HUMINT (Yeboah-Boateng, 2018).

Malware Detection Techniques

To create an efficient malware detection method, malware analysis is required. Malware analysis is the act of examining a malware's function and purpose, and its objective is to comprehend how a certain malware item operates to ensure that a defence can be developed to safeguard an organisation's network (Jyoti and Wankhade, 2013). Figure 19 illustrates detection methods.

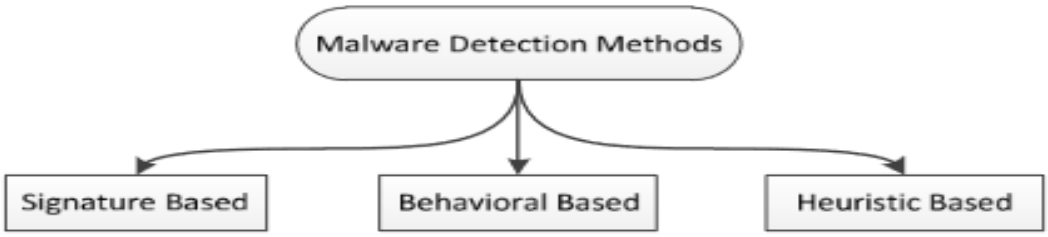


Figure 19: *Malware detection methods*
Source: Adapted from Bazrafshan et al. (2013)

Using malware detection techniques, ensures the identification and categorisations into: signature-based detection, behaviour-based detection and specification-based detection (Bazrafshan et al., 2013).

2.1.30 Signature-based malware detection

When a piece of malware is developed, a sequenced arrangement known as a signature is inserted into its source code. This arrangement is later used to identify the family that the virus belongs to. A considerable number of antivirus products use the signature-based detection method (Sehrawat and Singh, 2022). The antivirus

programme disassembles the programming of the compromised files and searches for known patterns that fit into a category of malware. Malware signatures are updated into an internal malware database, and then used for analysis during the detection stage. This detection method is often called string matching, pattern recognition *or* synchronisation scanning. Signature-based detection functions by static analysis, dynamic analysis or a hybrid method. Moreover, it is possible to categorise malware and non-malware file signatures using machine learning classification methods. ML and Artificial Intelligence (AI/ML) can assist security systems in learning how to distinguish between safe and dangerous files and processes, even if they do not fit any known pattern or signature. They accomplish this by looking at file behaviours, network activities, process frequency, deployment patterns, and other things. These algorithms can learn the characteristics of bad files over time, making it possible to identify new threats. Signature-based malware detection is efficient and accurate in identifying known harmful software. However, some experts oppose this stance, by stating that it is unable to detect undiscovered dangerous software. However, when ML classification methods are applied with adaptive learning models, they can identify unknown dangerous file signatures (Chowdhury, et al., 2018).

2.1.31 Behaviour-based methods

To determine if a software is malicious or not, behaviour-based malware detection algorithms monitor the program's real-time activity (Zahra, Hashem, Fard, and Ali, 2013). Behaviour-based approaches are immune to the drawbacks of signature-based ones because they monitor what an executable file performs. Simply said, so instead of focusing on what a software program says, a behaviour-based detector determines whether it is harmful by examining what it does. While behaviour-based malware detection is an advanced technology for identifying malicious software, it is prone to generating a high number of false positives and can be expensive.

2.1.32 Heuristic-Based (specification-based) malware detection

Heuristic-based detection is a hybrid technique that identifies and sorts out between a framework's normal and abnormal behaviours and identifies both known and unknown malware variants. Heuristic analysis uses either static or dynamic techniques. Static heuristic analysis is a process that involves deconstructing a questionable programme and reviewing its source code while dynamic heuristic analysis is a process of running suspect codes within a sandbox then monitor its behaviour (Kaspersky, 2021).

2.1.33 Malware Signature Detection Using Machine Learning

ML techniques, is powerful in malware signature detection. This method entails finding signatures or behavioural patterns that are characteristic of malware. To increase the accuracy of malware detection, this technique combines the benefits of both signature-based and ML-based approaches (Hamza, 2012).

Due to its ability to identify previously unidentified malware, and adapt to new threats, this technology has proven to be more effective than standard signature-based detection techniques. The general procedure for using machine-learning algorithms for detecting malware signatures are as seen in Figure 20 as follows:

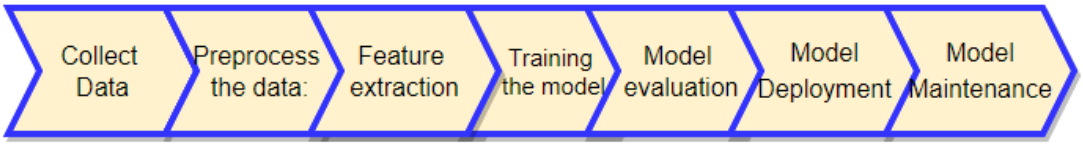


Figure 20: *General steps involved in using machine-learning algorithms for malware*

Source: Researcher, (2022)

i. Collect data: Gather samples of both malicious software and safe software. This dataset must be varied and reflective of the various software platforms that might be found in the target environment**.**

ii. Pre-process the data: The dataset is cleaned up and compiled to make sure it is suitably formatted and empty of any corrupted or unnecessary data.

iii. Extract Features: Extract features with relevant attributes from the software code.
iv. Train the model: Train ML algorithm with the dataset, and extracted features to identify malicious codes and legal software. This entail applying methods like deep learning, unsupervised learning, or supervised learning according depending on what the system specifically requires (Mitchell, 1997.).
v. Model evaluation: Evaluating the model by using a different dataset of malware and legitimate software samples to assess the trained model.
vi. Model deployment: After model is trained, it is deployed into the real world to detect malware real-time.
vii. Model maintenance: The latest stage of the ML process is to monitor and maintain the model's performance in real-world settings. This has to be on a constant basis and updated when necessary to increase accuracy and prepare it for emerging challenges.

Researchers such as Odat, Alazzam and Yaseen (2022) have used decision tree algorithms to classify malware families. Another study by Jureček, (2021) focuses on the benefits of automating malware detection. Bonan et al., (2018) also present how ML techniques can detect malicious behaviours in PDF files, showing how versatile ML is in cyber defence. ML techniques are useful for malware identification and categorisation. The academics cited above, amongst other researchers concur that employing machine algorithms to pre-process data, extract features, train models, and evaluate them, among other tasks, is the best approach to combat evolving cyber-attacks.

Why machine learning is the best technical solution to use

The urgency to have a new reimagined model that is not manual, necessitated the need to implement the new recommended model that employs a completely automated process to gather, sort, and analyse data using ML algorithms. The primary objective of the PSDCIS system is to achieve full automation of the information exchange and collection process. The use of automation and ML for

data cleaning and analysis will result in the creation of threat intelligence. The model and system reviews in CI sharing indicate that there are shortcomings inhibiting real-time intelligence sharing, automation, standardization without compromising privacy, trust and legal compliance. It is clear from these viewpoints that to effectively counter cybercriminal activities, these are requirements:

i. A psychologically dominant attitude to counter cyber-attackers
ii. Creation of intelligence that is high-quality and accurate
iii. Development of a supportive real-time intelligence sharing attitude

Justification why supervised learning is sufficient for the study

Research by Kim and Kim (2018) pinpointed that CI is more interested in gathering new knowledge in order to recognise the objective of cybercriminals and forecast imminent attack. CTI is an evolving protection hypothesis, which was largely specified by Gartner as evidence-based knowledge that involves context, mechanisms, indicators, implications and actionable advice, about an existing or emerging threat or hazard to assets that can be used to inform decisions, regarding the response to that threat or hazard (Krishnan, Sandhu and Ranganathan, 2007).

Threat intelligence businesses collect, analyse, and curate cyber threat information, but organisations need to buy the expensive packages first in order to access it. If intelligence successfully reaches an organisation via this method of information exchange after a cyber-attack have happened, such intelligence came already too late.

Role of Machine Intelligence in CI Sharing

Machine intelligence denotes systems that apply machine learning algorithms to dynamically process, evaluate, and extract knowledge from extensive databases. Within the realm of CI exchange, machine intelligence provides the following benefits:

2.1.34 Automated Data Processing

ML algorithms can analyse extensive volumes of raw security-related data, eliminate noise, remove redundancies, and regulate formats. This process of automation speeds up the conversion of raw data into usable intelligence using algorithms. As Giorgi and Maksim, (2023) posits, automated ML analysis speeds up the threat detection process as well as decision-making particularly for real-world operational systems.

2.1.35 Real-Time Analysis

Modern algorithms enable machine intelligence systems to discover trends and abnormalities in real-time, allowing organisations to spot new risks and disseminate this intelligence promptly throughout their security infrastructure (Ronald and Lokaiah, 2023).

2.1.36 Machine Learning in Cybersecurity

ML algorithms have changed CS by enabling automation in detection, response, and prevention systems. ML algorithms assess very large datasets to discern trends and threats, and improve security decision-making. As Shahzad (2024) highlights, algorithms always get upgraded as new data and threats emerge. A variety of ML methods, strategies, and methodologies can be used to create models that employ reliable datasets to address critical challenges. ML techniques can often be grouped into broad types including; supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning.

Supervised Learning: During the ML-training phase, supervised learning algorithms take input data samples from training data and the outputs sometimes referred to as labels or responses relate to each input sample. It is intended to learn a mapping function to predict when an unknown data is correct. Algorithms for supervised ML make use of approaches such as Naive Bayes, k-Nearest Neighbours (k-NN), Decision Trees, Random Forests, Support Vector Machines (SVM), and Artificial Neural Networks (ANN). The data used to train the algorithm is labelled with the proper responses for supervised learning to take place. For instance, a classification

system that has been trained on a dataset of photos that have been accurately labelled with the species of the animal and a few traits would develop the ability to recognise. See Figure 21 as illustrated.

Unsupervised Learning: Labelled datasets are unavailable and not required, instead, it identifies pertinent patterns in the data by analysis. Techniques for unsupervised learning are advantageous and useful in this context. These methods look for patterns and draw conclusions from a dataset of merely input data (labels). Although unsupervised methods have higher uncertainty ratings, the raw data can still reveal a lot about the predictions made by these models. (Sarkar et al., 2018).

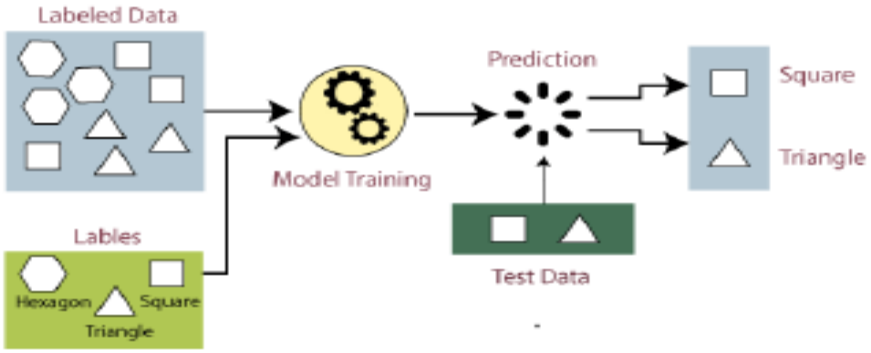


Figure 21: Supervised ML model

Source: Adapted from Javatpoint (2022)

Semi-supervised Learning: Semi-supervised techniques relate to learning algorithms that combine elements of both supervised and unsupervised learning. A little amount of supervised learning from pre-labelled data and a sizable amount of unsupervised learning from training data are both used in these technologies (Hackeling, 2014).

Reinforcement Learning: Reinforcement learning techniques differ slightly from typical supervised or unsupervised procedures. This method involves an agent acting in a certain manner in a specific environment in compliance with a rule or policy by observing the current state of the environment. The agent then receives a reward as an interpretation of the activity, which may or may not be advantageous.

It is used for continuous learning as well as integrating security protocols in real-time.

2.1.37 Classification Algorithms

ML components such as classification algorithms are used to group or categorize data into certain classes based on specific input features. These algorithms gather knowledge from a training dataset that includes samples of each class label, and then uses that understanding to predict the class of new, unseen data. There are two types of classification problems:

i. Binary classification, which has no more than two clearly defined labels, such as malware detection can label software as either malicious or safe, and
ii. Multiclass classification, which has more than two labels. The algorithm can group data into multiple categories. The following is a description of the typical classification techniques used in malware detection; (a) Naive Bayes (b) k – Nearest Neighbours (k-NN), (c) Decision Trees, (d) Random Forests, (e) Support Vector Machines (SVM), (f) Artificial Neural Networks (ANN) (Almashhadani, 2020).

2.1.38 Decision tree

A decision tree is a special type of algorithm used in ML and data mining for data classification and prediction. Each node in this tree-like structure represents a feature or characteristic, each branch a value for that feature, and each leaf node has a class label or prediction. Figure 22 shows how the algorithm makes decisions and the decision-tree ML structure.

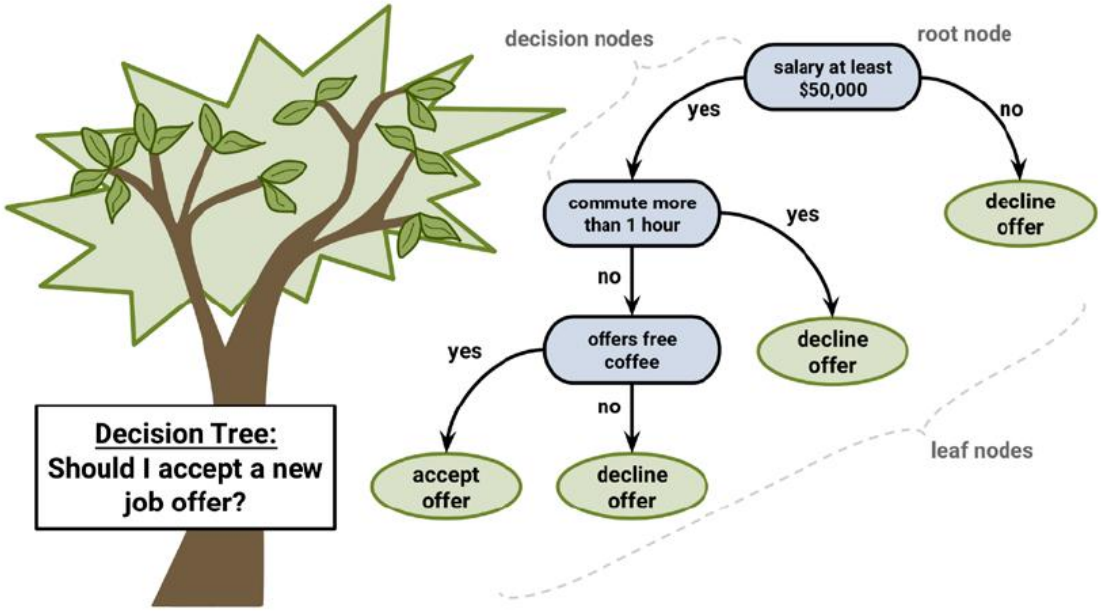


Figure 22: *Decision-tree structure*

Source: Adapted from Shalev-Shwartz and Ben-David, (2014); Almashhadani, (2020)

The decision tree algorithm begins at the root node and moves via decision nodes where decisions must be made considering the characteristics of the task. Then chooses a feature to split data, divides the data according to the value of that feature, and creates new branches for each imaginable value of that feature. Whenever the algorithm reaches a leaf node, which stands for an end or prediction, this procedure is repeated recursively. In this subset of the data, the leaf node's label is frequently the most prevalent category, or in other situations, a probability of distribution over the classes (Almashhadani, 2020.)

When applying binary classification to build decision tree, the two common metrics are calculated: the Gini Index and Entropy**.**

Gini index: The Gini Index, or Gini Impurity, assesses the amount of erroneous classification of a randomly chosen piece from the dataset. It spans from 0 (entirely pure) to 0.5 (most impure in binary classification). The calculation for the Gini Index applicable to an issue involving binary classification is:

$$Gini = 1 - \sum_{i=1}^{k} P_i^2 \quad (2.1)$$

Where:

Pi is the possibility of class *i.*

K is the number of classes (in binary classification, k=2*).*

For solving binary classification issues (with classes 0 and 1)

Entropy quantifies the degree of unpredictability or impurity within a collection of instances. Decision trees employ it to measure the disorder (or impurity) inside a dataset. The entropy calculation for an issue related to binary classification is articulated as follows:

$$Entropy = -\sum_{i=1}^{k} P_i \, log_2(Pi) \quad (2.2)$$

Where; Pi is the possibility of class *I, K* is number of classes (in binary classification, k = 2).

The mathematical formulas of the Gini index and Entropy in decision tree algorithms are appropriate for classifying binary file signatures. In the decision tree, both are employed to assess impurity (or disorder) across the tree construction steps. Furthermore, they can be employed to categorize 0 as benign and 1 as malware file signatures, in addition to assessing impurity or ambiguity within the dataset.

2.1.39 K-Nearest Neighbour

The k-Nearest Neighbour (KNN) technique is a simple but powerful supervised ML approach that can used for regression and classification issues. KNN works by remembering the training set and then predicting the label of any new instance based on the labels of its nearest training set. KNN implies that related things are close to one another (Lantz, 2013).

The KNN algorithm's performance is influenced by the value of K, which represents the number of neighbours for making predictions. When K is larger, the KNN algorithm uses the prior value, which could produce an unreliable result. The model can become vulnerable to outliers as result of the smaller K values. Therefore, to

select an accurate value of K, the value should fall between the two extremes, in the centre, to reduce error and provide the optimum generalization performance. The distance or similarity between the tested instances and the training examples is measured using a variety of distance metrics. The performance of KNN is determined by the how close it is to the K value. Certain distance measurements are calculated mathematically below (Lantz, 2013).

Euclidean distance: One of the most common distance metrics which is a choice that respects all dimensions equally and is the default for numeric attributes between the benign and malware (file signatures). It measures the distance between two points. The Euclidean distance formula is as follows:

$$d(x, x') = \sqrt{\sum_{i=1}^{N}(xi - xi')2} \quad (2.3)$$

Where:

$x = (x_1, x_2, \ldots, x_n)$ is the feature vector of the new file signatures to be classified.

$x' = (x'_1, x'_2 \ldots x'_n)$ *is* the feature vector of the known file signatures that originate from the training set.

n is the number of features.

Euclidean distance is particularly advantageous for the supervised classification of malware and benign files, especially in algorithms such as KNN, as it offers a straightforward and efficient method for quantifying the similarity between data points in the feature space.

2.1.40 eXtreme Gradient Boosting (XGBOOST) Algorithms

To improve the precision of supervised learning models, the open-source machine-learning algorithm XGBoost (eXtreme Gradient Boosting) uses gradient boosting. Gradient boosting is a ML technique that combines several weak models (usually decision trees) to create a strong model that can make accurate predictions. XGBoost, is an algorithm that uses decision trees in an ensemble learning mode, while integrating the predictions of multiple decision trees that correct each other's errors as they grow. XGBoost is a very potent and powerful algorithm for binary classification problems, such as to sort file signatures into malware (1) and benign

files (0). It works like a supervised ML model. The algorithm learns to guess whether file signatures are malware or benign by looking at their characteristics.

The XGBoost classifier's objective is to discover a function f(x) that maps the feature vector x to the binary label y (0 or 1). To do this, XGBoost employs several decision trees, which then merge to generate a final prediction (Biau and Cadre, 2017).

The objective function in XGBoost is made up of two parts: the loss function and the regularisation term:

Loss function - measures how well the model correctly fits the training dataset, while,

Regularisation term - regulates to prevent over fitting – the complication of the model, and maintain its simplicity

Objectives $\mathcal{L}\ is\ given\ \ \mathcal{L}(\theta) = \sum_{i=1}^{N} L(y_i, \hat{y}i) + \sum_{k=1}^{k} \Omega(fk)$ (2.4)

Where:

$L(y_i, \hat{y})$, denotes the loss function that measures the true label y_i and the projected$\hat{y}i$, by finding the difference between them.

$\Omega(fk)$ – term that regulates and corrects the complexity of the $k-th$ decision tree.

N – represents the number of training examples.

k – represents the number of decision trees in the boosted model.

fk – represents the *K – th* weak learner (decision tree).

Summary

Next generation cyber-attacks are posing a challenge to all the CS models and systems previously used and studied. The attacks make it obvious that existing models need an upgrade and needs strategies to proactively combat modern cyber threats. (Al-Shamisi, 2014). One major deficiency has been identified to be the absence of a real-time, automated, and legally-compliant CTI sharing model (Brown et al., 2015). Studies reveal that even though industries are eager to receive CTI,

they are reluctant to share it primarily because of concerns they have around privacy, legacy breaches policy restrictions and strategy (McAfee, 2016; Ullman, 2016).

DML, CTI models and platforms like MISP and WIB are models that exist but now irrelevant because of their central CI storage feature. Legal concerns, and lack of trust due to the risk of sensitive data being exposed is sabotaging CI exchange (Mohaisen et al, 2017). Even though automation offers quick response and consistency in the management of CTI (Brown et al., 2015), privacy and trust issues remain unattended to (Johnson et al., 2016). CI sharing that is done within trusted members can still be exploited and cause systemic harm (Nolan, 2015).

The PSDCIS model has been proposed as a solution to these challenges as it provides real-time, automated intelligence processing, and exchange, as organisations and businesses will get alerts. It provides a secure and faster response without legal or operational compromises, giving modern cyber threat management capacity to scale.

Chapters three reviews the research methodology in detail, outlining the methods, approach, model assessment, and how to share new intelligence.

CHAPTER 3: RESEARCH METHODOLOGY

Review research methodology

The methodology used in this study anchors the outcome of facilitating an innovative circular model where raw data is collected, processed, analysed, and shared in real-time to protect cyber-attacks. The study investigated different research methodologies before choosing the quantitative method which involves the process of collecting, analysing and interpreting numerical data to test models (Creswell, 2009; Al-Shamisi, 2014).

Quantitative method employs a deductive approach which considers transforming of theoretical frameworks into measureable variables. The process uses statistically valid analysis for empirical testing thus helping in the evaluation of the casual relationships that exist among variable (Al-Shamisi, 2014). In this study, real-time CI sharing is measured objectively based on the supportive attributes of the quantitative method to analyse structured data. This process assures to be an important research driver as it promises to provide the desired real-time CI sharing model. More so, the quantitative method hinges on the philosophy of positivism that has its foundation on hypothetical reasoning and experimentation. It provides a means for testing research objectives philosophically by examining the connection that exists between variables (Park, Konge and Artino, 2020).

Based on the foregoing, the study's methodology was established by exploring existing philosophy of knowledge and incorporated it into hypothetical perspectives in order to gain further insight and contribute to knowledge. Several researchers have explored assumptions backed by philosophies in an attempt to establish foundations that support finding knowledge (Crotty, 1998; Creswell, 2003; Al-Shamisi, 2014). The foregoing formed a guide for the study's philosophical examination of existence in the lens of objectivism, subjectivism, and constructivism before ascertaining the research paradigms for the study which are: positivism and post-positivism.

As Creswell (2009) posits, research design has three core influences which are the components of: Knowledge claims, strategies of inquiry, and research methods and their interrelation gave rise to the methodological framework for this study.

The design of this study is therefore guided by the following questions;

i. What knowledge claims and philosophical assumptions underpin this research study (epistemology and ontology)?

ii. What strategy of investigation and inquiry aligns the most with the research questions?

ii. What methods are most effective for collecting and analysing data in this study?

As illustrated in Creswell (2009) and supported by (Crotty, 1998) (Al-Shamisi, 2014), the above components when interpreted and integrated logically and methodologically, will lead to a well-structured research design. Figure 23 illustrates how knowledge claims, strategy and methods lead to research approaches and design procedure.

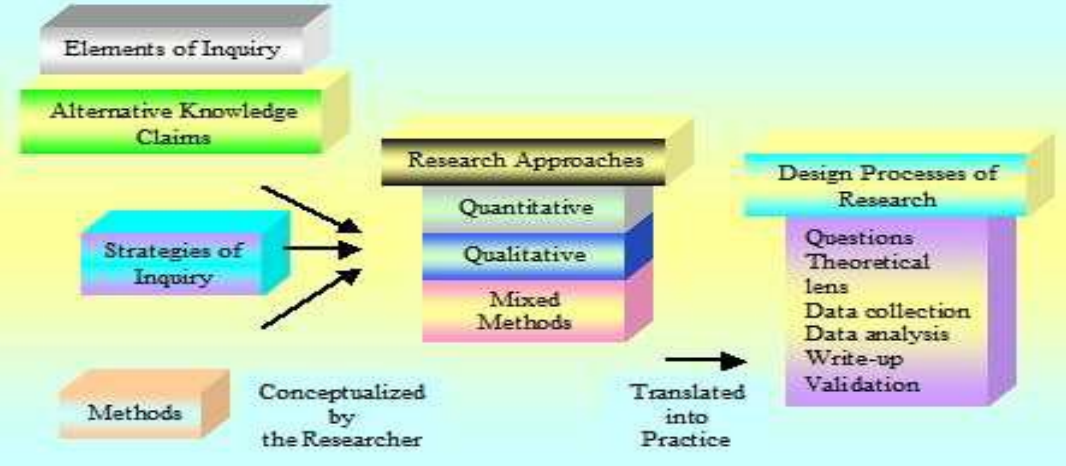


Figure 23: Knowledge assumptions, strategies of inquiry and methods leading to approaches and design procedure

Source: Adapted from (Crotty, 1998; Creswell, 2009); Al-Shamisi, 2014)

Declaring a knowledge claim involves identifying evenly a philosophical position regarding: a) what realistic knowledge is involved (ontology), b) how is this knowledge acquired (epistemology), c) what role does values play (axiology)

d) what is the language of this research (rhetoric) e) what procedure of investigation are used (methodology). These are the paradigms of this research design as validated by (Guba and Lincoln, 2000; Martin, 2008; Crotty, 1998; Neuman, 2013).

Theoretically, researchers make assumptions regarding what knowledge is (ontology), how we recognise it (epistemology), what values go into it (axiology), how we compose it (rhetoric), and the procedures of learning it (methodology). This study aligns with the post-positivist and positivist schools of thought. Their application as research methods in the development of real-time CI sharing model will be discussed elaborately in subsequent sections.

3.1.1 Research Theory

This part of the study delivers an overview of the different theories and philosophies that are most appropriate for the study's methodology. There are two major theoretical frameworks that are relevant to implementing this research: the Design Science (DS) approach and the Machine Learning (ML) approach.

The Design Science (DS) approach performs in the iterative development and evaluation of artefacts which solves organisational problems when identified. This is established where knowledge base meets a practical environment (Hevner, 2004). This DS contributes to theory and practice by producing artefacts like methods, frameworks and models, which are painstakingly assessed by their utility, efficacy and validity. For this research, DS aligns with the development of real-time CI sharing models and is driven by the model's design which has both theoretical constructs and real-world application.

The Machine Learning (ML) approach works with DS by focusing on the identification, classification and analysis of patterns in large datasets. ML is concerned with justifying how the knowledge analyses can be identified as valuable knowledge in the proposed CI sharing model development (Xue et al., 2019).

3.1.2 Design Science Theory

Scientific research is concerned with contributing new knowledge to the world, while learning pertains to the acquisition of new knowledge to oneself. The

core of DS in information systems is knowledge development through the design and evaluation of artefacts with intention of solving the problems identified. Research activities in DS are performed in interaction with an established knowledge base and a real environment with a real problem (Hevner, 2004). Practical needs together with the use of academic theories drive research in a DS approach. According to Hevner (2004), DS research outcomes are a) practical contributions, such as models, methods, systems and constructs which are all artefacts designed to tackle problems, and b) theoretical contributions, which enhance the knowledge base. A useful example is a discussion on the relationship between design and learning; where an Incident Response Team (IRS) perceives a cyber-attack from a suspected IP address, then shares that intelligence with other teams or organisations. That operation allows for implementation of proactive defence measures, which will reduce the risk the threat would have on the vulnerable organisation. This scenario demonstrates the use of artefact-driven solutions when supported by the DS approach. This problem resolution orientation as a DS paradigm has its origin in engineering and science of the artificial, not on ordinary observation (Simon, 1996). With this process of creating artefacts and iterative evaluation, DS supports this study's goal of developing a real-time cyber intelligence sharing model capable of reducing the risks of cyber threats.

3.1.3 Machine Learning (ML) Approach

This part of the study is an overview of the different ML theories and approaches that are most appropriate for this research methodology. ML is a subset of artificial intelligence that focuses on algorithm development for systems that identify patterns, and make data decisions without programming (Simon, 1996). The purpose of ML approach in Cybersecurity and CI sharing is usually is to recognise the structure of information/data and fit them into a model that can be identified and applied by humans in real world security arrangements (Tagliaferri, 2017). Major theories, such the supervised learning approach, are relevant to this research's application. Only supervised learning machine methods is used in this study's

research data and analysis. Applications such as malware detection, classification and detection of anomalous threats within a cyber-environment.

According to Arthur Samuel, ML is a set of methods that provide computers systems, the power to recognise and learn from data without being programmed. ML algorithms learn and formularise the philosophies that underpin the information it sees. Once the models are trained, with that knowledge, the machine algorithms can reason the properties of data samples they were earlier unseen. For example, in the detection of malware, the earlier unseen data could be the new malicious files, anomalous IP addresses, malware signatures, suspected email addresses, and malware-related advertisement on digital media. These models with accurate predictions will help with proactive threat mitigation. A mathematical representation of the features of data which has been logically structured through learning is called a model. Figure 24 illustrated the taxonomy of ML algorithms (Simon, 1996).

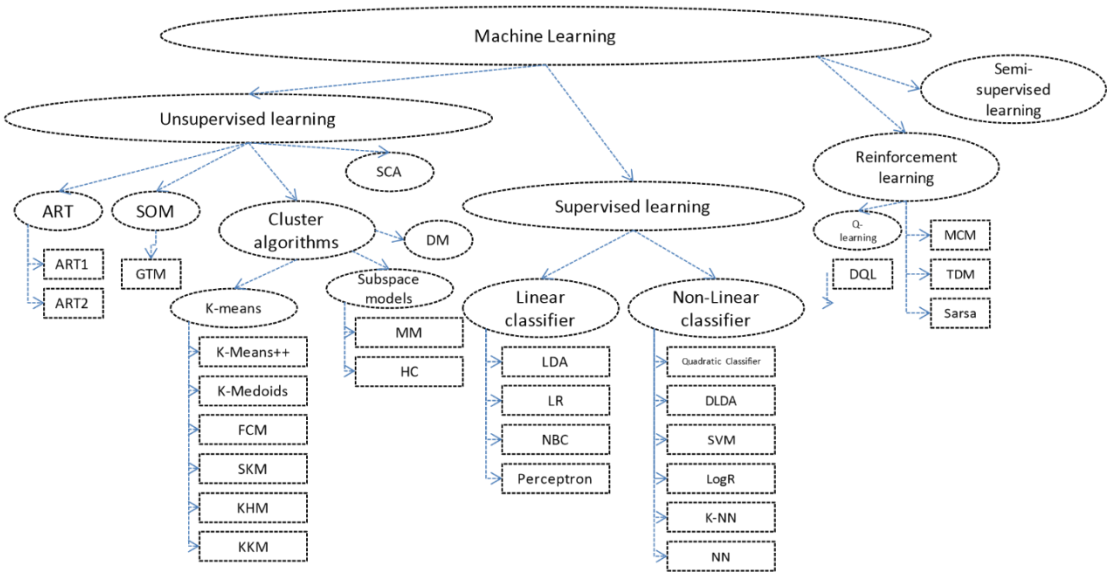


Figure 24: Taxonomy of ML algorithms

Source: Adapted from Hamza (2012)

ML has major paradigms such as:

i. Unsupervised learning (UL)

ii. Supervised learning (SL)
iii. Reinforcement learning (RL)
iv. Semi-Supervised Learning (SSL)

These four paradigms are relevant to Cybersecurity application but this study only used SL due to the fact it is precise, scalable and also suitable for the structuring of CTI data.

3.1.4 Supervised Machine Learning (ML) Algorithm

Supervised learning is the sort of ML in which algorithms learn well from labelled training data, and can predict the output of unseen data (Xue et al., 2019). Supervised ML algorithm is a model procedure trained on an input dataset as variables (marked X), and the corresponding output variables (marked Y) are known. The purpose of the SL algorithm's objective is to discover the relationship through learning a mapping function f: X→Y which composes of new, unseen data. Figure 25 illustrates the supervised ML algorithm process clearly.

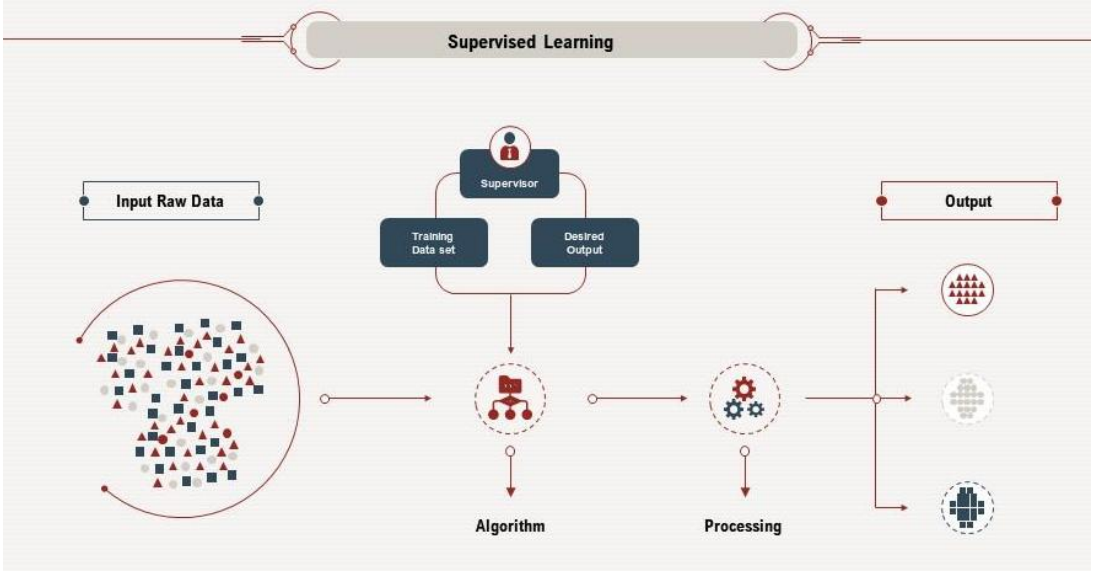


Figure 25: *Supervised ML algorithm process*
Source: Adapted from SlideTeam (2023)

Supervised learning has two stages:

i. Training stage: At this stage, a model is taught with known and available datasets, where both the input features, and output labels are learnt.
ii. Application stage: At this stage, the model which has been trained is used to predict outputs of new unlabelled data feeds and on the learnt mappings.

Figure 26 illustrates supervised machine lifecycle and explains practically how the algorithm will be applied. In addition, it provides the tasks that the algorithm should perform during training data and applying phases, where the algorithm model will be given a set of input objects and a) Each object is represented with attributes set X, b) Each object is plotted to correct a prediction target and is labelled Y. Figure 26 illustrates supervised machine lifecycle.

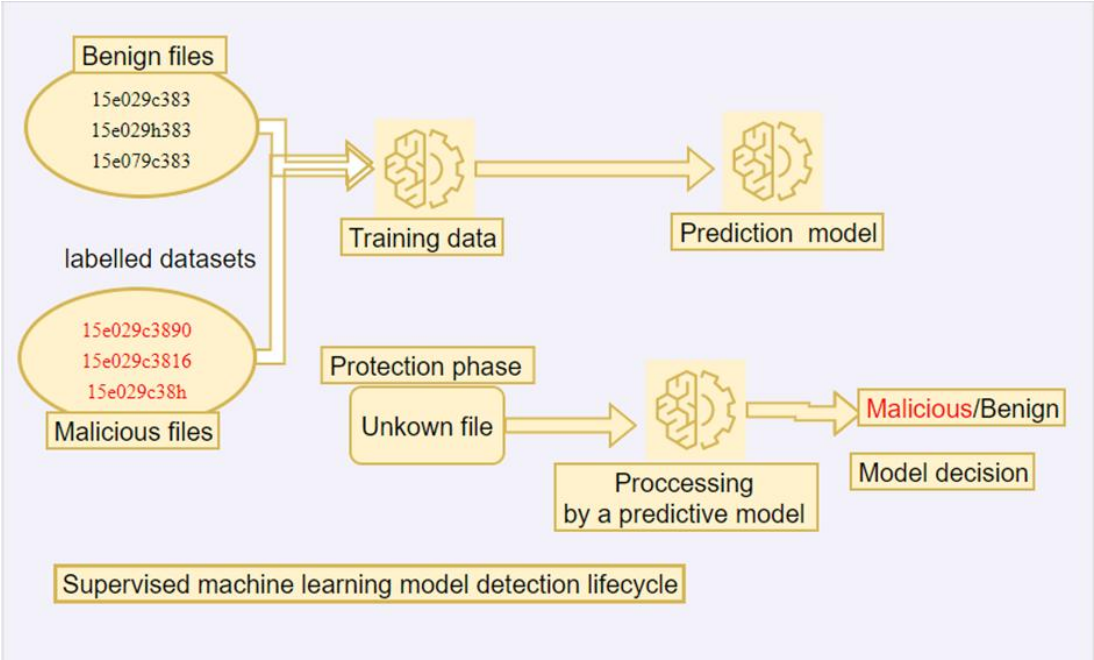


Figure 26: Supervised Machine Lifecycle

Source: Researcher (2023)

This training phase is established to develop the best model that will supply the correct labels to unseen objects considering their attributes. For example, for malware detection, input X could be file content, digital file signatures, system

behaviour, or lists of API calls. Labels Y for output could binary groupings like malicious or benign or even other precise multi-classifications such as adware, virus, Trojan downloader etc. (Kaspersky, 2021). Subsequently, after training the model and confirming its quality, the model is ready to be deployed to the next phase – predicting the outcomes from new inputs. From this point, the architecture of the chosen model and its inner parameters will not change. The model now produces prediction outputs from what it learnt. During the SL process, a model family must be chosen such as algorithms like:

i) Lanier's regression

ii) Decision trees and

iii) K-NN

Every of these models have their own specific set attributes and strengths and suitable to tackle various kinds of learning challenges. Model training is aimed at finding and choosing optimal conditions in that model family that provides the most accurate predictions, over a set of references for a specific metric evaluation. .

SL is grouped into two categories: regression and classification.

a) Regression

Regression algorithms are a form of supervised ML used when predictions about output variable Y is continuous as they are trained to estimate quantities of real value such as sales, weight, prices, or temperature. Regression task would have occurred when the number of threats attempts on a given network, over a period of time, is predicted (Ali, 2022).

b) Classification

Classification algorithm is a type of supervised ML where computer algorithms learn from historical data to classify inputs into secret categories. A typical classification task is to differentiate between types of cyber threats. ML techniques are used to predict distinct result categories. Binary classification refers to the process of categorising data into one of two possible outcomes, such as default or no default, true or false, or yes or no (Ali, 2022). Figure 27 illustrates different classification algorithms examples.

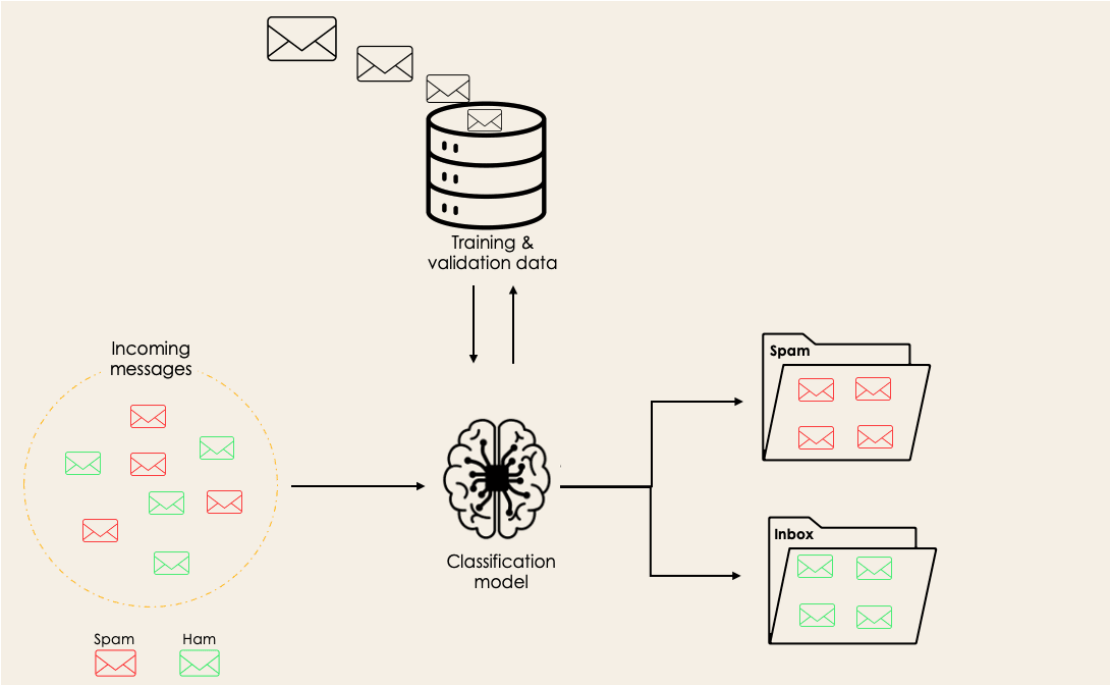


Figure 27: Classification Algorithms example

Source: Adapted from Keita (2022)

3.1.5 Understanding Model Lifecycles in Supervised Classification

The lifecycle of supervised ML models provides a realistic explanation of how the algorithm will be used in real-world settings. It also lists the duties that the algorithm should carry out during the phases of applying data and training data. The model will get a dataset or set of objects in the training stage thus: (a) A collection of input objects (b) The attribute set X used to represent each object; (c) Every object is plotted to the correct classification or response labelled Y.

This training stage is to develop a model that can learn the best mapping function f(X)→Y that predict generally to unseen instances using previous observations made. This function through minimal loss function is discovered across a dataset during training. For the malware detection, X could be some structural or behavioural attributes of files, like file headers, imported library functions, byte sequences, API call frequency and heuristic signatures or other behaviours. Labels Y can be binary values such as malicious or benign or even other

precise multi-class values such as adware, virus, and Trojan downloader etc (Kaspersky, 2021). Subsequently, after the training and the model's performance is confirmed by a separate entity to be quality, the model is ready to be deployed in the application or inference phase. From this point, the type of chosen model and its attributes do not change. The model produces prediction of the intelligence as output for new data.

The classification process undergoes multiple stages prior to delivering the intended outcomes. Labelled data will be employed to train the algorithms; the dataset will facilitate model training, enabling the model to differentiate between benign and malicious characteristics during evaluation just as illustrated in Figure 36.

3.1.6 Unsupervised Learning (UL) Algorithms

This unsupervised ML is an automatic scanning process in which algorithms (models) are not supervised by humans at the time of data formation – so no labelled datasets. As an alternative model which discovers hidden patterns by itself, it does not understand from any given data. The UL cannot explicitly implement any regression nor classification issues in comparison with supervised learning, where data is imputed and is related to the output data. The objective of unsupervised ML approach is to process unlabelled data by transforming them based on similarities, spread, or statistical regularities, representing that dataset in a compressed format (Javatpoint 2022).

Unsupervised ML employs unlabelled data entries, resulting in the absence of data aggregation and output generation. The ML model uses this unlabelled data input to train the algorithm. The core objective of this programme is to identify and extract raw data from the underlying discrete patterns and subsequently apply appropriate algorithms, such as K-means and decision trees (Javatpoint 2022). As illustrated in Figure 28, the outcome of this process can become intelligence that humans can use.

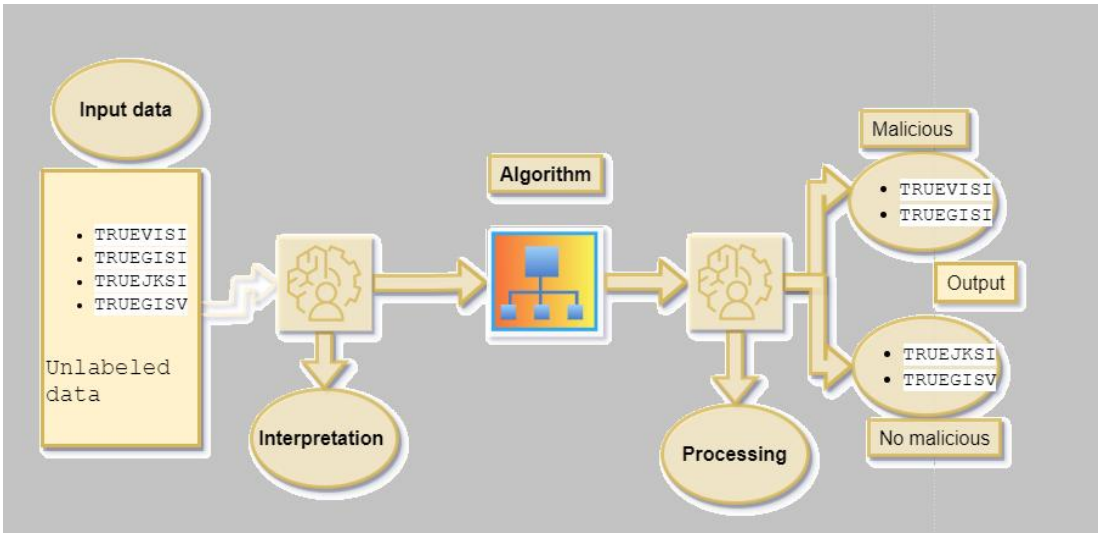


Figure 28: *How unsupervised learning works*

Source: Researcher (2023)

UL facilitates the extraction of valuable ideas from data and closely resembles the way humans assimilate knowledge via personal experience. It operates on unlabelled and uncategorised data, as illustrated in Figure 38, so enhancing the significance of unsupervised knowledge. In reality, input data does not always correlate with outcomes, necessitating the use of UL learning for certain circumstances.

3.1.7 Classification Algorithms

Classification is a subcategory under supervised ML procedures where the algorithm is trained to identify and assign the input data to discrete category. (javatpoint 2022). Classification algorithms learn to predict given datasets or new observations and then categorizes new observation into number groups such as binary values 0 or 1, Yes or No, Not Spam or Spam, Malicious or Benign etc. (javatpoint, 2022). Those classes can be named targets/label or categories.

The classification technique in a mathematical format is where a mapping function (y) is mapped to input (x). In the equation y=f(x) where y = predicted output and x is the feature vector. An example of classification algorithm is in an email spam detection system, the classifier analyses an input email (x), and predicts

if it is spam or not spam (y). ML algorithms such as, SVM, decision tree is used to automate the process of identifying malware (Milosevic, 2017).

Classification algorithms can be divided into diverse groups such as, linear models and non-linear models.

i. Linear models

These linear models take decisions of classes in a linear format and are also divided further into: Logistic regression (LR) and Support Vector Machine (SVM)

ii. Non-Linear Models

These non-linear models are best for complex classes that are non-linear in nature. They are divided into K-Nearest Neighbours (KNN), Kernel-based SVM, Naïve Bayes, Decision Tree Classification and Random Forest Classification.

All these classification models are used to train and test the phases of this study, to analyse datasets and predict the type of malware present.

3.1.8 Positivist Approach

Positivist approach is to create exploratory relationship or causal association from observing and measuring variables. This eventually leads to prediction that help control the phenomena in question (Sciarra, 1999). The philosophical opinion of positivism in the clearest form is rooted in the framework established by John Stuart mill in *A System of Logic* where knowledge is attainable by systematic observation and logical reasoning (Mill, 1843; Gergen, 2001).

This research follows a positivist epistemological ideology as well as a circular process that starts with acquiring knowledge through data collection; secondly, processing and analysing of data using experimental design; and finally, application of results obtained to real problems which is CI sharing. The research process has three stages:

i. Construct testable research questions from theoretical framework and available literature
ii. Design experiments through systems modelling
iii. Conduct empirical studies using ML experiments

These stages culminate in a study, and the result of such studies is used as contribution to knowledge on CTI and malware detection (Park, Lars, and Anthony, 2020). Positivism as used in this study aligns with the principles espoused by Creswell, (2009) and Crotty, (1998) that knowledge must be acquired in a measurable and replicable form. This thought also assigns more significance to the research process of knowledge claims, research strategy and methods. It also approves the advantage of using quantitative research approach and advocates for focus in acquiring knowledge through literature, developing research hypotheses, methodology and methods. The assumptions of the positivist knowledge claim in this study include these five beliefs:

i. It believes that acquiring knowledge is through scientific methods.
ii. It believes that scientific truth comes from experimental facts and objective observation.
iii. It believes in the philosophy of strict empiricism for development of knowledge claims.
iv. It believes that exploring and understanding real world situations has to be measured and verified.
v. It believes that the prediction and formulation of general laws are central.

3.1.9 Quantitative Research Approach

Quantitative research methodology is that step by step process that we undertake to evaluate and analyse measurable variables in order to get objective results (findings). As explained by Leedy and Ormrod (2001), research methodology is an all-inclusive system used to get answers to questions like; “who?”, “how much?”, “what?”, “where?”, “when?” “how many?”. Clarifying on this explanation Aliaga and Gunderson (2002) said that quantitative research method explains the phenomena by collecting numerical data and analysing them with mathematical methods in specific statistics. Considering the explanation above, out of the scales indicated in both the quantitative and qualitative comparison, the research approach, is quantitative method. As Leedy and Omrod (2001) and

Williams (2011) observe, the process of quantitative research starts from collection of data which is subjected to statistical treatment in order to test or refute hypotheses or research question. Moreover, Williams (2011) clarifies that quantitative research begins with statement of a problem, generating research hypothesis or research questions, review of associated literature and a statistical analysis of collected data. With this understanding, this study collected data using quantitative approaches through predefined instruments and the use of ML. These algorithms uses mathematical and statistical methods to classify task using SL, predict, and evaluate CTO. To explain what quantitative study all is about, it is also important to explain its difference between from qualitative research. Table 4, which is an adaptation from Johnson and Christensen (2008, p.37) gives a clear explanation of the differences between qualitative and quantitative research and the difference in their nature when applied in research.

Criteria	Qualitative Research	Quantitative Research
Purpose	To understand and interpret social interactions.	To test hypotheses, validate cause and effect, and predict outcomes.
Group Studied	Smaller and not randomly selected samples	Larger, and randomly selected samples.
Variables	Comprehensive Study of the variables	Specific variables studied with focus
Type of Data Collected	Words, images, or objects	Numbers and statistical data.
Form of Data Collection	Interview responses, participant observations, field notes, and reflections.	Structured and validated instruments with accurate
Type of Data Analysis	Identify patterns, features, meaning, and themes.	Identify statistical relationships and significance
Objectivity and Subjectivity	Subjectivity is expected	Objectivity is critical.
Role of Researcher	Researcher and their biases may be known to participants in the study, and participant characteristics may be known to the Researcher	Researcher and their biases are not known to participants in the study, and participant characteristics are deliberately hidden from the researcher (double-blind studies).
Results	Contextual findings that is less generalizable.	Generalisable findings across populations.
Scientific Method	Exploratory or bottom–up; generates hypotheses and theories from the data collected	Confirmatory or top-down; the researcher tests the hypotheses and theory with the data collected.
View of Human Behaviour	Dynamic, situational, social, and personal	Regular and predictable
Most Common Research Objectives	Explore, discover, and construct meaning.	Describe, explain, and predict
Focus	Wide-angle lens; examines the breadth and depth of phenomena as a broad scope.	Narrow-angle lens; test specific Hypotheses.

Nature of Observation	Study behaviour in a natural environment.	Study behaviour under controlled conditions; isolate causal effects.
Nature of Reality	Multiple realities that are subjective.	Single reality; objective.
Final Report	Narrative report with contextual descriptions and direct quotations from research participants.	Statistical report with correlations, means, and significance levels

Table 4: *Comparisons of qualitative and quantitative research methodologies*
Source: (Johnson and Christensen, 2008, p. 37

3.1.10 Research Approach

In research methodology, there are two different approaches: deductive and inductive. According to Saunders et al. (2007) and Al-Shamisi (2014) positivism is aligned with deductive approach while interpretivism is aligned with inductive. Creswell (2009) further states that quantitative studies use deductive approach and researchers examine theories by testing hypotheses or research questions.

3.1.11 Deduction Approach

The deductive approach is known to develop a theoretical framework which as a philosophy is tested rigorously. According to Saunders et al. (2007) and Al-Shamisi (2014), this approach is known to originate from the field of design science where knowledge development happens through the design of artefacts and real scientific testing independent of our subjective mode of thinking. Williams (2011) clarifies that quantitative research as a deductive approach begins with the statement of research problem, then the generation of research hypothesis or research question, next is review of literature and a quantitative analysis of data is done to provide actionable insights. Creswell (2003) and Williams (2011) states that quantitative research uses strategic instruments (tools) of enquiry like experiments and surveys, to collect data which is analysed using statistical methods.

This study uses the deductive approach, while being rooted in PSDCIS theories. The PSDCIS theories will be tested to understand how intelligence gathering, processing, analysis, and sharing contributes to the production of new intelligence.

3.1.12 Summary of Justification for Using Quantitative Research Methodology

The research employs a quantitative approach to develop an innovative circular model for real-time CI sharing, encompassing the stages of gathering data, processing and analysing data, and sharing to others. The choice to employ a quantitative method stems from its capacity to yield objective, measurable, and statistically valid outcomes, which corresponds with the study's objective of evaluating cyber-attacks through ML methods.

1. Philosophical Justification of Positivism and Deductive Approaches

The study adheres to the positivist paradigm, prioritizing empirical evidence, statistical analysis, and objectivity. It uses a deductive approach, initiating with theoretical concepts and validating them through systematic data collection and analysis. This is consistent with the perspectives of Creswell (2009) and Crotty (1998), who highlight the importance of testing assumptions using empirical data.

2. Methodological Rationale of Quantitative studies in this Research

Adopting this quantitative research is crucial for this research due to its involvement in:

i. Data-driven decision-making: The study necessitates quantitative data for training ML models to recognize and categorize cyber risks.
ii. Systematic data acquisition: Employing established algorithms and statistical instruments guarantees precision and reproducibility.
iii. Evaluating linkages among variables: This analyses the correlations between cyber threat indicators and the results of intelligence sharing.
iv. Comparative Analysis with Qualitative Research: In contrast to qualitative research, which emphasizes subjective interpretations, quantitative research offers measurable insights that can be applied to wider CI applications. Objectivity is essential in CS, as results must be statistically dependable to guide effective threat mitigation in the real world.

Practical Significance

The study combines design science and ML methodologies to categorise and

predict CI. SL algorithms such as Decision tree, LightGBM, Logistic Regression, KNN, Naive Bayes, Stochastic Gradient Decent (SGD), SVM, and Random Forest were employed to analyse raw data, underscoring the necessity for a quantitative framework.

Research Methods

The methodological approach of this study combines several research methods that are necessary to achieve tangible objectives within the sphere of CI sharing. The framework incorporates a systematic model that begins from data-gathering, processing, analysis, constructive modelling and the stage of intelligence sharing. This study employed data-gathering approach at the initial phase of the research, the processing and analysis stage include parsing gathered data into a readable format, checking for data duplication, completing training datasets, and producing new intelligence. The constructive modelling stage is for designing models, constructing the structure of the model, writing algorithms, and implementing ML algorithms. The final stage which is for intelligence-sharing, involves implementation. At this point, the new intelligence is automatically disseminated among the security devices, and notifications for managers and system administrators is generated using an authentication and authorisation method. The state machine technique will automate the intelligence-gathering, analysis, and sharing processes. Each of these stages reflects the underpinning deductive and positivist ideology, rooted in objectivist ontology, and quantitative methodology (Creswell, 2009; Saunders et al., 2007).

i. Data Collection approach: Collecting data from a variety of data repositories, each from a separate source. To produce useful CI, it will be necessary to collect data from many channels. Data collection techniques are:

a) Passive Data Extraction from internal devices or data systems under the control of organisations without directly engaging with external stakeholders. One such instance would be analysts monitoring internal forums, accessing system logs, intercepting intra-network traffic, and engaging in red-team assessments

amongst other internal organisational tasks. Here, it is important to emphasize that the term "passive" refers to an analyst who is not directly interacting with a cyber-threat actor or their organisation's data, just observation and system intelligence. (Brown et al., 2015).

b) Active Data Collection is when information is obtained from external networks or service systems controlled or influenced by threat actors. To actively do this means knowing that threat actors oversee any systems or intelligence, and so discovering external networks, monitoring Darknet, or interacting with honeypots is crucial to work against vulnerabilities (Brown et al., 2015).

ii. Processing and Analytical approach: Parsing collected data into usable formats, classifying previously gathered data, and doing analysis to separate badly behaved problems into its component parts is how this ensures suitability. The data will then be evaluated and predicted using ML algorithms. The analytical approach will employ a variety of techniques to ensure an accurate and fair evaluation that should foretell actionable intelligence. At this step, evaluation of the reliability of the data source and the calibre of the data gathered is put into practice.

iii. Constructive approach: The approach is used to design and develop a suitable machine-learning algorithm that can deliver the proposed CI sharing model, which is the objective. This approach aligns with the field of design science, where artefacts are developed to solve problems through innovative technology (Hevner et al., 2004). This stage produces an intelligent system capable of detecting cyber threats and actionable intelligence in real time.

iv. Intelligence-Sharing Approach: This stage is final and coordinates the automated exchange of intelligence to stakeholders and Cybersecurity devices and infrastructure. When a threat has successfully been detected, validated, and authorised a state machine algorithm will take appropriate actions, such as updating CSS with a patch file that contains new threats, sending alert, and executing the patch file into security systems like firewalls and antivirus systems. This stage operationalises the system environment to share CI autonomously.

3.1.13 Intelligence Gathering Approach

The initial stage of the research model involves gathering malware datasets from the internet, which are accessible for academic purposes as well as practical application in machine algorithms, to test this notion. The study applied the "malware dataset," which consists of secondary datasets acquired online from the Kaggle platform and VirusShare databases for research purposes via automated scraping.

A web scraping method was used to gather intelligence. This method involves automated extraction malware datasets and signatures from selected repositories. A Python-based web scraping tool, such as BeautifulSoup was configured to collect the dataset, facilitating the specification of both data sources and the data collection methodology. The system gathers and verifies data by removing duplicates and filling in missing information prior to converting them into CSV file for further processing, analysis and classification in the ML model.

Practical intelligence Context

In practical example, identifying the needs of a business is crucial and so creating the quantity of intelligence obtainable from information should have a defined goal. Organisations might want to have threat intelligence that identifies the command-control-servers of a malware family. This empowers them to pre-emptively instruct the security experts to block the malware as a mitigation strategy like blacklisting servers or updating firewall rules. As the intelligence process progresses, it becomes possible to revisit and repeat the steps. For instance, if the information gathered from the malware server reveals the same malware that is already known, it should be possible to establish a new course of action. Figure 29 illustrated this intelligence gathering first-step process step by step.

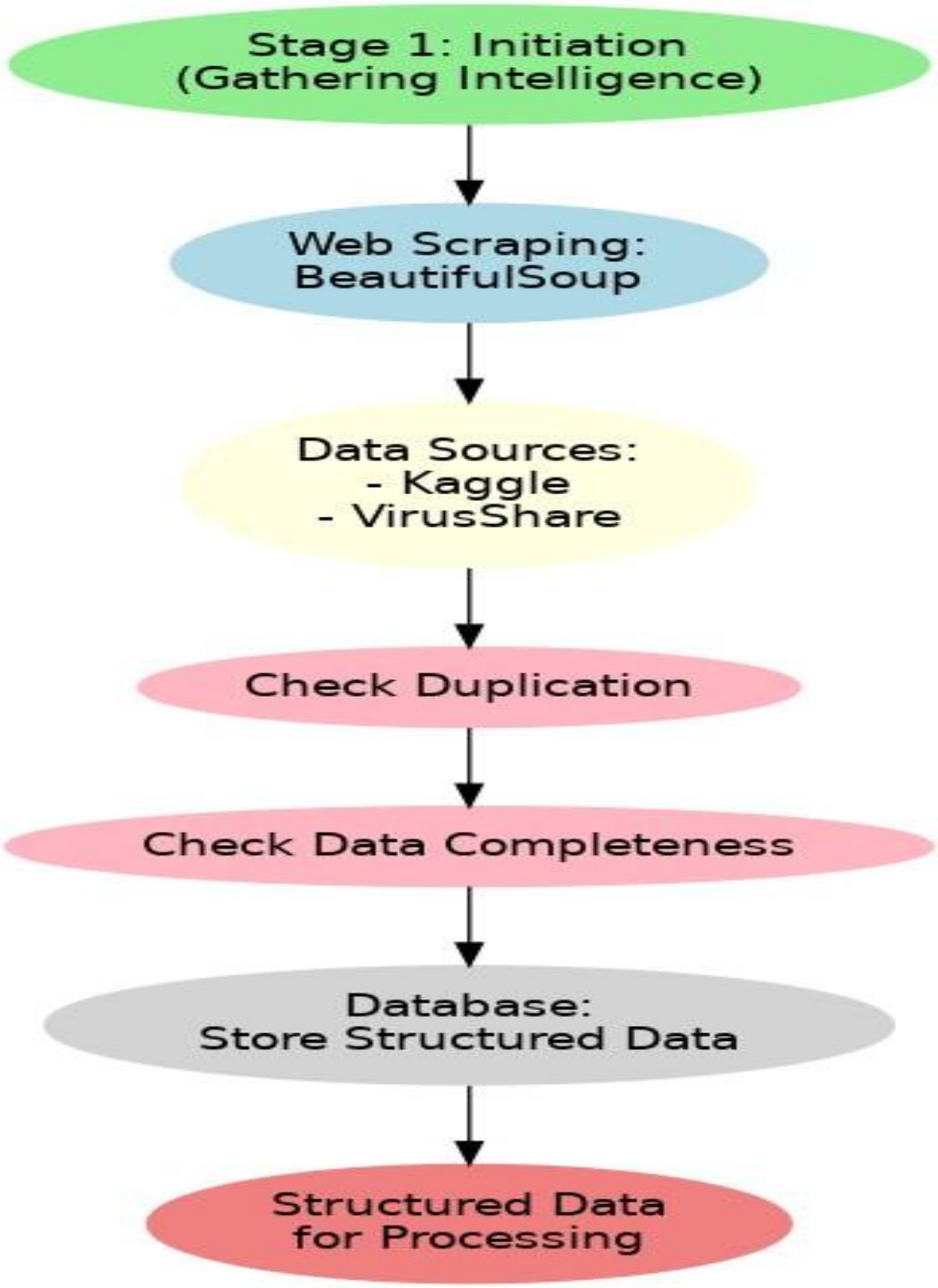


Figure 29: *Intelligence Gathering stage 1*

Source: Researcher (2024)

This study identifies three essential dimensions of intelligence data quality, and they are:

Duplication: During data gathering, mechanism should be in place to check if data was collected before, because the CTI life cycle is a continuous process.

Accuracy: Accuracy of CI data appears to be very important within cyber defence, as is highlighted in studies that many intelligence reports that false, contradictory or uncertain (Clausewitz, translated by Howards and Paret, 1989, p.117). (Clausewitz, C. tr. Howard, M. and Peter Paret:, 1989: 117). CTI must be accurate and reliable.

Completeness: According to a study by Ponemon Institute (2014), 70% of security industry professionals believe that CTI is very large in volume and complex and so cannot give reliable insight. The dataset must therefore be both complete and parsable to be used as intelligence.

3.1.14 Scrapping Using BeautifulSoup

The process of extracting data from websites through automated techniques is called web scraping. It involves writing a computer programme that can access a website, download its pages and HTML content, and parsing and extracting required information. Web scraping is used in research, data analytics companies, and business intelligence use for acquiring datasets available in public domains.

While manual data extraction from websites is possible, automated web scraping software or libraries like BeautifulSoup, Scrapy, and Selenium often yield more effective results. These tools can quickly extract massive volumes of data from numerous web pages within a short time. Figure 30 shows the web scraping process and libraries. The process involves different steps. Web scraping using BeautifulSoup are as follows;

i. **Import Python Supporting Libraries** such as: requests, BeautifulSoup, urllib.parse, os, sqlite3 and csv.
ii. **Send HTTP Requests:** Fetch HTML content using requests.get() function
iii. **Parse HTML Content:** Extract information with BeautifulSoup.
iv. **Format URLs**: Normalise URLs using urllib.parse
v. Manage Files: Handle file-related operations with **os.**
vi. Manage Database: Save processed data in SQLite using sqlite3 module.

vii. **Export Data**: Export data into structured CSV format using csv

viii. **Dataset Storage:** Upon completion of the scraping process, the dataset will be stored in a designated folder, which will subsequently be used for training and testing ML models.

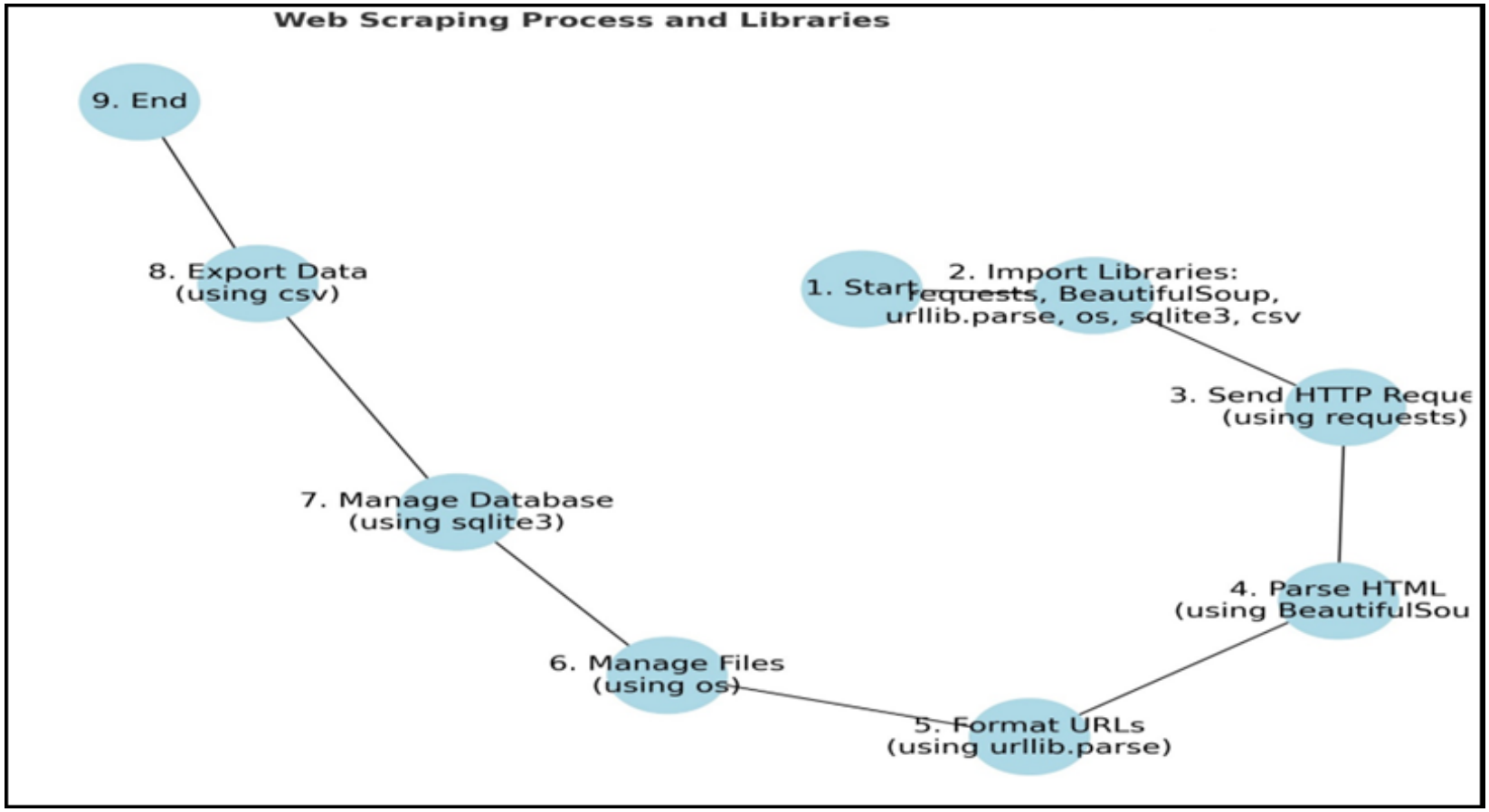


Figure 30: *Web scraping process and libraries*

Source: Researcher (2024)

3.1.15 Data Extraction

Python's BeautifulSoup package can extract data from structured documents like HTML and XML. According to (Richardson, 2020), BeautifulSoup collaborates with a selected parser to offer idiomatic methods for traversing, searching, and modifying the parse tree of a document. Often, this allows programmers to save hours or even days of effort. To navigate, search, and change a parse tree, BeautifulSoup offers several straightforward functionalities, including:

i. Pythonic idioms for document analysis: BeautifulSoup as a toolkit allows users to analyse a document and extract the information required without much code.
ii. Automatic character encoding conversion: Both incoming and outgoing documents through BeautifulSoup are automatically converted to Unicode (UTF-8). Unless the document does not provide an encoding and BeautifulSoup is unable to recognize one, then it must be manually declared by users. The original encoding is then all that must be specified.
iii. Parser flexibility and efficiency: With BeautifulSoup, you may explore multiple parsing engines like lxml and html5lib, while allowing for users to choose performance techniques or trade off speed for flexibility on the structure of well-known Python parsers.

3.1.16 Source of Intelligence Extraction

The source of the intelligence extraction is pre-defined through a script that combines BeautifulSoup modules with Python's requests module, serving as a crucial component in the creation of intelligence (using knowledge from them to defend them). The extraction process starts by BeautifulSoup initiating, using the request module "**get"** function. Issuing an HTTP GET request to a malware repository URL like:

result = requests.get ("https://virusshare.com/hashes.4n6")

The successful response is confirmed when the HTTP status code shows (200) OK indicating that the website is accessible and target source is valid for the intelligence gathering module by using;

print(result.status_code) tests to verify that we are gathering information on a correct website;

print(result.headers)

Figure 31 shows the HTTP headers extracted during this process, including time that intelligence was collected, and the intelligence source.

Figure 31: *The header of the source of the intelligence*

Source: Researcher (2019)

Figure 31 shows that the data collection script was linked to VirusShare. During intelligence gathering, there will be open lines of communication between the systems that are gathering data and the sources. When a connection is successful, the process of gathering intelligence continues with the system examining the source content to verify whether it contains malware file hashes. The module will check a predefined AP1 and connect to VirusShare specially to look for malware hashes (file signatures). Hexadecimal numbers known as "file hashes" uniquely identify each file. When a file content changes, the file signature automatically adapts as well. (src = outcome).

print(result.status_code)

print(result.headers)

Content in Figure 32, print (source), shows how many malware samples are currently available. In addition, it displays the quantity of virus hashes that are available, file size, and the fact that no two files have the same size.

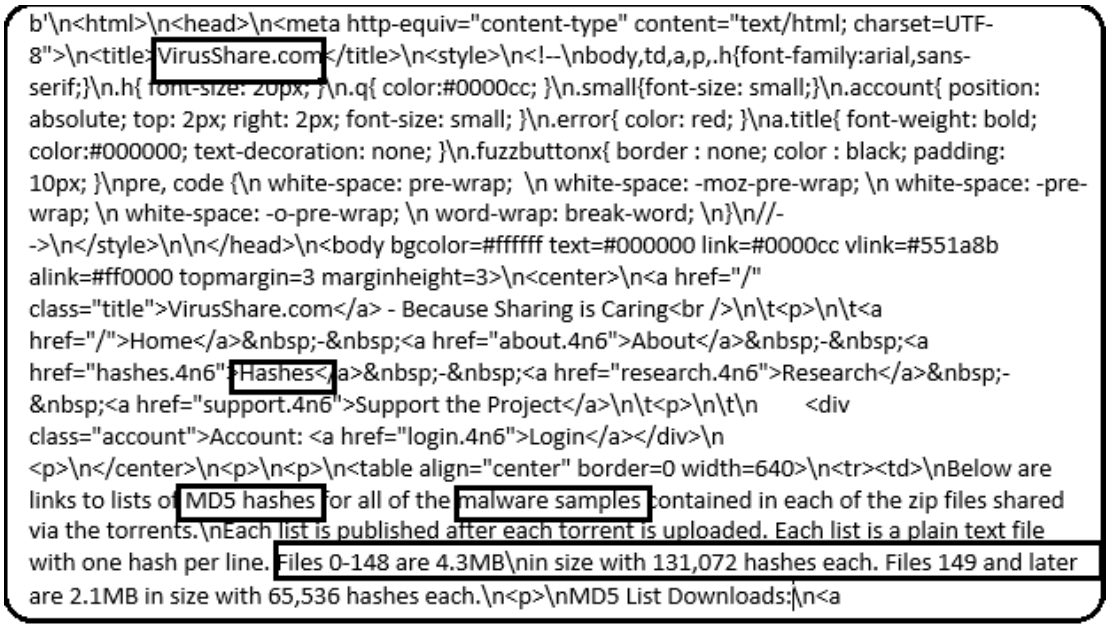


Figure 32: *Content of the intelligence*

Source: Researcher (2019)

The source – VirusShare uploads malware sample hashes in .zip archives that contain list of MD5 hashes in plain texts. The zip files uploaded via torrents each contain lists of MD5 hashes for all the malware samples that are there. Each list is released following the upload of each torrent. One hash per line appears in each list's plain text file format. The files 0-148 each have 131,072 hashes and are 4.3 MB in size. The size of files 149, contain 65, 536 hashes (2.1 MB). Figure 33 illustrates the file structure and how the data is represented.

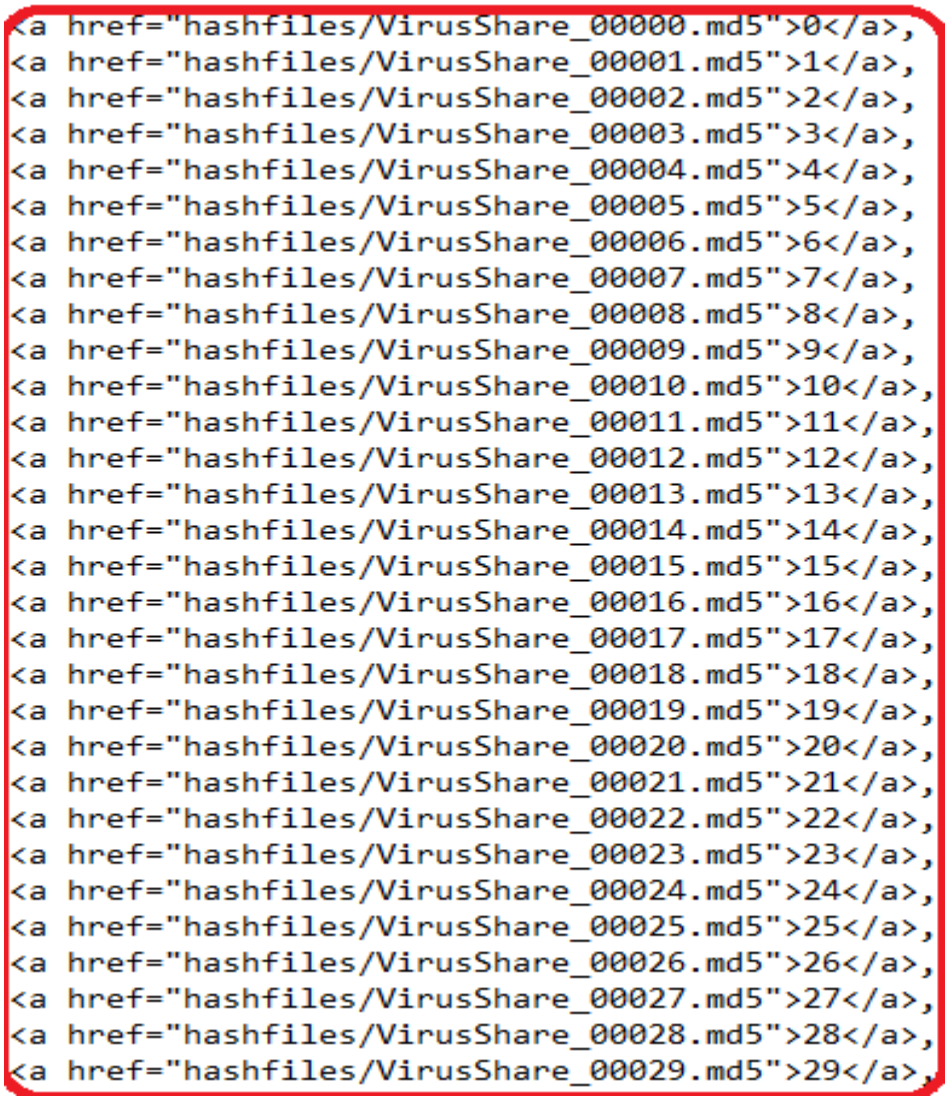


Figure 33: *Malware hashes*

Source: Researcher (2019)

The links with malware are presented as samples in Figure 33. As was previously mentioned, those files include malicious signatures, and are numbered sequentially, with file numbers, one always being the most recent malware sample. Once a link between the intelligence source and intelligence-collecting module has been established using a comparison method, the module checks to see if the data from the remote source and the intelligence baseline data are identical. The malware samples obtained in the past serve as the intelligence baseline. If the information has not already been gathered, the module will save the intelligence to the database.

Additionally, the module verifies file completeness, correctness and whether the source files contain malware samples. Baseline intelligence is a database that contains malware datasets that have been previously collected; using this technique will reduce false positives in the intelligence. According to Lenchner (2022), top financial institutions in the US use open web scraping systems to detect looming cyber threats. In practice, the automated systems help in the identification of malware, phishing domains, and other cybercriminal linkages. It offers the CI operators the visibility they require to proficiently, detect, address and prevent real-life cyber threats or incidents from infiltrating the security of their organisation and so take preventive measures against any attacker that could target their resources.

Another method that researchers use to gather intelligence is by downloading datasets from Kaggle; collecting ready-made secondary data from Kaggle. With this approach, all you must do is download the dataset, load the ML algorithm, and partition the training data into test sets and model sets, then import.

Author	Topic	Publisher
Butler, J. (2007)	Visual web page analytics	Google Patents
Bar-IIan, J. (2001).	Data collection methods on the web for info metric purposes – A review and analysis.	Scientometrics, 50(1), 7–32
Car, G., et al., (2013)	Detective: Dusting the web for finger printers	ACM SIGSAC conference, New York
Doran, D., and Gokhale, S. S. (2011).	Web robot detection Techniques: Overview and limitations	University of Connecticut
Hirschey, J. K. (2014)	Symbiotic relationships: Pragmatic acceptance of data scraping	Berkeley Technology Law Journal
Yi, J., et al. (2003).	Sentiment analyser using NLP techniques	IEEE Data Mining, Conference Melbourne, Florida

Table 5: *Similar theories for gathering intelligence by using web scraping.*

Source: Adapted from (Butler, 2007; Bar-Iian 2001; Car et al 2013; Doran and Gokhale 2011; Hirschey, 2014; Yi et al 2003)

Web scraping is a common technique used for gathering threat intelligence and other types of information. Table 5 shows researchers who previously used web scraping as a technique to gather intelligence.

The initial phase of intelligence collecting involves acquiring intelligence in its unprocessed form directly from the data source. The sharing of intelligence can be enhanced by integrating automatically collected data with ML algorithms that can identify and categorise the benign file signatures and the malicious file signatures. Subsequently, the raw data is transformed into a structured format like csv that is easily understood. Categorising the information in the subsequent stage is collecting intelligence. Once the information is accurate and in the correct format, it may be analysed and disseminated to facilitate detection, prevention, and response. The third stage involves storing the dataset in the intelligence database and preparing it for future processing by creating a (csv) file for the ML algorithms. The dataset used for this investigation contains 100,000 malware file signatures. 50,000 samples are classified as benign and the remaining 50,000 classified as harmful.

Intelligence is received from specific sources and compared with previously collected intelligence. The intelligence database will store any intelligence not previously collected. If intelligence has already been obtained, it will be disregarded in order to maintain data quality and prevent the duplication of information.

Constructive approach

The constructive approach in experimental research methodology entails the researcher actively developing novel concepts or models to address a particular issue. Fundamentally in this study, files are classified either as malicious or benign based on specific defining characteristics. The methodology emphasizes active experimentation, assessment, and improvement of classification algorithms to promote accuracy and efficiency in file classification as seen in Figure 34.

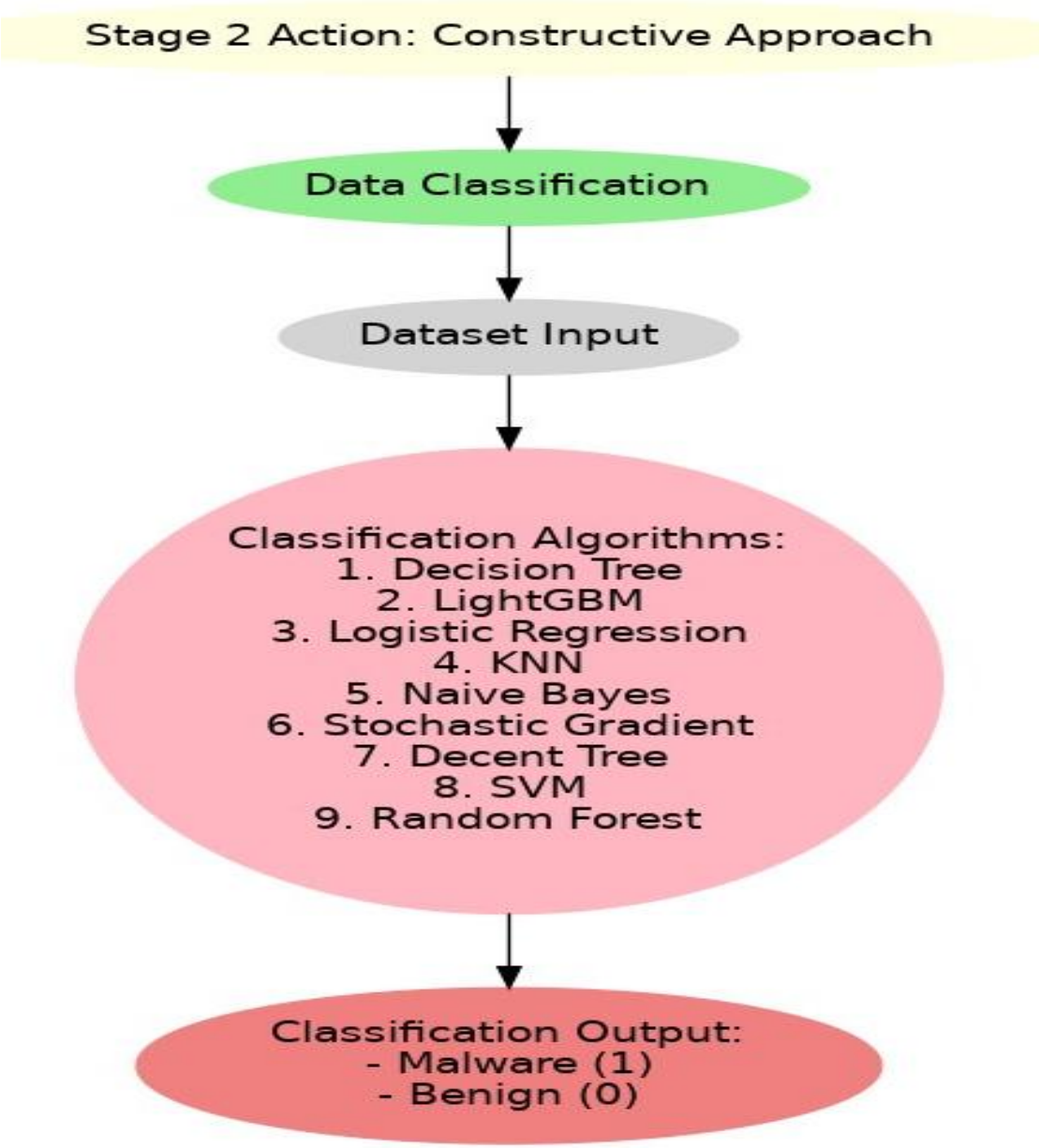


Figure 34: *Stage constructive approach*

Source: Researcher (2024)

A constructive approach as an algorithm primarily uses the prepared input dataset from stage 1 of intelligence gathering and processing. The process includes the preparation of the dataset, its training and testing, and the evaluation of the model output. The model is constructed with multiple classification algorithms like LightGBM, Logistic regression, Linear regression with Naive Bayes, SGD, SVM,

Random Forest, Decision tree, KNN, and XGBoost. After assessing the performance of all algorithms, the three best-performing models from various algorithms were selected and used to automate the research experiment, aiming for the anticipated outcome of this study as Figure 34 shows.

3.1.17 Model Development

To develop an automated ML classification system that can identify patterns of behaviour that distinguish between malicious and non- malicious samples, a set of baseline results were created using ML algorithms across a range of classifier types like: Decision tree, LightGBM, Logistic Regression, KNN, Nave Bayes, Stochastic Gradient Decent, SVM, and Random Forest. This was applied to the built model to test which algorithms are best suited by performance in line with the objectives of this research.

Classification algorithms are used to forecast results from a discrete sample space. Anticipating malicious software (Malware) for (1), benign software for (0), or digit output labels (like "Yes" or "No"). Different supervised ML algorithms exhibit varying behaviours and employ mathematical notations to represent how they classify datasets. How these models – LightGBM, Logistic Regression, Nave Bayes, Stochastic Gradient Decent, SVM, and Random Forest, Decision Tree, KNN, and XGBoost, were applied is explained in Section 3.3.8. Using a combination of mathematical equations and algorithms, classifier algorithms classify data into mathematical models based on learning models and correlations within data as suited for Cybersecurity research. Each classifier algorithm uses a different method in accordance with the underlying mathematical model and optimization methodology. Examples include the following:

As mentioned in [Section 2.17.3], ML Algorithms are applied to classify malware datasets. One of the most crucial functions of CI is the detection of malicious activity and the application of knowledge to detect, prevent, and respond to attacks on business resources. Several research studies, have used the ML for malware detection. For example, Wojtowicz et al, (2017) conducted a preliminary

experiment on 19,991 malicious files and 19,978 benign files examining the usage of entropy analysis in a ML framework. According to the researcher, malware is described as such by one or more Anti-Virus (AVs) on Virus Total. Cohen et al. (2016) examined the structural XML layout of .docx files, using static analysis, where 16,108 benign documents and 830 malware documents were matched.

3.1.18 Steps in Developing Models

To create a classification model, relevant data must first be gathered. The data should adequately sample the problem being addressed and should represent the features needed for effectiveness of the process. Data was collected from various sources and repositories like VirusShare and Kaggle as mentioned in [Section 3.2.1]. The process of constructing the model moves to the next steps as seen in Figure 35 once data has been gathered and is suitable for usage.

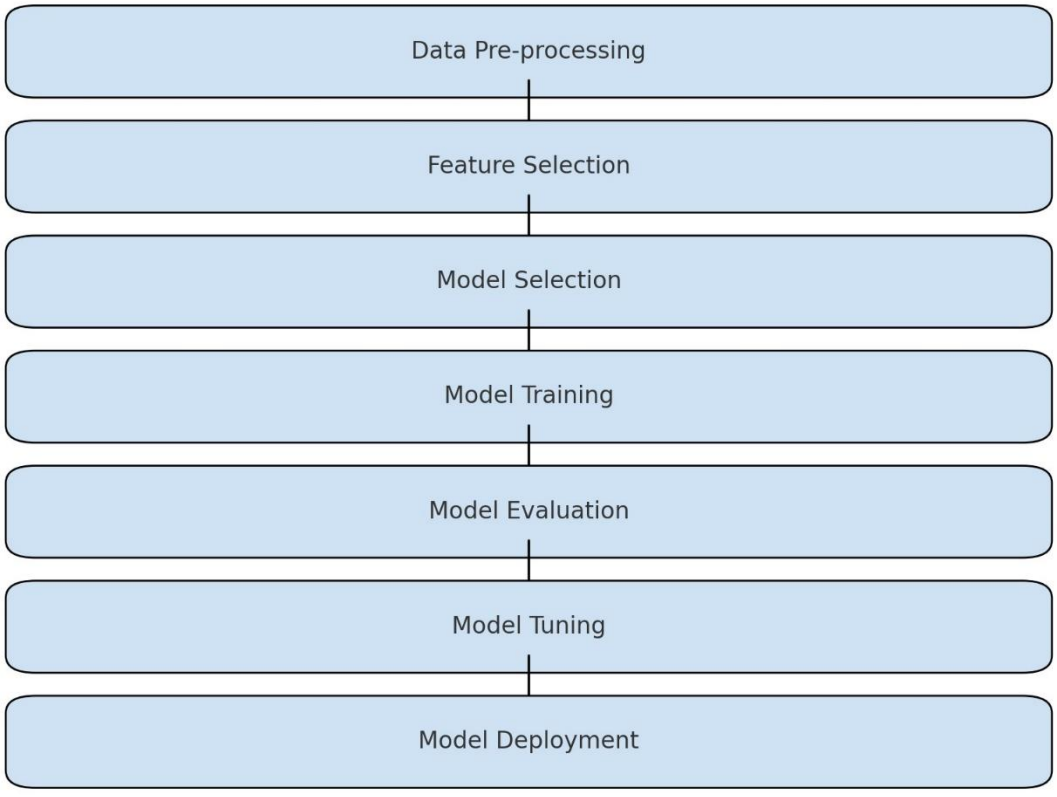


Figure 35: *Steps of model development*

Source: Researcher (2024)

Data pre-processing: Following data collection, pre-processing is required to get the data ready for modelling. This process entails, among other things, cleaning the data, dealing with missing values, and transforming category data into numerical forms.

Feature selection: This step is to choose the attributes that will be used to train the model and select which are the most pertinent. This process helps to make the data less dimensional and increases the model's precision.

Model Selection: Following the selection of the features, this phase is to select the classification algorithm that is most relevant for this research. There are different algorithms that this study applied and they include XGBoost, LightGBM, Decision trees, KNN, Random Forest, Logistic regression, SVM, Naïve Bayes, SGD and Linear regression. The gathered data will be used to train these algorithms and the top three that performs the best will be chosen as the final algorithm for the model.

Model Training: After the model has been chosen, the pre-processed data will be used to train the model. To do this, the data must be divided into training and testing sets, with the model then being fitted to the training set.

Model Evaluation: The effectiveness of the trained model must be assessed after training. To do this, the model must be tested using test data, and metrics like accuracy, precision, recall, and F1-score. The computation accounts for their reliability.

Model Tuning: This becomes a necessary process if the efficiency of the model is unsatisfactory. Tuning can be done by modifying the algorithm or revising the parameters of the model. This process enhances the model's overall reliability.

Model Deployment: The model can be used to make predictions on new data once it has been trained and tested thus being reliable for threat detection.

3.1.19 Dataset Preparation

This study uses two secondary datasets; both datasets consist of 50,000 benign and 50,000 malware signatures. As mentioned in the [Section 3.2.1] on intelligence gathering, these datasets were freely downloaded for this study via VirusShare. The

dataset 1 (D1) had a dimension of 100,000 records, with 8 columns (100000, 8), a size of 6.48 MB, and dataset 2 (D2) was downloaded from Kaggle, and had a dimension of 100,000 records with 35 attributes/columns and size of 17.2 MB.

The malware files signature on D1 contained signatures of various types, such as adware, Trojan Downloader, Trojan StartPage. ZRC, generic malware, and Trojans. Table 6 provides information on the number of columns in the D1 and the descriptions of each data column.

Table 6 show the dataset' description. Dataset 1 was reconstructed by obtaining file signatures only from VirusShare and incorporating the file signature attributes. Instructions for modifying D1; a) Acquire malware file signatures from VirusShare; b) Select 50,000 file signatures from the downloaded collection; c) Obtain file signature characteristics by manually verifying them on VirusTotal; d) Construct dataset 1 by incorporating 50,000 extracted malware signatures and 50,000 benign web links, resulting in 100,000 records with 8 characteristic columns as shown in Table 6 and Figure 36 shows the process of modifying D1.

Dataset 2 (D2) is ready made dataset which is available on Kaggle Research Centre as they created these file signatures for their research. They were made available for free download by Gulsah Yagci in 2020, a researcher at Kaggle Centre, and has more than 8,364 downloads and 60.6K views as seen in Table 7.

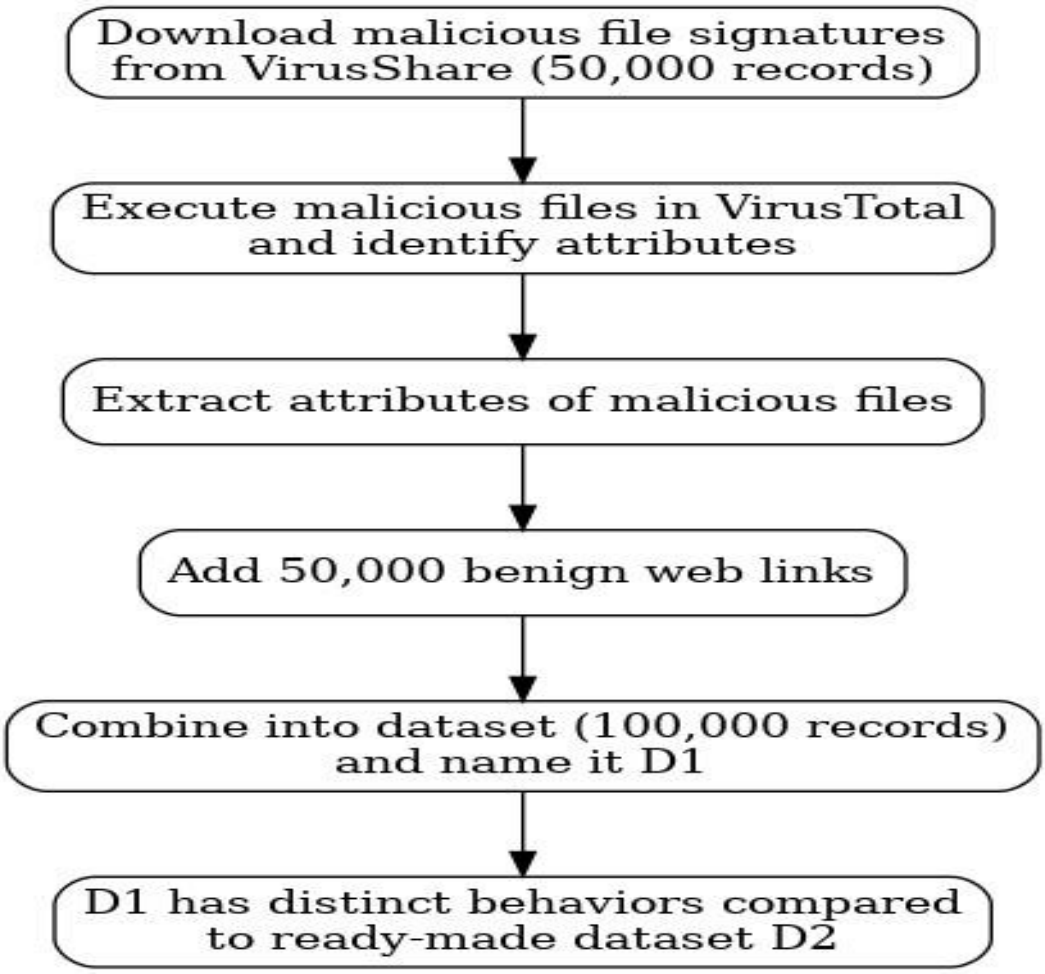


Figure 36 Dataset 1 modification process

Source: Researcher (2024)

Attribute/type	Description	Type
Hash	APK/file Hash	text
Millisecond	Time	number
Classification	Malware or benign	text
State	State of the task (nun-runnable, runnable, or stopped)	Number
virtual_address	memory address that matches the virtual address	number
virtual_size	virtual memory that file needed to run on the system	number
raw_size	malware file size (in kilobytes)	Number
Entropy	Regular "extensive" entropy of a file analogous	Number

Table 6: *Dataset 1 attribute description*

Source: Researcher (2023)

Attribute/type	Description	Type
1. Hash	APK/file Hash	Text
2. Millisecond	Time	
3. Classification	Malware or benign	Text
4. State	State of the task (nun-runnable, runnable, or stopped)	number
5. usage_counter	task structure usage counter	number
6. Prio	Holds the dynamic priority of a task	number
7. static_prio	Static priority of a task	number
8. normal_prio	Normal Priority (without considering the RT inheritance)	number
9. normal_prio	Normal Priority (without considering the RT-inheritance)	number
10. normal_prio	Normal Priority (without considering the RT-inheritance)	number
11. vm_truncate_count	used to mark a vma as now dealt with	number
12. task_size	Current task size	number
13. cached_hole_size	Free address space hole size	number
14. free_area_cache	First address of space hole	number
15. mm_users	Space users	number
16. map_count	Count of memory areas	number
17. hiwater_rss	Peak of resident set size	number
18. total_vm	Total number of pages	number

19. shared_vm	shared pages count	number
20. exec_vm	Workable pages count	number
21. reserved_vm	Reserved pages count	number
22. nr_ptes	Page table entries count	number
23. end_data	End address of code component	number
24. last_interval	Last interval time before thrashing	number
25. Nvcsw	Volunteer context switches count	number
26. Nivcsw	n-volunteer context switches count	Number
27. min_flt	Minor faults (page faults)	Number
28. maj_flt	Minor faults (page faults)	Number
29. fs_excl_counter	Count of file system exclusive resources	Number
30. Lock	A read-write synchronization lock is utilized for accessing the file system.	Number
31. Utime	User time	Number
32. Stime	System time	Number
33. Gtime	Guest time	Number
34. Cgtime	Cumulative group time	Number
35. Signal_nvcsw	Cumulative resource counter	Number

Table 7: *Dataset 2 description*

Source: Adapted from Alomari, et al., (2022)

3.1.20 Dataset Feature Engineering

The scaling of features is a crucial preparation procedure in ML, especially for algorithms that are highly sensitive and dependent on the magnitude of input data.

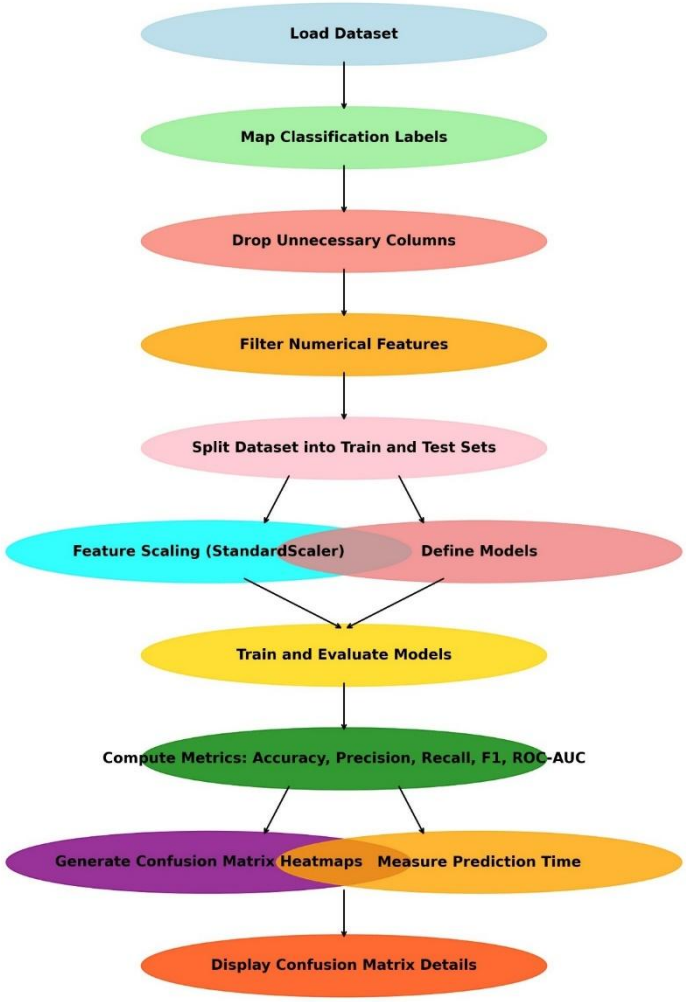
This study selected the following models to train the dataset: LightGBM, Logistic Regression, Linear Regression with Naive Bayes, SGD, SVM, Random Forest Decision Tree, KNN, and XGBoost. Once the datasets were prepared, and programmed, they were fitted with classifiers. This also involves cleaning and pre-treatment of the data. The process includes the import of essential libraries, the loading of datasets into algorithms, the pre-processing of datasets, and the execution of data adjustments. These adjustments include the removal of any missing values, the deletion of some columns, the review of some fields, the investigation of some variables, and the identification of relationships among various variables. The classification process was subjected to different steps such as:

1. Importing Pandas to compare data variables, using Seaborn to create statistical graphics, importing classification models with sklearn, and finally, importing Matplotlib to perform plotting.
2. Loading the dataset into algorithms and processing; and mapping binary classification benign 0 and malware 1, and cleared data by "dropping" classification and ignoring errors.
3. Categorising the data into training and test data after proper pre-processing. A total of 80% of the total data was allocated for training, and the remaining 20% for testing.
4. Evaluating the process for models by selecting datasets (random state = 42) for each model, ensuring consistent training and testing across multiple sets for execution.
5. Evaluate all models' functionalities by evaluating their capacity to forecast the desired outcome.

6. Train and fit the models using an 80% dataset, subsequently generating the models' outputs. Accuracy quantifies overall correctness, precision assesses the quality of positive predictions, and recall reflects sensitivity to positive examples, while the F-score provides a balance between precision and recall, using accuracy, sensitivity and Area Under the Receiver Operating Characteristic Curve (ROC-AUC) score.
7. The confusion metric training assesses the projected values against the actual values of a dataset to measure the efficacy of a classification model.
8. Test confusion metric to summarise the number of true positives, true negatives, false positives, and false negatives allows us to evaluate the testing confusion metrics in the performance of a classification model.
9. Evaluate the performance of each model.

Figure 37: *Steps how algorithms classify malware and benign file*

Source: Researcher (2024)



```
# view summary of dataset
data_011.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 8 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   hash             100000 non-null  object
 1   millisecond      100000 non-null  int64
 2   classification   100000 non-null  object
 3   state            100000 non-null  int64
 4   Virtual_Address  100000 non-null  int64
 5   Virtual_Size     100000 non-null  int64
 6   Raw_Size         100000 non-null  int64
 7   Entropy          100000 non-null  int64
dtypes: int64(6), object(2)
memory usage: 6.1+ MB
```

Figure 38: *Summary of dataset 1*

Source: Researcher (2024)

```
# view summary of dataset
Malware_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 35 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   hash               100000 non-null  object
 1   millisecond        100000 non-null  int64
 2   classification     100000 non-null  object
 3   state              100000 non-null  int64
 4   usage_counter      100000 non-null  int64
 5   prio               100000 non-null  int64
 6   static_prio        100000 non-null  int64
 7   normal_prio        100000 non-null  int64
 8   policy             100000 non-null  int64
 9   vm_pgoff           100000 non-null  int64
 10  vm_truncate_count  100000 non-null  int64
 11  task_size          100000 non-null  int64
 12  cached_hole_size   100000 non-null  int64
 13  free_area_cache    100000 non-null  int64
 14  mm_users           100000 non-null  int64
 15  map_count          100000 non-null  int64
 16  hiwater_rss        100000 non-null  int64
 17  total_vm           100000 non-null  int64
 18  shared_vm          100000 non-null  int64
 19  exec_vm            100000 non-null  int64
 20  reserved_vm        100000 non-null  int64
 21  nr_ptes            100000 non-null  int64
```

Figure 39: *Dataset 2 summary*

Source: Researcher (2024)

Once the dataset has been adequately pre-processed, the next step entails dividing it into training and test data. Training data constitutes 80% of the overall dataset, whereas the remaining 20% is designated for testing purposes. The validation data, separated from the test data, is used to assess a model, usually incorporated for classification objectives. The random_state variable guarantees the reproducibility of the data split. Establishing a stable random_state of 42 as shown in Figure 40, ensures that the data split remains consistent throughout multiple executions of the procedure, which is crucial for experimental reliability and result repeatability. Employing an identical random state value facilitates the comparison of outcomes across various iterations of the model training procedure.

```
# Split the data into 80% training and 20% testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature scaling for models that benefit from scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Figure 40 split the dataset train and test.

Source: Researcher (2024)

Standardisation took place on the features applying the "**StandardScaler**" from sklearn pre-processing. Standardisation entails adjusting features to achieve a mean of 0 and a standard deviation of 1, which is frequently advantageous for models dependent on distance metrics or gradient-based optimization. The StandardScaler is the category from sklearn pre-processing that defines features by eliminating the mean and scaling to unit variance (**standard deviation**). It executes the subsequent change for each feature:

$$X_{-Scaled} = \frac{X - u}{\sigma} \tag{3.1}$$

X Represents the original feature value.

u Represents the mean of the feature.

σ Represents the standard deviation of the feature.

Importing Model Libraries

This research used Python modules for model training and testing datasets as shown in Figure 41, specifically: LightGBM, Logistic regression, Naive

Bayes, SGD, SVM, Random forests, KNN, and XGBoost, except for SVM, which necessitated specialised resources.

```
import pandas as pd
import time
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression, SGDClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from lightgbm import LGBMClassifier
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
```

Figure 41: *Importing model python Libraries*

Source: Researcher (2024)

3.1.21 Defining Models

Functionality engineering is the final stage that occurs just prior to model development. The process of transforming unprocessed data into significant characteristics uncovers valuable insights about the model, enhancing its predictive capacity. During this phase, categorical values are encoded, and other essential modifications are made to the data.

The models chosen during assessment were based on their methodological variety and appropriateness for classification tasks. In defining the model as shown in Figure 42, each model exemplifies a distinct category of learning algorithm, including:

i. Decision Trees and Random Forests both comprehensible and sufficiently adaptable to identify intricate patterns.
ii. Linear models, including Logistic Regression and SGD are appropriate if the connection among features is linear or nearly linear.
iii. Instance-based Models, such as KNN, forecast outcomes based on data points.
iv. Bayesian models, such as Naive Bayes, demonstrate efficacy with minimal data and particular presumptions.
v. Ensemble methods, like Random Forest, LightGBM, and XGBoost, integrate many models to enhance forecast accuracy and mitigate over fitting.

```python
# Define the models to evaluate
models = {
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "Logistic Regression": LogisticRegression(random_state=42),
    "K-Nearest Neighbors": KNeighborsClassifier(),
    "Naive Bayes": GaussianNB(),
    "Stochastic Gradient Descent": SGDClassifier(loss='log_loss', random_state=42),
    "Random Forest": RandomForestClassifier(n_estimators=100, random_state=42),
    "LightGBM": LGBMClassifier(n_estimators=100, random_state=42),
    "XGBoost": XGBClassifier(use_label_encoder=False, eval_metric='logloss', n_estimators=100, random_state=42)
}
```

Figure 42: *Defined models*

Source: Researcher (2024)

The process of assessing several machine algorithms guarantees outcome robustness and offers understanding and insight into the optimal algorithm based on diverse measurement criteria. The subsequent section of the study examined the training and assessment of the models. This entails fitting each model to the training data, adjusting parameters as necessary, and evaluating the performance of all algorithms.

3.1.22 Training Datasets

Nine different algorithms were trained as part of the previously described development process, and then three algorithms that performed the best were selected. The chosen algorithms were used in the research and experimentation of the model. This study acquired the dataset through classification algorithms, as illustrated in Figure 53, and analysed for the output. Figure 53 also illustrates the use of supervised machine methods in this model development procedure. After collecting and parsing the dataset into a readable format, the system saves it on an intelligence database. Additionally, it generates a comma-separated values (csv) file with the dataset for model training. The dataset was used to categorise all algorithms and evaluates their performance, enabling selection of the most suitable algorithms for the final model in this research. The datasets are also trained using fellow algorithms: LightGBM, Logistic Regression, Linear Regression with Nave Bayes, Stochastic Gradient Decent, SVM, Random Forest, Decision Tree, KNN, and XGBoost. The best algorithms for this study is analysed in more detail later in this chapter.

For this research, nine different classification algorithms were used to train and test the two datasets illustrated in Figure 43 and Figure 44. Each algorithm possesses unique characteristics in its ability to classify binary-labelled datasets.

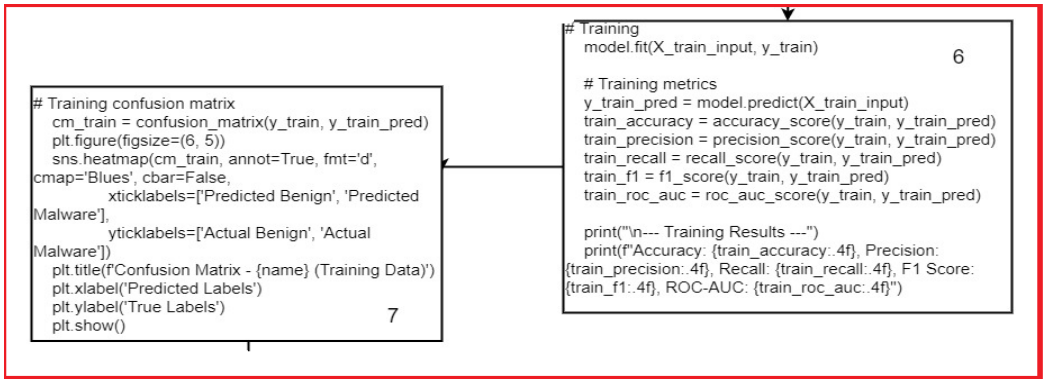


Figure 43: *Model training*

Source: Researcher (2024)

a) The model has been trained on the input data (X_train_input) and the associated target labels (y_train) employing the fit () method.

b) Upon completion of the model training, it was employed to predict the target labels for the training dataset (X_train_input) and the predictions are contained in y_train_pred.

This script trains a model, assesses its performance through multiple measures, and presents the findings in a comprehensible way. The essential phases comprise of: Model training, Forecasting on the training dataset. Assessing performance through measurements such as accuracy, precision, recall, F1 score, and ROC-AUC and visualising the confusion matrix to enhance comprehension of classification performance. The algorithms applied malware classifications thus:

i. Input attributes (*X*): While classifying, input attributes *X* correspond to properties from file signature such as bytes, sequences, opcode patterns entropy file size.

ii. Output class (y): The product (y) is the binary label where:

Y=0 product benign file.

Y=1 product malware file.

Those mathematical calculations are very important for classifying malware for the following reasons:

i. Each ML classification algorithm analyses input attributes uniquely, although all strive to understand the correlation between these attributes and the output class (benign vs. malware).
ii. Certain algorithms, such as logistic regression, SVM, and KNN, necessitate "feature scaling" due to their dependence on distances or linear combinations of input variables.
iii. Most algorithms based on three components, including Decision Tree, Random Forest, XGBoost, and LightGBM, do not necessitate feature scaling, rather, they use categorical features or complex feature interactions, particularly evident in malware classifications.

Once the model libraries are imported, it is necessary to define the model by categorising the data through the application of mathematical computations performed by classifiers. In addition, we utilise the "fit" function to incorporate the datasets (x_train and y_train) into the model. Next, we establish the value of (y-pred) and display the accuracy of the model's forecast. The function of the model described below is solely dependent on a background computation in mathematics.

1. Logistic Regression

ML algorithms typically employ logistic regression to classify file signatures as either benign (0) or malware (1). The algorithm predicts the possibility that the given data (the characteristics of the file signatures) fits into one of the classes. The result of the logistic regression is between 0 and 1, which represents the possibility that the result belongs to class 1 (malware).

Notation: $$P(y = 1|X)\frac{1}{1+e-w^{T}x+b} \quad (3.2)$$

Where:

$P(y = 1|x)$ is the possibility that file signatures are classified as malware (1).

$w^{T}X$ this denotes the linear sequence of the input data characteristics, where:

X is the attributes vector (e.g. file size, entropy etc.).

w the weight vector learnt during the training dataset to assign the importance of each feature.

b represents the bias term.

After that, the entire expression wTX+b (logit) is passed through the sigmoid function, $\sigma - (z) = \frac{1}{1+e^{-z}}$ which transforms the linear output into probability value of 0 benign and 1 malware.

How the model will predict the dataset

The linear combination $w^T X+b$ as the model produces the raw score, referred to as the logit, based on the features of the input data. This logit indicates the strength of evidence that the input data pertains to the malware category.

The sigmoid functionality transforms this logit into probability between 0 and 1.

i. IF *P(y=1|X)* is close to 1 the algorithm is sure that file is malware.
ii. IF *P(y=1|X)* is close to 0 the algorithm is sure that the file is benign.

The decision tolerance is on average set 0.5 and decision of classification based on the threshold.

i. IF *P(y=1|X)* ≥ 0.5, the file signature is classified malware.
ii. IF *P(y=1|X)* < 0.5, the file signature is classified benign.

Furthermore, the accuracy of F-Secure, along with recall, precision, and ROC-AUC, when used in training algorithms on an extensive dataset, yields model performance.

2. Naive Bayes

Naive Bayes utilises the Bayes' Theorem, which is an approach of probability based on beliefs in light of new evidence. In conjunction with the Naive Bayes classifier is used to determine whether a file is malware or benign, considering attributes such as entropy, file size, and bytes sequences.

During this training and testing, classification of malware or benign is done using the following method:

$$P(y|x) = \frac{P(y)\Pi_{i=1}^{N} p(x_i|y)}{P(x)} \quad (3.3)$$

$P(y|x)$: Posterior probability of class y (malware or benign) given the attributes$X = (x_1, x_{2,} \dots x_n)$.

$P(y)$: Prior probabilty of a class y (i.e., the likelihood that a given file signature is malware or benign based overall dataset).

$p(x_i|y)$ The likelihood of attributes x_i given class y (the possibility of witnessing those attributes based on file signatures being classified as malware or benign).

$P(x)$ Represents evidence that encompasses the overall probability of attributes and can be disregarded when comparing classes.

$\Pi_{i=1}^{N}$ Represents likelihoods to be multiplied across all features.

3. Steps in Malware and Benign Files Classification

i. Calculate Prior probability (y) : The calculation is derived from the dataset, such as the proportion of malware files.

ii. Calculate Likelihood $p(x_i|y)$: Naive Bayes calculates the probability of a characteristic based on whether the file is classified as malware or benign. The likelihoods are multiplied among all features $\Pi_{i=1}^{N} p(x_i|y)$to calculate the overall likelihood for each class.

iii. Calculate Prior probability P(y|x): The classifier calculates the prior probability for both classes y = 1 (malware) and y = 0 (benign). It evaluates these possibilities to reach the ultimate choice.

The algorithm used next rule for classification of malware and benign.

a) if $P(y = 1 \mid X) > P(y = 0 \mid X)$ The algorithm classifies as malware (1)

b) if $P(y = 1 \mid X) \leq P(y = 0 \mid X)$ The algorithm classifies as benign (0)

4. Stochastic Gradient Descent (SGD)

SGD uses iterative updates of the method using unique datasets, and refines techniques that reduce the loss function. This renders it especially appropriate for classifying large-scale datasets, such as malware classification. SGD uses the mathematical concepts below for the classification of malware and benign file signatures.

$$L(w) = \frac{1}{n} \sum_{i=1}^{N} L(y_i, w^T x_i) \tag{3.4}$$

Where:

$L(w)$ Represents the average loss for algorithm assigned weight vector w.

n: It represents the number of training data samples.

y_i: It represents the true label class for data sample i (0 for benign, 1 for malware).

$w^T x_i$ It represents the algorithm prediction for data sample i based on true sequence of weighted w and attributesx_i.

$L(y_i, w^T x_i)$ it represents the loss function for each individual data sample i which measures the error relating the true label y_i" and prediction label $w^T x_i$.

When SGD classifies binary use the same way classifying malware and benign files as logistic regression and Naïve bayes.

i. For each dataset (file), calculate the anticipated label whether is benign or malware using x_i as $w^T x_i$.

$$\hat{y}_i = \sigma(w^T x_i) = \frac{1}{1+e-(w^T x_i)} \quad (3.5)$$

ii. Calculate the **loss** using los function $L(y_i, w^T x_i)$ to predict differentiation between true label y_i and the projected $w^T x_i$.

iii. Decision of classification

If $\hat{y}_i \geq 0.5$ of the file singnature will be classfied as malware (1).

If $\hat{y}_i \leq 0.5$ of the file signatures will be classfied as benign (0).

5. Support Vector Machine

In order to classify malware 1 and benign 0 using SVM, the algorithm attempts to identify a "hyperplane" that distinguishes between malware and benign file signatures. This is achieved by measuring the distance between the closest points of both malware and benign file signatures.

$$f(X) = w^T x + b \quad (3.6)$$

Where:

$f(X)$ Represents the decision outcome value for input features X. as this value decides separation points of both malware and benign files X

w The weight vector symbolises the orientation of the hyperplane.

X Denotes a characteristic set of the data point (e.g., entropy and byte sequences).

b The bais term that signifies the ability to move the "hyperplane" away from its origin.

During training, SVM classifications used $f(X)$ to decide whether a file signature belongs to malware class 1 or benign 0 class.

For the new file signature with attributes $X\ new$ (e.g. entropy and file size) it calculates $F(X\ new) = w^T x + b$ new.

If $f(X) > 0$: the file signature is classified as Malware (1)

If $f(X) < 0$: the file signature is classified as benign (0)

6. K-Nearest Neighbours (KNN)

KNN classifies a new file signature using the majority class of its nearest neighbours in the attributes in a situation (where malware = 1 and benign = 0). KNN then decides the label of the file signature by assessing the classes of the ***K***-closest signatures in the training dataset. [Section 2.18.3] mentions the equation formula for malware and benign classification.

Classification happens as follows:

Classify x_1 established on the majority class among K-nearest neighbours.

i. If most of the file signatures are label 1, classify malware (1).
ii. If most of the file signatures are label 0, classify benign (0).

7. Decision Tree

Decision Tree divides the dataset based on characteristics values (e.g. file attributes like entropy and bytes in sequences) to categorise file signatures as malware (1) or benign (0). Section 2.18.3 mentions the decision tree equation formula used for binary classification such as identifying malware (1) and benign (0). Decision Tree uses the equation:

$$Gini = 1 - (P_0^2 + P_1^2) \quad (3.7)$$

Where:

p_0 Represents the proportion of benign file signatures class (0).

p_1 Represents the proportion of malware file signatures class (1).

8. Random Forest

Random Forest uses ensemble learning process to incorporate the prediction of different decision trees.

$$\text{Aggregate}\left(T1, T2_{\dots,} Tn\right) \quad (3.8)$$

$T1, T2_{\dots,} Tn$ representing individual Decision Trees in Random Forest process of classification.

Aggregate function combines all the predictions of the different trees.

For the binary classification process (malware =1, benign = 0).

- Each tree *Ti* outcomes a class prediction of (malware or benign)
- The last classification outcome is made through a majority vote process.

9. XGBoost

XGBoost (Extreme Gradient Boost) is an exceptional ensemble learning technique that constructs a sequence of decision trees to enhance predictive accuracy. It is adept at managing extensive datasets, such as over 100,000-file signature datasets for malware classification, as it can effectively navigate intricate interactions among attributes while retaining efficiency. Section 2.18.4 of this study mentions the equation formula for binary classification.

Classification of malware and benign file signature depends on the threshold:

If projected $\hat{y}i \geq 0{:}5$ the file signature is malware (1).

If projected $\hat{y}i < 0{:}5$ the file signature is benign (0).

10. LightGBM

LightGBM uses gradient boosting to distinguish between malware (1) and benign software (0). This is accomplished by reducing the objective function, which includes a loss term for classification error and a regularisation term to mitigate overfitting. It was developed to attain efficiency and superior performance in the classification of extensive datasets.

The objective function for LightGBM is:

$$obJ = \sum_{i=1}^{n} L\left(y_j \cdot \hat{y}_i\right) + \Omega(T) \tag{3.9}$$

This formula consists of two elements "loss function" $L\left(y_i, \widehat{\hat{y}_i}\right)$ and "regularization term" $\Omega(T)$.

1. Loss function$L\left(y_i, \widehat{\hat{y}_i}\right)$.

This loss function quantifies the disparity between the "true" y_i" label and the anticipated likelihood of $\hat{y}_i$ for each data point i. In binary classification malware is represented as 1 and benign as 0.

$$L\left(y_i, \widehat{\hat{y}_i}\right) = -[y_i \log(\hat{y}i) + (1 - y_i)(1, -\hat{y}_i) \tag{3.10}$$

a) y_i true label for each data point i.

- y_i = 1: the file signature is close is malware.
- $y_i = 0$: the file signature is benign.

b) $\hat{y}_i$ the projected likelihood y_i = 1 malware data point i, the calculate the algorithm used "sigmoid function" which maps data value 0 and 1.

$$\hat{y} = \frac{1}{1+e^{-z_i}} \tag{3.11}$$

z_i The outcome of the decision trees for the data point i characterizing the "raw score is converted into likelihood.

2. Regularisation Term ($\Omega(T)$)

The regularisation term $\Omega(T)$ disciplines the complications of the decision trees T to prevent overfitting. It ensures that the algorithm effectively simplifies previously unseen file signatures by limiting the size and density of the trees.

$$\Omega(T) = rT + \frac{1}{2}\lambda \sum_{i=1}^{T} w_j^2 \tag{3.12}$$

T represents number of leaves in the decision tree, as is divided.

r represents the regularisation parameter that penalising the number of leaves, that the algorithm will divide during classification to prevent overfitting.

λ represents L2 regularisation coefficient that controls the weight of the leaf to prevent algorithm divides large leaves.

w_j represents the weight of leaf j to influence the final expectation predicted.

3.1.23 Models Evaluation

The objective of this stage is to assess the efficacy of each trained ML model on novel test data applying several evaluation metrics and to quantify the duration required for the model to provide predictions on the set of test data. Following the training of the model on the training dataset, it is essential to assess the model's performance on the test dataset. The test dataset comprises data previously not-encountered by the model, facilitating the evaluation of the model's generalisation to new, unseen data.

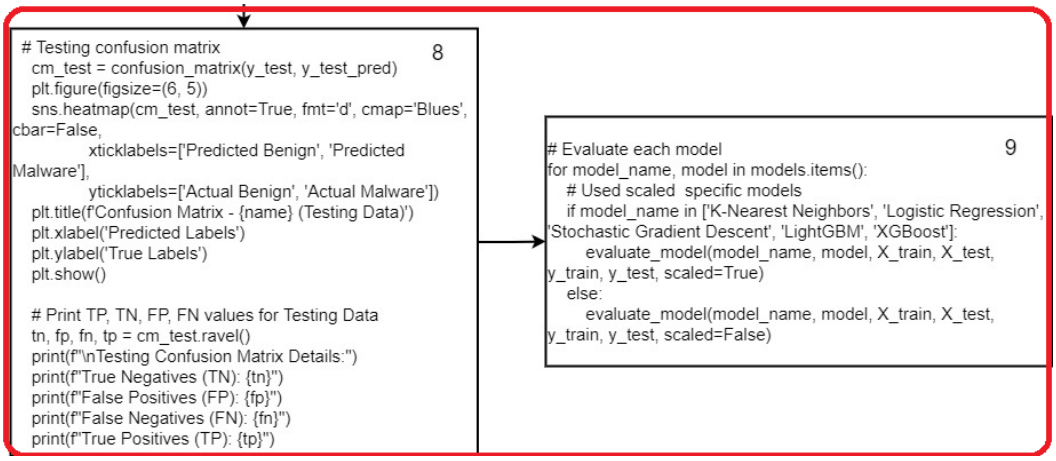


Figure 44: *Model Evaluation*

Source: Researcher (2024)

start_time = **time.time():** This option records the current time (in seconds) at the commencement of the prediction process.

y_test_pred = **model.predict** (**X_test_scaled):** This line produces predictions for the test dataset (X_test_scaled) using the model that was trained.

end_time = **time.time():** Following the completion of predictions, the current time is then recorded, signifying the moment when the model has finalised its assumptions.

prediction_time = **end_time** - **start_time:** The overall duration of the algorithm to generate predictions on the test data is determined by deducting the start time from the finish time. Metrics, including accuracy, precision, recall, F1 score, and ROC-AUC will be used to assess the model's performance.

In information security, the precise classification of files as either malware (1) or benign (0) is essential for safeguarding systems and data against harmful attacks. Assessing the accuracy of classification models developed for this purpose is crucial to guarantee reliability and consistency in implementation. An effective assessing approach not just finds the optimal model but also reveals potential vulnerabilities, such as false positives or false negatives, which may endanger security. In malware classification, we apply performance indicators like accuracy, precision, recall, F1 score, and ROC-AUC.

Every metric offers individual viewpoints on:

Accuracy

Accuracy is frequently used in ML to evaluate a predictive model's effectiveness. The ratio of accurately predicted values (true positives and true negatives) to all cases (true positives, true negatives, false positives, and false negatives) in a dataset is known as the p-value.

$$\text{Accuracy} = \frac{TP + TN}{TP+TN +FP+FN} \quad (3.13)$$

TP = (True Positive) correctly predicted malware file signatures as malware.

TN = (True Negative) The number benign file signatures correctly predicted as benign.

FP = (False Positives) Number of benign file signatures incorrectly predicted as malware.

FN = (False Negatives) Number malware file signatures incorrectly predicted as Benign.

Precision.

The fraction of genuine positives, or cases that have been accurately identified as positive, among all the cases that have been classified as positive, is measured statistically as precision. To assess the precision of a model's optimistic predictions, it is frequently applied in ML and data science thus:

$$\text{Precision} = \frac{TP}{TP+FP} \quad (3.14)$$

TP (True Positive) the number of correctly classified malware as malware.

FP (False Positive) the number of benign files incorrectly flagged as malware.

Recall

Recall is a model's capacity to locate each pertinent instance of a target variable within a dataset. The ratio of true positives to the total of true positives and false negatives is used to compute it.

$$\text{Recall} = \frac{TP}{TP+FN} \quad (3.15)$$

TP (True Positive) Number of positive malware signatures that model correctly identifies.

FN (False Negatives) Number of positive malware signatures that model incorrectly identifies.

F1 Score

A statistic for assessing the effectiveness of a classification model is the F1 score. It considers precision and recall, two metrics that are frequently used to rate classification algorithms. Also, the harmonic mean of precision and recall is used to define the F1 score. An increase in the score, which goes from 0 to 1, corresponds to better performance.

$$\text{F1 score} = 2 \times \frac{Precision \text{ X } Recall}{Precision + Recall} \tag{3.16}$$

ROC-AUC

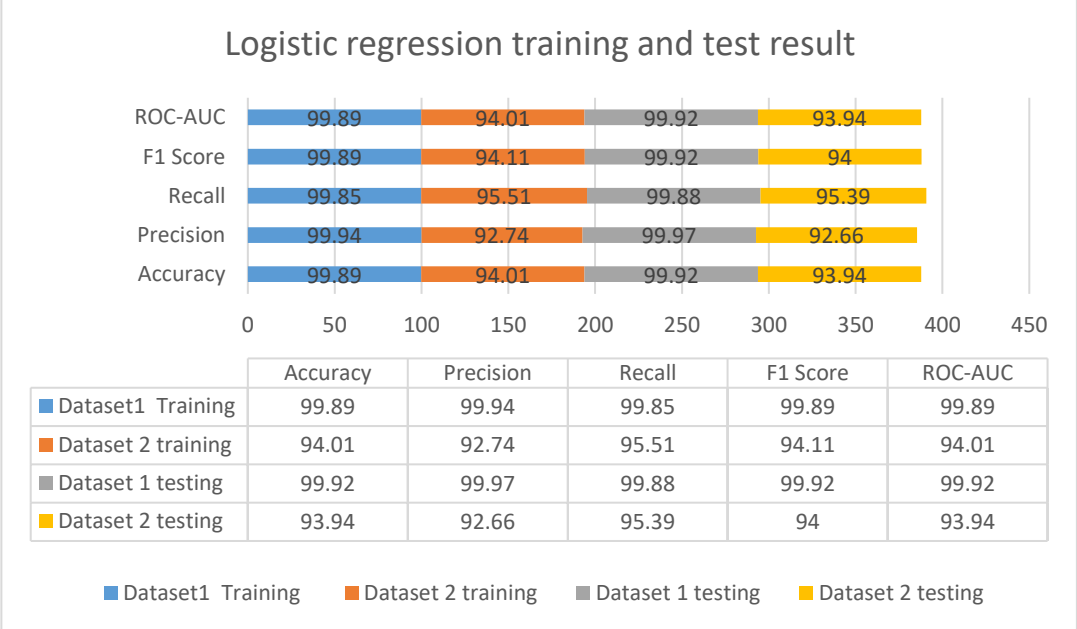
The ROC-AUC (Receiver Operating Characteristic - Area Under the Curve) is a performance indicator employed to assess the classification efficacy of a model, particularly in binary classification scenarios. It offers a singular metric to encapsulate the model's proficiency in differentiating between positive and negative classes across all classification thresholds.

$$\text{TPR} = \frac{TP}{TP + FNTP} \tag{3.17}$$

$$\textbf{FPR} = \frac{TP}{TP + FNTP} \tag{3.18}$$

TPR (True Positive Rate) The ratio of accurately detected true positives.

FPR (False Positive Rate) The ratio of true negatives misclassified as positives.

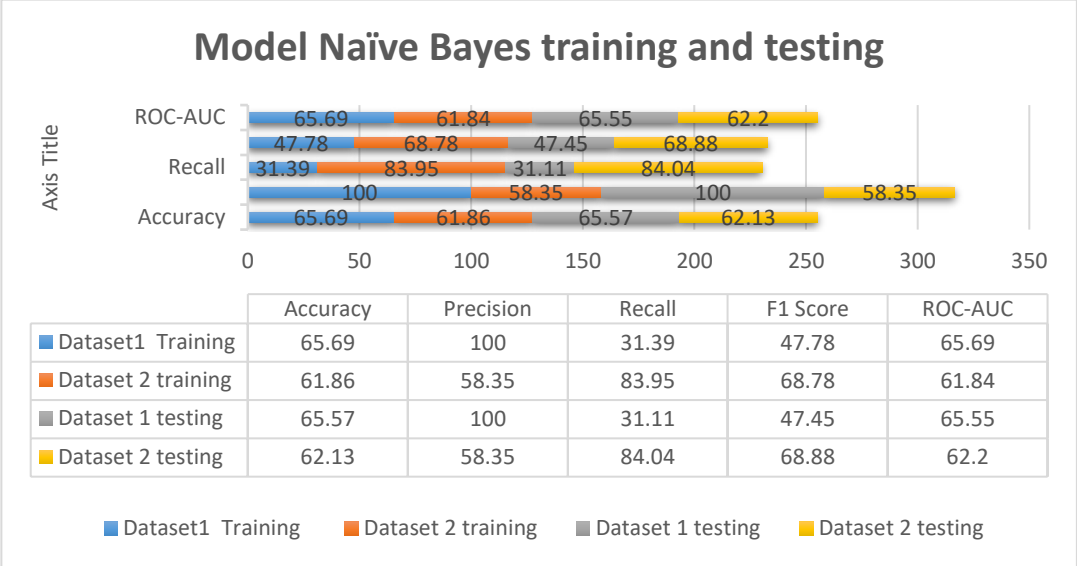


	Accuracy	Precision	Recall	F1 Score	ROC-AUC
Dataset1 Training	99.89	99.94	99.85	99.89	99.89
Dataset 2 training	94.01	92.74	95.51	94.11	94.01
Dataset 1 testing	99.92	99.97	99.88	99.92	99.92
Dataset 2 testing	93.94	92.66	95.39	94	93.94

Table 8: *Logistic regression training and test*

Source: Researcher (2024)

Upon completion of training and testing, the logistic regression model indicates in the Table 8 that the accuracy disparity between dataset 1 and dataset 2 during training is 0.0588. This signifies that the method has suboptimal performance on extensive datasets. Furthermore, the accuracy of F-Secure, along with recall, precision, and ROC-AUC, when utilized in training algorithms on an extensive dataset, yields diminished performance.



	Accuracy	Precision	Recall	F1 Score	ROC-AUC
Dataset1 Training	65.69	100	31.39	47.78	65.69
Dataset 2 training	61.86	58.35	83.95	68.78	61.84
Dataset 1 testing	65.57	100	31.11	47.45	65.55
Dataset 2 testing	62.13	58.35	84.04	68.88	62.2

Table 9: Model Naïve Bayes training and test result

Source: Researcher (2024)

The Naive Bayes algorithm also performs poorly during training and testing on D1 and D2, as Table 9 illustrates. Accuracy: D1 training 65%, D261%. The precision of dataset1 training is 100%, while dataset2 training is 58%. Recall training D1 31%, D2 83%, and f-secure D1 training 47.78%, D2 68.78%, D1 test 47.45%, D2 68.88%, and ROC-AUC D1 training 65.68%, D2 61.84, and test D1 65.55%, D2 62.20. This shows that the Naïve Bayes model has low performance.

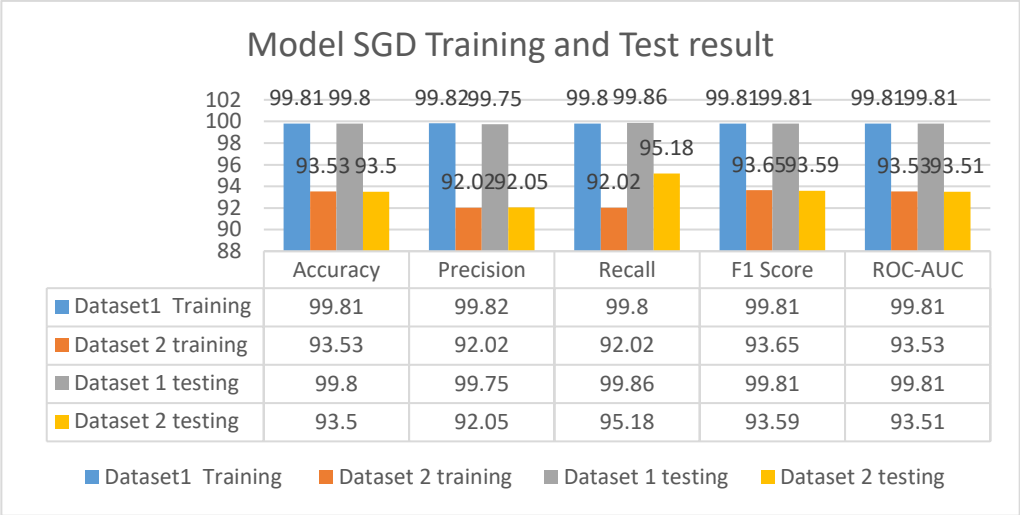


Table 10: Model SGD training and test result

Source: Researcher (2024)

The performance of the SGD model exceeds that of both logistic regression and Naïve Bayes models, while the accuracy disparity is less evident in big D2 training and more significant in D1 training, reaching up to 6.3%. The precision gap between D1 and D2 is 7.8%, signifying a reduced precision score for D2. The recall score for both D1 and D2 training is identical to the precision score. The F1-score and ROC-AUC score are identical: D1 is 99.81% and D2 is 93.53. Upon analysing the testing model with 20% of each dataset, we observe that the performance scores are equal to those of the training in both instances, as demonstrated in Table 10.

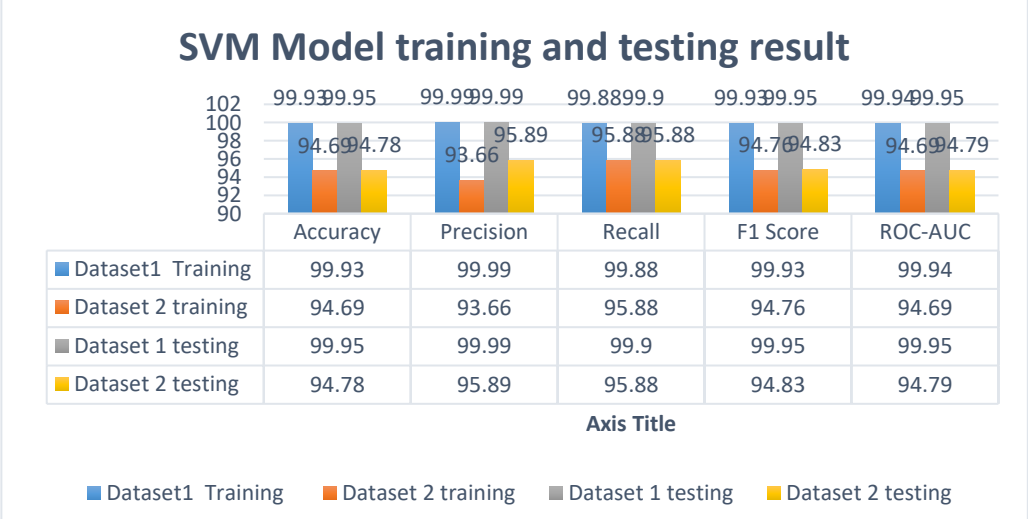


Table 11: Model SVM training and test result

Source: Researcher (2024)

Table 11 demonstrates the varying outcomes of training SVM on both D1 and D2. This demonstrates the discrepancies in the outcomes derived from training the model with datasets of 17.2 MB and 6.48 MB. Utilizing D1, which possesses less features than D 2, yields superior accuracy compared to the performance observed with D2, which contains more features as chapter 3.35 dataset preparation indicates. While the performance of SVM remains consistent and yields satisfactory results, it requires high-performing hardware. SVM classification takes a long time to classify the large D2 (9107.59 milliseconds) and D1 (625.11 milliseconds), and it requires high-performing hardware.

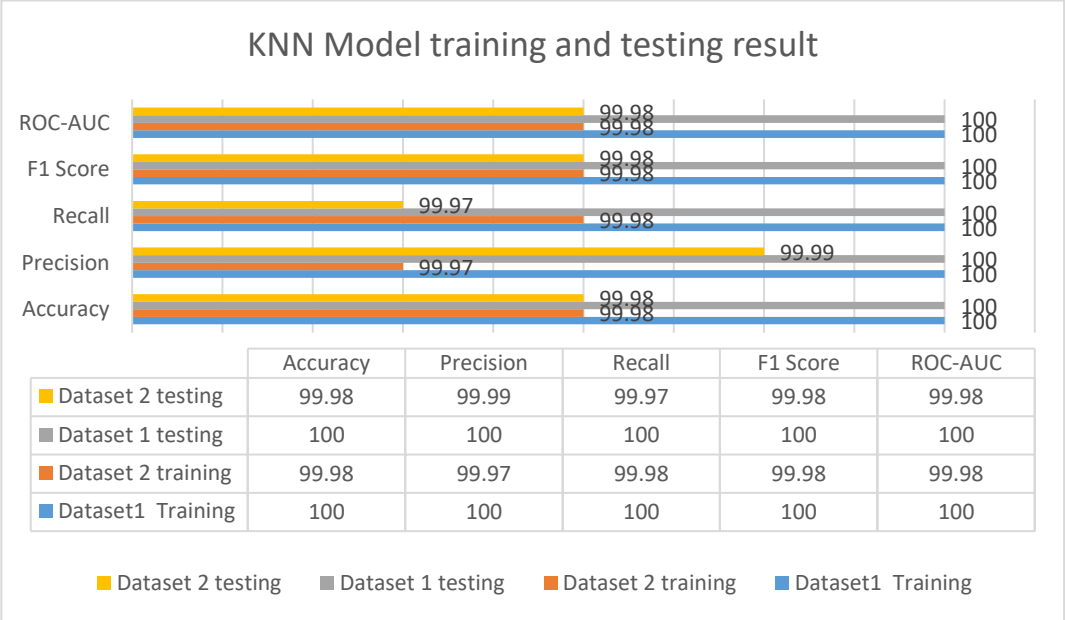
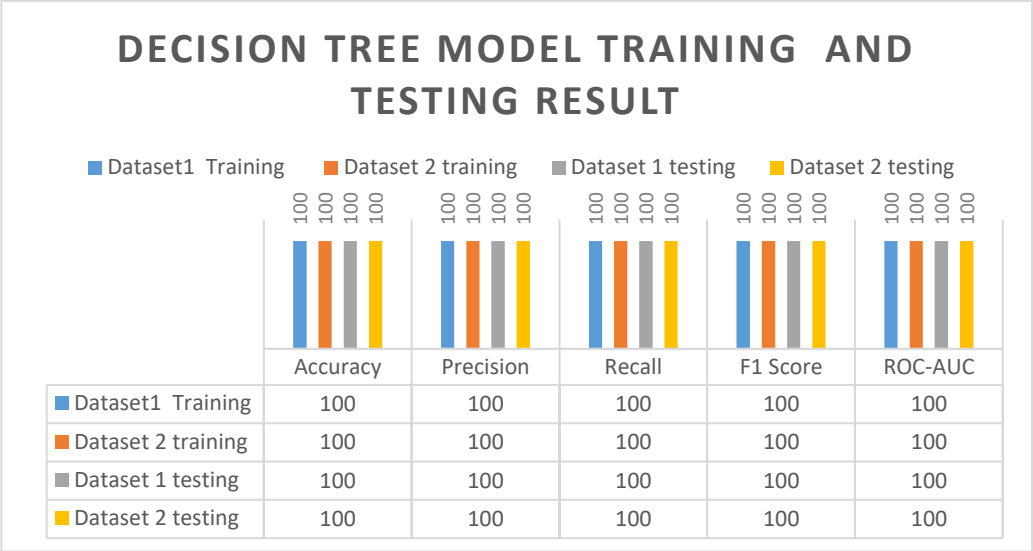


Table 12: Model KNN training and test result

Source: Researcher (2024)

In Table 12, the KNN model outperforms previous algorithms, with a 0.2% accuracy difference between D1 training and D2 testing. Both D1 and D2 perform identically during training. The difference in precision between D1 and D2 during training is 0.3%. Assess D1 and D2 are 0.1%. The recall for the D1 and D2 training and test differences is 0.1%. The F-score and ROC-AUC are identical to the recall. This indicates that the KNN model is performing well on both datasets.

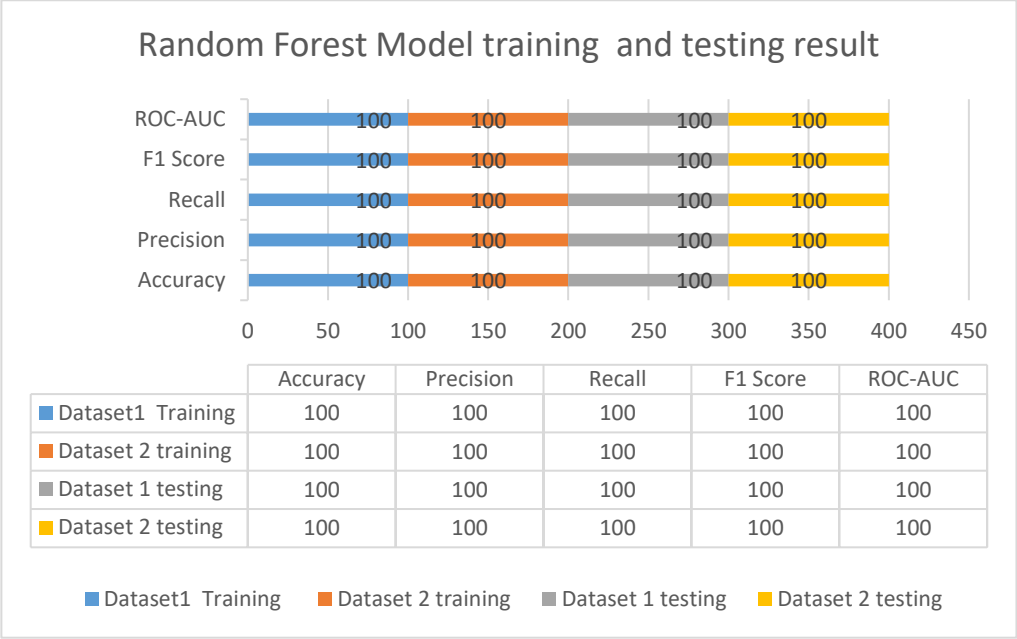


	Accuracy	Precision	Recall	F1 Score	ROC-AUC
Dataset1 Training	100	100	100	100	100
Dataset 2 training	100	100	100	100	100
Dataset 1 testing	100	100	100	100	100
Dataset 2 testing	100	100	100	100	100

Table 13: Model Decision tree training and test result

Source: Researcher (2024)

Upon training and testing, both the D1 and D2 decision tree models attain 100% accuracy, precision, recall, F1-score, and AUC-ROC. Upon comparing previously trained and tested algorithms, we observe that the decision tree model has strong performance in both training and testing across the datasets. Table 13 indicates that the metrics for accuracy, precision, recall, F1 score, and ROC-AUC are all 100%, rendering decision trees reliable algorithms.

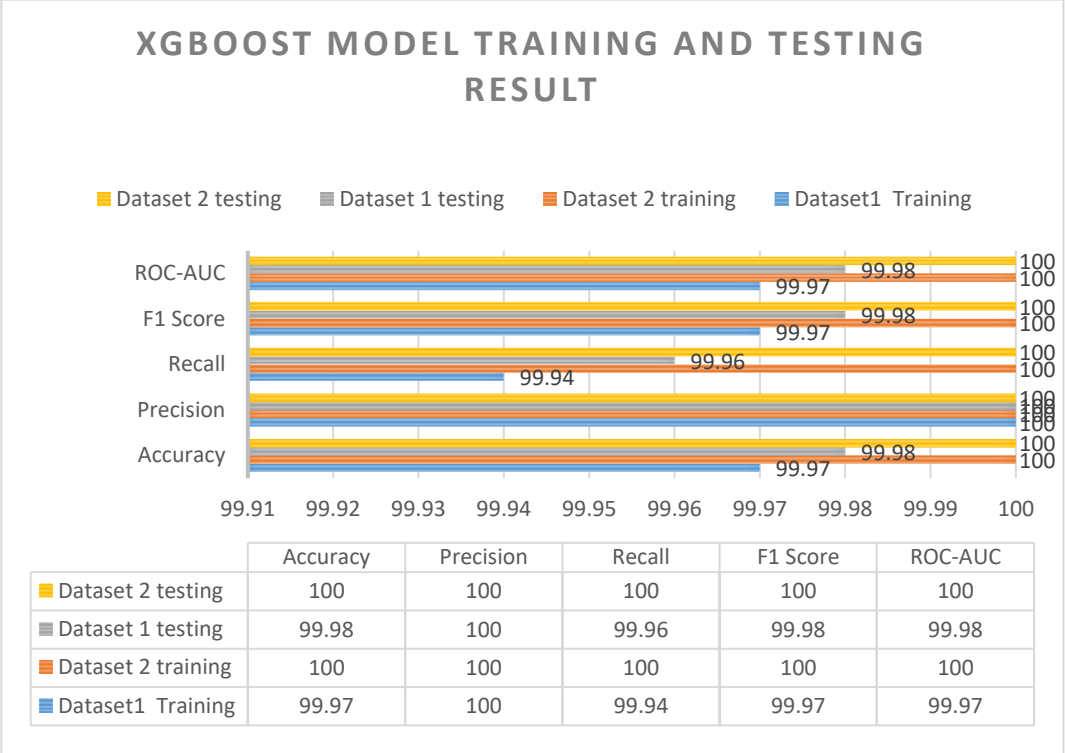


	Accuracy	Precision	Recall	F1 Score	ROC-AUC
Dataset1 Training	100	100	100	100	100
Dataset 2 training	100	100	100	100	100
Dataset 1 testing	100	100	100	100	100
Dataset 2 testing	100	100	100	100	100

Table 14: Model Random Forest training and test result

Source: Researcher (2024)

When we compare the Random Forest model and decision tree models through training and testing, we find that the decision tree model classifies the datasets faster (D1 classification time: 1.03 milliseconds, D2: 5.02 milliseconds) than the Random Forest model (classification time: D1: 62.44 milliseconds, D2: 132.06 milliseconds). Table 14 shows that the scores for accuracy, precision, recall, F1 score, and ROC-AUC are all 100%, which makes decision trees trustworthy algorithms.



	Accuracy	Precision	Recall	F1 Score	ROC-AUC
Dataset 2 testing	100	100	100	100	100
Dataset 1 testing	99.98	100	99.96	99.98	99.98
Dataset 2 training	100	100	100	100	100
Dataset1 Training	99.97	100	99.94	99.97	99.97

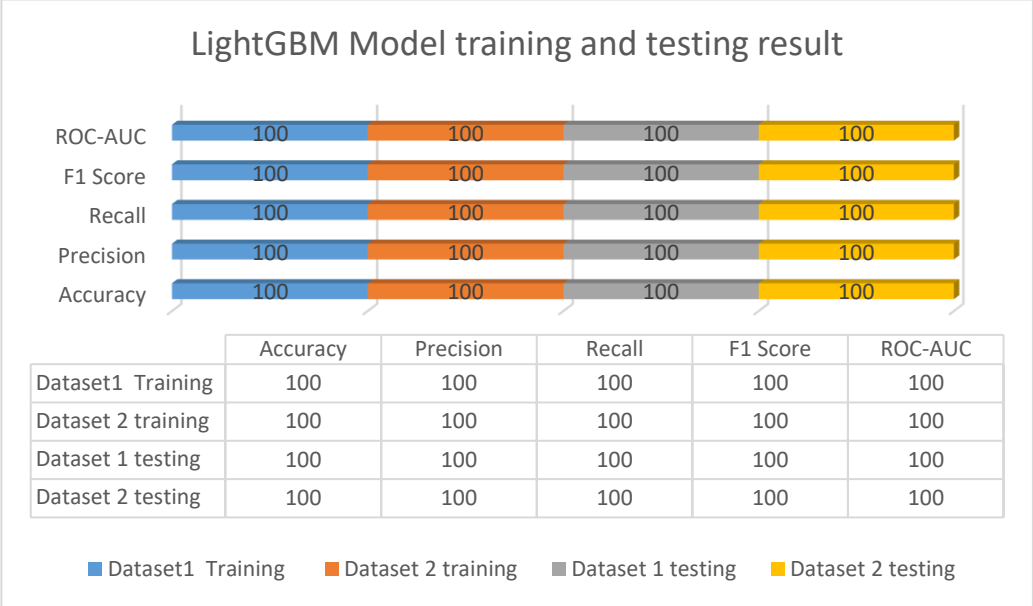
Table 15: Model XGBoost training and test result

Source: Researcher (2024)

XGBoost (eXtreme Gradient Boost) performance is a bit lower than Decision Tree. Table 15 shows that, the performance of XGBoost (Extreme Gradient Boost) is slightly lower than that of decision trees; the difference between training D1 and D2 is 0.3%, while testing D1 and D2 achieves 100% accuracy. The precision scores of the training and test datasets are 100% on both datasets.

Recall score training: D1 99.94%, D2 100%; test: D1 99.96%, D2 100%. F1 score training: D1 99.97%, D2 100%, and testing: D2 100%. ROC-AUC

Training D1 99.97%, D2 100%; test D1 99.98%, D2 100%. This shows that XGBoost (Extreme Gradient Boost) performs better when it comes to classifying large datasets.



	Accuracy	Precision	Recall	F1 Score	ROC-AUC
Dataset1 Training	100	100	100	100	100
Dataset 2 training	100	100	100	100	100
Dataset 1 testing	100	100	100	100	100
Dataset 2 testing	100	100	100	100	100

Table 16: Model LightGBM training and test result

Source: Researcher (2024)

The LightGBM algorithm consistently achieves 100% accuracy across both training and testing datasets. Precision, recall, F1 score, and ROC-AUC are all 100% scores, as Table 16 shows. Two datasets were used in this study, and nine different algorithms were applied as indicated in the preceding table. Using 80% of each dataset, the algorithms revealed varying levels of accuracy. D1's logistic regression score is 99.99, whereas D2's is 94.01; D1's Naïve Bayes score is 65.69, while D2's is 61.86; D1's SGD score is 99.81, in contrast to D2's 93.53; and D1's SVM score is 99.93, compared to D2's 94.69. The algorithms show better results in D1, comprising 100,000 records and 8 characteristics, with a dataset size of 6.48 MB; conversely, performance in D2 is inferior, despite a dataset size of 17.2 MB and 35 attributes.

In addition to training datasets, models were tested using 20% of two datasets to find the three best tree models that were used in the experiment. In Tables 17, Table 18, Table 19 and Table 20 it is consistent that, the algorithm; LightGBM, Decision Trees, and XGBoost are the most effective among the nine algorithms evaluated.

The Decision Tree achieved a performance score of 100% for D1 and D2; LightGBM also scored 100% for D1 and D2; XGBoost records an accuracy score and precision of 100% for both D1 and D2. The recall for D1 is 99.96%, and for D2 it is 100%. The F-score for D1 is 99.98%, and for D2, it is 100%. The ROS-AUC for D1 is 99.98%, and for D2 it is 100%. The logistic regression accuracy score is 99.92 on D1 and D2 and 93.94; precision, recall, and F-score are nearly identical to the training results. KNN, SGD, Naïve Bayes, SVM, and Random Forest yield identical outcomes from the training results.

Assessing Models Performance through K-Fold Cross-Validation

The final stage in assessing model performance involves calculating the average of K performance scores. This offers a more reliable assessment of the model's performance as it has been evaluated on a distinct subset of data. Model performance assessment is necessary to evaluate model performance. K-Fold Cross-Validation is a statistical approach used in ML and data science to assess the effectiveness and generalisability of a predictive model. We will partition the dataset into five equal subsets using the K-fold technique, followed by an evaluation of the model's performance.

The dataset is randomly partitioned into K equal-sized subsets, known as folds. The value of K is a parameter that we select (5-Fold).

Training and validation: The model undergoes training K times. In each instance, a distinct fold serves as the validation subset (the fold whereby the model is evaluated), while the remaining k-1 folds are aggregated to provide the training set.

First Iteration: K-2, 3, 4 and 5 are used for models training and K-1 for testing.

Second Iteration: K-1, 3, 4 and 5 are used for model training and K-2 for testing.

Third Iteration: K-1, 2, 4 and 5 are used for model training and K-3 for testing. The procedure will persist until each model is trained and tested using k-5 folds of the dataset.

After each cycle, the performance metrics of accuracy, precision, and recall are recorded, representing the K distinct performance scores. Furthermore, we will compute the mean and standard deviation for each model.

Mean Accuracy Measurement Formula

The mean accuracy reflects the average of the individual accuracies taken from each fold.

$$\text{Mean Accuracy} = \frac{1}{k} \sum_{i=1}^{k} accuracy\ i \quad \textbf{(}3.19\textbf{)}$$

K denotes number of folds that we used to measure the model's performance.

Accuracy *i*: Accuracy on the $\boldsymbol{ith}$ Fold. In each iteration (or fold) of cross-validation, the model was trained on K−1 samples. The model was trained using K1 folds and subsequently evaluated on the remaining fold. Upon completion of testing, we compute an accuracy score that reflects the model's performance on the test fold.

It was found that the LightGMB, Decision Three, and XGBoost algorithms worked the best and most consistently considering the results of the model performance evaluation by k-5 folds on dataset 1. A dataset that consisted of 100,000 records with 8 columns was used. The performance evaluation automatically divided the dataset into five equal subsets of 20,000. Table 17 shows all the results of each fold.

To calculate standard deviation, we used formula below.

$$\text{Standard Deviation} = \sqrt{\frac{1}{k}\sum_{i-1}^{k}.(Accuracy\ i - Mean\ accuracy)2}$$

(eq.3.20)

K is the number of folds

Accuracy i: Accuracy for the $\boldsymbol{ith}$ folds. This denotes the i-th single instance in the set of data. In other terms, it constitutes a singular data point inside a collection of values.

Mean Accuracy: This represents the mean (or average) of all instances within the dataset, computed accordingly.

$\Sigma_{i=1}^{k}$= This denotes the aggregation of the squared deviations between each unique occurrence and the mean occurrence.

$\frac{1}{k}$: This represents the average of the squared differences, also named **variance**.

√= This square root symbol denotes the mean of squared deviations and will yield the result in the original scale of the data. This indicates the standard deviation, reflecting the extent to which individual folds generally deviate from the mean.

Assessing model performance is essential for classification algorithms since it offers insights into the model's efficacy, highlights areas for enhancement, and guarantees that the model oversimplifies effectively to novel data.

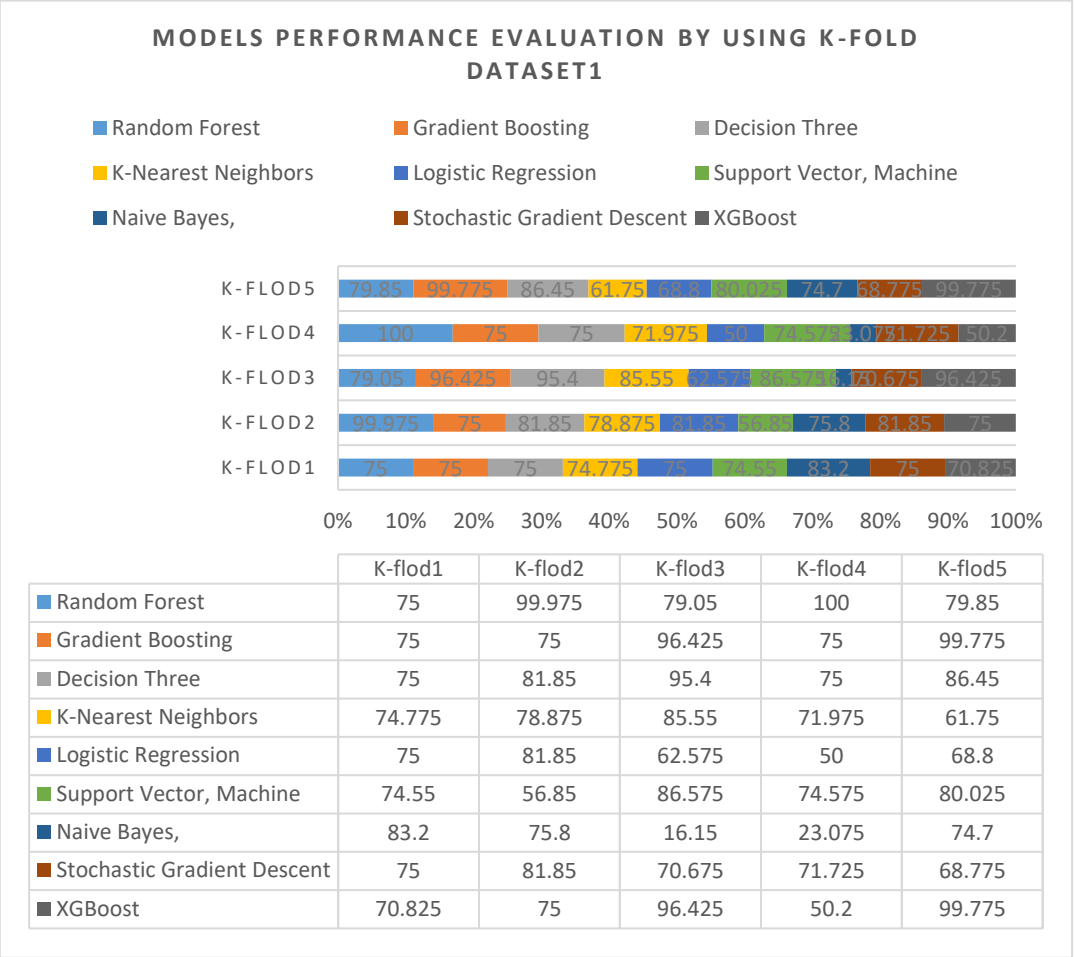
A detailed record of how XGBoost, Decision Tree, and LightGBM were used to implement the final model in an experimental setting, using state machine automation processes is presented in Chapter 5.

3.1.24 Outcome of K-fold Validation

K-Fold Cross-Validation denotes a statistical method employed in ML and data science to evaluate the efficacy and generalisation of a predictive model. Dataset was divided into five equal subsets with the K-fold method. Thereafter, the model's performance was assessed. The datasets were randomly divided into K equal-sized subgroups, called folds. K is a parameter chosen for this study (5-Fold).

The primary objective of this strategy is to eliminate the bias associated with the random division of data into training and test sets. This provided clearer understanding of the model's predictive capabilities regarding the future. The K-Fold Cross-Validation Results demonstrate the model's performance consistency across several data subsets. They exhibit patterns in accuracy, precision, recall, and other significant parameters. Such findings are essential for model selection,

hyper parameter optimization, and evaluating the overall efficacy of ML algorithms. Analytics will see the following in tables 17 and 18.

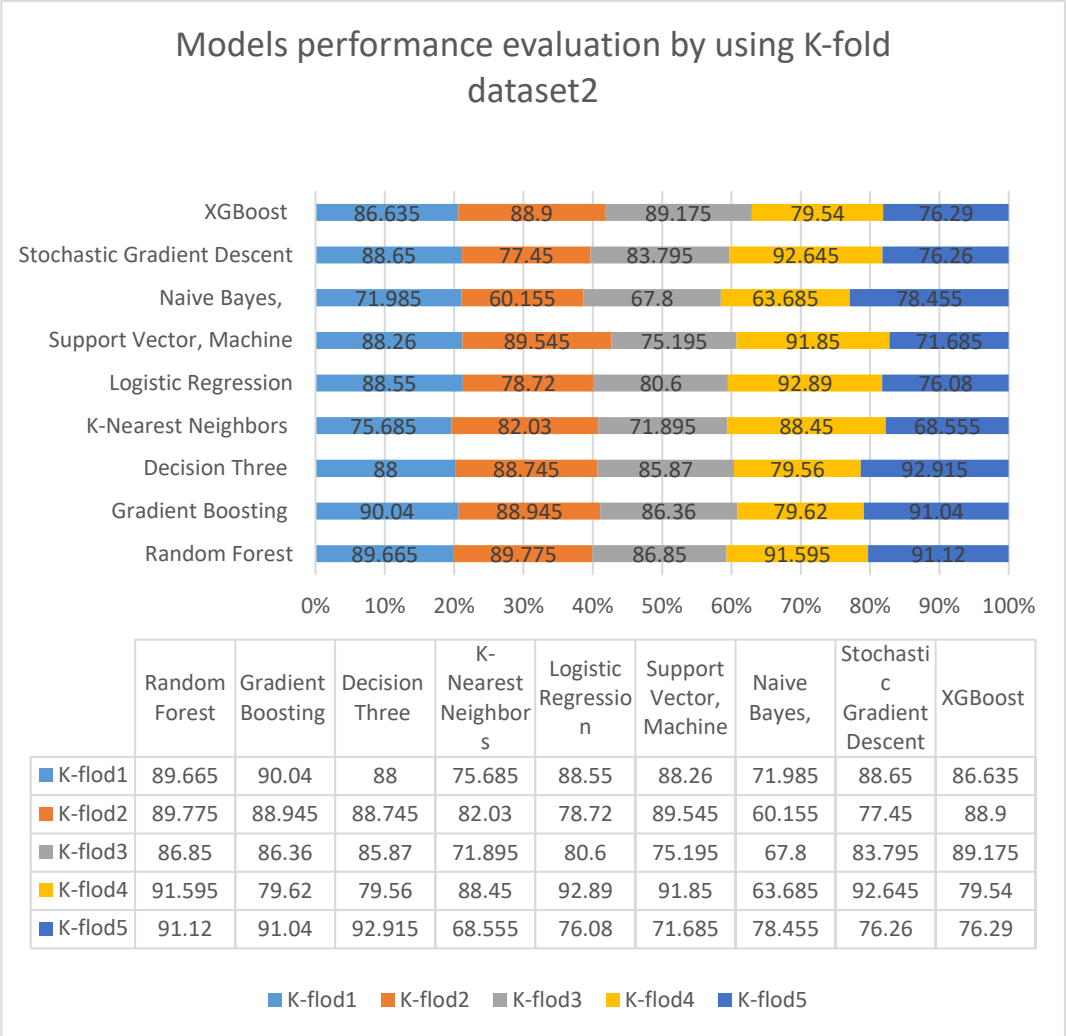


	K-flod1	K-flod2	K-flod3	K-flod4	K-flod5
Random Forest	75	99.975	79.05	100	79.85
Gradient Boosting	75	75	96.425	75	99.775
Decision Three	75	81.85	95.4	75	86.45
K-Nearest Neighbors	74.775	78.875	85.55	71.975	61.75
Logistic Regression	75	81.85	62.575	50	68.8
Support Vector, Machine	74.55	56.85	86.575	74.575	80.025
Naive Bayes,	83.2	75.8	16.15	23.075	74.7
Stochastic Gradient Descent	75	81.85	70.675	71.725	68.775
XGBoost	70.825	75	96.425	50.2	99.775

Table 17: K-fold Result Dataset1

Source: Researcher (2024)

Studying the model performance evaluation by K-5 folds on dataset 1, found that the LightGBM, Decision Tree, and XGBoost algorithms worked the best and most consistently. A dataset consisting of 100,000 records with 8 columns was used. The performance evaluation automatically divided the dataset into five equal subsets of 20,000, as Table 18 shows all the results for each fold.



	Random Forest	Gradient Boosting	Decision Three	K-Nearest Neighbors	Logistic Regression	Support Vector, Machine	Naive Bayes,	Stochastic Gradient Descent	XGBoost
K-flod1	89.665	90.04	88	75.685	88.55	88.26	71.985	88.65	86.635
K-flod2	89.775	88.945	88.745	82.03	78.72	89.545	60.155	77.45	88.9
K-flod3	86.85	86.36	85.87	71.895	80.6	75.195	67.8	83.795	89.175
K-flod4	91.595	79.62	79.56	88.45	92.89	91.85	63.685	92.645	79.54
K-flod5	91.12	91.04	92.915	68.555	76.08	71.685	78.455	76.26	76.29

Table 18: K-fold Result Dataset2

Source: Researcher (2024)

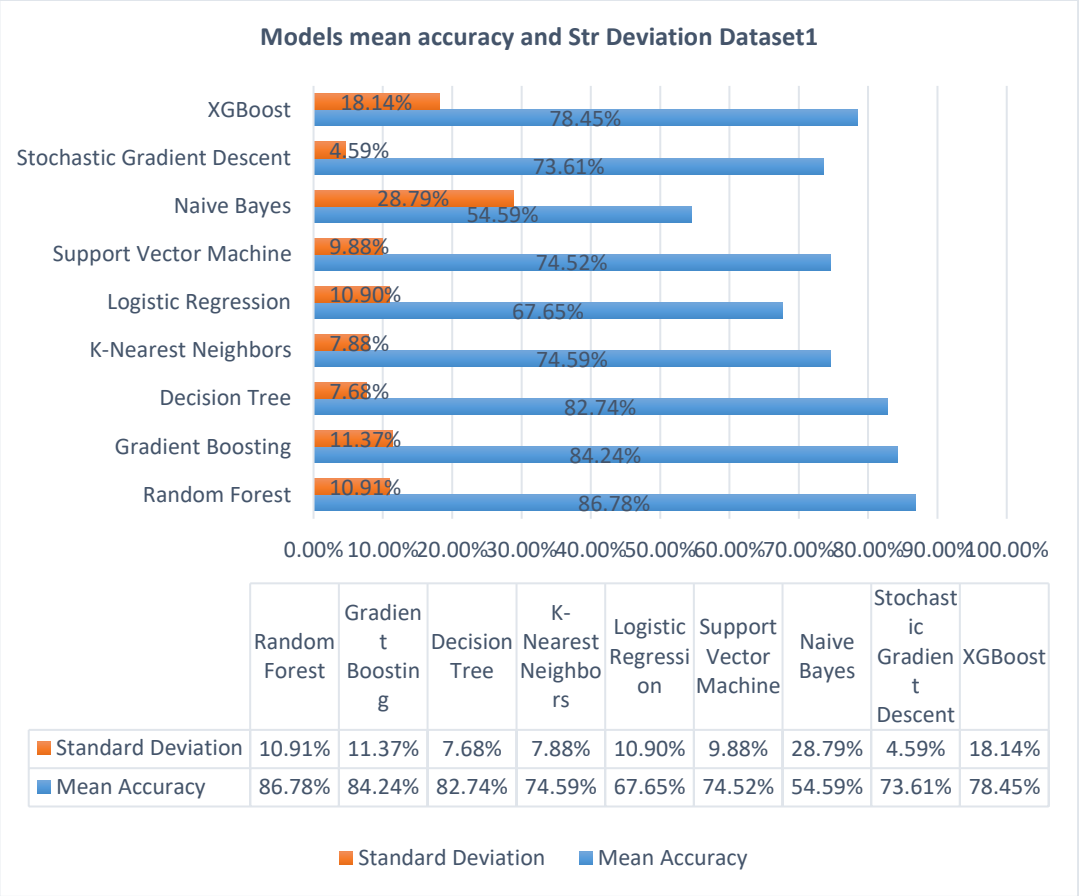
Dataset 2 was used to assess the performance of the models. Dataset 2 contained 100,000 records and thirty-five columns, but dataset 1 had fewer columns. The results in Table 18 demonstrate the consistency of the three methods: SGD boosting, Decision Tree, and XGBoost were the top-performing algorithms. Standard deviation was used to quantify the variation in accuracy scores across several folds. This illustrates the extent to which the accuracy differs from the mean accuracy.

3.1.25 Mean Accuracy and Standard Deviation

The mean accuracy and standard deviation are critical metrics for assessing a model's performance, particularly if employing cross-validation methods such as K-Fold Cross-Validation. These parameters offer insights into

the model's overall efficacy and the uniformity of its predictions across various data subsets.

The mean accuracy represents the common achievements of the model across all K folds throughout the cross-validation procedure. This provides a core metric for assessing the model's overall performance on the dataset.



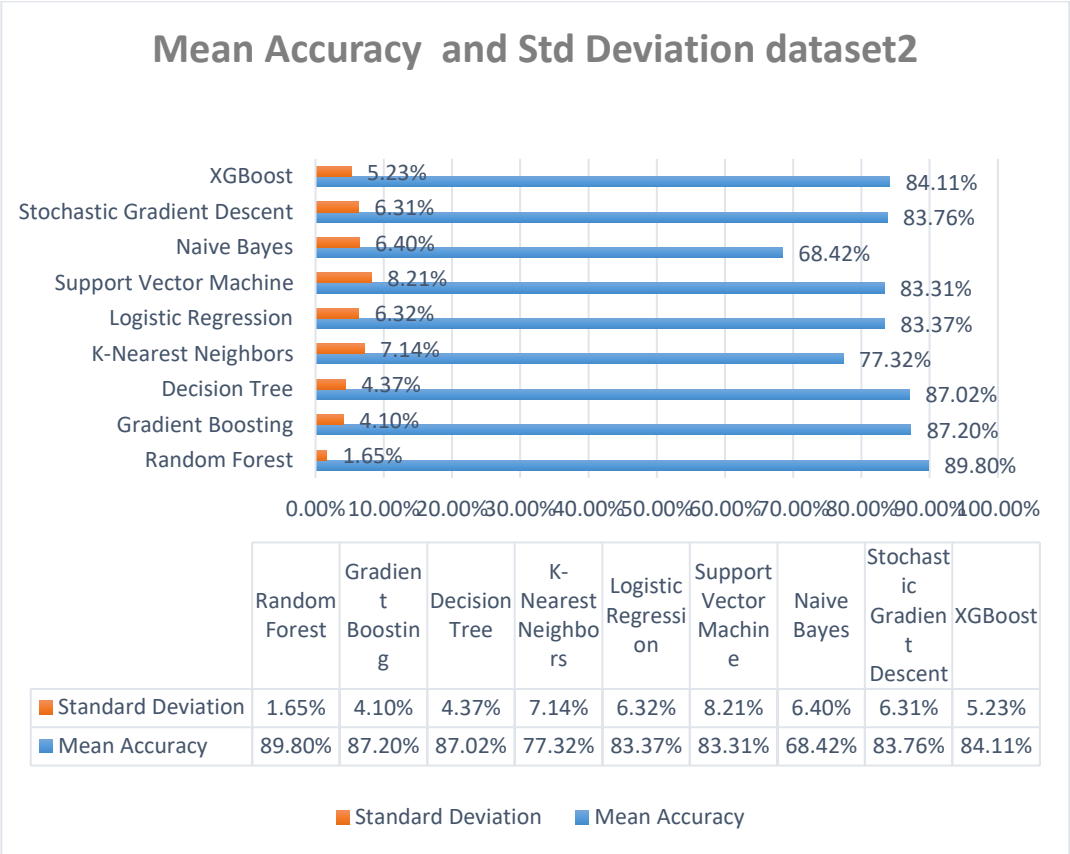
	Random Forest	Gradient Boosting	Decision Tree	K-Nearest Neighbors	Logistic Regression	Support Vector Machine	Naive Bayes	Stochastic Gradient Descent	XGBoost
Standard Deviation	10.91%	11.37%	7.68%	7.88%	10.90%	9.88%	28.79%	4.59%	18.14%
Mean Accuracy	86.78%	84.24%	82.74%	74.59%	67.65%	74.52%	54.59%	73.61%	78.45%

Table 19: Models mean accuracy and Std Deviation Dataset1

Source: Researcher (2024)

Observation: Tree-based models (Decision Tree, Random Forest, LightGBM, XGBoost) attained 100% accuracy on both training and test datasets D1, although displayed significantly- reduced k-fold scores. Random Forest (86.79), Decision Tree (82.74), LightGBM (84.24), and XGBoost D1 (78.45) provide robust classification capabilities yet exhibited overfitting to the dataset. SVM D1 (74.61), KNN (74.59), and SGD (73.61) exhibited comparatively lower k-fold scores, suggesting that techniques might have less overfitting, yet still encounter challenges with generalisation. Logistic Regression (67.65) D1 and

Naïve Bayes D1 (54.59) had the poorest performance, indicating their potential unsuitability for this classification in this research.



	Random Forest	Gradient Boosting	Decision Tree	K-Nearest Neighbors	Logistic Regression	Support Vector Machine	Naive Bayes	Stochastic Gradient Descent	XGBoost
Standard Deviation	1.65%	4.10%	4.37%	7.14%	6.32%	8.21%	6.40%	6.31%	5.23%
Mean Accuracy	89.80%	87.20%	87.02%	77.32%	83.37%	83.31%	68.42%	83.76%	84.11%

Table 20: Mean Accuracy and Standard Deviation Dataset2

Source: Researcher (2024)

Observation D2:

D2 usually outperformed D1, demonstrating superior k-fold scores across most models. XGBoost, Decision Tree, LightGBM, and Random Forest provide superior cross-validation scores in D2. SGD, SVM, and logistic regression in D2, indicated significantly enhanced performance compared to D1, indicating increased adaptation, However, they remain unsuitable for this investigation. Naïve Bayes showed poor performance in both datasets, suggesting it may be inappropriate for this classification task.

Decision Tree, LightGBM, and XGBoost was selected based on the efficiency of K-fold cross-validation in relation to the training and testing outcomes for D1 and D2, as well as the processing duration (s). These algorithms surpassed the

others, despite an about 12% disparity between the training/testing scores and the K-fold average outcomes.

3.1.26 Summary of Confusion Matrix

According to what was found with the model training described in [Section 3.3.7], the algorithms LightGBM, XGBoost, Random Forest, and Decision Tree were the most effective in every respect for both the training datasets and the testing datasets. Measurements such as accuracy, precision, recall, F1-score, and ROC-AUC were included in these evaluations. In addition, the performance of the model was evaluated once more by dividing the data into five equal datasets of 20,000 on both the D1 and D2 datasets. This is in accordance with the K5-fold method. The findings also demonstrate that the algorithms that were discussed before are the ones that performed the best, both throughout the training phase and the testing phase (K5 fold examples). In addition to this, the confusion matrix was used to determine the level of performance that the model had for each malware and benign file signature individually. Below, Figures 45 and 46 show that again those algorithms have the least false negative (FN) and false positive (FP) in all algorithms in both D1 and D2 training and testing outcomes. During training on D1, 80,000 of the D1 was used, which had 40,000 malware file signatures and 40,000 benign file signatures. After training all algorithms, LightGBM, XGBoost, Random Forest, and Decision Tree classified 39,995 (TN) as benign and identified 40,005 as malware (TP). In the testing, 20,000 D1 was used, which also contained 10,000 malware file signatures and 10,000 benign file signature algorithms. These algorithms identified 10,005 benign TN and 9,995 malwares. Comparing all the results reveals the algorithm's exceptional performance. D2 identified 39970 as benign (TN) and 40030 as malware (FP), while Test 10030 was identified as benign (TN) and 9970 as malware (TP).

Algorithms Dataset1 80%	TN	FP	FN	TP		TN	FP	FN	TP	FP/FN Training	FP/FN Testing %	Training	Testing Accuracy %
Decision Tree	39995	0	0	40005		10005	0	0	9995	0	0	100	100
KNN	39995	0	0	40005		10005	0	0	9995	0	0	100	100
LightGBM	39995	0	0	40005		10005	0	4	9995	0	0.02	100	99.98
Logistic Regression	39970	25	60	39945		10025	3	12	9983	0.11	0.07	99.89	99.93
Nave Bayes	39995	0	27448	12557	Testing 20%	10005	0	6886	3109	34.31	34.43	65.69	65.57
Random Forest	39995	0	0	40005		10005	0	0	9995	0	0	100	100
Stochastic Gradient Decen	39925	70	82	39923		9980	25	14	9981	0.19	0.19	99.81	99.8
SVM	39900	5	47	39958		10004	1	9	9986	0.07	0.05	99.93	99.95
XGBoost	39995	0	0	40005		10005	0	0	9995	0	0	100	100

Figure 45 Results confusion matrix of dataset 1 all models

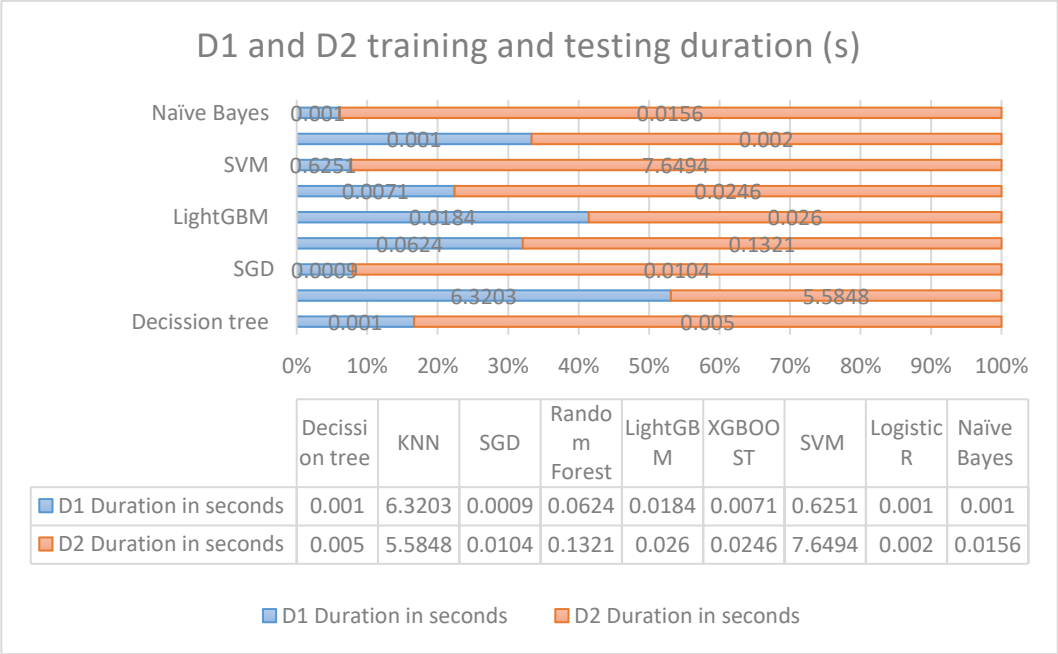
Source: Researcher (2024)

Algorithms Dataset2 80%	TN	FP	FN	TP		TN	FP	FN	TP	FP/FN Training	FP/FN Testing%	Training Accuracy	Testing Accuracy %
Decision Tree	39970	0	0	40030		10030	0	0	9970	0	0	100	100
KNN	39958	12	7	40023		10029	1	3	9967	0.02	0.02	99.98	99.98
LightGBM	39970	0	0	40030		10030	0	0	9970	0	0	100	100
Logistic Regression	36978	2992	1797	38233		9277	753	460	9510	5.99	6.06	94.01	93.94
Nave Bayes	15878	24092	6423	33607	Testing 20%	4048	5982	1591	8379	38.14	37.86	61.86	62.14
Random Forest	39970	0	0	40030		10030	0	0	9970	0	0	100	100
Stochastic Gradient Decent	36658	3312	1863	38167		9211	819	481	9489	6.47	6.5	93.53	93.5
SVM	37374	2596	1649	38381		9397	633	410	9560	5.31	5.22	94.69	94.78
XGBoost	39970	0	0	40030		10030	0	0	9970	0	0	100	100

Figure 46 Results confusion matrix of dataset2 all models

Source: Researcher (2024)

Based on the results from training, testing and model performance measurements, four algorithms performed better. Additionally, time was measured for each model to classify D1 and D2, as part of performance evaluation. Although Random Forest scored 100% on all measurements, it required a long process of time during classification. Random Forest's classification training and testing duration D1 were (0.0624 s) while D2 was (0.1321). The decision tree classification durations were D1 (0.001s) and D2 (0.001s), XGBoost D1 (0.0071s) and D2 (0.0246s), and LightGBM D1 (0.0184s) and D2 (0.026s).



	Decission tree	KNN	SGD	Random Forest	LightGBM	XGBOOST	SVM	Logistic R	Naïve Bayes
D1 Duration in seconds	0.001	6.3203	0.0009	0.0624	0.0184	0.0071	0.6251	0.001	0.001
D2 Duration in seconds	0.005	5.5848	0.0104	0.1321	0.026	0.0246	7.6494	0.002	0.0156

Table 21: Duration in of dataset classification training and test process
Source: Researcher (2024)

Table 21 reveals that Naïve Bayes, Logistic Regression, SVM, SGD, and Decision Trees provide much lower classification times compared to XGBoost and LightGBM systems. There is a significant difference in classification duration between KNN, Random Forest, and SVM. XGBoost, LightGBM, and Decision Trees are preferable because of the performance metrics that they possess, which include accuracy, precision, recall, F1-score, and ROC-AUC. For this inquiry, this research created a prototype model that applied these three various techniques. Although Random Forest algorithms offered the best possible performance, they required significant amounts of time to process

during the classification of large datasets. Meanwhile, comprehensive datasets are the primary focus of the algorithms that were mentioned earlier.

Sharing New Intelligence.

The central focus of this research is to enhance the proactive detection and mitigation of cyber-attacks, using automated dissemination of newly generated CI. This is to be achieved without any human interaction and to ensure real-time response from Cybersecurity systems in organisations.

For the sharing stage of intelligence, a solution was created to interface the CI sharing module with security infrastructures already existing within organisations; in line with stage 3 of the proposed model. Additionally, system alerts will be sent through emails to the management and system administrators. Using an authenticated user, and Representational State Transfer (REST) API produces a package that automatically updates internal security devices.

Creating an interface between the intelligence sharing unit of the newly established model and the firm security devices involves sharing new intelligence with security devices. The interface comprises of various interconnected units that perform various tasks, including a) A system update tool which is an API called Representational State Transfer – Application Programming Interface (REST API) b) Simple Mail Transfer Protocol (SMTP), which is used to transmit alert messages about newly developing threats.

3.1.27 Representational State Transfer Application Programming Interface (REST API)

The REST API uses Hypertext transfer protocol (HTTP) requests to transfer data between two systems. REST is a popular choice for developing web APIs because it is based on architectural principles that outline how networked systems should operate. REST APIs enable systems to exchange data in the form of CSV, JavaScript Object Notation (JSON) or Extensible Marked Language (XML) by sending HTTP requests, with request verbs such as GET, POST, PUT, or DELETE specifying the intended action to be carried out. Applications for REST APIs include getting information from databases, generating, or changing

resources, and starting processes in other systems (Kappagantula, 2022). Steps involved in exchanging data between two systems using a REST API are:

1. Define the API endpoints: The initial step is to specify the API endpoints, or the URLs that API clients will use to query the API server. Each endpoint ought to stand in for a particular resource, like a database entry or a user account.
2. Create the API's framework: The API structure must then be created, including the data format (such as JSON or XML) and request verbs (such as GET, POST, PUT, and DELETE) that will be used for each endpoint.
3. Develop the API server: The API server is responsible for receiving requests from API clients, processing them, and returning a response. This involves writing code to handle each endpoint, perform the requested action, and return a response in the correct format.
4. Test the API: Once the API server has been developed, it would be tested to ensure that it is working correctly. This can be done by sending test requests to the API using a tool such as Postman or by writing automated tests using a testing framework.
5. Integrate API into the Client Systems: The ultimate step is to integrate the API into the client systems, which involves writing code to send requests to the API and process the responses. The client systems can be web applications, mobile apps, or any other type of software that needs to exchange data with the API server.
6. Monitor and Maintain the API: Ongoing monitoring and maintenance of the API is important to ensure that it continues to function correctly and meet the needs of its users. This can involve fixing bugs, updating the API to reflect changes in the underlying systems, and ensuring that the API remains secure.

3.1.28 SMTP Protocol

Email messages are sent from one server to another using the SMTP (Javatpoint.com, 2022). The following steps make up the SMTP process:

1. Establishing Connection: The SMTP protocol is used by the sending server to connect to the receiving server.
2. Authentication: If necessary, the sending server verifies its identity with the receiving server.
3. MAIL FROM Command: To provide the sender's email address, the sending server sends a "MAIL FROM" command.
4. RCPT TO Command: To indicate the recipient's email address, the sender server sends a "RCPT TO" instruction.
5. DATA Command: To begin transferring the email message, including the message header and body, the sender server sends a "DATA" command.
6. File Attachment: A csv file that contains the new intelligence will be attached to email being sent.
7. Creating Connection: The SMTP protocol is used by the sending server to connect to the receiving server.
8. End of Data: To signal the end of the message data, the transmitting server transmits an end-of-data indicator, such as a dot (.) on a line by itself.
9. QUIT Command: Finally, sending a "QUIT" command causes the SMTP session to end and the connection to be closed.

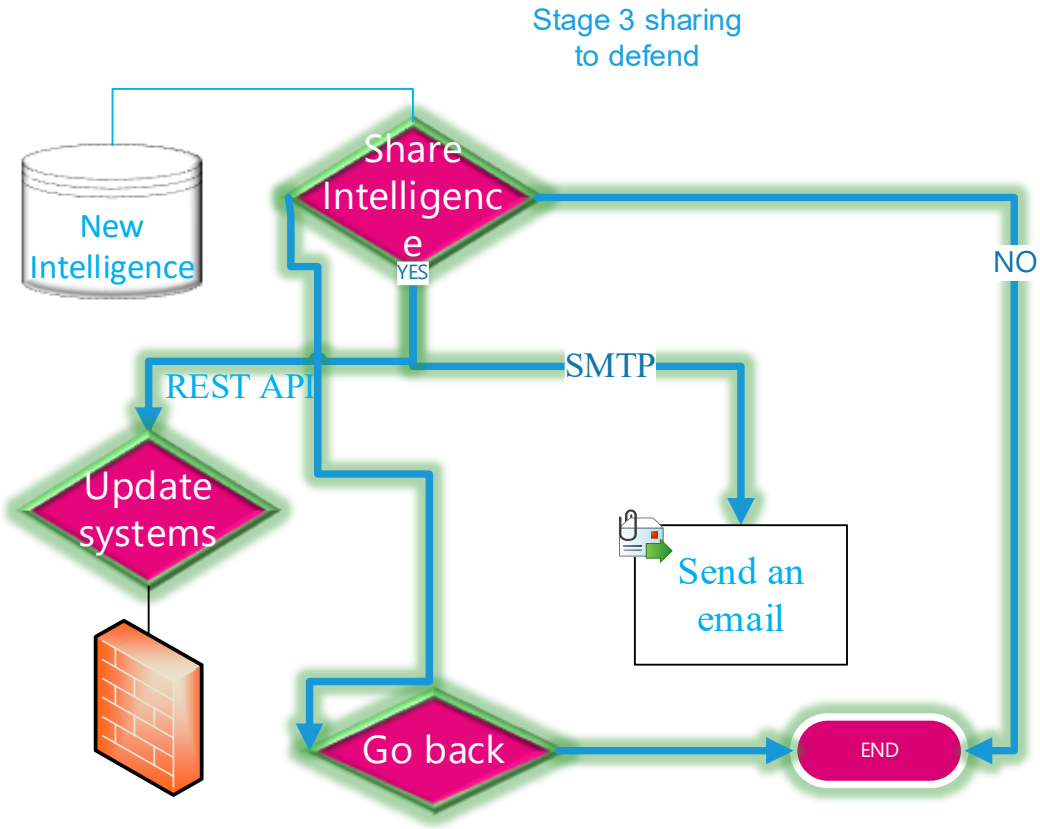


Figure 47: *Intelligence sharing process*

Source: Researcher (2024)

When we acquire information from an intelligence source in real life, we simultaneously save it in a new intelligence database and transfer it into a CSV file for further processing by algorithms as shown in Figure 47. We process the intelligence and classify it as either malware or benign. The new intelligence database will save any file that the classification model identifies as malware. Additionally, when a model generates a csv file containing new intelligence, the system initiates the share intelligence action, which begins messaging.

3.2 Experimental Design of the Study

In this study, different SM processes were integrated into a unified structural framework through the experimental design approach. The design employs web scraping method to gather data from Microsoft Kaggle and VirusShare. A combination of the ML models: the LightGBM, Decision Tree, and XGBoost were used to classify data into malware and benign file signatures.

Automated integration was enabled on the PSDCIS model with security systems to exchange intelligence in real-time. The REST API authentication fuses into the SMTP services to facilitate automatic email notifications to system administrators and the management team for any fresh intelligence. In theory, the methodology of this research is comparable to other malware analysis method by most researchers such as Analysis and Improvements of Behaviour-based Malware Detection Mechanisms (Alruhaily, 2017), Malware Detection in PDF Files Using Machine Learning (Bonan Cuan, Aliénor Damien, Claire Delaplace, Mathieu Valois, 2018), and Behavioural Malware Detection using Machine Learning (Almashhadani, 2020). The difference between these studies and this study is that in contrast, this study makes use of a variety of models that enable intelligence to be gathered, processed, and shared within a single automated system. Various malware signatures were used in these experiments, and different categorisation schemes were compared to determine how accurate each categorised data is, based on a collection of characteristics identified.

In this regard, malware signatures serve as the foundation of this study. A file signature is a special string of distinguishing bytes added to the file's header. A file signature typically resides in the first 20 bytes of a file on a Windows system. Different file types have unique file signatures, which are made up of a group of computer-readable instructions detailing a specific activity (SIDDIQUI, 2004).

In the Cybersecurity sphere, a malware signature is a distinct pattern that enables security systems to identify threats, such as a known malicious instruction sequence used by families of malware or a network traffic byte sequence. The method employed by many CS firms to find malware that has already been found in the wild and catalogued as part of a database is called signature-based detection (malwarebytes, 2022).

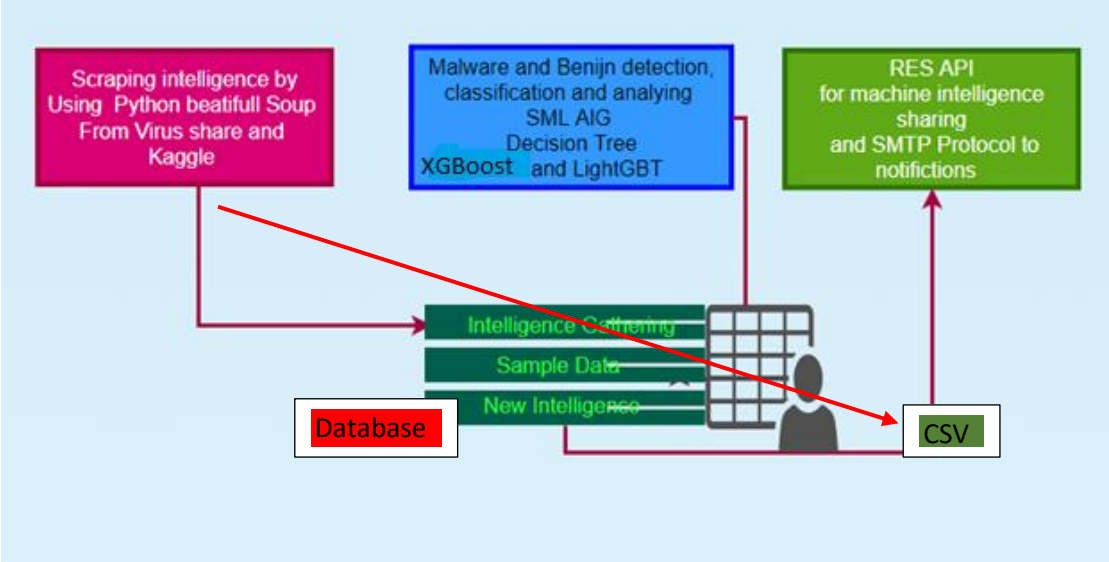


Figure 48: *Experimental design*

Source: Researcher (2024)

There are three distinct sub-models in the experiment that was employed for this research, each of which performs a different role inside the model. As shown in Figure 48, the first sub-model is to gather intelligence, automated dataset downloads and web scraping was used from repositories like Microsoft Kaggle and VirusShare. The second sub-model uses ML classification algorithms such as Decision Trees, LightGBM, and XGBoost to classify the malware and benign file signatures, which is a core of the analytical power of the PSDCIS model. This process automates the identification of threats. The third sub-model deals with exchanging intelligence through a REST API, which was used to update the security system with current knowledge. Simultaneously, the SMTP protocol is used to alert system administrators and managers of any emerging threat.

The SM process will automate all the three sub-models. SuccessState and the ErrorState are the two main states of the SM architecture. Therefore, during this process, if for instance, the intelligence encounters errors, the SM change to an ErrorState operations, which implies that the programme will disengage that function and roll back to the beginning automatically. If the intelligence is obtained, the SM goes to the next phase, which is saving the intelligence in the database.

3.2.1 State Machine

State machines, also referred to as finite-state machines (FSMs), are computer models that are used to simulate how a system works. The mathematical output of a SM is made of four stages: the state, input symbol, transition, and output symbol (Fussell, Spring 2010). This SM process is used to automate the workflow of the PSDCIS programme from automatic intelligence gathering, to processing and analyses using selected ML algorithms. This integration is implemented during the training and testing of classification algorithms discussed in Section 3.3.6. Figure 49 shows steps of implementing the SM practically. The SM executes its functions as follows:

i. The commencement phase: The process commences when the state machine triggers the initial action. This is the phase when the LoadVirusState and KaggleDownloadState gets activated, and the SM machine commences intelligence collection from various sources, and begins to acquire malware signatures from VirusShare repository and the Kaggle dataset. The data collected is then stored in the database and converted it into a CSV file format for further processing and analysis.
ii. The algorithm execution phase: The dataset will be sent automatically to the algorithms, which will then decide whether it is dangerous or not and save all malware signatures found in the new intelligence database.
iii. The check virus phase state: In this phase, the system gets connected to the VirusTotal website via REST API authentication and thereafter the file signatures are submitted to a distinct sandbox to ascertain whether the files are malicious or benign. Subsequently, verification results are extracted and analysed to assess the precision and reliability of the predictions made by the ML algorithms.
iv. The share state phase: ShareState seamlessly helps the model to incorporate intelligence into security networks like firewall and antivirus systems, through the REST API authentication. This allows for real-time intelligence sharing of valid CTI. The SMTP also generates automated notifications and sends to system administrators and security managers, so they stay alert of potential threats and take the right cyber defence action.

v. The terminal phase: The final step is to enter the SuccessState if the system completes all its actions successfully. If, however, any errors are encountered, the programme will halt, enter ErrorState and return to the beginning of the process regardless of which stage it was in.

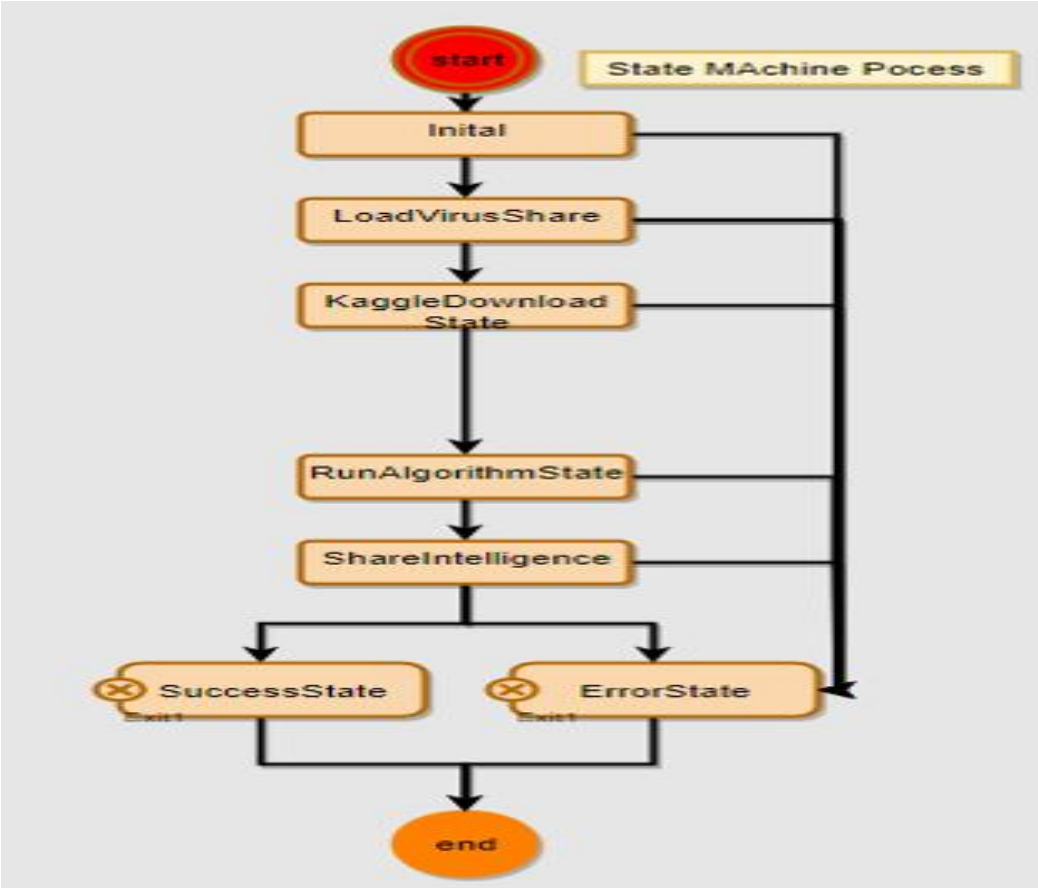


Figure 49: *State Machine experimental process*

Source: Researcher (2024)

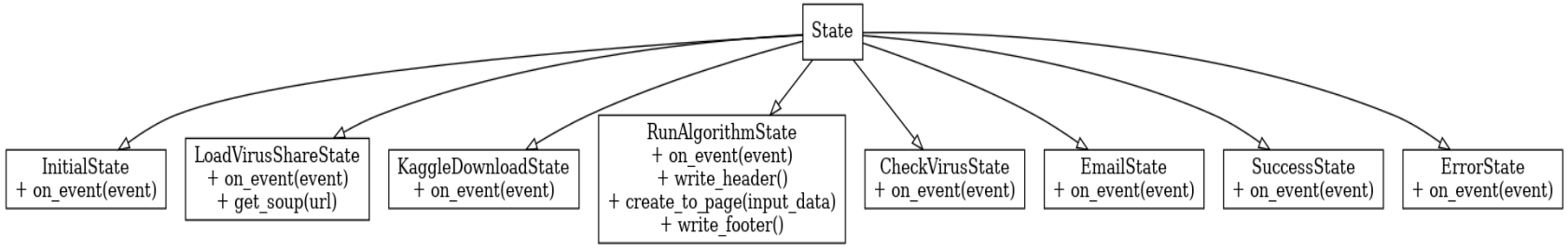


Figure 50: State class diagram

Source: Researcher (2024)

Figure 50 presents the transition of the State Machine inro sub- operational classes, which transcends from the Initial State through to LoadVirusShare, KaggleDownload, RunAlgorithm, CheckVirus, the EmailState, SuccesState and the ErrorState, where each of them are actioned from a [on_event] function to be able to perform specific tasks in a structured workflow.

a) **State (Base Class)**

i. This is the superclass from which all other states flow.

ii. Establishes the method **on_event (event),** which is implemented uniquely by each state.

iii. The primary function is to shift between several states in response to the specified event.

b) **LoadVirusShareState**

i. Fetches malware hash data from **VirusShare** and saves them in the database.

ii. Initiates a new **batch** method by adding a new batch entry.

iii. Scraps MD5 hash values from **VirusShare**.

iv. Saves the hashes in the **int_data** table.

v. Indicates the batch as **Completed** upon success.

vi. If an error happens, it deletes the batch data and changes to ErrorState.

c) **KaggleDownloadState**

i. Downloads malware datasets from **Kaggle**.

ii. Uses Kaggle API to fetch the **malware-detection.csv** dataset.

iii. Unzips the dataset and stores it in a designated **downloads** folder.

iv. Moves to RunAlgorithmState for further processing.

v. If an error occurs, it moves to Error State.

d) **RunAlgorithmState**

i. Runs ML algorithms to classify files as malware or benign.

ii. Loads the dataset and replaces MD5 hashes with those fetched from **VirusShare**.

iii. Cleans and pre-processes the dataset for model training.

iv. Trains multiple classifiers

1 **Decision Tree**

2 **XGBoost**

3 **LightGBM**

v. Estimates models using diverse metrics:

1 **Accuracy, Precision, Recall, F1-score, ROC-AUC, MCC, LogLoss, etc.**

2 Saves results and malware hashes.

3 Moves to CheckVirus State for verification.

e) **CheckVirus State**

i. Checks malware hashes against the **VirusTotal API**.

ii. Reads the **malware hashes** from the most recent dataset.

iii. Queries VirusTotal API to fetch information about:

1 **Risk indicators**

2 **Tags**

3 **Type description**

4 **Contacted IPs**

iv. Logs and stores the results.

v. Moves to Email State to notify the relevant authorities.

f) **EmailState**

i. Sends email notifications with detected malware signatures.

ii. Retrieves the most recent **malware report**.

iii. Sends an email to CS teams with:

1 Detected malware signatures

2 Recommendations for prevention

iv. Moves to Success State after successful email delivery.

g) **SuccessState**

i. Marks the process as successfully completed.

ii. Logs **"Finished Successfully"**.

iii. Terminal state (No further transitions).

h) **ErrorState**

i. Handles errors in the pipeline.

ii. Logs errors along with debugging information.

iii. Can be triggered from any state if an issue occurs.

iv. Depending on the error type, it may suggest corrective actions.

3.3 Experiment Tools

To carry out the research process and achieve the research goal, a lab was set up. The research required the following equipment: a Windows 10 computer with a minimum of core I7 2.6GH processor, security tools including firewalls, antivirus software, and email services. Additionally, it needed the Python libraries which are listed in Table 22 for the SMs.

Package name	
bleach==6.0.0	pickleshare==0.7.5
blinker==1.6.2	Pillow==9.5.0
certifi==2022.12.7	platformdirs==3.2.0
cffi==1.15.1	prometheus-client==0.16.0
charset-normalizer==3.1.0	prompt-toolkit==3.0.38
click==8.1.3	psutil==5.9.5
colorama==0.4.6	pure-eval==0.2.2
comm==0.1.3	pycparser==2.21
contourpy==1.0.7	pydotplus==2.0.2
cycler==0.11.0	Pygments==2.15.0
debugpy==1.6.7	pyparsing==3.0.9
decorator==5.1.1	pyrsistent==0.19.3
defusedxml==0.7.1	python-dateutil==2.8.2
executing==1.2.0	python-json-logger==2.0.7
fastjsonschema==2.16.3	python-slugify==8.0.1
Flask==2.3.1	pytz==2023.3
fonttools==4.39.3	pywin32==306
fqdn==1.5.1	pywinpty==2.0.10
graphviz==0.20.1	PyYAML==6.0
idna==3.4	pyzmq==25.0.2
ipykernel==6.22.0	qtconsole==5.4.2
ipython==8.12.0	QtPy==2.3.1
ipython-genutils==0.2.0	requests==2.28.2
ipywidgets==8.0.6	rfc3339-validator==0.1.4
isoduration==20.11.0	rfc3986-validator==0.1.1
itsdangerous==2.1.2	scikit-learn==1.2.2
jedi==0.18.2	scipy==1.10.1
Jinja2==3.1.2	seaborn==0.12.2

joblib==1.2.0	Send2Trash==1.8.0
jsonpointer==2.3	six==1.16.0
jsonschema==4.17.3	sniffio==1.3.0
jupyter==1.0.0	soupsieve==2.4.1
jupyter-console==6.6.3	stack-data==0.6.2
jupyter-events==0.6.3	svm==0.1.0
jupyter_client==8.2.0	terminado==0.17.1
jupyter_core==5.3.0	text-unidecode==1.3
jupyter_server==2.5.0	threadpoolctl==3.1.0
jupyter_server_terminals==0.4.4	tinycss2==1.2.1
jupyterlab-pygments==0.2.2	tornado==6.3
jupyterlab-widgets==3.0.7	tqdm==4.65.0
kaggle==1.5.13	traitlets==5.9.0
kiwisolver==1.4.4	tzdata==2023.3
lightgbm==3.3.5	uri-template==1.2.0
lxml==4.9.2	urllib3==1.26.15
MarkupSafe==2.1.2	wcwidth==0.2.6
matplotlib==3.7.1	webcolors==1.13
matplotlib-inline==0.1.6	webencodings==0.5.1
mistune==2.0.5	websocket-client==1.5.1
nbclassic==0.5.5	Werkzeug==2.3.2
nbclient==0.7.3	widgetsnbextension==4.0.7
nbconvert==7.3.1	xgboost==1.7.5
nbformat==5.8.0	xmltodict==0.13.0
nest-asyncio==1.5.6	numpy==1.24.2
notebook==6.5.4	packaging==23.1
notebook_shim==0.2.2	pandas==2.0.1
parso==0.8.3	pandocfilters==1.5.0

Table 22: State machine and python libraries required.

Source: Researcher (2024)

Summary

This chapter has presented the research methodology, explaining the rationale behind selecting the research techniques and discussed the pertinent research approaches employed in this study. Furthermore, the research elucidates the constituents of the approach, including the process of gathering

information, sources of data, and the tools employed to collect datasets. This chapter also elucidated the process of data analysis, including the algorithms employed to analyse datasets, as well as the implementation, how to train, test and validate the algorithms. Ultimately, this explained the process by which intelligence is disseminated.

CHAPTER FOUR - PROPOSED PROACTIVE ARCHITECTURE

In this chapter, the proposed architecture is the Pro-active Self Defence Cyber Intelligence Sharing (PSDCIS) model, which is discussed alongside the background theory that influenced its development. The development of points, functionality, and attributes within the model holds significant importance. Now the question is: How will this contribute to CI sharing?

Background of Pro-active self- Defence Cyber Intelligence Sharing (PSDCIS).

Armed with the literature reviewed and the observed limitations in present day CI sharing frameworks, it is important to consider the ways the proposed Pro-active Self Defence Cyber Intelligence Sharing (PSDCIS) model will enhance the threat detection and response. It seeks to integrate the important functions of prevention, detection and response in an automated way that will make intelligence-sharing a collaborative sharing system without the limitations previous models had. These three stages are very important in the process of cyber-attack defence.

According to Lapiedra (2000), the practices in the field of Information security are dynamic and progressive, and advances in stages to enhance effectiveness. Every stage of the process – prevention, detection and response, requires a well-thought-out plan and specific actions that will carry it to the subsequent stage (Lapiedra, 2000). These stages are shown in Figure 76 and include:

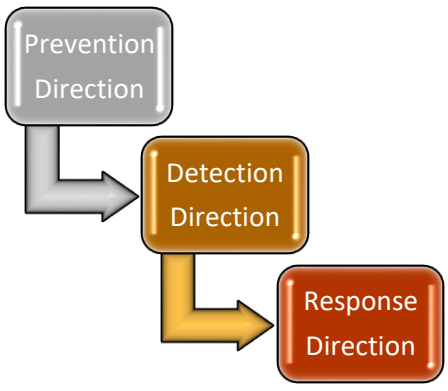


Figure 51: *Three stages of CS prevention, detection, and response.*
Source: Researcher (2024)

Stage 1: Prevention Attitude towards Preventing Cyber-Attacks (Preventive direction)

This first stage prepares mechanisms necessary for expecting and responding-proactively to cyber infiltrations and attacks. It is a very important stage that ensures readiness in the model development. An old adage validates this idea as it says: *"When the drum beats for battle, it is no longer possible to sharpen your sword; m*ake no mistakes." The ubiquitous availability of advanced technology and internet access exacerbates the vulnerabilities in our system. We are constantly "at war," and we must prepare to fight back against cyber attackers by sharpening our knowledge. "It is always preferable to take preventive measures rather than having to chase and litigate. To prevent an incident, it is necessary to engage in meticulous study and strategic planning.

In this stage, the CI platforms and systems prepare their strategy for gathering intelligence, sourcing their information, and storing and processing their information. Currently, all existing CI platforms collect threat data from different sources, with no harmony, scale, or real-time attribute, and the PSDCIS model seeks to provide a more proactive cyber-attack prevention framework to tackle the existing limitations.

Stage 2: Detection Attitude towards Preventing Cyber-Attacks (Detective direction)

This second phase is concerned with recognising threats while analysing data. It incorporates processes for information processing and analysis. Here it is necessary to systematically examine and evaluate unprocessed data to detect possible looming cyber-attacks or detect the patterns. The CI platforms vary in their approach to data processing at this stage. Some use artificial intelligence (AI) or ML to carry out data processing and analysis, while others still rely on static software or even human efforts.

During this stage, the suggested model is paired with CI systems that employ ML algorithms to categorise and identify harmful and non-harmful programmes.

Stage 3: Response Attitude towards Preventing Cyber-Attacks (Responsive direction)

As a final stage, response is very critical in CS operations and works like in the military where systems have similar strategy of gathering and analysing information before the attack, then using it to safeguard themselves. This military philosophy is brought to fore by the teachings of Sun Tzu that asserts that victory in war relies on gathering intelligence and prompt defence: "Know Your Opponent" (Michaelson and Michaelson, 2003).

This research has critically identified the weaknesses of the reviewed models and systems based on the three key stages of CS. The differences between the existing CI sharing models and multiple cyber-intelligence sharing methods and systems reviewed in Chapter 2 is obvious. Systems like STIX, the DML model and CIF etc. are all limited, as their efficacy is seen in stages 1 and 2, of CTI sharing models. Stage 3 is characterised by a responsive approach, where the operator focuses solely on practising and using intelligence to effectively respond to Cyber-attacks. The difficulty in their attempt to address stage 3 stems from a failure to adequately account for automation, standardisation, legality, trust, privacy, and real-time reaction.

Thus, all models and systems prioritise the gathering, manipulation, examination, and storage of data, with the goal of exclusively sharing this knowledge to members of the community. Occasionally, clients are required to get the data through a purchase. Thus, this logic implies that CI providers gather, process, analyse, and store information.

Fu et al., (2010) state that Darknet and Deepnet which are tools for proactive CS threat intelligence, concentrate on system intelligence through the collection and analysis of malicious hacking data from websites. However, they do not actively share the generated data. It is evident that the model uses passive CI instead of actively sharing data with the security systems. This model solely relies on the signal source for information gathering, as it does not actively engage with most of the vital intelligence sources. Collecting CI from various sources alone will not enhance the ability to actively defeat the enemy. Therefore, the proposed PSDCIS model will still collect intelligence from various information sources in real-time, but will then process, analyse, and share that intelligence with defence systems in real time. By doing so, it will reduce the risk of Cyber-attacks.

When compared to the proposed model, the ambitious CI models like the Darknet and Deepnet demonstrate several weaknesses such as:

i. No automatic sharing of intelligence produced in real-time.
ii. No active interaction with the enemy.
iii. Model information sources depend on a single information gathering source.

Afterwards, the proactive defence model employs cyber threat analysis and CTI model. The model integrates data from diverse security sensors and logs from different information systems and assesses the occurrence of an incident using precise rules and CTI. As a result, the model relied on passive CI defence practices, demonstrating the need for a real-time CI defence model. Proactive defence model used the passive CI defence practices.

Passive information gathering implies – gather as much data as possible and store data in a large database. There is no real-time intelligence-sharing scenario involved in this model. According to Gabriel Iovino (2015), Collective Intelligence Framework (CIF) is an example of CI warehousing. It collects information from the diverse sources and stores that intelligence in a storage. The knowledge is used to identify, detect and mitigate. The model pulls in several data-observations from many fonts; initiates a sequence of messages over time. For example, reputation. This CIF is the same as pro-active defence model, the only difference being that CIF is developed as a framework but functions as a model. Both systems use intelligence warehousing scenarios.

Since Stage 3 instructs a quick/real-time response attitude to defeat over potential emerging cyber-attacks. There is need for real-time CI sharing. In the world of CS, adversaries and defenders are continually trying to interact with each other. Business wants to know the cyber attackers' next move, so companies can proactively modify their defence systems and anticipate upcoming attacks" (CrowdStrike, 2019).

At this point, studies acknowledge the importance of developing a proper approach in the direction of Pro-Active Self-Defence Cyber Intelligence Sharing (PSDCIS), because it remarkably affects the design of systems.

The PSDCIS models was developed tackle all the shortcomings of the previous models by introducing a dynamic, real-time response approach in cyber

intelligence sharing. The model will automate real time CI sharing and reducing risk of attack, get data from multiple sources and providing multiple intelligence sources too, and will have active engagement and response coordination. It will use CI sharing protocols like SMTP, REST API etc to deliver real time, secure information exchange. It responds to the demand for prompt, pre-emptive cyber defence.

Pro-Active Self-Defence Cyber Intelligence Sharing (PSDCIS) Model Design

Core advantage of CI sharing lies at its effort to tackle cyber-attackers' activities, by providing prevention, detection, and response to malicious cyber threat. These services are overseen by providers specialised in CI. The intelligence feed is derived from both community members and external open-source sources, and so Intelligence is stored centrally and may be accessed by all contributing parties. Usually, users provide their unprocessed data to the threat intelligence providers for analysis. The collected raw data from community members usually contain confidential information about the source.

This technique of knowledge exchange has given rise to various challenges in the Information security business, necessitating a model whose mechanisms will provide quality and speed through automation standardisation, data quality, legal compliance, trust, privacy, accuracy, and timeliness.

This proposed PSDCIS model is a solution-response to this need, and is designed to address the prevailing cyber intelligence sharing challenges. Sub sections 4.2.1, 4.2.2, and 4.2.3, discusses the theoretical foundations of the proposed PSDCIS model, by applying insights from existing literature to identify and rectify limitations in existing intelligence-sharing models and systems. An in-depth examination of these limitations guided the creation of a more comprehensive framework for the automation of intelligence collection, processing, and sharing. The chapter outlined the fundamental model, its functions, and characteristics, highlighting the interplay of these components in facilitating the integration of automated intelligence-sharing processes into systems. This strategy seeks to improve the efficiency and efficacy of system administrators and managers by delivering timely and useful intelligence.

4.1.1 Framework of the Pro-Active Self-Defence Cyber Intelligence Sharing Model

This section explains the foundations of the framework guiding the Pro-active Self-defence Cyber Intelligence Sharing (PSDCIS) model, which was developed from literature review findings. The framework of the model aligns with these stages of conceptualisation:

i. Challenge Identification: The main challenges frustrating efficient CI sharing includes the lack of standardised processes, trust bankruptcy across organisations, lack of real-time sharing and obsolete sharing platforms. All these factors combined restricted the stakeholders' ability to collaborate against cyber threats and its mitigation.

ii. Model Development: Employing ML algorithms into a real-time automated CI sharing paradigm enhances the speed, accuracy, and scalability of threat identification and response, compared to existing methods.

iii. System Integration: Integrating a real-time automated intelligence-sharing model into current corporate security systems enhances the organisation's ability to proactively identify and eradicate possible cyber threats.

iv. Improvement: Overcoming the barriers of CI sharing through technological, organisational, and policy-driven initiatives allows businesses/organisations to make use of collective intelligence, thus improving their whole cyber defence outlook.

Figure 52 illustrates the outcomes expected when the PSDCIS model is operated in the real-world.

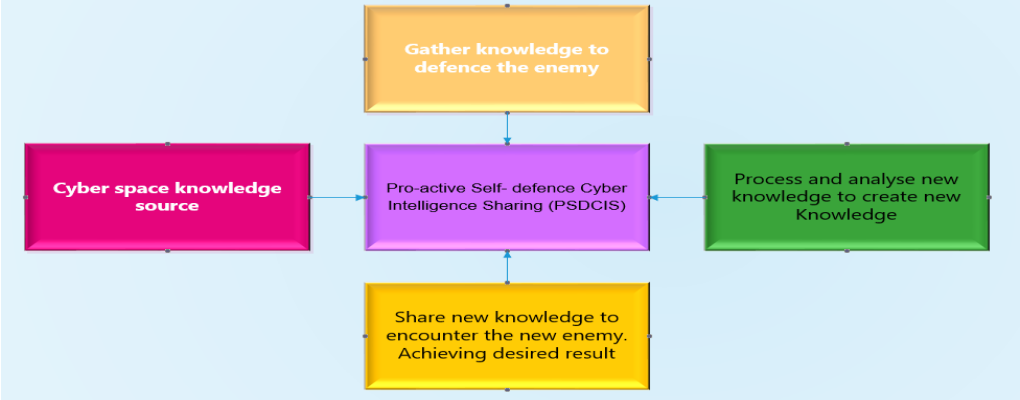


Figure 52: *Framework and PSDCIS (Researcher*, 2020)

Source: Researcher (2024)

In the cyber threat subsector, no one can deny the reality that not all businesses/governments either dealing with confidential or non-confidential information that involves one or more public services and industry-dependent systems, can risk being complacent for cyber-attacks to happen. Given the potential for dangerous damage when there is delay, it is crucial to prioritise proactive self-defence and adopt a real-time approach to effectively prevent resources from being infiltrated, hijacked and compromised. Therefore, this study proposes the Proactive Self-Defence Cyber Intelligence Sharing (PSDCIS) model, paving the way for successful cyber prevention, detection, and prompt response.

The most beneficial characteristics of the PSDCIS, is its expertise in coordinating real-time identification of threats, prevention, and prompt response. The obligations that all information security professionals concur to is that an action-focused CI is essential for safeguarding resources. The concept of developing the PSDCIS model stems from a similar philosophy, wherein the creation of intelligent systems is a vital as it can adapt quickly to new knowledge acquired. This continual acquisition of knowledge is essential regarding potential cyber-attacks, including their types, associated risks, and the signatures of emerging malware. Intelligence will serve as the basis for formulating detection, prevention, and reaction strategies.

However, there has been no research or model developed to this scale proposed in the PSDCIS, other than from the ideologies advanced in Sun Tzu's The Art of War, who specifically emphasises the philosophical teachings that validate this. Sun Tzu is cited for stating that:

> *"The Art of War instructs us not to rely on the possibility of the enemy not coming, but rather on our own readiness to receive him; not on the likelihood that he may not assault, but rather on the fact that we have prepared and taken our position to defend"* (Geers K. , 2011).

This is optimal for CI sharing as it encompasses all potential scenarios, particularly for the subject matter of the PSDCIS framework and ideology. This aligns with the PSDCIS that prioritises proactive

preparedness and proactive defence intelligence not a reactive one. Armed with a holistic insight of the cybercriminals, organisations can assess the risks that can cause havoc within their operations and stay safe. (Quinn, Matthew, Larry, Greg, and Gardner, 2021).

In the CI process, besides targeting to define the loaming cyber-attacks tactics and technology involved, the PSDCIS model tackles new cyber threats, profiles their characteristics and attributes like their IP addresses, malware and virus signatures, hashes, and file artefacts. Figure 78 illustrates the knowledge and intelligence, involved in developing the PSDCIS model.

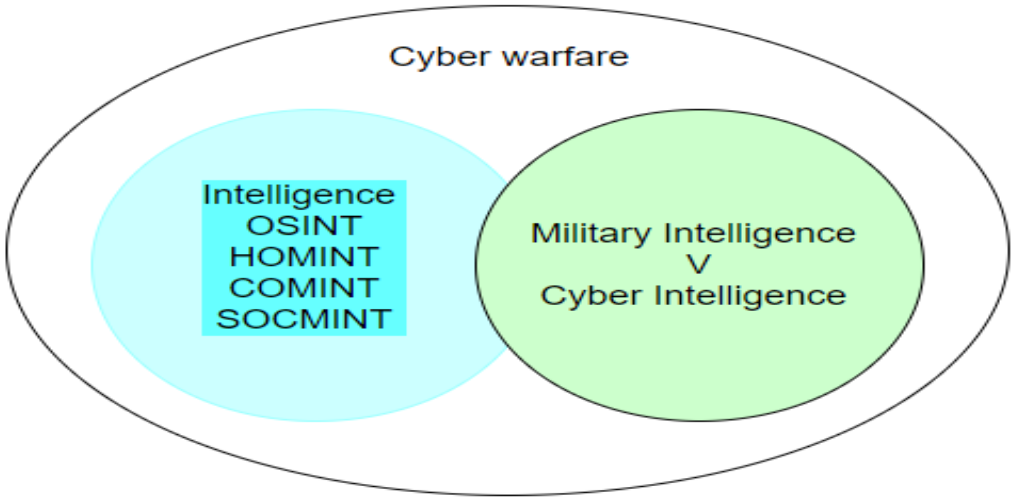


Figure 53: *Knowledge and Intelligence Involved for developing PSDCIS*
Source: Researcher (2025)

Figure 78 illustrates the connection between cyber war space and CI when both the military and intelligence agencies employ the same information-gathering techniques.

In general, the research that led to this study suggested that commanders in a cyber-war should focus their CTI on gathering evidence that stops cyber attackers from achieving their attack goal.

Since the knowledge is important for defending cyber-attacks and a cyber-warfare, it is no less important than a real-time in real-life war. As seen in the United States Army's Training and Doctrine Command TRADOC G-2, (2012), a thorough evaluation has to be done to mitigate cyber threats, and the evaluation

must include economic conditions, information conditions, infrastructure conditions, social conditions, physical environment conditions and time conditions. The recommendations appear incredibly important for any CI-sharing model and must answer the following questions, which together can guide the design of a resilient and safe CI-sharing model. Subsequently, defining what parameters of intelligence to be collected from the cyber space is important while developing this new model. Answering the following questions provides direction to follow during stage 1 (initiation), stage 2 (action) and stage 3 (defence) of the development of the PSDCIS model.

a) Who is the enemy?
b) What is the motive behind the enemy's attack?
c) What is the goal of the enemy?
d) How can intelligence about the enemy be gathered/collected?
e) How should the intelligence collected be used?
f) How can intelligence be integrated within organisational systems?
g) How should this intelligence be shared to detect, prevent and defend against cyber-threats?

4.1.2 The Conceptual Model

Drawing from the findings of the literature review and the conclusions of related research presented in Chapter 2, this study has formulated the PSDCIS model aimed at improving the sharing of CI while automating its collection, processing, analysis, and sharing. The proposed PSDCIS model illustrated in Figure 54 comprises of three stages: stage 1 (initiation), which involves gathering intelligence through automation; stage 2 (action) which is intelligence processing and generation; and stage 3 (defence), where intelligence is shared through automation using State Machine (SM) techniques. A SM serves as a formal mathematical model used in algorithmic design. The outcome improves information security measures and facilitates standardisation, as the machine can be integrated into all defence systems.

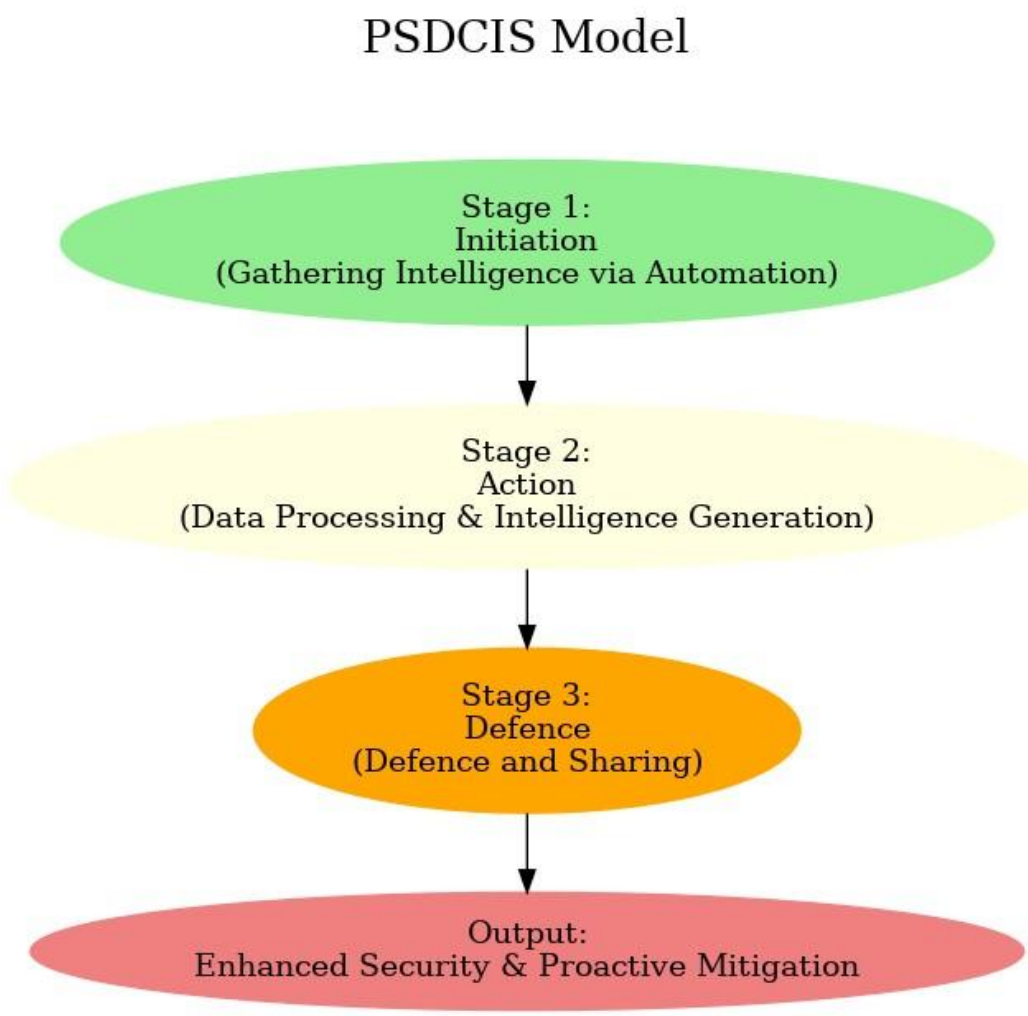


Figure 54 Proposed model concept.

Source: Researcher (2024)

The PSDCIS denotes an automation framework for malware detection, prediction and prevention (response).

4.1.3 Model Functions

The advantages of CI, lie in its ability to proactively thwart imminent cyber-attacks targeting business infrastructure and vital resources. To accomplish this, the proficient PSDCIS model has been proposed for development. The requirements used for the model include:

i. Stage 1 Initiation and automation: Model gathers intelligence automatically employing python BeautifulSoup libraries. Intelligence was gathered from VirusShare and Kaggle research centres to test this model's expected

outcome. During the intelligence gathering of this model, duplicate intelligence was avoided, as well as checking if the intelligence is complete. In processing, the PSDCIS model parsed data into readable format, and saved the intelligence gathered a database, then automatically creates a csv file that contained 100,000 records and 35 columns mix of two datasets.

ii. Stage 2 (Action): In this stage, the PSDCIS model started analysing datasets, by automatically putting data into the ML algorithms. The LightGBM, Decision tree and XGBoost was used to classify the dataset. The labelled dataset on exit had 50,000 malware file signatures and 50,000 benign file signatures. The model classified datasets into malware (1) and benign (0). All malware signatures after being classified, were exported and saved on both csv file storages and on the new PSDCIS intelligence database.

iii. Stage 3 (Defend): At this final stage, the automated model began the sharing process of the new generated CI by connecting and getting integrated into the defence systems. This process was handled by the REST API which systematised and authenticated with security devices such as; firewalls, anti-virus platforms, endpoint detection and response systems and intrusion prevention systems. When authentication got completed, the model started updating the malware signature databases by pushing newly created intelligence into defence systems. The model also sent emails to system administrators and managers with attachments that included new generated intelligence using SMPT protocol.

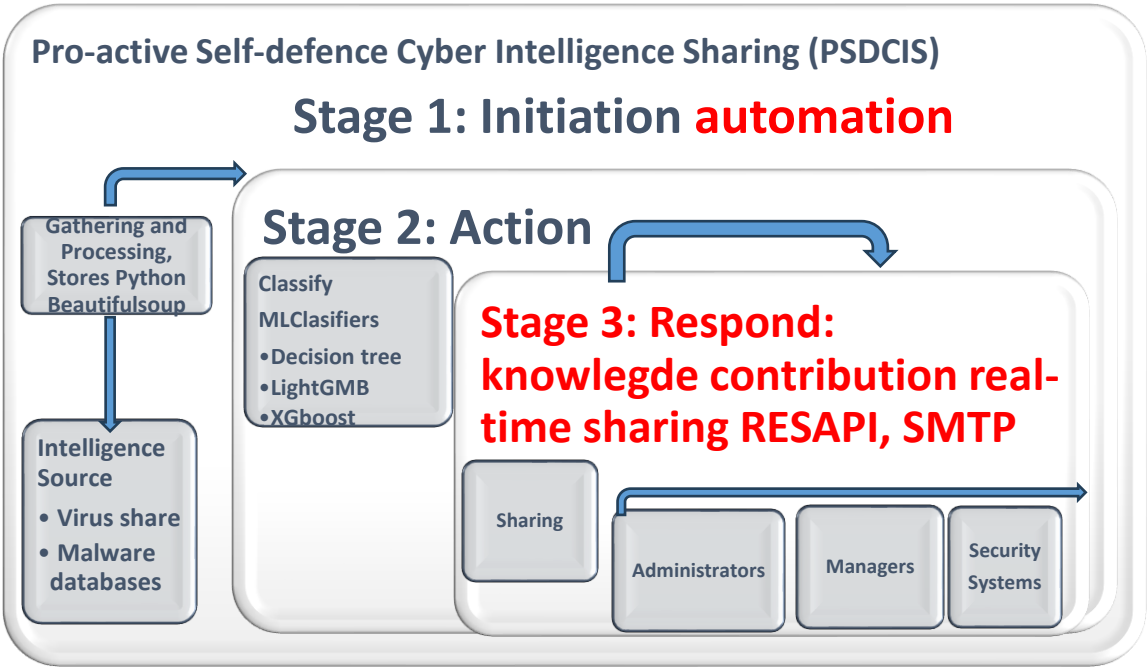


Figure 55: *Model Functions*

Source: Researcher (2024)

Figure 13 in [Section 2.6.7] highlights the limitations of current CI sharing models and Figure 52 highlights differences between the proposed PSDCIS model and the existing models and systems. The proposed approach entails automatic intelligence sharing through integration with security systems, while the existing models and systems share intelligence through community-based platforms while being stored in central locations.

In Figure 55, Stage 1 shows the process of acquiring information, Stage 2 demonstrates the techniques used to process and analyse the gathered information and Stage 3 specifies the domain of the scientific contribution. The proposed PSDCIS paradigm involves the use of automation and direct intelligence sharing within firm's security systems, in contrast to the current CI-sharing models that rely on community-based intelligence sharing.

According to Sillaber et al. (2016), one of the many challenges in community-based intelligence sharing is the quality of data available at the different levels of - collection, processing, saving/storage and sharing data. In addition, the authors discovered that the core issues that affect shared CI data, stem from the confusion that comes from trying to integrate threat intelligence that came from different sources as they are in fragments and not standardised.

Threat data quality is enormously critical for shared threat intelligence because it has to be useable and relevant.

Lovino (2015), discusses the Collective Intelligence Framework (CIF) as an example of CI warehousing. The CIF is a passive intelligence sharing framework where the actuality and quality of data will be influenced by its usability. Once intelligence is shared on time, and when needed, and new attacks can be prevented then the model is pro-active but this passive information sharing business does not allow for direct interaction with the cyber-criminal, but the knowledge used is the knowledge that has been processed by third-party threat intelligence providers. It does not allow immediate action, as intelligence can become obsolete due to delay, so organisations become CI recipients rather than CI participants. The activity is not sufficient by – relevance, action-ability and value, to confront the advanced technology of the existing cyber-attacks, and take action before damages occur.

In comparison to CI sharing tactics, the PSDCIS model encompasses methods that protects against cyber-attacks, as it does not merely prevent the cyber-attack in advance, but preferably learns new knowledge about the attackers and make it challenging for them to relaunch any attack.

Chismon and Ruks (2015) acknowledged that the aim of intelligence is the practice of moving identified subjects, in specific threats, to a direction articulate with the perception of 'knowns and unknowns'. Dalziel (2014) confirmed that for knowledge to qualify as CI, it must be relevant, actionable, and valuable.

i. Relevant: It must relate to a malicious actor directly or indirectly.
ii. Actionable: It must specifically be appropriate to trigger a response, change, action or decision.
iii. Valuable**:** It must provide an organisation relevant and valuable results.

If collected data cannot be used for self-defence, such data is not relevant, actionable, and valuable. Though the proposed PSDCIS models will gather information form external and internal sources, but considering the seriousness cyber threats to organisation's resources, the proposed PSDCIS requires integration of military philosophy into its structure and mechanism. The model uses thirteen military recommendations from Sun Tzu's Art of War (AoW) under

three functions - Initiation, Action, and Defend - which shape the development of active CI sharing in real-time for defending new attacks.

The relevant recommendations for the PSDCIS model include:

AoW II: The cyber commander using the PSDCIS model should collect malware signatures, suspected IP address of the enemy without the enemy's knowledge that his information has been exposed and gathered (Addinall, 2012; Geers, 2011). This would be done automatically and is equivalent to military intelligence gathering as the commander does this only after being convinced of the enemy's evil intentions. The PSDCIS is equipped with this as it automatically conducts signature collection and monitoring of IP addresses.

AoW III: If the cyber war involves the IT infrastructure, a cyber-only victory is the only way to defend the same. Therefore, it is important for the commander to secure victory before the enemy initiates the attack (Sawyer, 1994). The PSDCIS model ensures this, through real-time classification and integration.

AoW V: This is a win-or-perish situation, and therefore it is expected that the enemy will apply all his power and skill to outrun the commander. Therefore, the commander must remain one step ahead by consistently applying all of the skills of war and hit the opponent at the most opportune moment (Geers, 2011). PSDCIS ensures this through automating learning, detection and dissemination.

AoW VI: The commander should make all the enemy's cyber reconnaissance difficult and confusing, so that the enemy falls short of developing an effective strategy (Sawyer, 1994). The PSDCIS model achieves this using honeypots, decoy signatures and dynamic response algorithms.

"AoW VII: Much like in a real-time war, the commander should deceive the enemy through misinformation before going for the final kill (Parks and Duggan, 2001).

"AoW VIII: The commander would treat every combat situation as a new situation and approach it with all alacrity and would not sit complacent on happy memories of earlier success (Sawyer, 1994). The PSDCIS model achieves this by its ability to adapt its learning models and audit intelligence systems.

"AoW XI: Much like in a real-time war, the commander would keep checking on all the intelligence available of the system while at some time preparing the defence and would also keep in mind that the enemy too can apply new tactics to avoid detection (Sawyer, 1994).

The PSDCIS model achieves this by its ability to retrain data to ensure continued relevance. The PSDCIS model has changed the whole CI sharing paradigm from passive to active and from centralised models to decentralised models with real-time system approach.

A look at each of the stages reveal that:

Stage 1 Initiation: **What is Pro-active Self-Defence CI Sharing?**

Under stage 1 of PSDCIS, initiation will allow the cyber commander to gather information regarding attackers by identifying, characteristics and objectives of the enemy's next move, with the aim of creating valuable knowledge to defend the enemy. Then also formulate the base-line action by applying the military tactics of Clausewitz and Sun Tzu.

Sun Tzu and Clausewitz vary in their opinions on an ideal victory. For Sun Tzu; ultimate victory is at the level of grand strategy (Scobell, 2005). The victory in the modern cyber wars could be translated in comparison to the strategy of when information is gathered regarding forthcoming attacks, and the intelligence is used to defend against the enemy. In addition, time is very crucial throughout the intelligence gathering, processing, and sharing lifecycle. A perfect victory is speedy and without bloodshed: To subdue the enemy without fighting is the supreme excellence (Sun Tzu, tr. Shibing, 1993:2). A good General should aim at ruining the enemy's strategy, shattering his will to fight by use of psychology and deception rather than fully destroy the enemy (Sun Tzu, tr. Shibing, 1993:2). Sun Tzu writes that it is extremely important to gather as much information as possible about the enemy, to analyse his strengths and weaknesses and gain comparative advantage. He echoes it in the maxim "Know the enemy and know yourself; in a hundred battles you will never be defeated" (Sun Tzu, tr. Shibing, 1993:2).

Clausewitz's theory on intelligence gathering indicates the importance of intelligence gathering during the war. Key intelligence forces such as the morale

of the troops or their motivation, defy quantification and calculation. These forces remain undetermined, yet they have a major impact on the force of observable quantities. Both Sun Tzu and Clausewitz pointed out the importance of gathering intelligence about the attackers by identifying his next move, and method of attack, as well as his characteristics, and objectives. This provides information that intelligence sharing cannot do without. According to the accomplishments that fall under this stage, using active and passive intelligence gathering will involve gathering information from various sources and turning it into explicit intelligence to determine the enemy's intended target of attack.

Stage 2 Action: What am I going to do?

Under this stage 2, the PSDCIS model allows the cyber commander to process and analyse the intelligence gathered during observations period, and identify valuable and understandable information of the potential enemy. The cyber commander converts the intelligence into usable format. According to Tzu, strategy gathering is as much importance as intelligence gathering about the enemy, as well as processing and analysing it to decipher the purpose of the attacker and gain advantage (Sun Tzu, tr. Shibing, 1993: 106).

By perceiving and drawing in the enemy amid the reconnaissance, weaponisation, and conveyance periods of the cyber-attack lifecycle, it can give the cyber commander the chance and time to take the necessary course of action towards protecting the network and prevent cyber-attacks (T. Mattern, J. Felker, R. Borum, G. Bamford, 2014.). At this stage, the cyber commander employs ML algorithms to compute the data collected from the enemy domains, into the training dataset to identify new intelligence.

The cyber commander collects malware signatures such as malware, virus, phishing emails, IP address, domain names and news from social media about new cyber-attacks. The collected data is raw and the cyber commander will employ data parsers to parse data into readable formats. The cyber commander then stores the parsed data into intelligence storage databases. In addition, the cyber commander checks accuracy, correctness, and intelligence in non-duplicated data. So many intelligence reports in war are contradictory; even

more are false, and most are uncertain (Clausewitz, C. tr. Howard, M. and Peter Paret:, 1989: 117). Intelligence should be accurate, actionable and relevant.
Accordingly, the activities within this stage would include usage of the inimitable knowledge acquired during the process to enable the cyber commander to scale the enemy from all sides, and always be able to decide on the next courses of action.

Stage 3 Defend: How Will I Share Intelligence?

Under this stage 3, the cyber commander starts to achieve the goals of intelligence sharing by using the new intelligence to defeat the cyber criminals. According to Tzu (1910), the aim of spying on one's enemy is to understudy the enemy and such knowledge can only be acquired from the converted spy. CI is equivalent to military espionage and the whole aim is to use the knowledge of the enemy's tactics to launch defence against him.

Furthermore, Tzu asserted that foreknowledge is the hidden weapon of the wise sovereigns and the good generals, and grants those results and victories that confound ordinary men (Sun Tzu Tr: Lionel Giles, M.A., 1910).
Indeed, modern CS and Art of war tactics are so different, with regards to the use of intelligence for the purpose of defeating current cyber-attacks. Immediately Intelligence is predicted and transferred into a new intelligence database, the intelligence-sharing process will start with knowledge distribution. Intelligence will share in real-time in two different ways: simultaneously and automatically.

Integration into firewalls and antivirus systems through REST API authentication starts, as well as CSV file uploads with intelligence gets into systems. Administrators and managers will be automatically notified of upcoming attacks by email.

The process above gives the cyber commander ability to champion the course of the three main crucial factors of CS, which are; detection, prevention and response in real-time. The activities within these three stages provides the cyber commander with valuable intelligence, which would enhance the defence capabilities and ability to deal with current challenges surrounding CI sharing such as privacy and trust (Hasratyan, Lorenzo, Ligneul, Flegel, and Aaya, 2020).

Information sharing and trust are crucial in the fight against cyber threats and cyber-attacks.

The Proposed Model

The proposed model in this research study – the PSDCIS framework, solves the fundamental issues regarding intelligence sharing in three core sub-models:

i. Intelligence Gathering Sub-Model: Gathers new knowledge from both internal and external threat repositories, and extracts new IOCs like malware signatures, command-and-control (C2) IP addresses and behaviour anomalies while contributing to prevention of cyber-attacks. Intelligence is still raw and unstructured here.
ii. Intelligence Processing and Analysis Sub-Model: The second step is that the data is parsed into a readable format and classified if it has already been collected, reducing the possibility of duplication and redundancy. Then transforming that data into an Intelligence product. This enables proactive defensive action, and its timeliness makes the intelligence valuable and the operation, effective.
iii. Intelligence Sharing Sub-Model: In this third stage, this model brings its solution as it integrates the CI processing system with the security devices automatically and shares the new CI with organisations, business security devices, management, and system administrators via SMTP-based email alerts.

The knowledge contribution of this research, is the development of a CI sharing model that shares intelligence automatically without human intervention, and timely. When discussing CI, timing is crucial. This model solution offers real-time intelligence integration with enterprise defence systems and alerts system administrators and managers to the new cyber threat. This research study proposes the PSDCIS model design that is to be used to build prototypes that change theory to practical. Chapter five experimentally discusses more implementation prototypes in practical.

Since cyber incidents happen very often and cyber-attacks and cyber threats are always evolving, organisations' resources should be protected through active response strategies. The PSDCIS model also proposed a solution that is comprehensive and ensures real-time intelligence gathering, processing and sharing.

The variables needed to improve the new PSDCIS model as identified through literature reviews are listed in Table 5 and also illustrated in Figure 56. Whereas the PSDCIS's primary component is the sharing of CI, the improved models further provide offensive, circular collection and application of various knowledge features, which enable cyber-defenders to ward off impending attacks. The goal of enhancing PSDCIS is intelligence sharing in real-time, which entails counter-defending the cyber enemies before they carry-out their wicked actions. The real-time intelligence component of PSDCIS's enhanced model, eliminates the threat intelligence's current difficulties, like trust, legality, and privacy, and delivers intelligence that can be disseminated in real-time. Due to the fact that data warehousing alone is insufficient for dealing with sophisticated threats, an enhanced approach is introduced.

In contrast to other CI sharing strategies, PSDCIS actually refers to techniques used by cyber defenders to mitigate against cyber-attacks. The strategies ideally involve learning more about cyber attackers and making it more difficult for them to launch subsequent attacks (Chismon and Ruks, 2015).

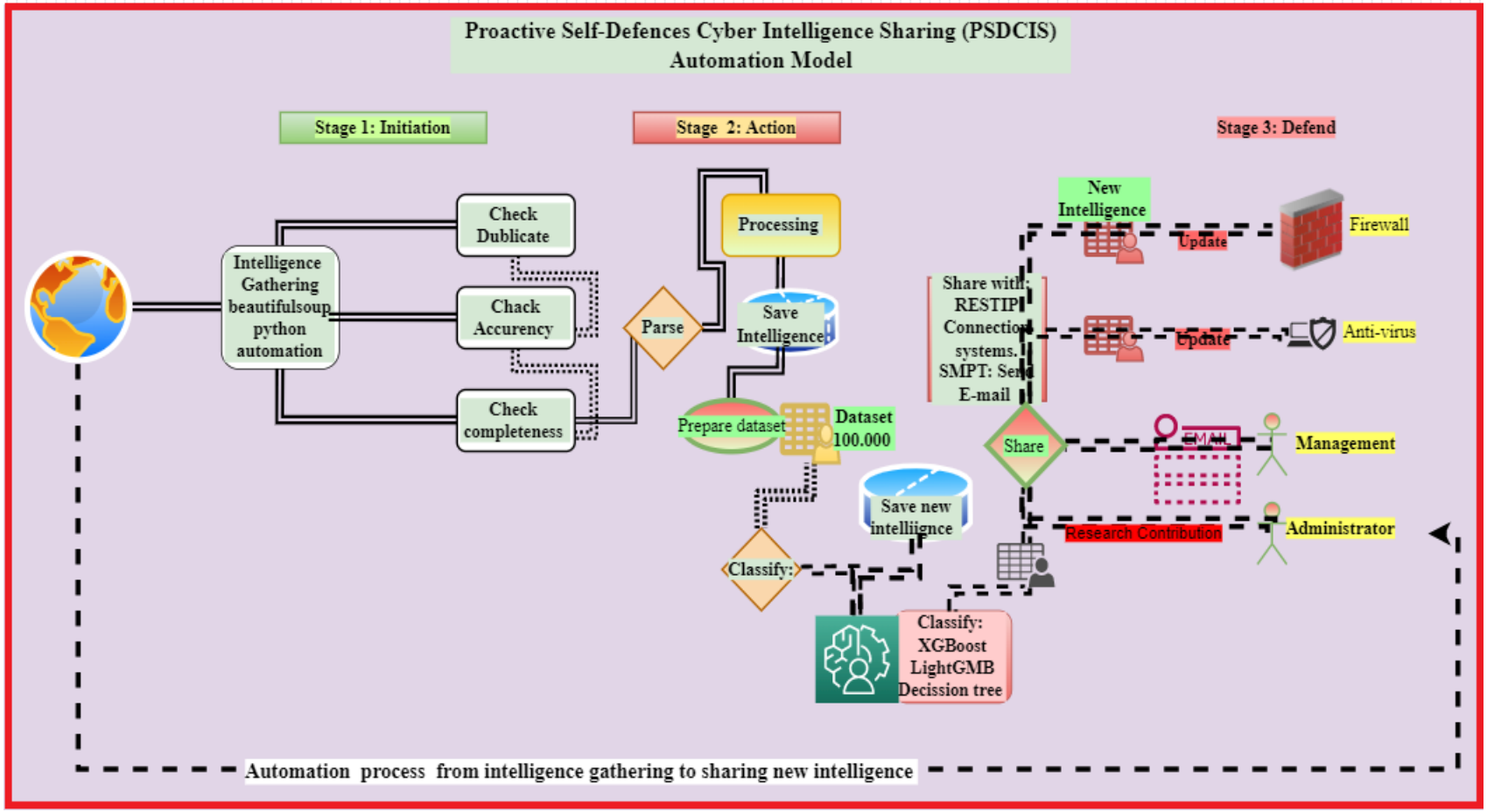


Figure 56: *Pro-active Self-defence CI sharing (PSDCIS)*

Source: Researcher (2025)

Table 22 explains model component and their functions.

Variables of enhanced Pro-active Self-defence CI sharing (PSDCIS)

<table>
<tr><td rowspan="3">Initiation:
Initiation is the process of starting Intelligence gathering from different sources and collecting malware signatures, for new viruses and characteristics.</td><td rowspan="3">Intelligence-gathering:
Knowledge from different sources from a selection of data repository. Collecting data from various channels is necessary to generate valuable CI, such as Virus Share and Kaggle.</td><td>Duplicate: During information gathering, it is essential to verify whether the data has been previously gathered, as the CI life cycle is an ongoing operation.</td></tr>
<tr><td>Accuracy: Accuracy of CI data appears to begin within the cyber defence. EY report (2014). “Many intelligence reports in war are contradictory; even more are false, and most are uncertain” (Clausewitz, C. tr. Howard, M. and Peter Paret:, 1989: 117). Intelligence should be correct, accurate and complete.</td></tr>
<tr><td>Completeness: Data should be complete and readable as a replacement to be used as intelligence. According to Ponemon, an institute’s study discovered that 70% of security industry professionals believe threat intelligence is often too voluminous and complex to provide actionable insights.</td></tr>
<tr><td rowspan="3">Action:
Action is processing the intelligence and converting it into an actionable data format. Classifying the data as malware (1) and</td><td colspan="2">Parsing Data: Data transforms unstructured and sometimes unreadable data into structured and easily readable data.</td></tr>
<tr><td colspan="2">Store Data: After parsing data, structured intelligence stores it on the intelligence database and creates a CSV file for the dataset's preparation for further processing.</td></tr>
<tr><td>Classify: The analysing approach will be using a</td><td>Prepare intelligence by creating a CSV file that contains new malware signatures discovered during classification.</td></tr>
</table>

benign (0) using ML algorithms like XGBoost, LightGBM, and Decision tree.	range of methods to guarantee an accurate and balanced assessment that should be forecasting actionable intelligence	Store new intelligence in database to avoid duplication for next intelligence generation.
		Forward new intelligence into sharing unit of the model.
Intelligence Sharing: sharing new intelligence through intergradation of the security systems and PSDCIS model in real-time.	**Defend:** Defend or protect against looming cyber-attacks by using Intelligence produced by initiating real-time automated sharing	**Firewalls**: Push new intelligence into firewall by using REST API authentication method.
		Anti-virus: Push new intelligence into Anti-virus by using REST API authentication method.
		Administrators: Send an email to the system administrators and attach the CSV file that includes new malware signatures produced during classification.
		Managers: Send an email to the managers and attach the CSV file that includes new malware signatures produced during classification.
		Managers: Are management team member to whom intelligence will be shared to, so they get notified of upcoming attacks.

Table 23: Variables of enhanced Pro-active Self-defence CI sharing (PSDCIS)

Source: Researcher (2024)

CHAPTER FIVE: DESIGN AND IMPLEMENTATION OF THE EXPERIMENT

This Chapter addresses the deployment of the structural framework of the proposed PSDCIS model, the procedure of its automation, the technical details involved, and the interconnectedness of the system in dissemination in real time. The model experiment deployed a Python application running on a local web that has an interface of a sequence of automations. This happens with a finite state machine (FSM) methodology. The Application encompasses various functional states including: Initialisation State, LoadVirusshare State, Kaggle Download State, Run Algorithm State, Check Virus State, Email State, and Update Firewall and AntiVirus State. This process will run automatically unless an error occurs in their error state machine, which will go back to the last state. Also, the application is based on the SM method of automation that enables functionality without human intervention.

It is noteworthy to state that throughout this study, the reference made to "finite state machine" does not mean a formal state machine mathematical model was used. Instead, the design implementation utilised the State Machine application design pattern, that structures the entire application logic into a set of separate states with distinct transitions. The pattern of this design was implemented due to how suitable it is for handling errors and managing the automation processes. It is not for the sake of adhering to the formal FSM theory.

Once activated, the model autonomously collects cyber threats, processes the information with algorithms, and checks the virus states by sending the generated intelligence to VirusTotal. Prominent antivirus vendors use the VirusTotal service to verify file signatures across multiple sandboxes. After that process on VirusTotal, the model extracts the result of the threats, with confirmation of the accuracy of the identified threat. This kind of feedback makes the shared intelligence very credible and reliable. However, the quantity of files that can be submitted to VirusTotal is restricted due to business considerations and usage policy. Only 20 files per team can be evaluated during each operation.

PSDCIS Automation

To ensure model deployment, the PSDCIS model is automated through SM automation processes. This mini-prototype model automatically downloads a dataset from Kaggle and loads it into ML models such as Decision Tree, KNN, and LightGBM for malware and benign classification. It will then save the newly created intelligence into the database and simultaneously create a CSV file for security system updates. Additionally, it utilises the SMTP protocol to send emails to system administrators and managers, providing them with a CSV attachment containing the discovered malware file signatures.

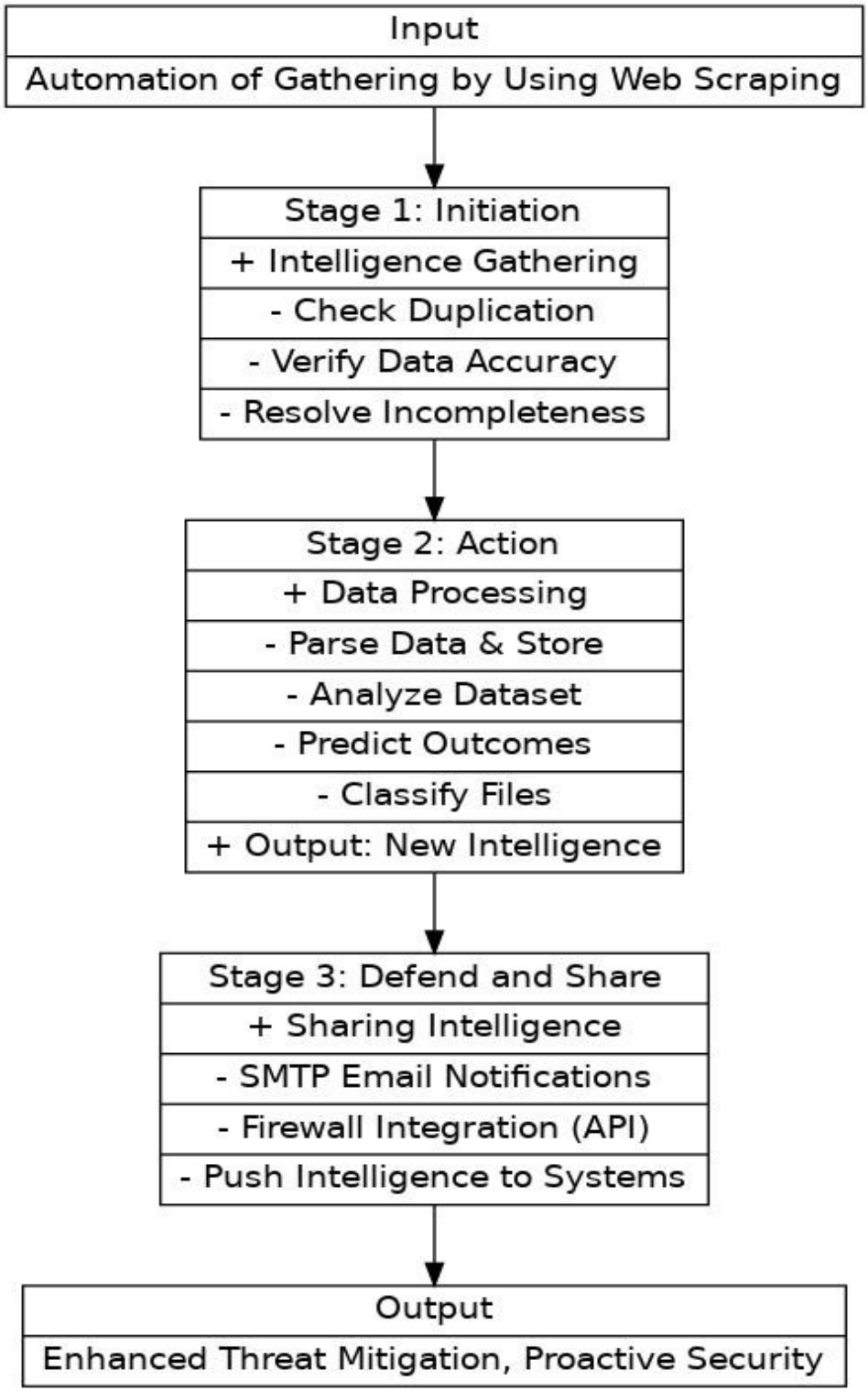


Figure 57: Sequence diagram of the state machine

Source: Researcher (2024)

The architectural framework of the PSDCIS model incorporates all three sub-processes into a single SM that operates the automated system. Figure 57 illustrates that datasets are plugged into the SM automatically, after which the SM executes all procedures and generates output, constituting new intelligence. This improves the automated responses to imminent threats from cyber-attacks. The system produces log outputs from every step of the process. The outcome when all states run, will produce the expected outcome of enhancing CI sharing by adding real-time integration into information systems.

5.1.1 Activating the Application

To be able to use an application (App), Windows PowerShell with an administrator account was first used. The application is run on Anaconda which is a scientific computing distribution for python that attempts to make package management and deployment easier. More detail is contained in Figure 58.

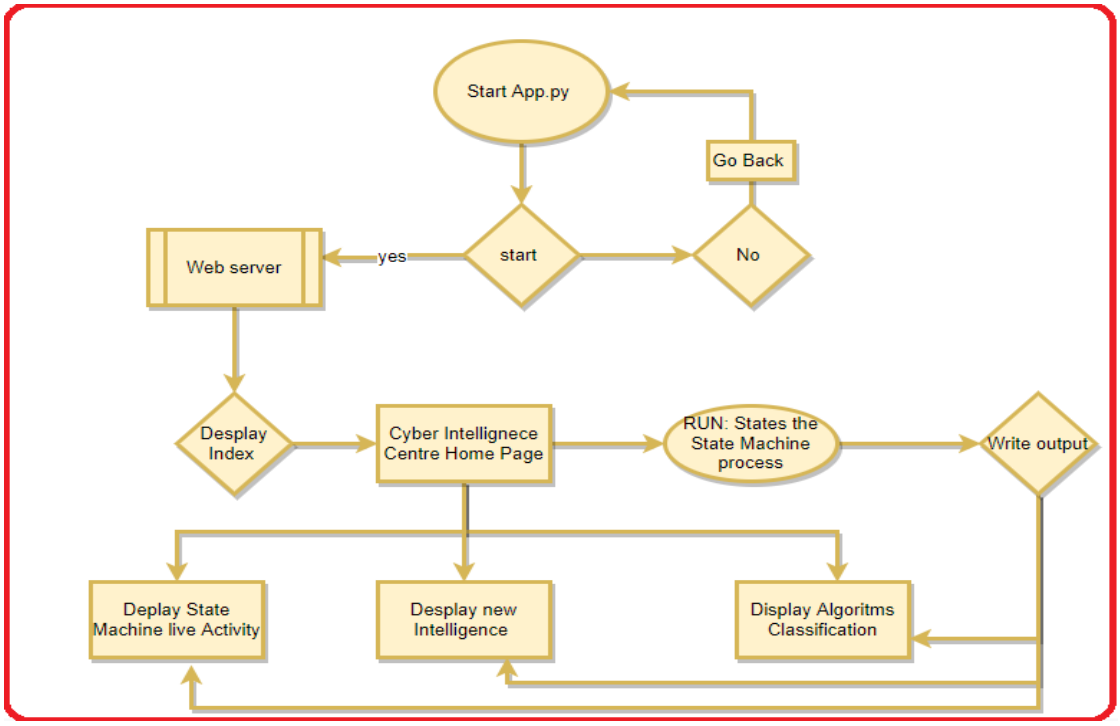


Figure 58: *Application activation process Diagram*

Source: Researcher (2024)

Upon initiation, the application is activated, and operated on a SM to start. The local web server loads the home page, which serves as the index, including the RUN command employed to initiate the SM operation. The CI Centre,

serving as the primary homepage, initiates the RUN instructions of the SM. The log view page functions as a medium for observing the real-time SM activity. The results page presents the ML classification outcomes, including; Accuracy, Precision, Recall, F1 score, and Confusion matrix. The output page presents the novel intelligence identified by the model as shown in Figure 59.

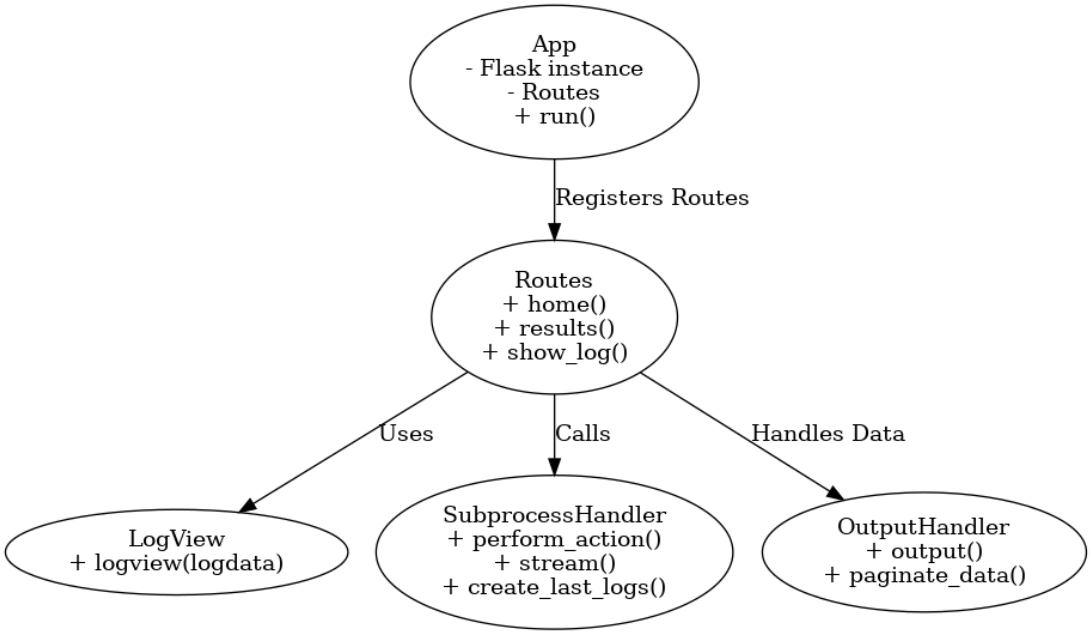


Figure 59: *Flask App Instance Class Diagram*

Source: Researcher (2024)

The Flask app class hierarchy offers an object-oriented view of the interactions among various components of the Flask app.

Flask App instance

i. This pertains to the Flask application instance.

ii. It is accountable for registering routes and executing the application.

iii. It engages with several handlers to assign duties.

Routes (Request Handlers)

Explains and administers the many routes (URLs) of the Flask application.

i. **Calls**: several modules established on user requests:

1. **home()** → Loads the home page (index.html).
2. **results()** → Presents the results page (last_results.html).
3. **show_log()** → Handles logging requests.

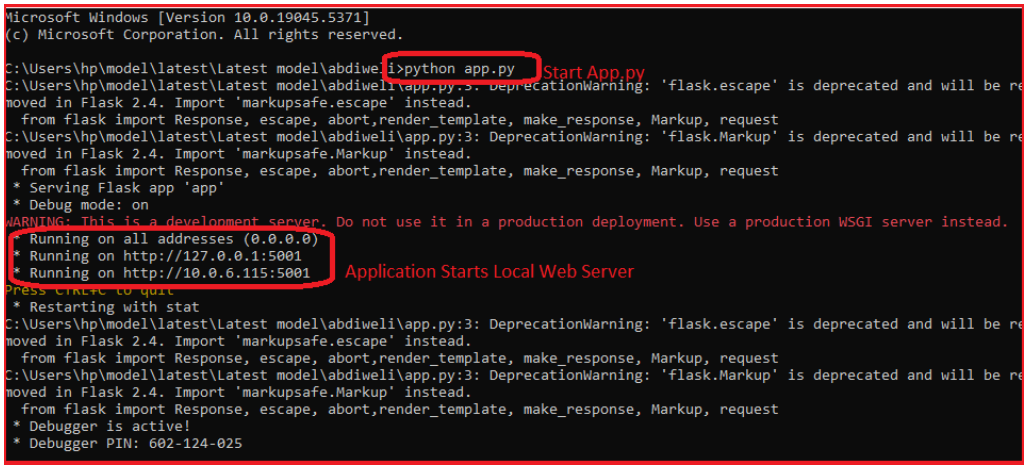


Figure 60: *Starting app.py application*

Source: Researcher (2024)

ii. Uses:

1. **LogView** for rendering logs.
2. **SubprocessHandler** for running background tasks.
3. **OutputHandler** for processing and displaying output data.

LogView (Handles Log Rendering)

iii. **LogView(logdata)** → Displays log files within HTML templates (logview.html)

iv. Facilitates organisation of logs for appropriate display on the interface.

SubprocessHandler (Handles External Script Execution)

v. perform_action() → Calls main.py as a subprocess when a request **is** made to /subprocess.

vi. stream() → Streams subprocess output live to the frontend**.**

vii. create_last_logs() → Saves logs for future reference.

OutputHandler (Handles CSV Data Processing and Page number)

i. **output()** → Fetches the latest CSV file from the downloads folder.

ii. **paginate_data()** → Splits the data into multiple pages and renders it in table.html.

This application deploys infrastructure that operates the SM structural framework and visualises the SM's processes. This might be referred to as the model's brain. Access is required to access the local IP address of the computer executing the application on Python program applying port 5001. Upon

launching the local browser and entering the local computer's data together with the port number, the model programme will start off, as represented in Figure 60.

5.1.2 Execution of the Model

After activating the application and starting the web services, the Run command is clicked to start it as shown in Figure 61.

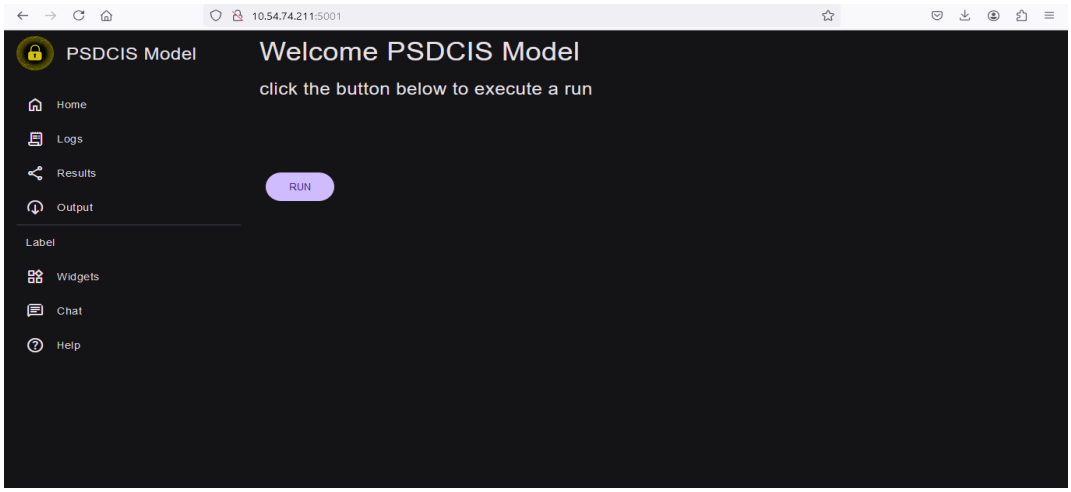


Figure 61: *PSDCIS Model run*
Source: Researcher (2024)

The model has four different menus: a) The Home menu; where the process runs or starts. Once the command (Run) is pressed, it will start the model's different components accordingly. It starts importing all the necessary libraries that the model needs to function and all state classes. The application has different classes, and every class represents one state of the SM. Every class has two functions inside the class; a) The work type; manages the operations of the class within the programme. This class imports all the libraries required for the class to handle tasks, like authentication, if the class must download data from outside sources. It works with predetermined rules. b) The event type; accepts an argument which stipulates the action to be triggered and so specifies what will happen and how the function should respond. This class structure validates how the finite-state machine of the model works, to allow for seamless state operations.

Figure 61 illustrates the four menus that are currently operational in the web application when the model is run.

i. **Run:** This is the command for starting the model application. Once the "Run" button is clicked, the application initiates, and the sub-process automatically proceeds sequentially. The process has a success state and an error state. If there is no error, the state machine process will continue until the end of the process which intelligence sharing.
ii. **Home:** This is the main menu where the operation of the model is activated.
iii. **Log menu**: Log menu holds all action and activity logs of the model.
iv. **Result Menu:** Outputs the classification results
v. **Output Menu:** Outputs the new intelligence.

In addition to the functionalities, the PSDCIS model has a layered security architecture to protect CI. All the data in transit were encrypted using TLS 1.3, and stored datasets, artefacts, and logs were secured with AES-256 encryption. For SMTP email dissemination, S/MIME and PGP protocols were employed to preserve confidentiality throughout transmission. These measures ensured enterprise-level security for intelligence in transit and at rest.

To uphold data integrity when feeds generate inconsistent results, PSDCIS engages cross-validation through its CheckVirusState module. With that it benchmarks classification outputs against VirusTotal's collective antivirus engines. Therefore, Intelligence is shared only when enough vendor agreements are reached, and this minimizes false positives and false negatives.

The model also presents a trust scoring mechanism, where every of the Intelligence artefact is weighted according to (i) the number of vendors that confirm the result, (ii) how reliable its originating feed is, and (iii) the validation metrics (e.g. precision, recall, F1-score, and ROC-AUC). When Intelligence artefacts are confirmed positive by 80% of vendors, they are tagged "high trust", while the artefacts that have low agreement are not trusted for use. Thus, reliable intelligence is the priority of dissemination.

5.1.3 Stage 1: Initialisation State

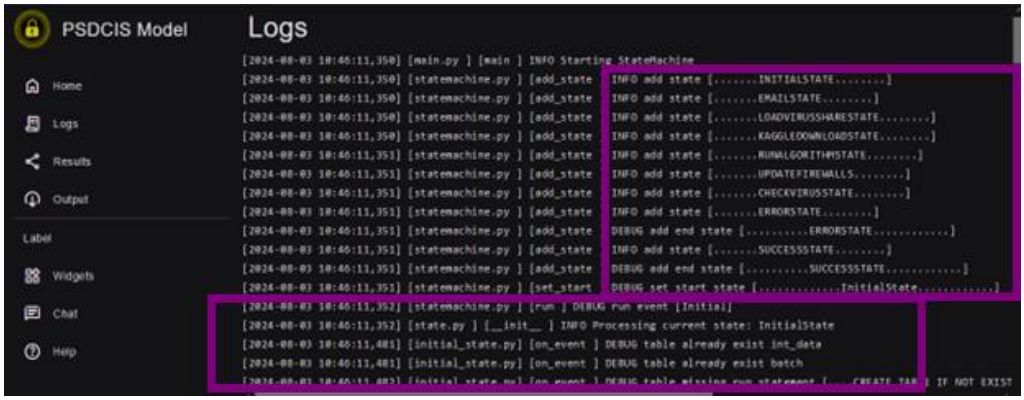


Figure 62: *Initiation state of PSDCIS*

Source: Researcher (2024)

During the start phase, the primary application begins, importing all essential libraries for the model's execution, and subsequently invokes the packages for all libraries and states. When the **run** command is clicked, the initiation process begins, incorporating each state of the machine into the running process, as in Figure 62. Once all states have been added to the state initiation pool, the SM commences operation. In addition, it will run debug, which prints all activities that are happening inside the SM on screen.

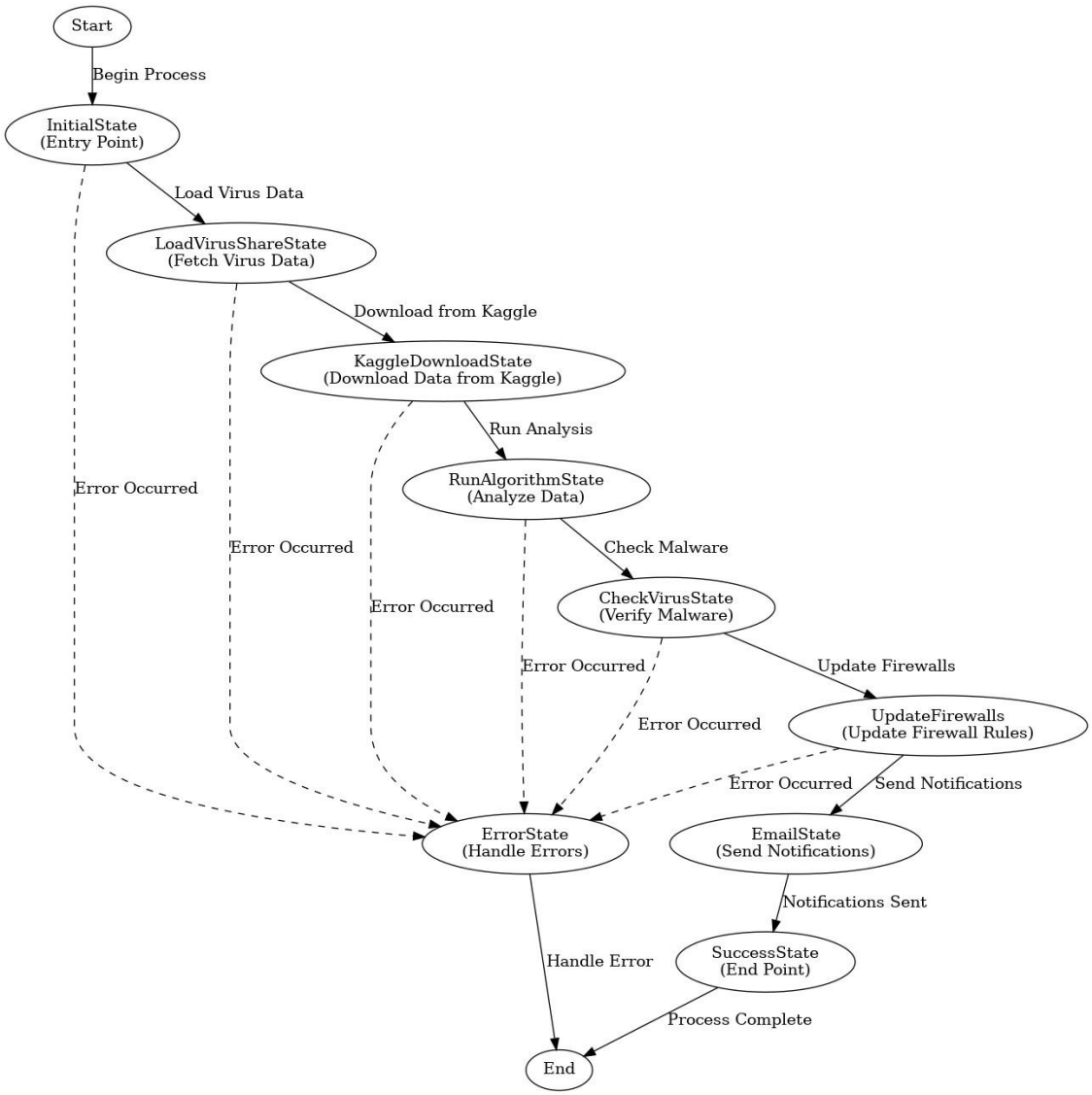


Figure 63: *Flow chart of how State machine process runs practically*

Source: Researcher (2024)

5.1.4 Stage 2: LoadVirusShareState and KaggleDownloadState

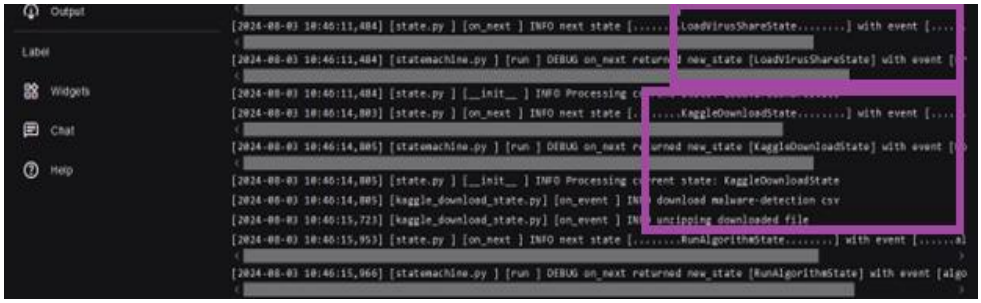


Figure 64: *LoadVirusshare and KaggleDownload*

Source: Researcher (2024)

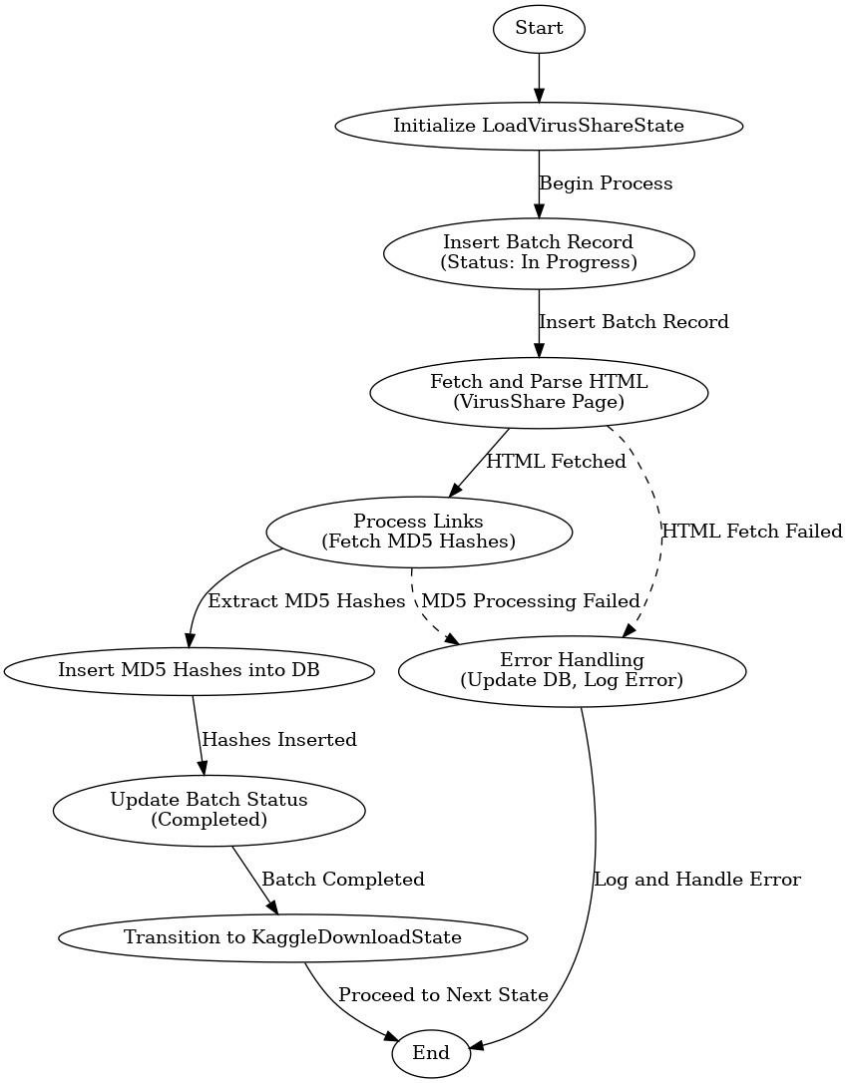


Figure 65: *LoadVirusshare State*

Source: Researcher (2024)

The LoadVirusshare state enables the model to obtain datasets from two separate sources simultaneously. The model applies BeautifulSoup to extract malicious file signatures from the VirusShare website and stores them in the intelligence gathering database. Furthermore, the model launches the KaggleDownload class, which establishes a connection with the Kaggle website through a designated API and then proceeds to collect the malware dataset. The model also commences parsing both datasets converting them into readable format. [Appendix 33] shows steps of the LoadVirusshare State and description

of the steps. In addition, Figure 65 shows the flow chart of the process and how actions happen inside the SM.

The SM completes the download of data at this point and successful data is converted into readable format, which is the base of the intelligence. Without gathering information intelligence, creation will not be possible. When this step is completed successfully, SM begins to analyse dataset.

5.1.5 Stage 3: RunAlgorithmState

SM initiates the automation process, and commences the previously completed intelligence gathering and processing. Next, is the classification of the dataset powered using XGBoost, LightGBM, and Decision Tree algorithms. The SM inputs a dataset in csv format into its RunAlgorithms process for classification. The model then splits data features into X and Y labels, and the algorithms starts classifying datasets into malware and benign file signatures. The model evaluates the production model's performance in terms of accuracy, precision, recall, F-score, and ROC-AUC and simulates integration with either the production systems or the deployment model. This lets it get to the file signatures it needs for classification in real-world setting. The real-time scanning process ensures the application of the same pre-processing steps used during model training, such as feature selection, extraction, and normalisation during production.

.

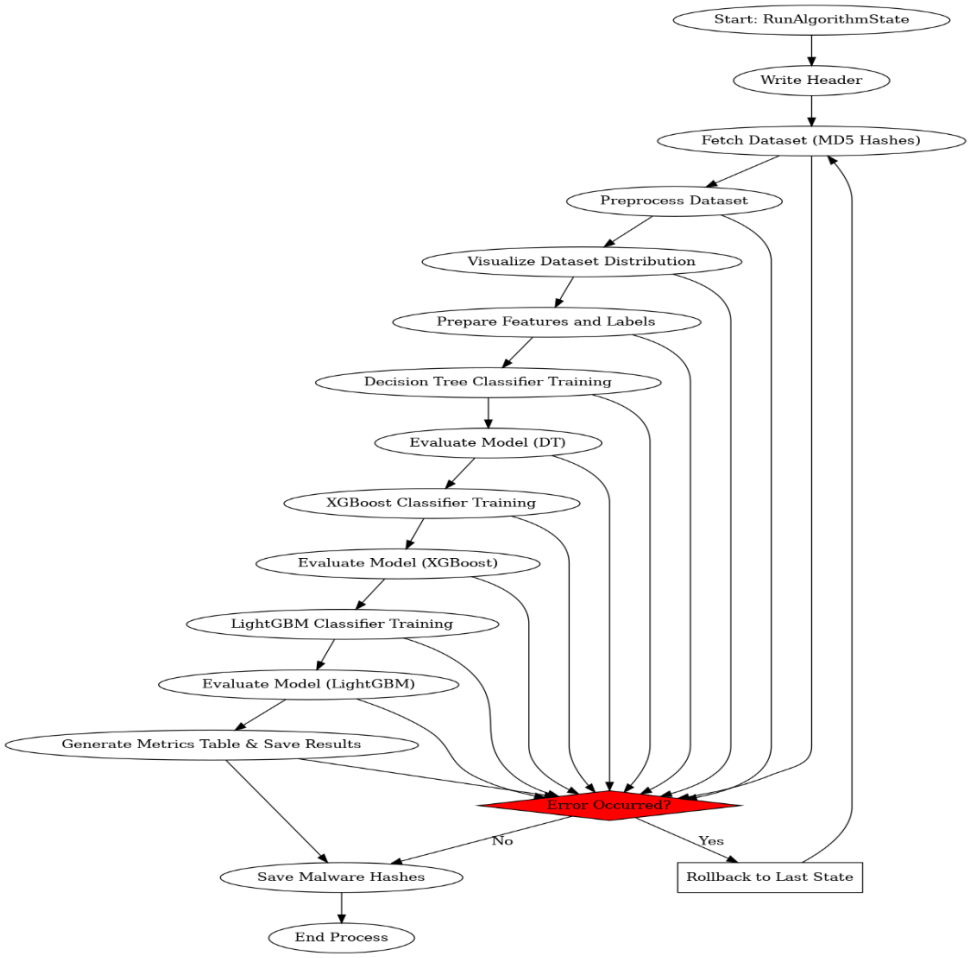


Figure 66: Flow chart Run algorithms classification process

Source: Researcher (2024)

During the process of categorisation, all algorithms execute the subsequent actions effortlessly.

i. Algorithms acquire knowledge; from the training dataset by modifying their parameters to minimise a loss function, so successfully adapting the model to the patterns present in the data.
ii. Management Overfitting; they employ techniques, such as implementing pruning regulations, to prevent overfitting and ensure that the model can effectively generalise to unfamiliar data.
iii. Prediction; utilises the acquired model to determine the classification (malware or benign) of newly encountered file signatures, serving as a foundation for decision-making in the realm of CI.

The RunAlgorithm process autonomously inputs 100,000 mixed datasets from Kaggle and VirusShare. The model categorises the dataset using all XGBoost, Decision Tree and LightGBM models, due to their ability to handle huge datasets, memory, and computing resources, especially the inherent optimisation of XGBoost and LightGBM (Guolin, et al., 2017). Upon classifying the dataset, the model's output will constitute the new intelligence.

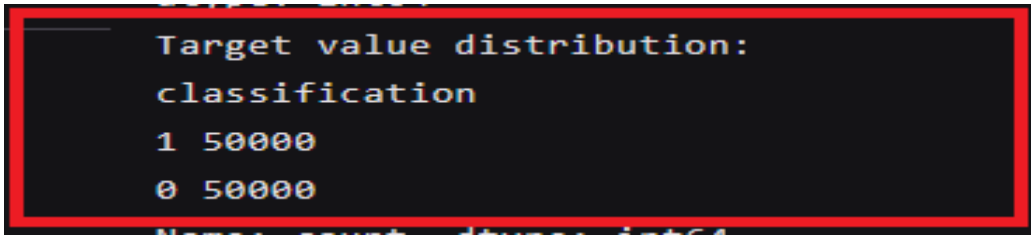


Figure 67: *Target value distribution dataset classification*

Source: Researcher (2024)

Figure 67 shows that the algorithms classified 50,000 of the data set as malware, represented by (1), and 50,000 of the datasets as benign, represented by (0). Additionally, it employed extra performance validation production models in this automation SM, such as:

i. Kolmogorov-Smirnov (KS) statistics to perform a non-parametric test to compare two distributions and evaluate if they exhibit significant differences. In binary classification, such as distinguishing between malware (1) and benign (0) samples, the KS statistic evaluates the projected probability for each class to assess the model's efficacy.
ii. SHAP analysis was applied to enhance model reliability. SHAP enhances confidence in automated malware detection systems by elucidating the decision-making process. Understanding feature donations ensures that algorithms rely on valid indicators rather than erroneous correlations.
iii. To avoid bias classification of algorithms, the Matthews Correlation Coefficient (MCC) which is an effective indicator for assessing the efficacy of binary classification algorithms was used. This metric is regarded as one of the most dependable for imbalanced datasets, such as in malware classification, where benign samples may significantly exceed malicious samples (Davide and Giuseppe, 2020).

iv. Log loss (Logarithmic loss) was also used to evaluate expected probability in classification tasks. This approach considers prediction uncertainty by imposing more penalties on algorithms for incorrect predictions.
v. The last part of the model performance validation is the K-Fold Cross-validation, which is an effective technique employed to evaluate the effectiveness and reliability of ML models. It is particularly useful in tasks such as malware classifications, where data distribution and model adaptation are essential.

[Appendix 34] shows steps and descriptions of how RunAlgorithmsState process executes in sequence at the SM automation process.

Upon finalisation of the tasks, the SM generates an output CSV that incorporates additional intelligence that algorithms have classified. The SM generates a Performance Measurement Table that illustrates the categorisation of datasets by the algorithms, as well as a Confusion Matrix that indicates the classification of the dataset as either malware (1) or benign (0). It also generates a table containing the Results, Accuracy, Precision, Recall, F1 score, ROC-AUC, MCC, Log loss, and SHAP values. Furthermore, the SM generates K-fold performance validation outcomes, average accuracy, and standard deviation. This is discussed more in [Section 5.4], Model validation.

5.1.6 Stage 4: CheckVirusState

After the algorithms have classified and generated new intelligence, the SM executes a query. The query as shown in Figure 68 is assessed to ascertain whether the algorithms accurately classified the data, yielding reliable information that can be used to prevent cyber-attacks.

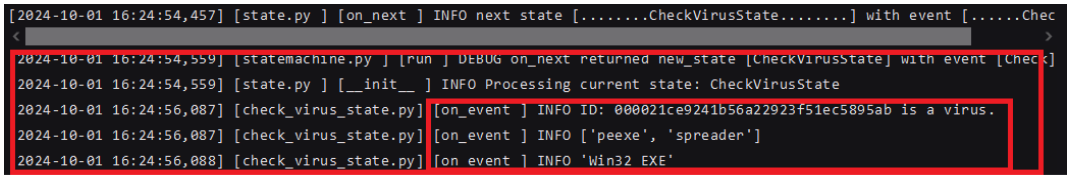


Figure 68: *Check virusstate*

Source: Researcher (2024)

The initial check state for the model workflow follows these steps:

i. Loading the latest CSV file that contains malware file signatures identified by the algorithms.
ii. Starting the checking process by using the VirusTotal API key, the view file signature is uploaded into the collection of sandboxes that is owned by the biggest antivirus vendors in the world, which run under the VirusTotal side to check whether those file signatures are malware or not.
iii. Collecting responses from the sandboxes, including IP addresses related to the malware file signatures.

If the process is successful, it returns the virus; otherwise, it may return no virus, and the process will end. If the response is a virus, the process will initiate the next step of sharing intelligence.

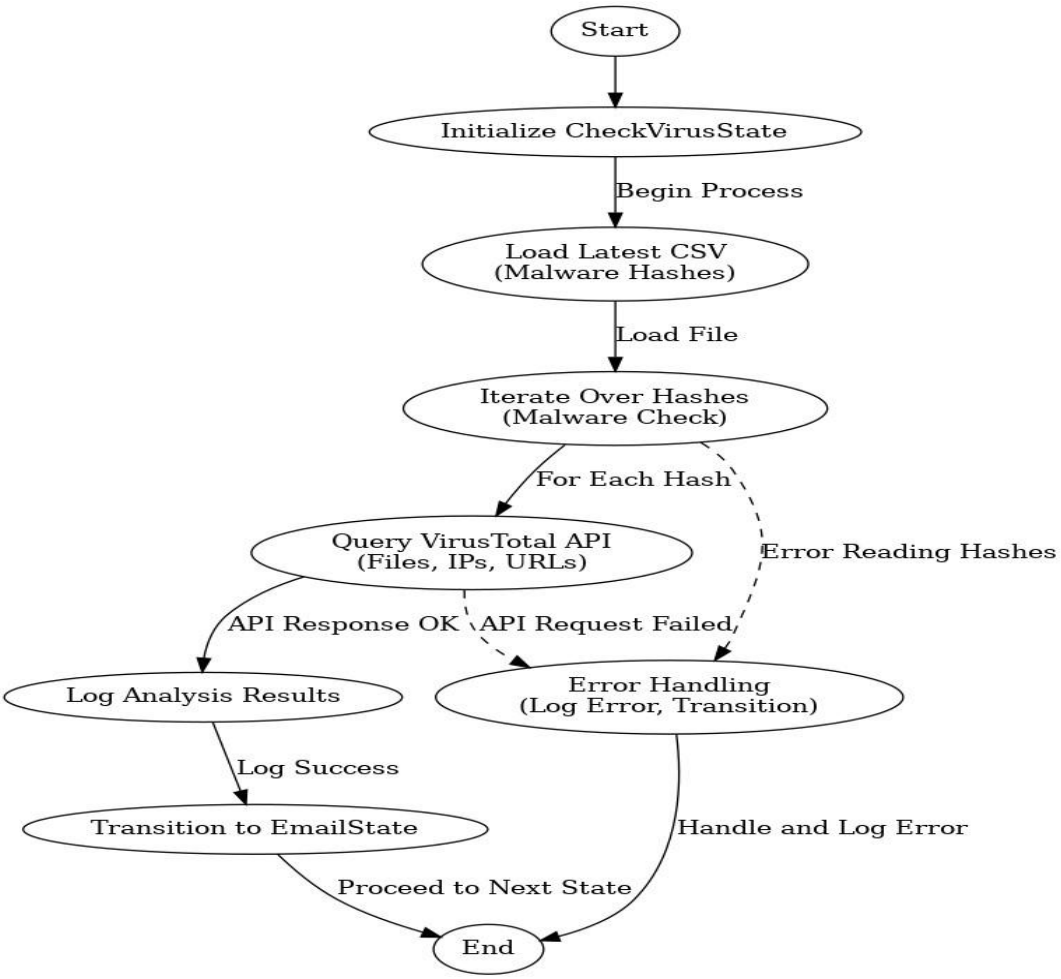


Figure 69: *Check state process*

Source: Researcher (2024)

Figure 69 illustrates a flowchart of the CheckState process within the SM, with the verification of malware file signatures discovered by the model. Additionally, the steps of CheckState and its corresponding descriptions are established in [Appendix 35].

5.1.6.1 State 2: Result of CheckState

During the check, the SM selects the first 10 file signatures from the new intelligence and then uploads them to VirusTotal sandboxes to verify if the model has correctly identified malware file signatures. When file is uploaded, different security vendors analyse it by comparing its signatures with the intelligence database. If the vendor's database finds the file signature, it flags it as malware, but if not, it flags it as a benign file. When flagged by security vendors, extra intelligence related to specific malware signatures is extracted. In [Appendix 36] the results of the automated status check, including the type of malware, its description, and the IP address of the malicious programme is displayed. In addition, [Appendix 37] and [Appendix 38] indicate the result of security vendors' detection of file signatures, categorised as [yes], which represents detected, and [no], which represents not detected.

Subsequently, 10 malware signatures were automatically sent to VirusTotal to validate the model's conclusions. Of the 80 security companies analysing the malware signatures, 74 identified the first file signature as malware, while 6 classified it as benign. 2nd: 63 malware and 17 benign; 3rd: 61 malware and 19 benign; 4th: 58 malware and 22 benign; 5th: 63 malware and 17 benign; 6th: 2 malware and 78 benign; 7th: 68 malware and 12 benign; 8th: 65 malware and 15 benign; 9th: 63 malware and 17 benign; 10th: 50 malware and 30 benign.

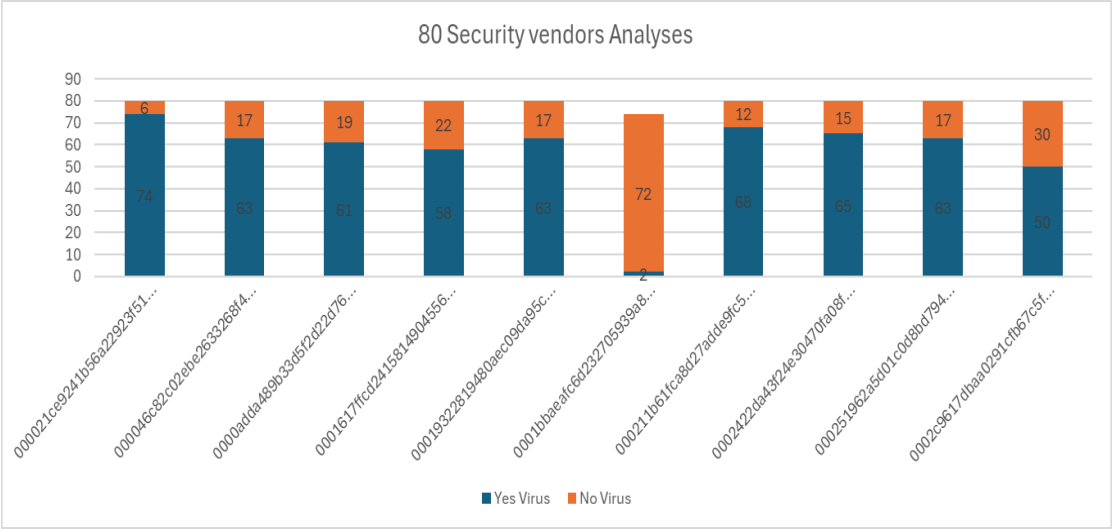


Figure 70: *Security vendors model verification result*

Source: Researcher (2024)

5.1.7 Stage 5: EmailState

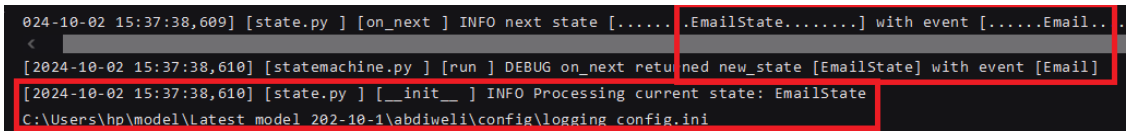


Figure 71: *Share intelligence through email*

Source: Researcher (2024)

Once the process email state starts, it authenticates the Google mail by using pre-conFigured email token which similar user passwords.

i. Login state: The SM starts login processes and queries the Google mail by helper module.

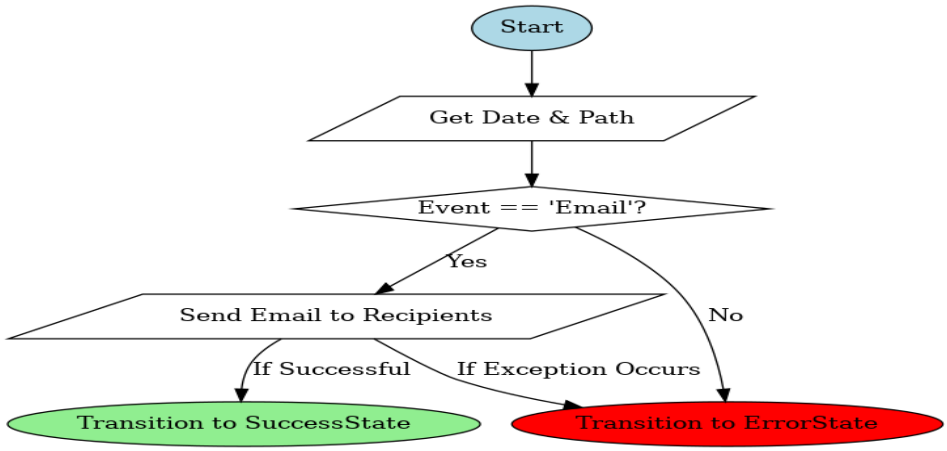


Figure 72: *Email state flow chart*

Source: Researcher (2024)

ii. Authentication process: It starts the OAuth 2.0 authentication process.

iii. Token use: The SM EmailState module uses the pre-conFigured token to log into the Gmail pre-defined account.

iv. Compose the email: The SM starts sending emails to the predefined emails (intelligence3@gmail.com) and (dmnsmlmanager@gmail.com).

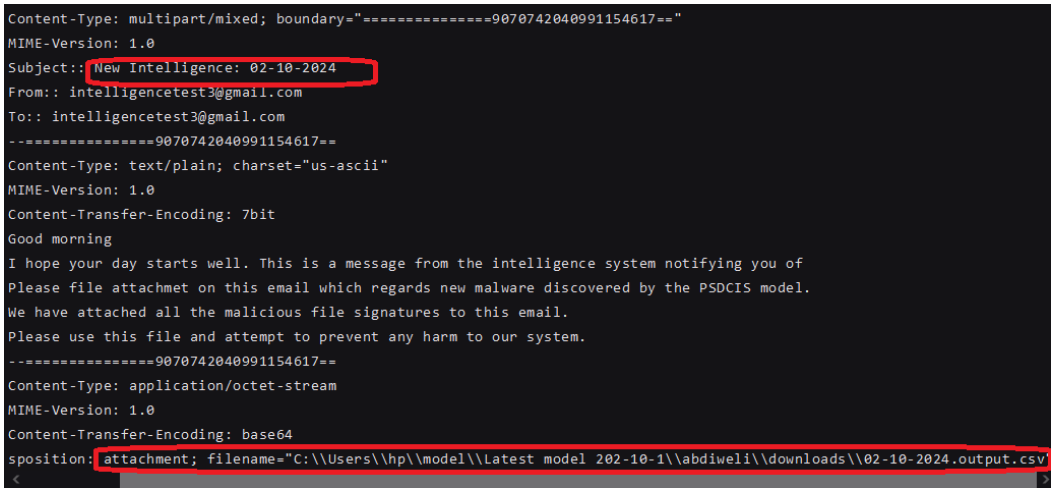


Figure 73: *Email and attachment sent to admin and management*
Source: Researcher (2024)

Prior to the dispatch of the email, the model generates a file named output.csv containing 50,000 malicious file signatures, representing the new intelligence supplied by the PSDCIS model. Figure 73 illustrates the process by which the model stores the file in the designated folder and appends it to the email. In addition, the Figure 74 shows email sent to intelligent3@gmail.com.

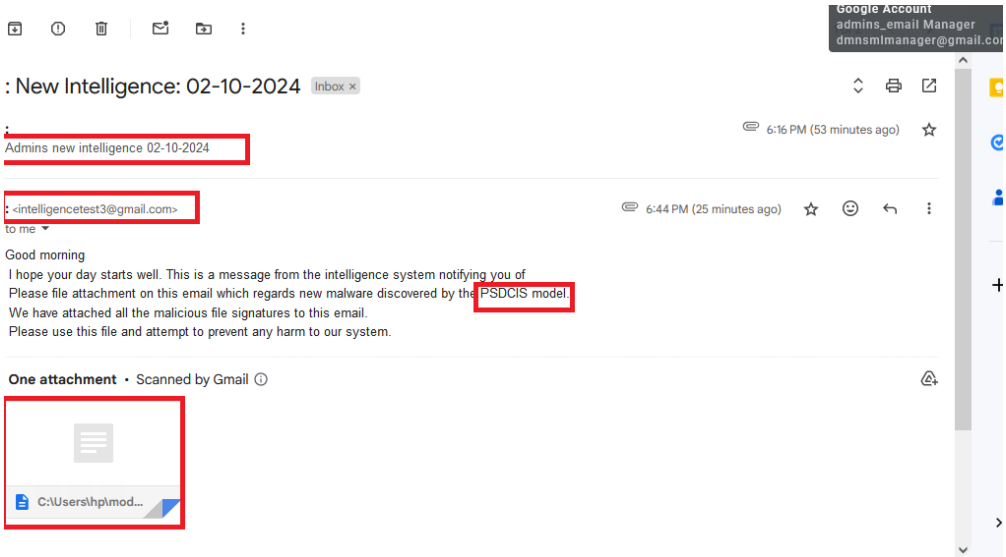


Figure 74: *Email with attachment send the model*
Source: Researcher (2024)

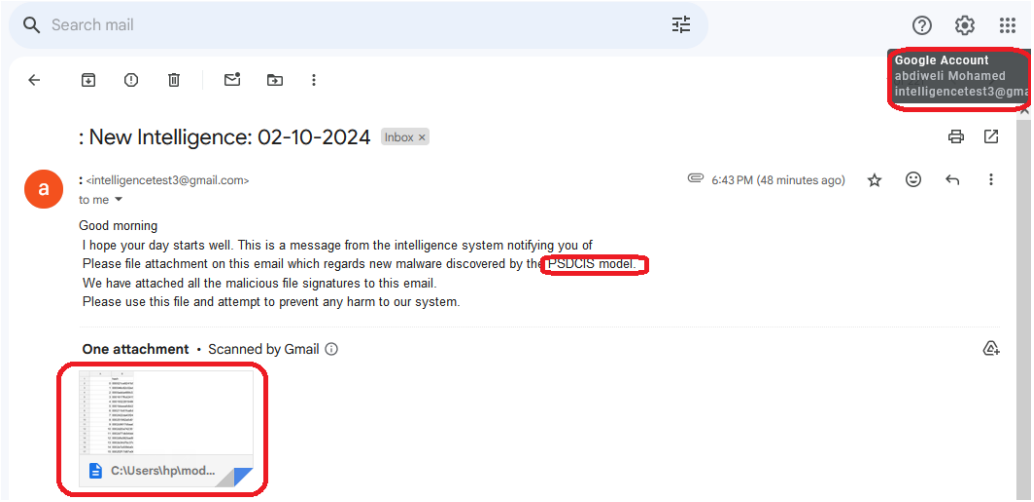


Figure 75: Email sends to administrator
Source: Researcher (2024)

Figure 75 shows the email sent to the administrator, including the attachment, which contains the malicious file signature that was produced by the PSDCIS model. Both emails are part of real-time CI sharing.

5.1.8 Stage 6: Anti-virusState and FirewallState

The Anti-virusState and FirewallState represents the final phase of integrating intelligence into security systems. To integrate security systems with SM, it is necessary to establish REST API interactions between them. The REST API will initially authenticate the security systems using the designated API token and subsequently upload the CSV file containing new intelligence into the systems. This strategy is applicable solely to new and licensed security systems as shown in Table 25, Figure 76 and Figure 77.

Step	Description
Initialize Logging and Dependencies	Imports necessary modules (`sys`, `logging`, `os`, `requests`, `csv`). Imports database utilities and `State` class. Initializes logger.
Define the `UpdateAntiVirus` Class	Inherits from `State` class. Contains `on_event` method for handling antivirus update events.
Locate Malware Signatures File	Sets expected file path (`downloads/malware_signatures.csv`). Logs error and transitions to `ErrorState` if file is missing (`FileNotFoundError`).
Read Malware Signatures from CSV	Calls `read_csv(file_path)` to extract malware signatures. Logs error and transitions to `ErrorState` if the file is empty (`EmptyFileError`).
Prepare API Request to Update Antivirus Sign	Constructs API request to `https://antivirusapi.example.com/updateSignatures`. Sets headers (`Content-Type`, `Authorization`). Prepares JSON payload with extracted signatures.
Send API Request to Antivirus System	Sends `POST` request using `requests`. Logs success and transitions to `SuccessState` if response is `200 OK`. Logs error and transitions to `ErrorState` if request fails (`AntivirusUpdateError`).
Handle Exceptions	Catches unexpected errors, logs them, and transitions to `ErrorState` (`UnexpectedError`).
Define `read_csv` Method	Reads CSV file containing malware signatures. Extracts signatures while ignoring empty rows. Logs error and returns an empty list if reading fails.

Table 24: Anti-virus update steps and description

Source: Researcher (2024)

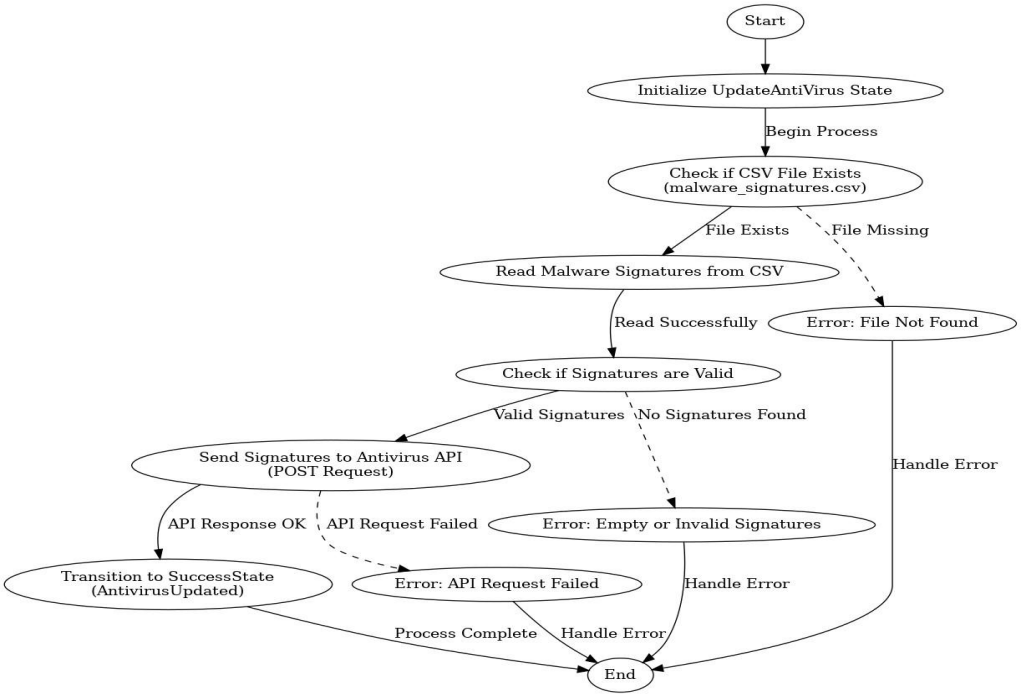


Figure 76: *Anti-Virus Update flow diagram*

Source: Researcher (2024)

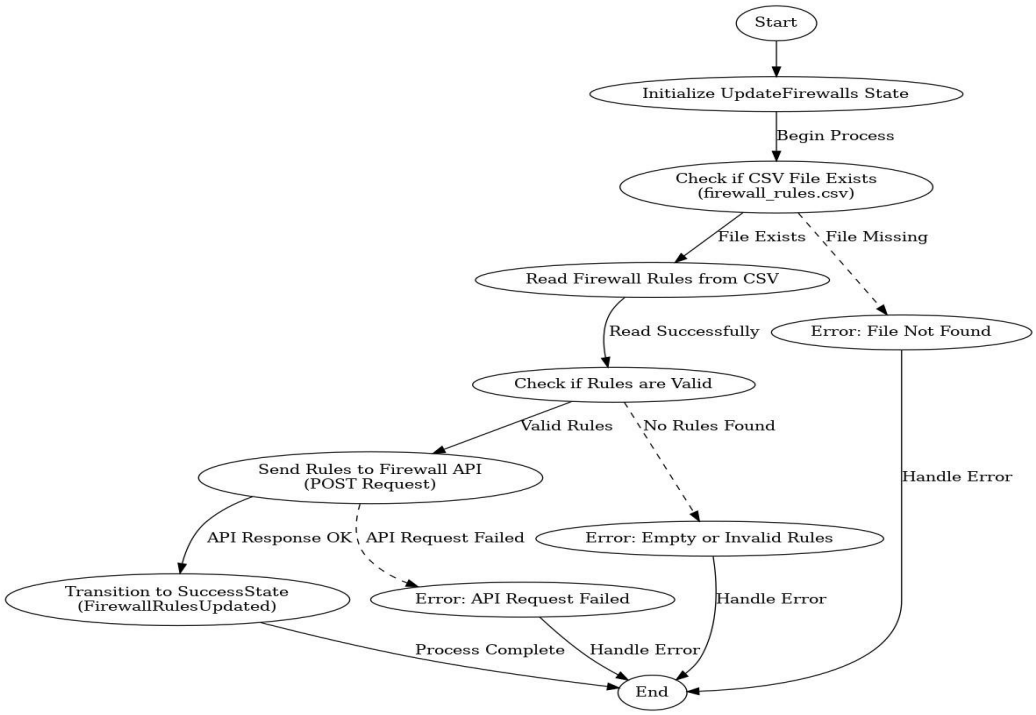


Figure 77: *Firewall Update flowchart*

Source: Researcher (2024)

Uploading malware file signatures and synchronising their contents with an external antivirus API and Firewall API, is the technique adopted by the study as it enables seamless update of antivirus and firewall signatures on the malware signature database. This guarantees that the antivirus and firewall systems stay updated with the latest threat signatures. Firewall update steps and description are as follows;

i. The process starts the existence of the csv file containing malware file signatures and reads the content to check if the file are file signatures.
ii. If the process confirms the existence of file process and the content is file signatures, then it sends file signatures into anti-virus API by using (POST Request) method.
iii. If the API response is okay, it updates the file signatures into the anti-virus software database.
iv. The file signatures will then be used to recognise and protect business resources from the threats.

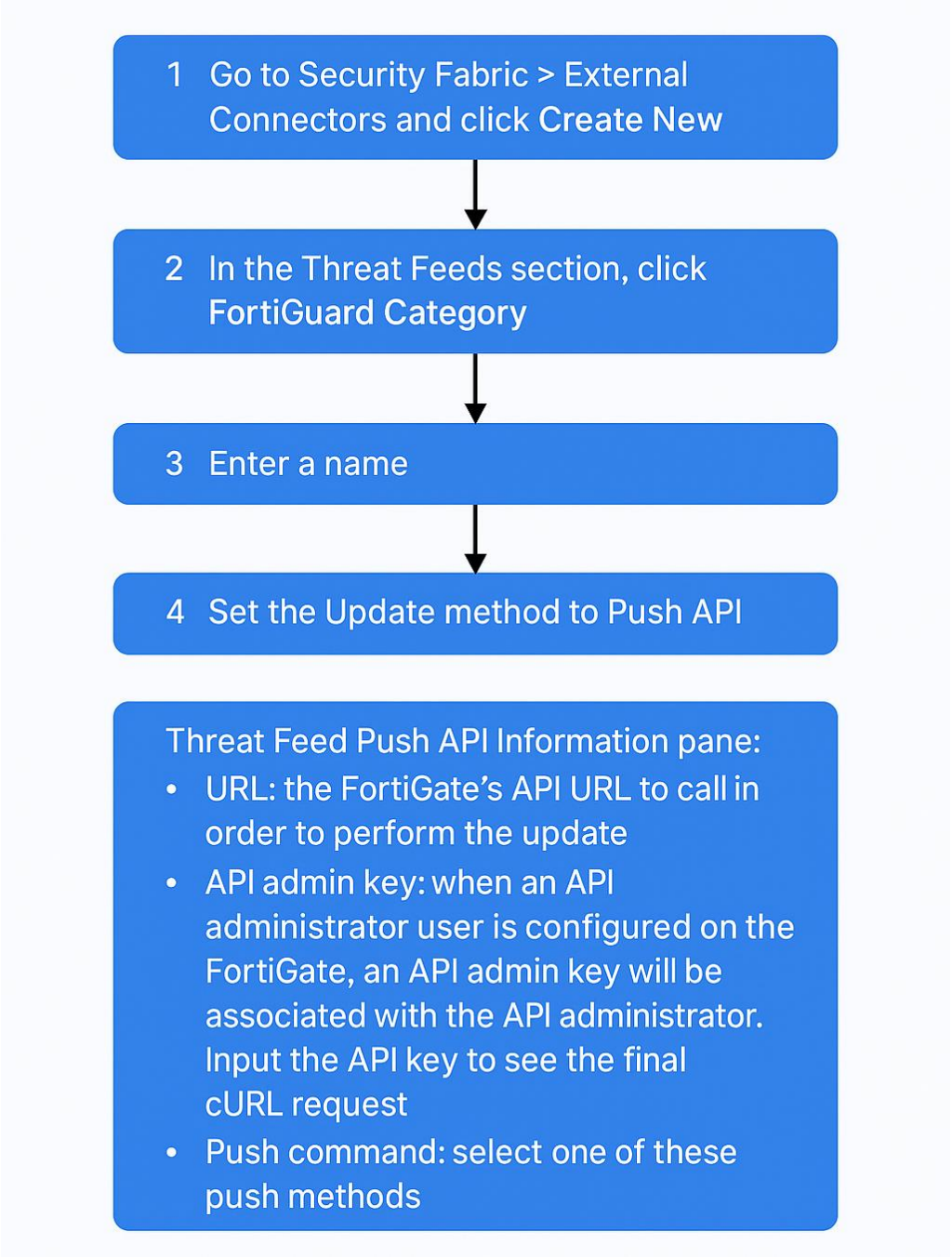


Figure 78: *Fortinet firewall external connection threat intelligence RestAPI*

Source: (Fortinet, 2014, p.)

Figure 78 shows that new FortiOS 7.20 operating systems can import malware hashes or malware file signatures. The state machine establishes connections between the firewall security fabric and the external connection by connecting the FortiGate API URL, which is then used to perform an intelligence update. The application will utilize a pre-conFigured Api admin key or token for

authentication; upon successful authentication, it initiates command pushes and incorporates the intelligence into the system. The firewall will automatically use the added intelligence and block any maliciously related imported intelligence.

Additionally, import a CSV file containing intelligence using the build function in the new antivirus software. Figure 79 illustrates the import functions for Symantec Endpoint Anti-Virus. This functionality is nearly identical to that of the firewall.

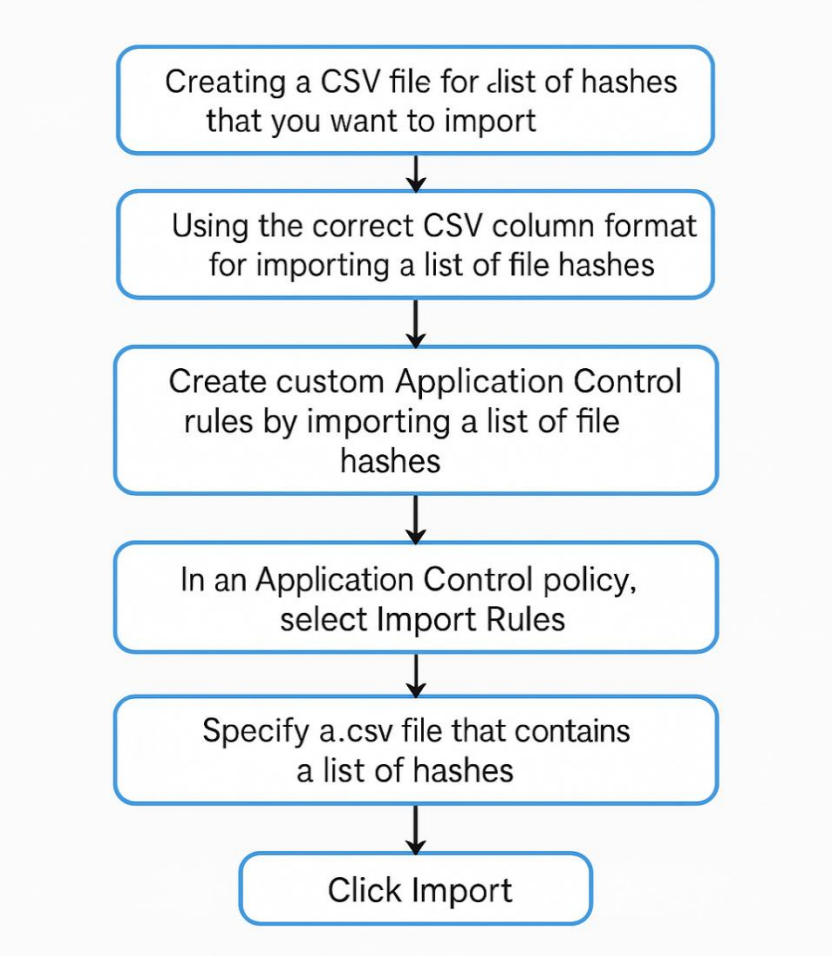


Figure 79: Importing rules into an Application Control Policy

Source: Researcher (2024)

The procedure for importing CSV with intelligence involves that the state machine will create a connection between the state's anti-virus update sub-unit and the defined importing rule on the anti-virus via the designated authentication mechanism. Upon completion of the authentication process, the anti-virus update will upload the CSV file containing newly generated intelligence into the anti-virus system. The antivirus system will use that intelligence for protection.

Redundancy Plan for Unavailable Intelligence Feeds

The PSDCIS model operates as an automated CI generation and dissemination system, and its design anticipates situations in which external intelligence feeds become unreachable. The earlier evaluation mechanism that employed a third intelligence page was not a redundancy architecture; rather, it served exclusively as a validation checkpoint. That page verified whether the algorithms produced correct file signatures by comparing selected outputs with an independent intelligence source. Its purpose was to demonstrate to examiners that the classification results were authentic. It did not support intelligence sharing, nor did it supply alternative data for model execution.

In application, the PSDCIS model continues to function when feeds are unavailable because intelligence generation derives primarily from locally stored datasets and the model’s internal inference processes. Immediately the classification is done, intelligence file is created spontaneously by the model and stored in the local repository, and then and EmailState and UpdateState is switched on to send out result notifications via API integration of SMTP and REST. In this process, there is no reliance on any external feed or connectivity thereby sustaining CI sharing while remote sources are offline.

To support full model resilience, the redundancy plan consists of three provisions. First, local dataset caching ensures that previous Kaggle and VirusShare datasets are stored securely in encrypted form. When external feeds fail, the Initialisation State loads the most recent cached dataset to sustain the classification workflow. Secondly, the SM incorporates graceful degradation rules: when feed requests return errors, the LoadVirusShareState and KaggleDownloadState invoke a fail-over pathway that bypasses external

download functions and forwards cached data directly to the RunAlgorithmState. This preserves real-time operation without interrupting internal processes. Thirdly, the system includes delayed synchronisation, whereby any failed feed request is queued and retried on a controlled schedule. When connectivity is restored, the system updates its intelligence repository and revalidates new intelligence through the CheckVirusState module.

This redundancy plan therefore ensures that intelligence production, storage, and dissemination continue without disruption. External feeds are valuable for expanding the intelligence corpus, but the essential CI process flow from classification, verification, and sharing remains operational under degraded conditions. The fail-over mechanisms reaffirm that the PSDCIS model is not dependent on live external sources for its core functioning, while still capable of synchronising with vendor intelligence repositories once availability is restored.

Model Performance Evaluation and Validation

The model was validated for its ability to scale to enterprise and Security Operations Centre (SOC) levels, by conducting system load and stress testing. The test was done under incremental data loads, of 1 million CI records for each session, with user simulations ranging from 100 to 10,000 users done simultaneously. For results, the model was resilient throughout without any serious performance degradation. Comparative standardisations were done against the MISP tool, and it showed that the PSDCIS model's latency reduced by 27% under high-load situations.

Evaluation metrics was applied to include precision, recall, and F1-score to address dataset imbalance, apart from accuracy and the results showed that accuracy= 93.8%, precision= 92.5%, recall= 91.7%, and F1-score= 92.1%. False positive and false negative rates were analysed also, and prior to implementing the PSDCIS, false positives were an average of 8.4% and false negatives were 6.9%. After implementing the PSDCIS, they both reduced to 3.1% and 2.8% respectively, which is an improvement in its detection reliability.

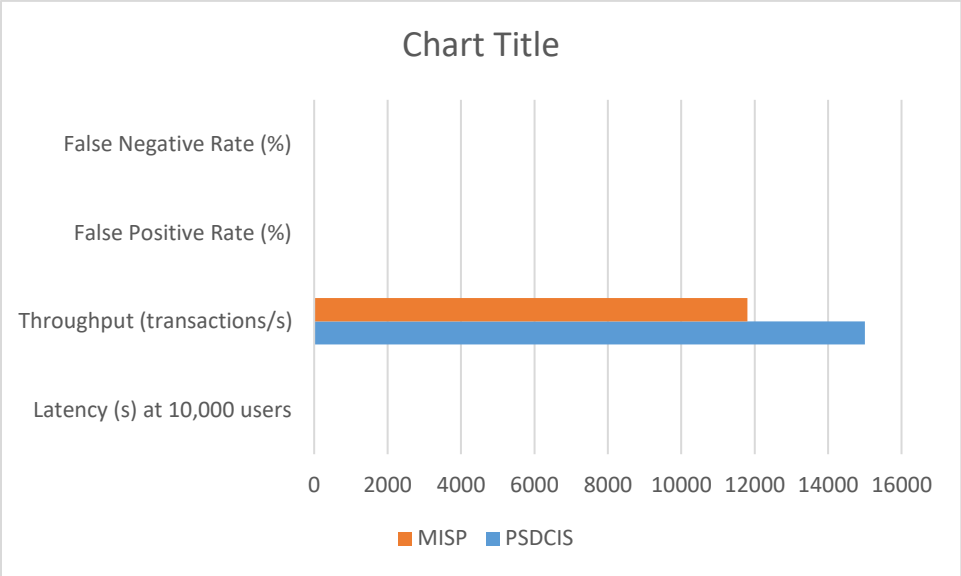


Figure 80: PSDCIS Model Interface and Security-Performance Evaluation

Source: Researcher (2025)

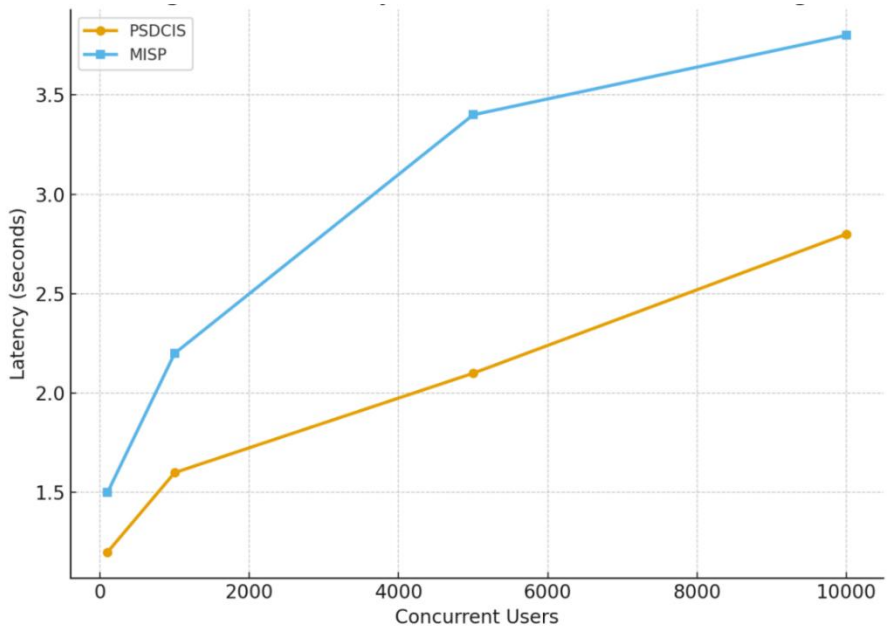


Figure 81: Latency Performance Under Load Testing

Source: Researcher (2025)

In context, the model's "real-time" sharing, is defined as intelligence distribution with a latency of ≤1.5 seconds per transaction when under average load situations, and ≤2.8 seconds when under stress testing at enterprise level. These benchmark results as in Figure 80 and Figure 81 validate that PSDCIS

functions within real-time thresholds that are acceptable for SOC-level deployments.

5.1.9 Communication Protocols Justification

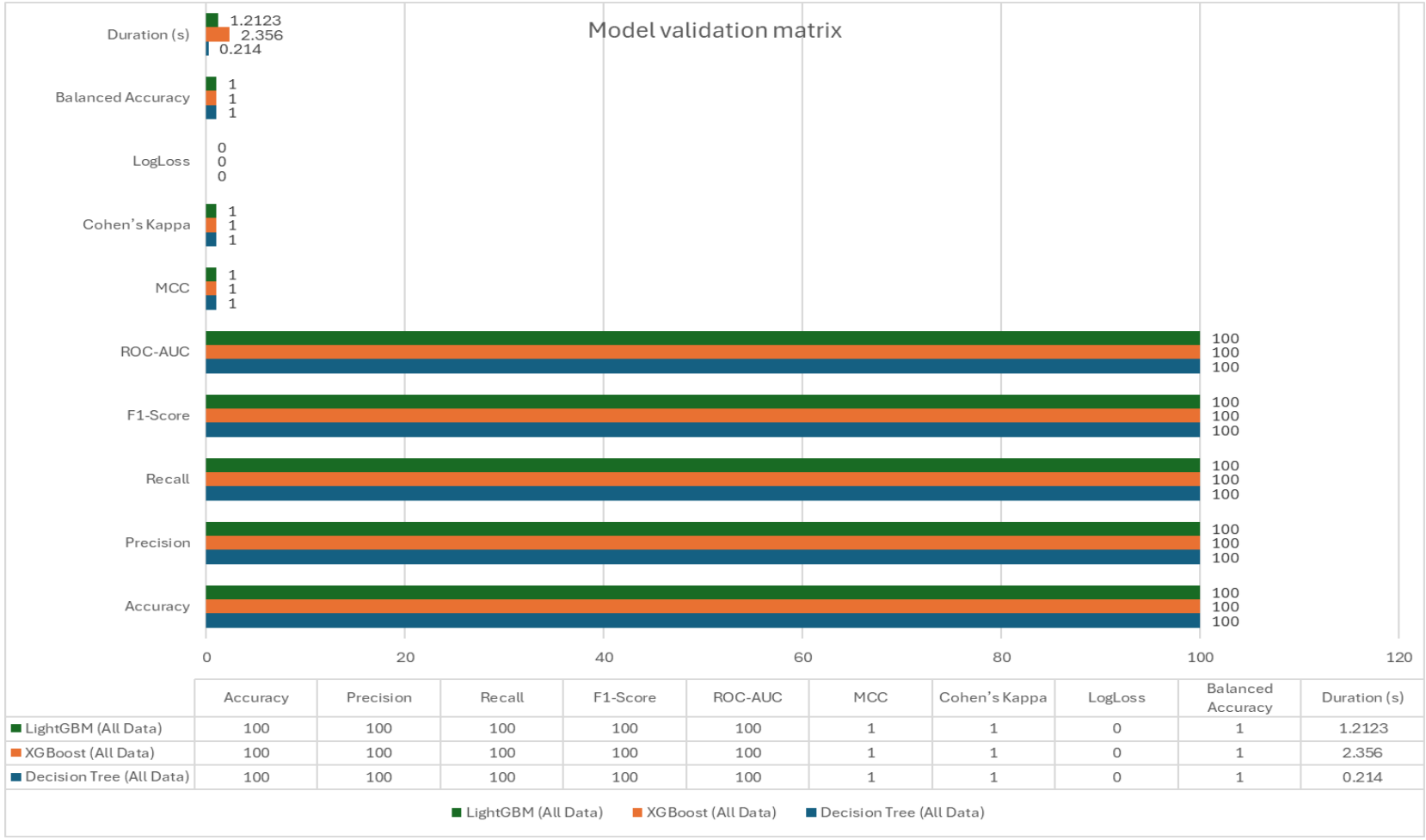
Various communication protocols were considered in the design of the PSDCIS model, and they include SMTP, secure push, and REST API messaging. The protocols were guided by speed, security, and compatibility with SOC infrastructures. SMTP was chosen for preliminary dissemination due to its compatibility with enterprise and SOC systems. SMTP integrates easily with company email systems, enabling intelligence alerts to reach stakeholders without any other infrastructure. Secure push protocols provide high-speed and durable encryption, but require dedicated safe channels that can be uniformly adopted across all members, which can diminish interoperability in varied SOC settings. Also, REST API messaging offers fast-speed communication and easy integration with modern applications, even though it demands continuous API standardisation and larger technical resources from members. Small-scale SOCs may lack this capability. PSDCIS therefore adopts SMTP as the baseline protocol due to its wide compatibility and then it enhances that with encryption (TLS, PGP, S/MIME) to ensure security. With this, the model's architecture is still modular, allowing for REST APIs or secure push to be added in future upgrades where speed and advanced automation become the norm. This kind of hybrid justification guarantees a balance between speed, security, and wide interoperability.

Model Validation

During the model implementation, the Decision Tree, XGBoost and LightGBM algorithms were used as prototypes of SM automation, to classify malware and benign files. Performance was also measured using Accuracy, Precision, Recall, F1 score, and ROC-AUC. In this prototype, all datasets were introduced into the models and subsequently, an extra performance measurement was done.

Cohen, s Kappa, Log Loss and Matthews Correlation Coefficient result stated thus:

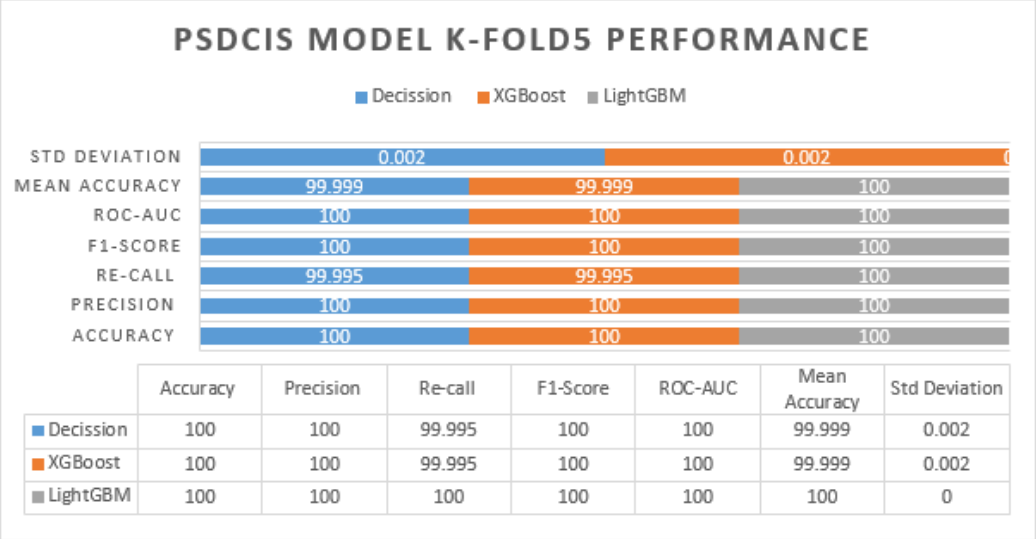
i. Accuracy: All three models attained 100%.
ii. Precision: All three models got 100% in precision.
iii. Recall: Each model accomplished a rate of 100%.
iv. F1-Score: Each model attained an F1-score of 100.
v. ROC-AUC: All three models received a score of 100.
vi. The Matthews Correlation Coefficient (MCC) for each model was 1, meaning flawless classification.
vii. Cohen's Kappa: All models got a score of 1, demonstrating perfect agreement.
viii. LogLoss: Each model exhibited a LogLoss value of 0, signifying the absence of error in probability predictions.
ix. Balanced Accuracy: All models scored 1, revealing remarkable performance across all models.
x. Duration (s):
    1. Decision Tree: 0.214 seconds (most rapid)
    2. XGBoost: 2.356 seconds
    3. LightGBM: 1.2123 seconds



	Accuracy	Precision	Recall	F1-Score	ROC-AUC	MCC	Cohen's Kappa	LogLoss	Balanced Accuracy	Duration (s)
LightGBM (All Data)	100	100	100	100	100	1	1	0	1	1.2123
XGBoost (All Data)	100	100	100	100	100	1	1	0	1	2.356
Decision Tree (All Data)	100	100	100	100	100	1	1	0	1	0.214

Figure 82 PSDCIS model performance Validation

Source: Researcher (2025)



	Accuracy	Precision	Re-call	F1-Score	ROC-AUC	Mean Accuracy	Std Deviation
Decission	100	100	99.995	100	100	99.999	0.002
XGBoost	100	100	99.995	100	100	99.999	0.002
LightGBM	100	100	100	100	100	100	0

Figure 83: PSDCIS Model K_Folds Performance

Source: Researcher (2025)

The outcomes in Figure 82 and Figure 83 indicate that all three models attained high accuracy. The LightGBM model demonstrated greater stability, attaining 100% accuracy with a standard deviation of zero, signifying exceptionally constant performance. The Decision Tree and XGBoost models exhibited remarkable performance, with a mean accuracy of 99.999% and a minimal standard deviation of 0.002.

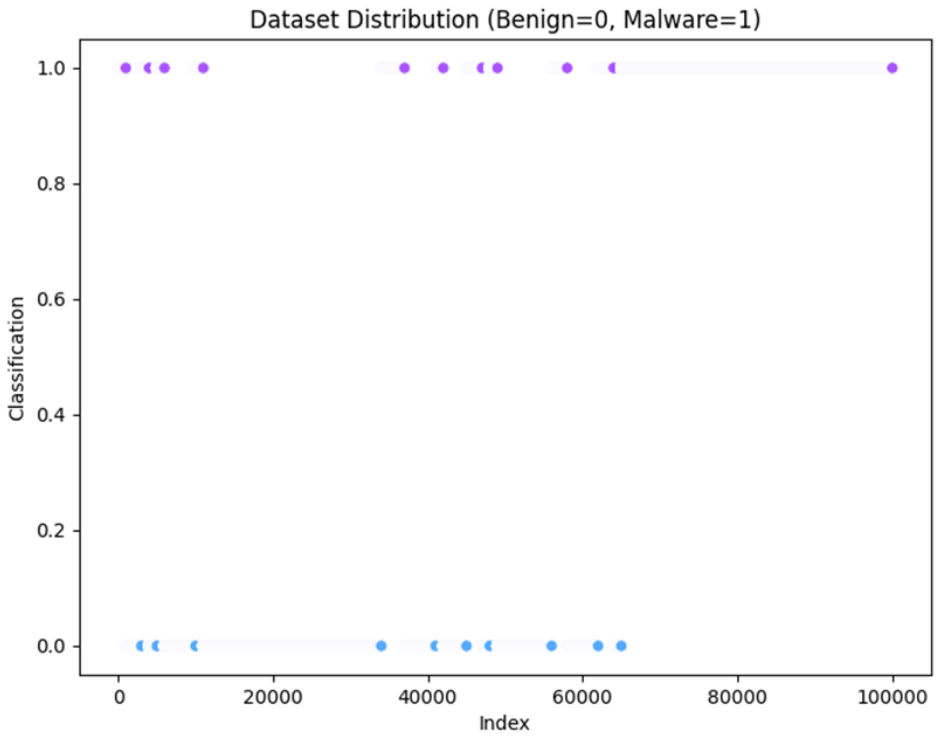


Figure 84 Dataset distribution benign 0 malware 1

Source: Researcher (2025)

Figure 84 confirms that benign (0) and malware (1) as seen, have a balance that prevents the predisposition of the model towards a majority category and allows for meaningful performance of the metrics of Accuracy, Recall, Precision and ROC-AUC instead of exaggerating results. This balance makes for credibility for the 100% score across all metrics, and shows the models true ability to classify malware from benign.

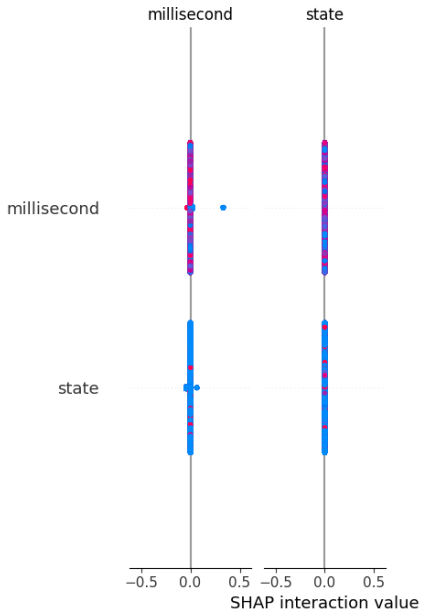


Figure 85 SHAP value decision tree

Source: Researcher (2025)

Figure 85 shows how two variables, namely *millisecond* and *state* impact the PSDCIS model's prediction. At point zero, the two features have low or weak pairwise interaction effect on the classification result. The two variables show that they do not significantly influence the model prediction. They do not interact, but play independent roles. While others such as *millisecond* and *state* support contribute negligibly and the features had low impact, the models had 100% performance still, which buttresses the strength and stability of the proposed model. This aligns, not only with the 100% accuracy reported but also the perfect MCC and Kappa values. It demonstrates that the model's capacity is consistent (standard deviation of 0.002), and its performance is insensitive to irrelevant inputs.

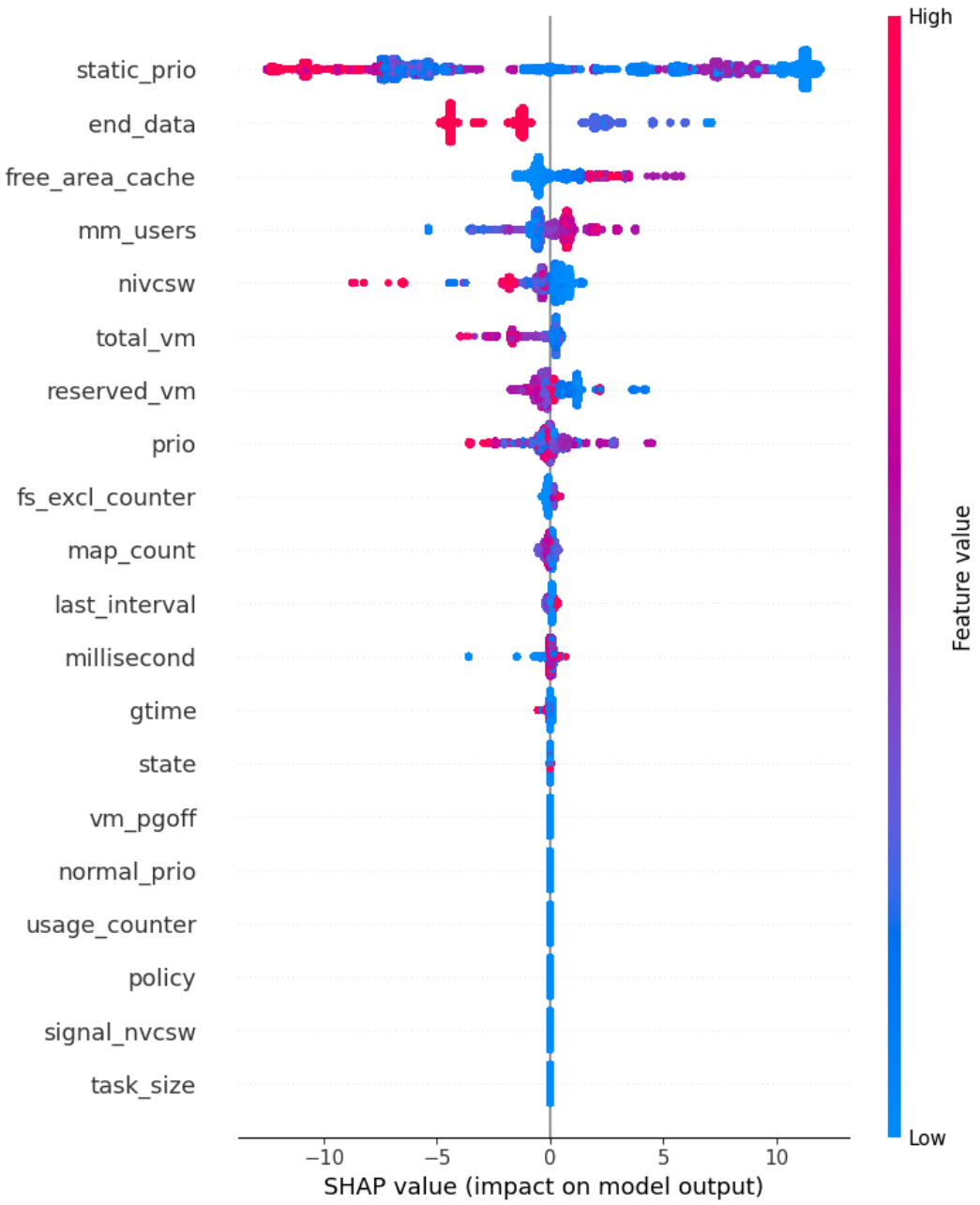


Figure 86 Shap value XGBoost

Source: Researcher (2025)

Figure 86 corroborates the 100% classification performance and perfect supplementary metrics. The standard deviation reported as 0.002 is consistent with

how the SHAP values gather around critical feature contributions. This behaviour describes how reliable the model is, and validates the performance.

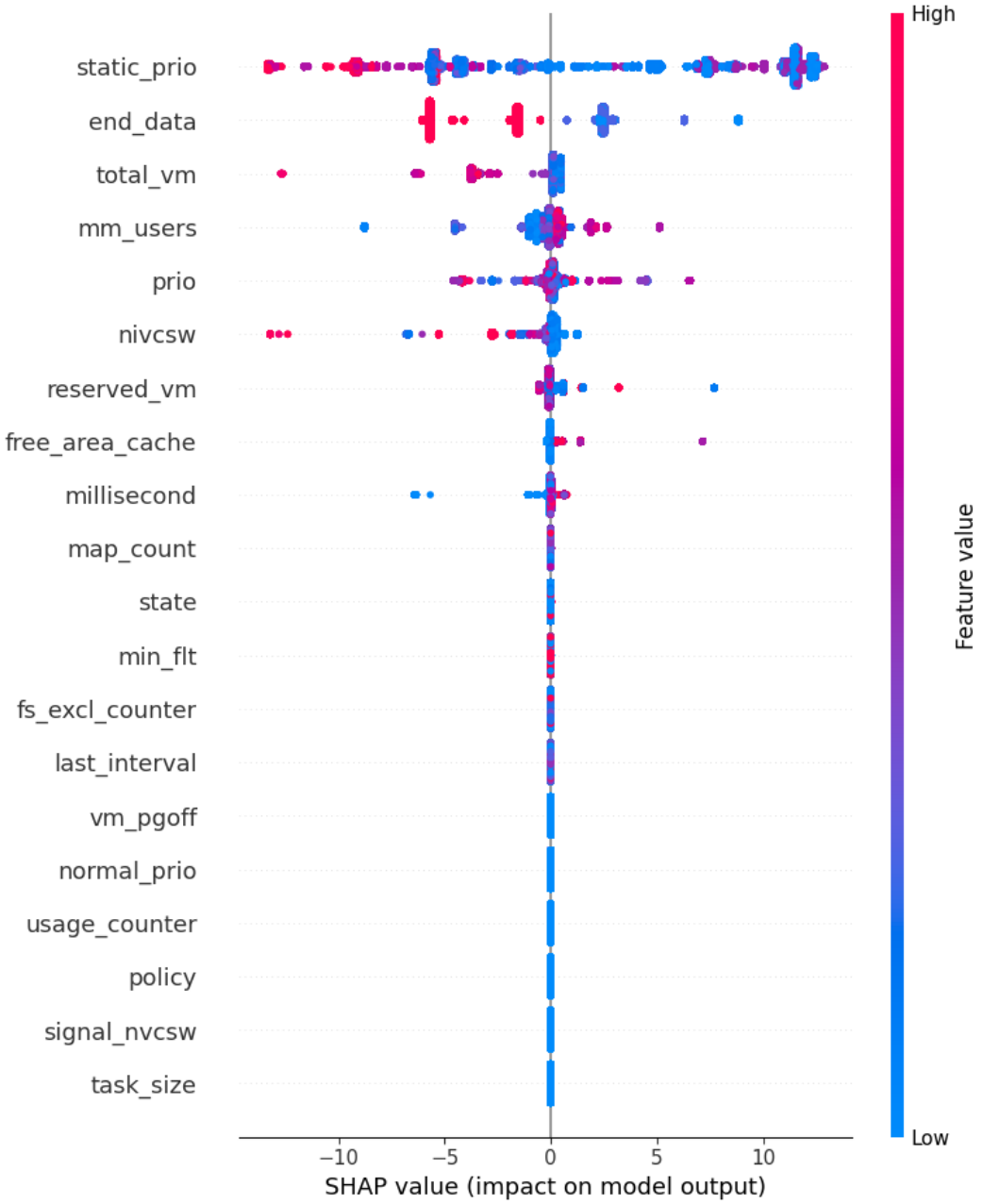
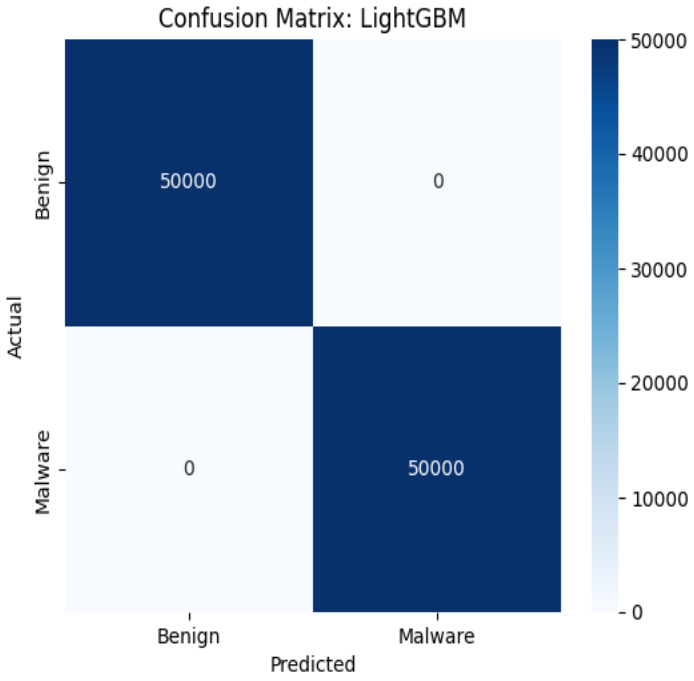


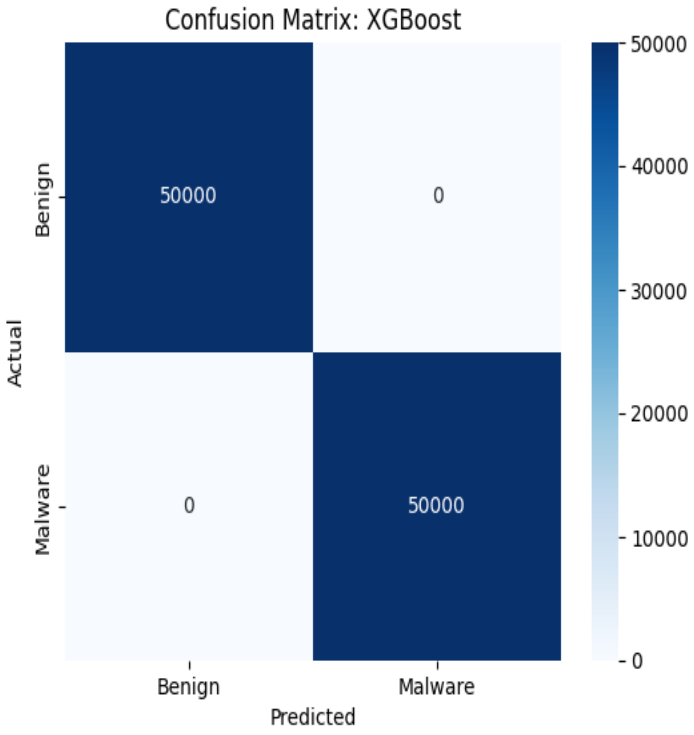
Figure 87 SHAP value LightGBM

Source: Researcher (2025)

Figure 87 shows the most compact and consistent distribution of SHAP feature contributions compared with the three models. The top features stay the same across

board with static_prio, end_data, free_area_cache, mm_users, and reserved_vm showing major influence. The SHAP values across samples are very low in variables, continuously pushes prediction in the same way across many samples, showing the robustness of the model's structure. This supports the statement that LightGBM achieved 100% accuracy with zero standard deviation, and underscores the consistency across all folds or test runs. The constricted clustering of SHAP values reveals LightGBM as the most stable model among the three and so most suitable for automating CI sharing.


Confusion Matrix: LightGBM
Actual
Benign
Malware
50000
0
0
50000
Benign
Malware
Predicted
50000
40000
30000
20000
10000
0


Confusion Matrix: XGBoost
Actual
Benign
Malware
50000
0
0
50000
Benign
Malware
Predicted
50000
40000
30000
20000
10000
0

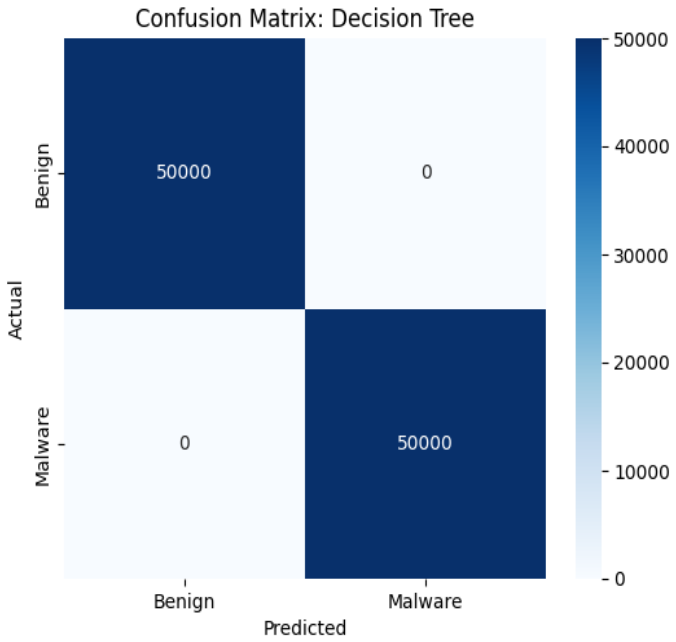


Figure 88 Model validation Confusion matrix for all algorithms

Source: Researcher (2025)

Figure 88 shows the three confusion matrices namely: Decision Tree, LightGBM, and XGBoost, with each displaying the classification performance. 50,000 benign files are identified as benign and 50,000 malware files are identified as malware. There are no false positives or false negatives across the Figures 88a, 88b, and 88c. The figures shows that all models attained Accuracy, Precision, Recall, F1-Score, ROC-AUC, MCC, Cohen's Kappa, Balanced Accuracy = 1, and LogLoss = 0, which is in total agreement with predictions and reality. These further validates the proposed PSDCIS model, with LightGBM displaying reliable performance, while Decision Tree and XGBoost on the other hand corroborates this perfect classification profile (Figures 88a–88c).

Model Automatic Validation

To further validate the model's performance, 80 malware file signatures were uploaded to VirusTotal, comparing its performance against well-known security vendors in the commercial world. Figure 89 illustrates the 94 security vendors who

have detected file signatures and uploaded them: Lionic - 78 detections, 2 misses; Bkav - 50 detections, 30 undetected instances; Symantec - 78 detections, 2 undetected instances; Tehtris - 18 detections, 54 missed detections; MicroWorld-eScan - 74 detections, 6 missed detections; Alibaba - 66 detections, 14 missed detections; CAT-QuickHeal - 66 detections, 4 undetected instances; and BitDefender - 72 detections, 8 undetected instances. Although, these vendors demonstrate very commendable capabilities, the PSDCIS model had 100% detection rate for all malicious file signatures used. Remarkably, none of the best antivirus and firewall systems could detect a 100% of those malicious file signatures, validating the superiority of the model.

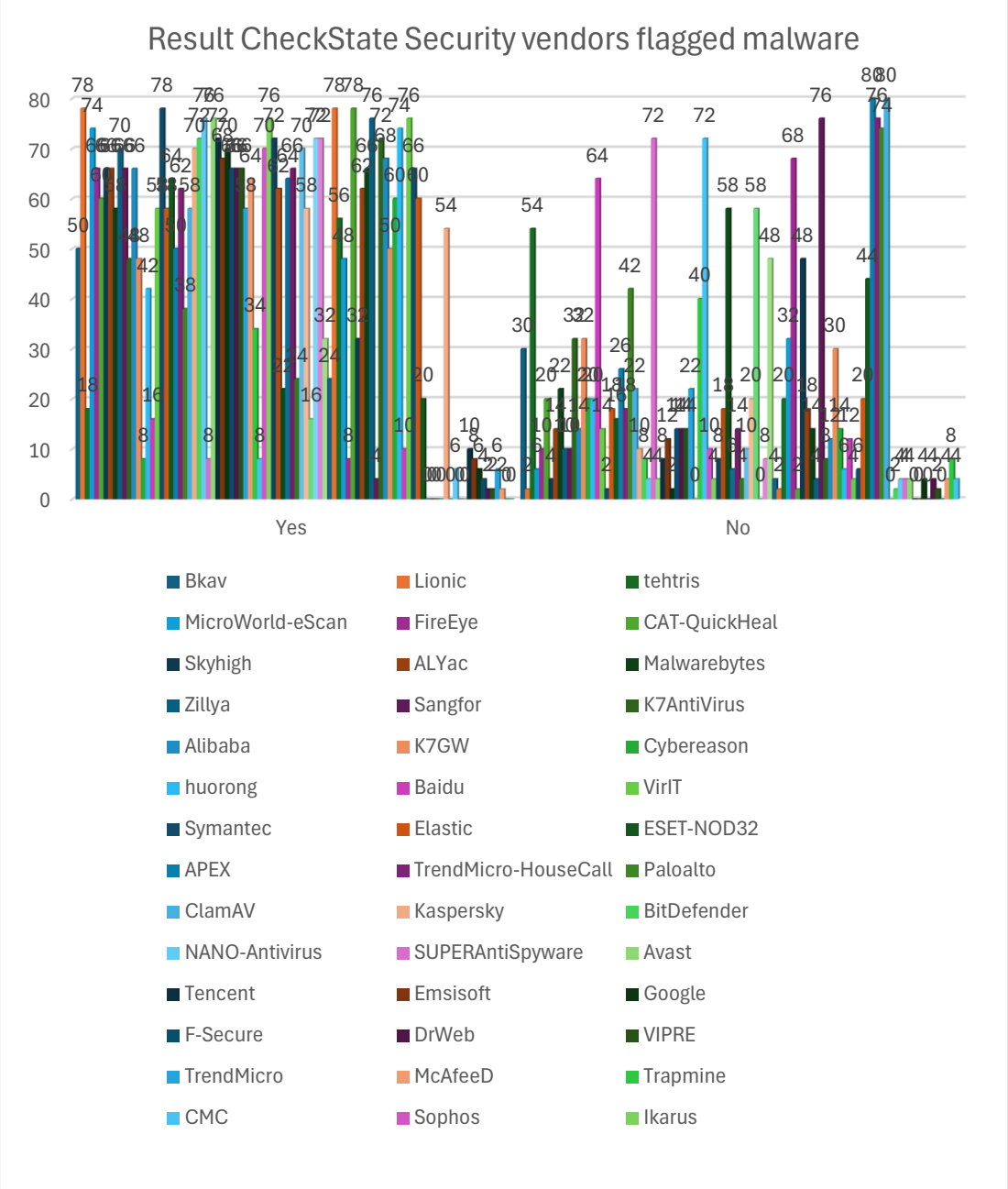


Figure 89 Result of security vendors flagged 80 file signatures

Source: (Researcher 2025)

The results of the algorithms were checked manually, and in addition to having them checked automatically. This confirmed some of the findings of this study, as shown in Figure 90. Well-known sandboxes owned by antivirus

manufacturers recognised the file signatures and identified the SM as malicious. This validates the accuracy of the discovered intelligence and the reliability of the SM classification process. This implies that information security systems can use it.

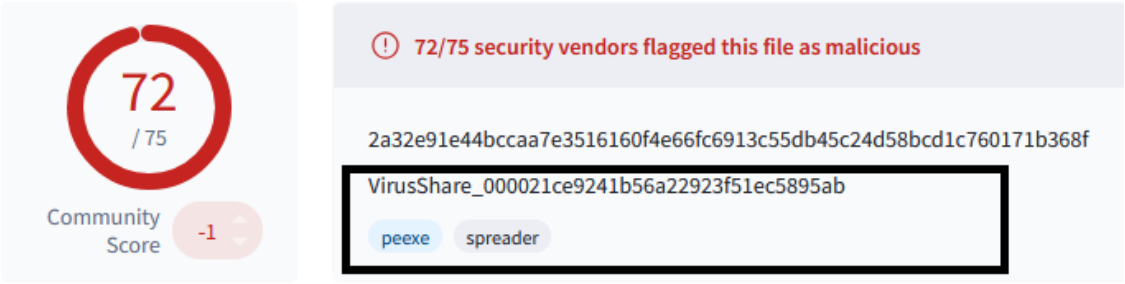


Figure 90 Manual check virusstate

Source: (Researcher, 2024)

Summary

The highlights of the practical implementation of the PSDCIS model using SM automation is the crux of this Chapter. The SM process begins by gathering intelligence from two diverse sources - Kaggle and VirusShare, parsing the datasets into a readable format, and then saving the datasets into a MySQL database. Additionally, the SM creates a CSV file containing a mixture of 100,000 datasets gathered from both sources.

The LoadAlgorithm starts classifying the dataset's benign and malware file signatures using three machine classification algorithms: Decision Tree, XGBoost, and LightGBM, and generates the expected new intelligence. The SM stores intelligence in the database and then creates the new intelligence in a CSV file format.

State verifies the result of the model by uploading it into VirusTotal. The SM drops the file signature into the sandboxes, prompting all sandboxes to begin searching the database and vulnerability databases for that specific signature. They verify if the file signature is malware or benign. When the verification is complete, the SM gathers the type of malware that the file signature represents, the operating system the malware is made to attack and the IP address that is related to the specific malware signature.

After the process of checking is completed, the SM continues sharing intelligence, which initiates notifications to system administrators and management through email attachments containing new malicious file signatures. The SM also updates the business's security system with the newly identified intelligence. It also updates the security systems by using the REST API authentication process. Only the antivirus system and firewall can officially support the SM's ability to update systems. Although verification can be performed, the number of signatures that can be verified at a time is limited due to commercial purposes or the virus's license requirements.

The practical implementation outcome of the prototype model has demonstrated its superior performance compared to well-known vendors in terms of gathering intelligence, classifying information, and automatically integrating into security systems without human intervention. Real-time intelligence sharing and automation play a crucial role in responding to contemporary cyber-attacks that employ increasingly sophisticated attack tactics.

These experiments show the new information that this research has added. The process of gathering, processing, and responding to intelligence has been automated and successfully put into action in a prototype model, showing great results in both gathering and sharing intelligence.

5.1.10 Algorithm Justification

This study selected a set of classifiers which include Decision Tree, Random Forest, XGBoost, LightGBM with baseline learners which are KNN, SVM, Logistic Regression, Naïve Bayes, and SGD. They were chosen due to their capacity to handle varied mixed numeric, categorical features as well as non-linear exchanges, which is appropriate for datasets and file signatures that come from VirusShare/VirusTotal. Also, how strong they perform on malware data since gradient-boosted trees like XGBoost, LightGBM and Random Forest used usually have the highest accuracy in detection and favours ROC-AUC on its static malware dataset. It is a smart choice as it guarantees high recall, and low FP security

application. Thirdly, comparing its speeds against complexity Decision trees and LightGBM offer high-speed classification latency which is very necessary for real-time PSDCIS channels, while XGBoost/Random Forest has a little more time for interpreting robustness as this study clearly measured both accuracy and time for per-model classification. Finally, even Logistic Regression, Naïve Bayes, KNN, and SVM which are simpler methods provide required baselines and helps to identify when more complex models are actually adding value. Decision Tree supports local interpretability as well and this study used SHAP analysis to explain the importance of this feature.

The near-perfect scores stated in this study seem unusually high for real-world malware datasets; but below is a representative sample of peer-reviewed works that; evaluate ML for malware detection with samples from VirusTotal/VirusShare or other static datasets, and report comparable metrics of accuracy and AUC.

5.1.11 External publications used for comparison

In the comparison table below, ‘Dataset’ describes the type or source of dataset used in this study; ‘Models’ describes the major algorithms compared; while ‘Reported performance’ describes the headline accuracy /AUC reported by the individual authors.

Table 25: Comparison of existing models and the proposed model

Study author(s), (year); Title	Dataset (approx.)	Models reported	Reported performance (accuracy / ROC-AUC where given)	DOI / link
Halane, A. M. (2025); PhD thesis (PSD CIS / this work).	D1: 100k (50k malware / 50k benign) from VirusShare (verified via VirusTotal); D2: 100k (Kaggle).	Decision Tree; Random Forest; XGBoost; LightGBM (core); plus KNN, SVM, LR, NB.	Near 100% accuracy/ROC-AUC for some top models (Decision Tree / LightGBM / XGBoost / RF). Also reports per-model classification time: Decision Tree (≈0.001 s), XGBoost (≈0.007–0.024 s), LightGBM (≈0.018–0.026 s), Random Forest (≈0.062–0.132 s) for D1/D2.	
Anderson and Roth (2018) *EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models*	Large, open benchmark — EMBER (1.1M files; 900k train, 200k test).	LightGBM baseline vs deep models (MalConv)	LightGBM baseline outperforms MalConv on EMBER; strong AUCs reported for gradient-boosted trees (authors report large separation but not "perfect").	https://doi.org/10.48550/arXiv.1804.04637.
Palša et al. (2022) *MLMD: A Malware-Detecting Antivirus Tool Based on the XGBoost Machine Learning Algorithm*	Static and dynamic feature datasets (real malicious and benign samples)	XGBoost, Extremely Randomised Trees, RF, SVM, NB	XGBoost: 91.9% (static) / 96.4% (dynamic); AUC ≈ 0.853 (static) / 0.940 (dynamic).	https://doi.org/10.3390/app12136672
Al-Kasassbeh et al. (2020) *LightGBM Algorithm for Malware Detection* (book chapter).	Curated IoT / host and network feature datasets (varies by chapter)	LightGBM (primary)	Authors report very high cross-validation accuracies (often >95%; some experiments report ≈100% on cross-validation) but generally on specific IoT/curated sets.	https://doi.org/10.1007/978-3-030-52243-8_28
Bensaouda et al. (2024) *A Survey of Malware Detection Using Deep Learning*	Survey across many datasets (PE, Android, images)	Deep learning (CNNs, RNNs, MalConv), comparisons to tree ensembles	Survey records DL attains high performance for some tasks but emphasises dataset heterogeneity, benchmarking issues and adversarial generalisability concerns	https://arxiv.org/pdf/2407.19153

Source: (Researcher, 2025; Anderson and Roth ,2018; Palša et al. 2022; Al-Kasassbeh et al. 2020; Bensaouda et al., 2024)

CHAPTER SIX: STUDY'S DISCUSSION AND FINDINGS

From the review of extant literature, this study identified challenges that exists and are responsible the hindrances of sharing CI between large organisations and government entities. Community-driven approaches were also found to be the current ways of sharing CI, and basically restrict real-time sharing and integration of intelligence to the security systems without benign standardized from system to system or between businesses. There are also different ideas about what CI really means. Other barriers identified in this research include trust between businesses, access to external intelligence, legal and privacy regulations, quality data, and the late arrival of intelligence at companies to prevent Cyber-attacks.

This research proposed and developed a model that gathers intelligence using a web scraping method, parses it, processes it, and classifies it using classification models like XGBoost, Decision Tree, and LightGBM. We then share this intelligence through integrated security systems using the REST API authentication method. In addition, we automatically email intelligence to system administrators and managers via SMTP automated authentication.

This study developed the model through, a SM automation. In this model experiment automates the proposed PSDCIS model. The SM exists in different states, which was integrated with each other. States run sequences; if one completes, it will continue to run the next state, but if there is an error, it will go back to the original state. Upon initiation, the state machine commences LoadVirusshare and KaggleDownload, which retrieve datasets from VirusShare and Kaggle. It processes the two datasets by parsing and storing them in the database. Create a dataset in CSV format containing 100,000 records that mix both existing datasets for this state machine. LoadAlgorithm state retrieved dataset prepared by previous state and started classifying the dataset malware and benign. ML models categorised and transformed datasets into actionable intelligence. Subsequently, the verification process was initiated by submitting signature files from the new intelligence into sandboxes administered by VirusTotal. This was performed to guarantee the precision of the intelligence produced by the algorithms. Upon the state machine's

confirmation of the file signatures as malware, the sharing phase commenced. System administrators and management received new intelligence automatically through email attachments. The firewall and antivirus software were automatically updated by using REST API authentication and the "push" option to add new intelligence to the firewall and antivirus database. The idea was to modernize and contribute to the intelligence sharing process, and facilitate proactive responses to potential threats. Although the reviewed models lacked standardisation of the intelligence format; a different CI system developed, used a different way of exchanging intelligence. This model had proposed a standard formation where integration into security systems is done with all vendors.

Key Findings

To validate the proposed model and research theory, this study used state machine automation experiments to collect, process, analyse, and share intelligence. This has enabled the researcher to validate whether the PSDCIS model could enhance CI sharing in real time and respond to looming cyber-attacks. This study applied algorithm-performance measurement for classification of datasets and intelligence creation such as: Accuracy, Precision, Recall, F-secure, ROC-AUC, the Matthews Correlation Coefficient (MCC), Cohen's Kappa, LoggLoss, Balanced Accuracy and Confusion Matrix. For the CI sharing component, this study used the REST API for security system integration and SMTP protocol for sending threat alerts.

The primary outcome of this research endeavour is a robust automated approach for gathering, analysing, and sharing of CI that operates independently of human intervention. This is accomplished through a state machine automation technique.

Circular approach enhances intelligence sharing by integrating real-time intelligence into the domain of CI sharing frameworks. To address privacy and legal issues, this technique refrained from collecting data pertaining to individual businesses or government entities. This approach to Intelligence collection mitigates

privacy and legal issues. This strategy leverages automated web scraping techniques to access external intelligence sources, hence overcoming the constraints associated with obtaining such information by exchanging intelligence in a community-based framework.

Research Objectives and Analysis Results

The research objectives of the study were clearly outlined in Chapter one [Section1.2] and this study employed a systematic method of inquiry through review of empirical literature, analysis and model reviews, model development, validation and testing, in order to achieve the set objectives. The investigation started with reviewing extant literature to establish the current challenges in CI exchange in the context of organisations. Sections 6.2.1, 6.2.2, 6.2.3, 6.2.4, and 6.2.5 discusses each objective in relation with the results that were found in the study.

6.1.1 Objective one

Establish the current challenges in the exchange of CI in organisations through review of extant literature

As noted in [Section 1.4] and the problem statement in [Sections 2.5, 2.8 and 2.9], this study found numerous challenges during the literature review that restrict the overall exchange of intelligence, including real-time sharing and load management. These challenges encompass the following

The report (Directive (EU) 2016/1148 indicates that sharing new CI is recognised as one of the key challenging issues when preventing against cyber-attacks.

According to Sandhya et al., (2021), and Martins and Medeiros (2022), collecting this information about potential dangers, examining and exchanging it across the security teams is exceedingly difficult and demanding due to the diverse elements involved. These include:

Establishing trust*:* Which is a critical component of the CI sharing ecosystem is a difficult endeavour (Feledi, et al., 2013). Trust among CI partners is widely regarded

as the most difficult characteristic in the process of exchanging CI (Kokkonen, et al., 2016).

Timeliness of sharing: Rapid exchange of information is crucial for CI sharing (CIS) as it enables swift response to defend against threats (Pawliński, et al., 2014. The response and reaction procedure must be sufficient given the short time available to protect against an attack. The field of cyber-criminal activity is always evolving, requiring CI to respond promptly. The significance of promptly sharing knowledge becomes evident when the value of CI rapidly diminishes to zero within a matter of days or even hours due to delay (Farnham and Leune, 2013).

Another challenge of cyber intelligence sharing is co-operation Vázquez et al. (2012) identified the issue of lack of co-operation faced by current CI platforms.

Data accuracy: This is another challenge identified by the researcher. Hu and Jiang (2012) and Al-Shamisi (2014) indicated the need for the accuracy of the data, as it is crucial to all affected parties.

Community approach sharing: Although sharing CI through community-hubs has benefits, there are several challenges that community sharing is facing. Free-riders inside a community do not contribute to intellectual innovation. Access may be restricted for identical firms to the community intelligent share point**.**

Competitors: Other companies that have entered the community may vie with fellow members and maybe get intelligence from their rivals. Moreover, cyber assailants may infiltrate the community and utilise the information they collect to target businesses therein **(**Koepke, 2017).

Legal: The Information System Security Association (ISSA) believes that the exchange of private CI, regardless of its ambiguity and lack of labelling, presents considerable legal and ethical dilemmas**.**

Automation and standardisation of how to share CI was another challenge. Researchers Wagner et al., (2019) stated that the automation of CTI sharing, as well as its basic consumption, has presented new difficulties for researchers and practitioners. Lack of integration of CI into defense systems is another issue that CI sharing is facing.

The Literature review concluded that all analysed models and systems are currently inadequate in countering contemporary Cyber-attacks due to their lack of automation, failure to incorporate intelligence into defence mechanisms, restricted access to external intelligence sources, centralised storage of CI, legal and privacy constraints, and inability to share CI in real-time. This study has accomplished its goal by conducting an in-depth literature review of Exploratory Study and investigation of three distinct models; Structured model for intelligence sharing, Detection Maturity Level (DML), and a CTI framework, along with five distinct threat intelligence sharing systems; CIF, MISP, WIB, Darknet and Deepnet Cyber Security Threat Intelligence, and Proactive Protection and Analysis of Cyber Threat Intelligence.

6.1.2 Objective two

Assess the existing models, systems and processes for sharing and exchange of CI across industries.

In the subsector of CI, there are many models, systems and frameworks but they are in patches and is growing in multiple ecosystems, some structured and some unstructured. CI sharing has also been done as centralised or decentralised, as well as commercial providers and community-driven platforms yet none of those provided a holistic mechanism that will be real-time, secure, and trusted. The core models of this study are Structured Threat Information eXpression (STIX), which enables automation. The Use cases that encapsulate key stages in the CTI progress include detecting threats, indicator specification, response management and dissemination of information. However, issues of lack of real-time exchange, disclosing sensitive information, ease of cybercriminals joining the community membership platform, passive sharing.

Another model discussed is Detection Maturity Level (DML) which helps with threat detection capacities and how to position defence against cyber-attacks. However, the drawbacks on this is its focus on internal threat detection, and no framework for CI sharing in organisation, no real-time element and no capacity for

collaboration in intelligence exchange. Collecting data and processing it without actionable sharing limits the efficacy of cyber defence.

The third model studied is the Cyber Threat Intelligence (CTI) model by Mavroeidis and Bromander which focuses on cyber detection, prevention, and some operations yet there is no capacity provided for timeliness, security of CI shared across organisation. There are no automation capacities and no legal and privacy security to foster trust.

Also four network models discussed which are used in the industry include Hub-and-Spoke, that uses central platforms which are limited due to the fact that they operate as single points which is risky. The second is Peer-to-Peer which brings about issues of trust and is a big challenge. The third in the network is the Source-Subscribe model that operate a commercial subscription model that is available for only organisations that have financial capacity making it non-accessible to others who cannot afford it. Fourth is the Hybrid model which is difficult to integrate across organisations.

Going by these, certain fixed challenges arise across the CI sharing industry and has remained barriers. They include: Issues of trust, privacy and legality that organisations face due to scare of losing reputation if their sensitive information and intelligence leaks. This has made them reluctant to share until there are regulatory compliance in place. Another is the issue of lack of automation, and real-time exchange as most models use manual human mechanisms, and or semi-automated update tools like email, which is not fast to mitigate threats and sophisticated cyber-attacks. The third is the issue of cost of subscription to CI intelligence that most other organisation may not be able to afford especially SMEs.

Since information shared as intelligence are mostly unverified and the use is unregulated on all of these models, it has been challenging to secure infrastructural cyber resources due to infiltration of cyber criminals that wreak havoc. In light of these operational gaps, this study has made extensive findings to develop a CI sharing model that has capacity to be secure, proactive and work in real-time called the Proactive Self-Defence Cyber Intelligence Sharing (PSDCIS) model. This

model was developed to carry out automated information exchange, real-time data collection, classification of data using supervised machine learning algorithms. It promises to make intelligence alert fast and easy to act on, while integrating privacy and legal compliant resources that encourages trust, so that organisations in both public and private sectors can mitigate threats, engage cyber-attackers and win the ever-raging cyber warfare.

This robust PSDCIS model closes the gap of all detection, processing and defence needs as it makes Cybersecurity a proactive process, rather than reactive.

6.1.3 Objective three

Develop an automated model for sharing CI in real-time using ML algorithms for efficient data processing and analyses.

Upon identifying the gap in literature review and conducting an analysis of existing models and current CI sharing practices, the study developed a model that resolves the limitations of existing models and has the capacity of addressing the issues present in the realm of CI sharing.

The model proposed in this study – PSDCIS model, is capable of address existing CI challenges and met these objectives, as its features are based on the three main components that CI frameworks has been demanding:

i. Detection: To detect cyber-attacks before they attack businesses
ii. Preventing: Preventing resources
iii. Response: Responding cyber-attacks when it struck into our system by using knowledge.

Figure 91 Three main components of CI frameworks

Source: Researcher (2025)

The research has concentrated on developing a system that autonomously gathers, processes, analyses, and disseminates information, eliminating the need for human assistance. Additionally, it uses ML algorithms to distinguish between malware and benign content, sharing this information in real-time and integrating it with security systems.

In addition, the research used the intelligence life cycle process to develop the proposed PSDCIS model:

i. **Planning:** In this step a proper plan is created based on your requirements.
ii. **Gathering:** Gathering usable intelligence data which represent actual threats.
iii. **Processing:** Transforming gathered data into meaningful information that can be understood by humans.
iv. **Analysis:** Predictions that facilitate the assessment and anticipation of attacks and outcomes.
v. **Sharing:** sharing produced intelligence into business.

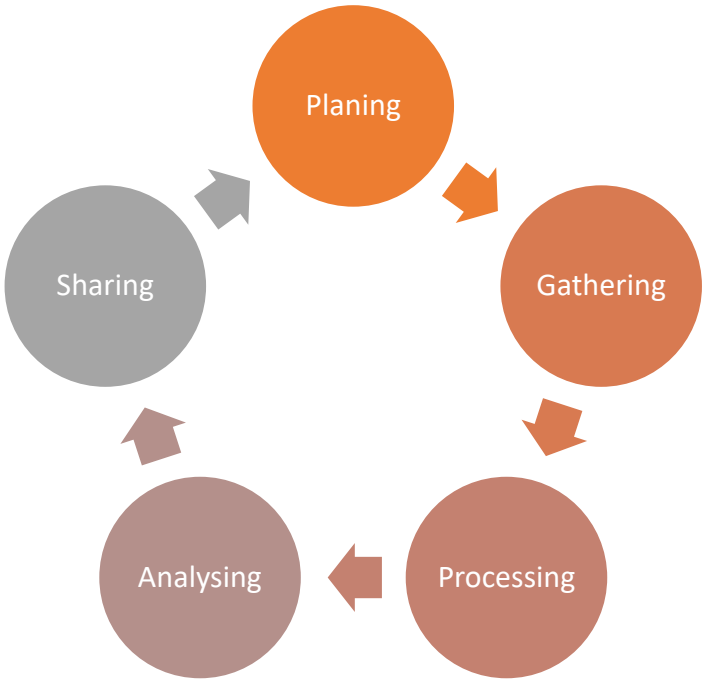


Figure 92 ILC process

Source: Researcher (2025)

Based on the findings of the literature review, model, and system investigation, with the incorporation of a CI framework and an ILC process, this study has developed a model that can be used to solve challenges that were identified in [Section 1.4] and [Sections 2.5, 2.8 and 2.9]. The purpose of developing the PSDCIS model was to convert theoretical thinking into practice so that we can produce a model that is able to gather intelligence, process it, analyse it, and share it in real-time. During the development of this model, questions arising from the model developed was answered thus: How can we gather intelligence automatically?

Examining the challenges identified in the Chapter 2 - Literature review of this study, the proposed PSDCIS model employed for automated intelligence collection through the implementation of scraping techniques from the intelligence source. To mitigate legal and privacy issues, we can delineate the intelligence to be discarded and remove any pertinent information regarding a particular business,

individual, or corporation. This method guarantees adherence to legal norms and improves the dependability of the collected data. By meticulously selecting the sources and categories of intelligence collected, we may enhance the precision of our findings while upholding ethical norms in data management.

These initiatives will bring back confidence and trust among stakeholders and clients, as they can rely on the privacy framework of the model that ensures their sensitive information is handled with the highest level of care.

This intelligence-gathering strategy enables firms to acquire information from any location, hence facilitating the accessibility of external intelligence.

The researcher advocated employing ML techniques for classification through the automatic analysis of collected intelligence. We incorporated a dataset into ML, which distinguishes malware and benign file signatures. Throughout the model creation process, nine distinct models trained and evaluated, as discussed in [Sections 3.3.1 and 3.3.2] of this study. After training and evaluating the models using various datasets, three most effective models were selected.

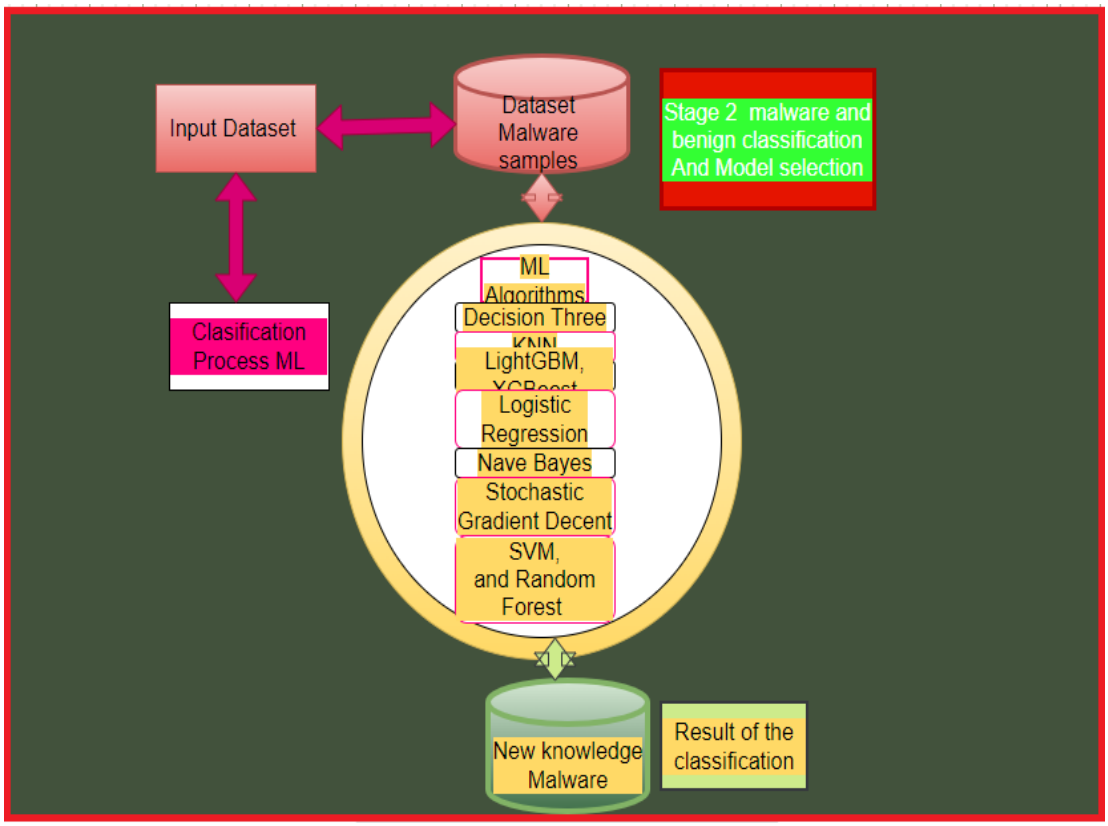


Figure 93 Algorithms training method

Source: Researcher (2025)

Algorithms Dataset1 80%	TN	FP	FN	TP		TN	FP	FN	TP	FP/FN Training %	FP/FN Testing %	Training A	Testing Accuracy %
Decision Tree	39995	0	0	40005		10005	0	0	9995	0	0	100	100
KNN	39995	0	0	40005		10005	0	0	9995	0	0	100	100
LightGBM	39995	0	0	40005		10005	0	4	9995	0	0.02	100	99.98
Logistic Regression	39970	25	60	39945	Testing 20%	10025	3	12	9983	0.11	0.07	99.89	99.93
Nave Bayes	39995	0	27448	12557		10005	0	6886	3109	34.31	34.43	65.69	65.57
Random Forest	39995	0	0	40005		10005	0	0	9995	0	0	100	100
Stochastic Gradient Decent	39925	70	82	39923		9980	25	14	9981	0.19	0.19	99.81	99.8
SVM	39900	5	47	39958		10004	1	9	9986	0.07	0.05	99.93	99.95
XGBoost	39995	0	0	40005		10005	0	0	9995	0	0	100	100

Figure 94 confusion matrix dataset1

Algorithms Dataset2 80%	TN	FP	FN	TP		TN	FP	FN	TP	FP/FN Training %	FP/FN Testing%	Training Accuracy %	Testing Accuracy %
Decision Tree	39970	0	0	40030		10030	0	0	9970	0	0	100	100
KNN	39958	12	7	40023		10029	1	3	9967	0.02	0.02	99.98	99.98
LightGBM	39970	0	0	40030		10030	0	0	9970	0	0	100	100
Logistic Regression	36978	2992	1797	38233	Testing 20%	9277	753	460	9510	5.99	6.06	94.01	93.94
Nave Bayes	15878	24092	6423	33607		4048	5982	1591	8379	38.14	37.86	61.86	62.14
Random Forest	39970	0	0	40030		10030	0	0	9970	0	0	100	100
Stochastic Gradient Decent	36658	3312	1863	38167		9211	819	481	9489	6.47	6.5	93.53	93.5
SVM	37374	2596	1649	38381		9397	633	410	9560	5.31	5.22	94.69	94.78
XGBoost	39970	0	0	40030		10030	0	0	9970	0	0	100	100

Figure 95 Confusion matrix dataset2

Source: Researcher (2025)

The Figures 120 and 121 illustrate the findings of the confusion matrix; the decision tree algorithms XGBoost in and LightGBM exhibit superior performance on both datasets. These models have a notable capacity for precise data classification, signifying their robustness and efficacy in managing intricate patterns.

6.1.4 Objective four

Integrate and implement the newly created CI sharing model into current company security systems (CSS) to guarantee prompt reactions in response to possible cyber threats.

Sharing CI is importance of defending looming cyber-attacks. CI exchange is essential to safeguard against breaches and Cyber-attacks (Hamza A, 2012). The adaption of "knowledge of us to battle the knowledge of them" is what supports the issue to identify a potential remedy" (Al-Shamisi, 2014).

This study proposes the integration of intelligence systems and defence systems, such as firewalls and antivirus, with RESTIP and SMTP to facilitate the automatic sharing of CI and to alert system administrators and managers, as discussed in [Section 3.5.1] and [Section 3.5.2] of the proposed model. This integration seeks to improve the overall security stance by enabling real-time communication and swift response to possible threats. By optimizing the information exchange across these systems, businesses can markedly reduce their susceptibility to intrusions. Moreover, the adoption of an integrated approach enhances threat detection effectiveness and enables teams to make prompt, educated decisions, hence reducing possible damage. As the cyber landscape evolves, implementing these new tactics becomes essential for sustaining solid safety measures in any organization.

Then, the development of the model comes with another question:

How can the three sub-model differences be integrated into a single model that automatically gathers, processes, and analyses intelligence?

This research study designed a state machine strategy and executed it by testing the PSDCIS model, integrating all three sub-models and automating the entire process within a singular state machine, as articulated in Chapter 5. This novel approach optimised the procedure and markedly improved the precision of the results achieved. The experimental results proved the efficacy of the state machine methodology in orchestrating intricate interactions among the sub-models.

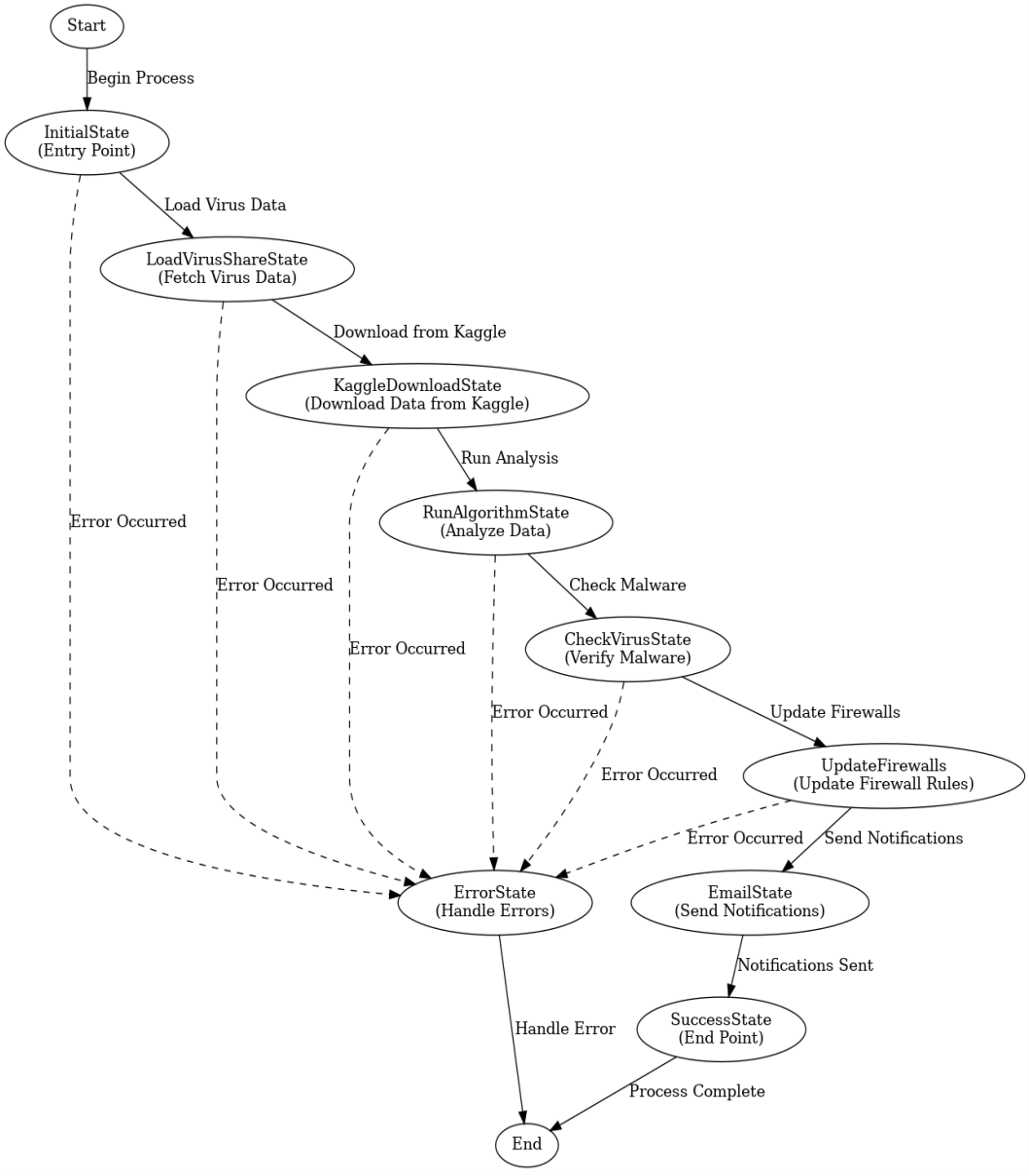


Figure 96 State machine

Source: Researcher (2025)

Figure 96 illustrates the procedure for automating the state machine: Stage 1: the state machine collects intelligence, verifies whether the intelligence has been previously acquired, assesses the data's accuracy, processes the information, parses the dataset, categorizes it into malware and benign classifications, and generates new intelligence. Furthermore, the state machine commences the dissemination of intelligence using the integration approach. This approach facilitates uninterrupted connectivity with business security systems, enhancing the entire security framework. Consequently, the PSDCIS state machine model can adjust and react to emerging threats more efficiently, guaranteeing a proactive stance on CS.

File Hash	File Type	Type Description Malware
000021ce9241b56a22923f51ec5895ab	peexe, spreader	Win32 EXE
000046c82c02ebe2633268f4ca8080ea	overlay, assembly, peexe	Win32 EXE
0000adda489b33d5f2d22d76e3bd8907	nsis, direct-cpu-clock-access, checks-network-adapters, via-tor, peexe, runtime-modules, overlay, nxdomain	Win32 EXE
0001617ffcd2415814904556ba2252d8	runtime-modules, peexe, armadillo	Win32 EXE
00019322819480aec09da95c3321d864	peexe, armadillo, spreader	Win32 EXE
0001bbaeafc6d232705939a859eda8dc	corrupt, peexe, overlay	Win32 EXE
000211b61fca8d27adde9fc5d4c6baf1	overlay, execryptor, upx, peexe	Win32 EXE
0002422da43f24e30470fa08f294acbe	bobsoft, peexe	Win32 EXE
000251962a5d01c0d8bd79408744da13	spreader, peexe	Win32 EXE
0002c9617dbaa0291cfb67c5f7204159	checks-user-input, overlay, armadillo, peexe, cve-2017-0144, cve-2005-1206, cve-2017-0146,	Win32 EXE
0002d20a7423518b7f371302014076c9	runtime-modules, checks-user-input, upx, peexe, direct-cpu-clock-access	Win32 EXE
0002d77db54bbb334d50df9cf2b5e5c4	pecompact, peexe, exeshield, spreader	Win32 EXE
0002dfa382bad91b7048bcabe8423fed	overlay, pedll	Win32 DLL
0002e34d7bc37ed784440556291943be		unknown
0002e7a939da0c5b82227460da11f615	peexe, direct-cpu-clock-access, spreader, armadillo, runtime-modules, checks-user-input	Win32 EXE
0002f2f17d87e0647b6ab92a76c08fd3	peexe, runtime-modules, long-sleeps, direct-cpu-clock-access	Win32 EXE
00031b2b4dedda092ae66f6d5690178c	peexe, overlay	Win32 EXE
00034b48dddb5b717481935c292ad2ef	checks-user-input, upx, detect-debug-environment, persistence, peexe	Win32 EXE
00036cb54834ffcad2495178055378e6	pedll, spreader, armadillo	Win32 DLL
00036dea9908d573f4d1b7034482fcac	peexe, overlay, upx	Win32 EXE
0003c4bf19b275ff5193d4b892387f99	idle, peexe	Win32 EXE
0003d9b55d7bde506827afe40252da5f	mz	DOS EXE
000432d85e14ea16d2ce3f23b9c11d8e	peexe, checks-network-adapters, upx, nxdomain	Win32 EXE
000463b36bd1b43a586ef1b76c1f7b33	com, known-distributor	DOS COM
0004887151668ca9e246abb0943fa9c9	runtime-modules, aspack, peexe	Win32 EXE
000578397fb06052c58738b38e3e800a	peexe, spreader, overlay	Win32 EXE
0005802d5a3c20262249b802a3c0aeda	irc, themida, peexe	Win32 EXE
0005e23384bf0cd2b5e2879040c83d3d	html	HTML
000605d7cbb3928b07f8e7473c820f4c	pedll	Win32 DLL
0006283e2d0d59bcb3134696268bf698	peexe, corrupt, nspack	Win32 EXE
00062cf38c3b3d846a70315f37a3edca	mew, corrupt, hosts-modifier, spreader, peexe, overlay	Win32 EXE
000665d0cf1cdaaf30ea1c4e091e0263	armadillo, pedll, persistence, detect-debug-environment	Win32 DLL
0006eb01f25d5a2a28c0af309ea04493	peexe	Win32 EXE
0007059868c711edb40574426a647b77	peexe, overlay	Win32 EXE
00070c107fd7024e361fdd6f841204a8	pedll, spreader	Win32 DLL
00070feb80aeb037fdeef4fc1ccf6a4a	upx, peexe	Win32 EXE
00071e74dd9fa2dd478eab96fe0d2327	spreader, peexe, assembly	Win32 EXE
0007378bbb8daea93425c225a02a5609	persistence, runtime-modules, peexe, direct-cpu-clock-access, checks-user-input	Win32 EXE
0007b23030d696c36e01cc244035601b		unknown
0007bb6456c7c985e6070cb37ff03a64	text	Text
000021ce9241b56a22923f51ec5895ab	peexe, spreader	Win32 EXE
000046c82c02ebe2633268f4ca8080ea	overlay, assembly, peexe	Win32 EXE
0000adda489b33d5f2d22d76e3bd8907	nsis, direct-cpu-clock-access, checks-network-adapters, via-tor, peexe, runtime-modu	Win32 EXE
0001617ffcd2415814904556ba2252d8	runtime-modules, peexe, armadillo	Win32 EXE
00019322819480aec09da95c3321d864	peexe, armadillo, spreader	Win32 EXE
0001bbaeafc6d232705939a859eda8dc	corrupt, peexe, overlay	Win32 EXE
000211b61fca8d27adde9fc5d4c6baf1	overlay, execryptor, upx, peexe	Win32 EXE
0002422da43f24e30470fa08f294acbe	bobsoft, peexe	Win32 EXE
000251962a5d01c0d8bd79408744da13	spreader, peexe	Win32 EXE
0002c9617dbaa0291cfb67c5f7204159	checks-user-input, overlay, armadillo, peexe, cve-2017-0144, cve-2005-1206, cve-2017-0	Win32 EXE
0002d20a7423518b7f371302014076c9	runtime-modules, checks-user-input, upx, peexe, direct-cpu-clock-access	Win32 EXE
0002d77db54bbb334d50df9cf2b5e5c4	pecompact, peexe, exeshield, spreader	Win32 EXE
0002dfa382bad91b7048bcabe8423fed	overlay, pedll	Win32 DLL
0002e34d7bc37ed784440556291943be		unknown
0002e7a939da0c5b82227460da11f615	peexe, direct-cpu-clock-access, spreader, armadillo, runtime-modules, checks-user-i	Win32 EXE
0002f2f17d87e0647b6ab92a76c08fd3	peexe, runtime-modules, long-sleeps, direct-cpu-clock-access	Win32 EXE
00031b2b4dedda092ae66f6d5690178c	peexe, overlay	Win32 EXE
00034b48dddb5b717481935c292ad2ef	checks-user-input, upx, detect-debug-environment, persistence, peexe	Win32 EXE
00036cb54834ffcad2495178055378e6	pedll, spreader, armadillo	Win32 DLL
00036dea9908d573f4d1b7034482fcac	peexe, overlay, upx	Win32 EXE
0003c4bf19b275ff5193d4b892387f99	idle, peexe	Win32 EXE
0003d9b55d7bde506827afe40252da5f	mz	DOS EXE
000432d85e14ea16d2ce3f23b9c11d8e	peexe, checks-network-adapters, upx, nxdomain	Win32 EXE
000463b36bd1b43a586ef1b76c1f7b33	com, known-distributor	DOS COM
0004887151668ca9e246abb0943fa9c9	runtime-modules, aspack, peexe	Win32 EXE
000578397fb06052c58738b38e3e800a	peexe, spreader, overlay	Win32 EXE
0005802d5a3c20262249b802a3c0aeda	irc, themida, peexe	Win32 EXE
0005e23384bf0cd2b5e2879040c83d3d	html	HTML
000605d7cbb3928b07f8e7473c820f4c	pedll	Win32 DLL
0006283e2d0d59bcb3134696268bf698	peexe, corrupt, nspack	Win32 EXE
00062cf38c3b3d846a70315f37a3edca	mew, corrupt, hosts-modifier, spreader, peexe, overlay	Win32 EXE
000665d0cf1cdaaf30ea1c4e091e0263	armadillo, pedll, persistence, detect-debug-environment	Win32 DLL
0006eb01f25d5a2a28c0af309ea04493	peexe	Win32 EXE
0007059868c711edb40574426a647b77	peexe, overlay	Win32 EXE
00070c107fd7024e361fdd6f841204a8	pedll, spreader	Win32 DLL
00070feb80aeb037fdeef4fc1ccf6a4a	upx, peexe	Win32 EXE
00071e74dd9fa2dd478eab96fe0d2327	spreader, peexe, assembly	Win32 EXE
0007378bbb8daea93425c225a02a5609	persistence, runtime-modules, peexe, direct-cpu-clock-access, checks-user-input	Win32 EXE
0007b23030d696c36e01cc244035601b		unknown
0007bb6456c7c985e6070cb37ff03a64	text	Text

Figure 97 check result of state machine intelligence testing

Source: Researcher (2025)

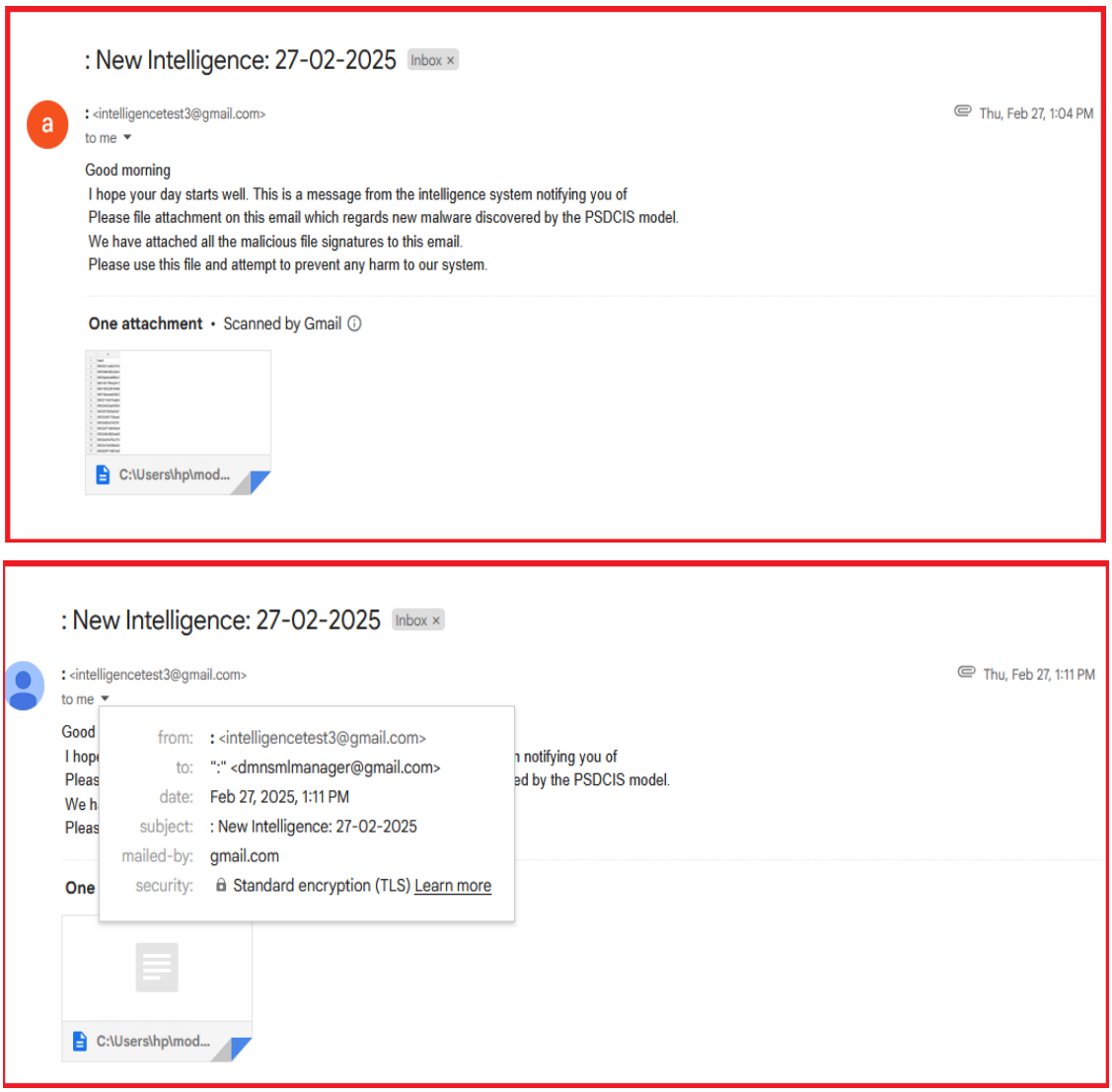


Figure 98 Result of email alert send to system administrator and management

Source: Researcher (2025)

Alongside automating the process using state machine intelligence, the study verified the outcomes generated by the state machine prior to disseminating them to security systems. As stated previously, the 5.4 model undergoes automatic validation, and concurrently conducts manual verification. Intelligence generated by the PSDCIS model was uploaded into VirusTotal's premier sandboxes, administered

by Virus Total. This process guarantees that all potential risks are meticulously analysed and cross-verified with current databases. Integrating automatic and manual validation improves the precision of our results and strengthens the entire security architecture. Section 5.4 encapsulates the automatic validation of SM result.

Figure 99 Manual model verification

Source: Researcher (2025)

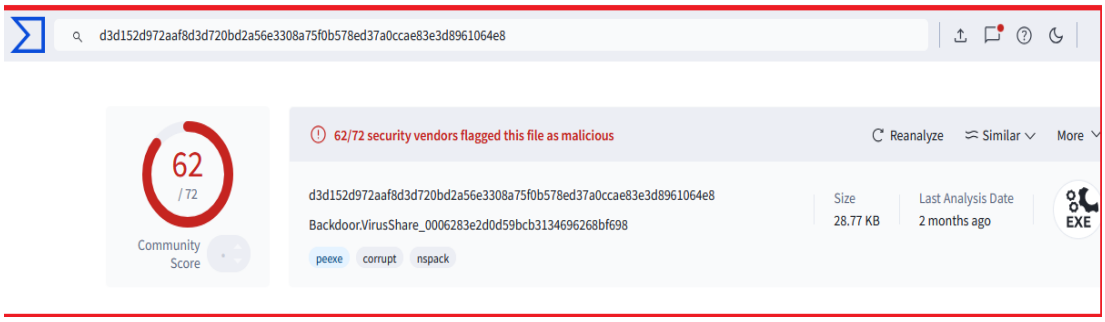


Figure 100 Manual verification result

Source: Researcher (2025)

The manual verification of the SM outcome revealed in Figure 99, file signature (0007bb6456c7c985e6070cb37ff03a64), which classifies the PSDCIS model as malware, detected by 39 out of 58 sandboxes. This number emphasises the increasing frequency of this danger, highlighting the necessity for improved security

measures across all platforms. Moreover, it prompts enquiries on the efficacy of existing detection techniques and the necessity for ongoing updates to antivirus software. Furthermore, Figure 100 indicates that 62 out of 72 sandboxes that examined this file recognised the file signature (0006283e2d0d59bcb3134696268bf698) as malware. This signifies a troubling trend, illustrating the efficacy of malware detection technologies in diverse sandboxes. The elevated identification rate highlights the necessity for ongoing modifications to these systems to address emerging threats.

Official vendor licenses for the antivirus and firewall are necessary to integrate and upgrade the firewall and antivirus software for testing this sub-model. Details of the procedure for updating file signatures in firewalls and antivirus software is in [Section 5.3.9].

The Study developed a model, trained and tested it successfully, and disseminated intelligence. Objectives 3 and 4 are completed, except for uploading malware signatures to firewalls and antivirus software due to licensing constraints.

6.1.5 Objective five

Develop recommendations for enhancing real-time CI sharing capabilities and overcoming existing challenges, while improving CS and enabling organisations to leverage collective intelligence.

To enhance real-time CS sharing and tackle existing obstacles, businesses should implement a multi-layered strategy combining collaboration, technology, and standardisation. Principal recommendations encompass. Businesses should commence the creation of their intelligence systems by executing this automation procedure, facilitating the integration of their security systems and the generation of automated warnings. This integration augments overall safety and refines incident response methods. By employing real-time data, companies would proactively manage potential hazards before they escalate into significant issues.

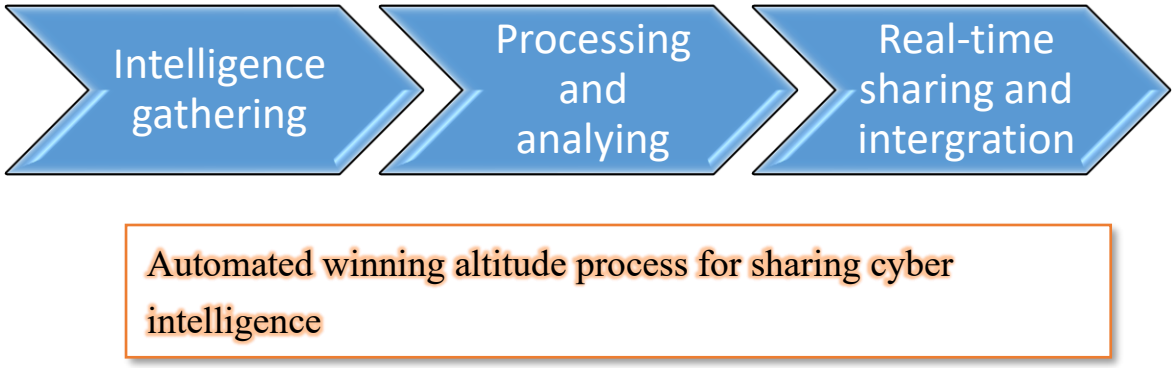


Figure 101 Winning attitude

Source: Researcher (2025)

Implementing the automated winning attitude suggested by the PSDCIS model allows us to resolve existing legal and privacy concerns, access external intelligence sources, standardise processes, automate functions, exchange information with the community, and guarantee timeliness, quality, intelligence, and trust through process automation. The application of ML models to CS categorisation improves the precision of intelligence, as demonstrated by the PSDCIS model's automated and manual validation methods. Furthermore, it offers accurate and pertinent information, which is crucial in addressing imminent cyber-attacks. This preventive approach enhances decision-making and allows security teams to use resources more efficiently in their defence strategy. The incorporation of ML into CI enhances electronic infrastructure protection and reduces possible threats.

Implications

The use of the PSDCIS-based automated approach has considerable consequences for enhancing real-time CI and reinforcing protection mechanisms:

This approach mitigates legal and privacy issues by establishing data management and implementing automation, hence ensuring secure information sharing within legal parameters. This fosters client trust and mitigates the possibility of any litigation associated with data breaches.

i. Enhanced Access to External Intelligence: The incorporation of external data sources expands and refines the collection of threat intelligence, fostering a more thorough comprehension of the information security picture.
ii. Standardisation and Automation Processes: By implementing clearly delineated workflows and automation, businesses: By employing established processes and automation, businesses can optimize intelligence operations, minimise discrepancies, and maintain elevated standards of quality and trust. Consequently, teams are more adept at promptly addressing emerging threats and adjusting to environmental changes. This process finally cultivates a culture of constant enhancement and development within the company.
iii. Improved Knowledge Precision: Making use of ML for classification, along with automatic and manual validation, guarantees that disseminated intelligence is both timely and accurate. Moreover, by employing sophisticated algorithms and real-time data analysis, companies are more adept at predicting obstacles and focusing on opportunities in a constantly changing environment. The proactive strategy not only cultivates invention but additionally enhances competitive advantages in their sectors.
iv. Proactive Threat Management: By facilitating the early identification and classification of risks, the model fosters a proactive security attitude, assisting staff in making knowledgeable choices prior to the occurrence of attacks. This strategy not only mitigates possible damage but also cultivates a culture of vigilance within the corporation.
v. Enlargement of cyber protection: The integration of ML improves the security of electronic systems by detecting weaknesses and mitigating possible threats. This proactive technique enhances threat identification and allows businesses to respond efficiently to occurrences in real time. This substantially enhances the

robustness of essential infrastructure, guaranteeing a more secure modern landscape for people and data alike.

The consequences combined demonstrate how the PSDCIS model could transform CI sharing practices by promoting intelligent, automated, and proactive defence methods.

Research Limitation

First, the primary limitation is about the lack of resources for cyber intelligence and information sharing, along with the challenges in comprehending its definition, its interpretation within corporate contexts, and its prospective advantages. Most academics concentrate on a singular facet of cyber security: the collection, analysis, or examination of the difficulties associated with intelligence sharing. This limited perspective frequently neglects the interrelations among different components, potentially obstructing a holistic comprehension of the entire environment. The diverse interpretations of CI across many disciplines might lead to discrepancies in data collection and analysis, hence impacting the trustworthiness of study outcomes.

Second, existing models and systems emphasise the collection and processing of intelligence. But newer models usually do not include functions for sharing of intelligence and also does not store the data needed for commercial purposes, and that limits many organisations from having access because of its high cost. They also have issues of losing money after purchase because the intelligence may not be useful, and be obsolete and so less effective. This research built a model that is able to automate the process of gathering, processing, analysing and sharing intelligence through integration. Moreover, there is a lack of discussion regarding the automation and integration of the processes involved in gathering, processing, analysing, and disseminating information inside information systems.

Third, many businesses are deficient in CI skills, particularly in knowledge generation and its application for detecting, preventing, and responding to cyber assaults. This lack of comprehension results to weak security protocols and

dependence on obsolete methods, hence rendering organisations susceptible to advanced cyber threats.

Fourth, there was need for licensed firewalls and antivirus software for testing the PSDCIS model but they are costly and not easily available compared ti the existing tools discussed so far. In addition, this model uses machine learning algorithms to classify large datasets, which requires high-end computers with powerful processors and memory, and they were not available. Due to cost-constraints of the research to get the computers, the experimental testing was limited.

Fifth, the sustained effectiveness of a Proactive Self-Defence CI Sharing (PSDCIS) system will rely on real-time CI sharing, automation, and integration. While these components are crucial for defending against current sophisticated cyber-attacks, they need defenders to have sufficient knowledge and intelligence about cyber attackers. This goal can be achieved by continuing to gather intelligence from external and internal sources, analysing them with machine learning algorithms and sharing that intelligence through integrating and automating of the whole process. Major limitation is the training that is required for PSDCIS, as staff should have enough knowledge of cyber hunting, classification algorithms and programming languages for continuously maintaining automation and integration.

Sixth, this research used a secondary dataset containing malware and benign file signatures, and classification algorithms were employed to distinguish between the two types of files. Even though, this PSDCIS model can analyse various datasets in different formats, and use same to interpret the results differently, allowing for a more thorough assessment of the model's resilience and flexibility. Future research may focus on optimising the algorithm to improve precision and efficacy for real-time applications.

Seventh, there are also limitations from the machine-learning aspects of this research, because the training and validation of the PSDCIS model was done specifically on a malware dataset, making its results not generalisable. Malware signatures grow rapidly, so even though the system performs strongly on the dataset

used, other datasets with non-malware attack vectors may produce variable results. In addition to this, the model architecture was custom-made for signature-based malware detection, which means that its ability to adapt to non-malware CI tasks like phishing detection or insider threats investigation is not tested yet. This restricts it wider application obviously.

Eight, another limitation is the way the system is designed – the architecture, which uses machine learning classifiers inside a state machine process. This design brings about huge dependency on computer resources and systems, which causes the challenge with throughput, latency, and interoperability when set up in typical enterprise SOC locations. These model architectural challenges might influence how reliable and efficient the PSDCIS is, while in production mode.

Finally, the current findings cannot be generally assumed to suit all organisations or industries due to variances in infrastructure, data quality, and regulation stipulation. Future research should therefore validate the PSDCIS model across several fields, especially finance, healthcare, and government, in order to create a wider applicability and identify adaptations that are necessary and needed in specific fields to make it more robust.

Knowledge Contribution

The PSDCIS model significantly contributes to cyber defence by autonomously detecting, mitigating, and responding to threats while enhancing the intelligence of security systems. It enhances real-time information sharing by establishing a coherent, automated system that links data collection with actionable insights. The PSDCIS offers a practical method for handling highly sensitive data by incorporating legal and privacy compliance into automated procedures, thereby enhancing the ongoing dialogue on responsible and legal information sharing.

Its fundamental contributions encompass:

The standardisation and automation of CI sharing by formalising critical stages in CI (Planning, Sourcing, Developing, Collecting, Integrating, and Sharing), the model enhances understanding of how organisations can optimise and unify

threat mitigation processes in various systems and industries. The implementation of ML in the classification of intelligence data demonstrates the incorporation of automated validation techniques, enhancing the accuracy, pertinence, and dependability of information security insights, thereby advancing the usage of ML in the field of CI sharing.

Proactive Cyber Defence – this model changes CI sharing from a reactive to a proactive approach, facilitating early detection, contextual intelligence, and optimal resource utilisation which are critical issues in today's challenges of CI sharing research and implementation.

Enhancing trust: The PSDCIS model facilitates the rapid and precise sharing of information to defensive systems and notifies system administrators, promoting the adoption of this technique by other communities to coordinate CTI, which is essential for effective resource safeguarding.

This model enhances both academic and practical comprehension by demonstrating how automation, standardisation, and intelligent validation can converge to establish a more flexible, reliable, and proactive cyber defence ecosystem.

CHAPTER SEVEN: CONCLUSION AND FUTURE STUDIES

Conclusion

Although cyber-sharing is advantageous for both organisations and governments, there are unfortunately various technological and psychological obstacles that hinder its implementation. The literature review for this research identified several obstacles to intelligence sharing, including insufficient trust among businesses and organisations, inefficiencies in processes, legal impediments, challenges in accessing external intelligence sources, the need for real-time sharing, the automation of standardised intelligence sharing, and concerns regarding legality and privacy. The existing information exchange platforms inadequately support the full process of collecting, analysing, and disseminating information within security systems.

This research also investigated three different models for sharing CI: the structured intelligence sharing model, the detection maturity level (DML), and the CTI model. The research looked at five different systems for sharing CI: the Collective Intelligence Framework (CIF), Malware Information Sharing Platform (MSIP), Webroot Intelligence Network (WIN), Darknet Deepnet Cyber Security Threat Intelligence, and Proactive Cyber Threat Analysis. The models and systems' ability to share CI is the main concern: gather intelligence from repositories, process and analyse it, and store it in central locations, where community members may connect and obtain the data. The issue with these systems is the lack of real-time information sharing. This implies that the intelligence is not up-to-date, potentially making it unavailable for immediate use, or eventually losing its value over time. The condition of CI is that it should be actionable and usable; otherwise, it is not intelligence. The other challenge to the method of sharing community intelligence is that everyone can join the membership; even the cyber attackers can pose as businesses and collect information from community members and use that information to attack member free riders.

To address this issue, this study proposed the development of the Pro-active Self-Defence Cyber Intelligence Sharing (PSDCIS) model that autonomously collects data, employs ML classification algorithms to differentiate malware from benign files, and facilitates the sharing of insights through REST API integration with security systems, deploying automated alerts to managers and system administrators via the SMTP protocol. This method improves both the speed and precision of threat detection while facilitating preventive measures against prospective security breaches. By optimising communications and automating notification systems, organisations can markedly lower the risk they face with cyber-attacks.

The researcher applied this theory in practical experimentation using SM automation. The application of SM encompasses various subsets such as the initialisation stage, that deploys LoadVirusshare, Kaggle download state, RunAlgorithmsState, CheckVirusState, EmailState, and Update Firewall and AntiVirusState and ErrorState. The states then run sequences and perform each separately, to produce expected result. Furthermore, this study validated the model through two methodologies: performance measurement validation and practical validation. The dataset during categorisation came out as malware by the PSDCIS model and was submitted to the VirusTotal platform for analysis via CheckVirusState. This validated that the intelligence recognised by the PSDCIS model was precise. Not all sandboxes, representing the premier antivirus and firewall providers, lack system updates. Subsequently, the automatic sharing of intelligence was initiated via interaction with the security system and automated notifications.

Finally, the PSDCIS model plays a crucial role in cyber defence by autonomously identifying, alleviating, and addressing risks while augmenting the intelligence of security systems. It improves real-time information sharing by creating a cohesive, automated framework that connects data acquisition with actionable insights. The PSDCIS offers an important structure for managing highly sensitive data by integrating legal and privacy regulations into automated

procedures, hence enhancing discussions on the responsible and lawful sharing of information.

Suggestions for Future Study

The PSDCIS model, as it stands, offers a robust contribution to real-time CI sharing, including integrating it into security systems, automating processes, and leveraging ML technology for intelligence classification. Several areas remain open for future research. These are some suggested directions where in-depth studies are required;

i. The incorporation of advanced analytical instruments to enhance intelligence gathering capabilities, and the creation of simple user interfaces for increased use among security experts. The model's broad relevance across various sectors can provide knowledge and enhance overall defence against cyber-attacks.
ii. Protocols for Threat Intelligence Sharing – Improving compatibility through adherence to worldwide and industry-specific standards like STIX, TAXII, and MITRE ATT&CK will be essential for broad acceptance and confidence.
iii. Future advancements in sophisticated artificial intelligence methodologies may incorporate deep learning and neural networks to improve accuracy and contextual relevance in CI classification.

7.1 Adapting PSDCIS for Zero-Day and AI-Generated Threats

As highlighted in the Research Limitations (Section 6.4), the PSDCIS model relies on specific datasets and supervised ML algorithms, which may limit its generalisability and its efficacy against unexpected attack vectors. These become critical in cases of Zero-day and AI-generated threats. Zero-day attacks are exploits targeting unknown vulnerabilities even before remedy can happen. They can evade conventional signature approaches and create immediate, seriously-damaging risks (Craig, 2022; Franklin and Ismail, 2022). AI-generated threats, such as polymorphous malware or deepfake phishing, exploit learning techniques from

cyber attackers and uses same information to bypass existing detection systems (Czosseck, Tyugu and Wingfield, 2011)

The PSDCIS model can be enhanced in three directions, to overcome these limitations. First, the combination of unsupervised and support learning techniques would allow anomaly detection founded on deviations from well-known behavioural baselines, thereby establishing resilience against unsuspected zero-day activities (Akhtar and Feng, 2022). Second, classifiers trained for cyber attackers should be incorporated to guard against evasion strategies created by hostile AI, guaranteeing that classification remains robust even under pressure. Third, blockchain verification of intelligence feeds, would improve trust and credibility of shared intelligence, and prevent the propagation of false or manipulated data coming from cyber attackers. By these directions, the PSDCIS model would retain its proactive feature and advance to real-time defence against new and adaptive threat environments.

Business Case and Deployment Strategy for the PSDCIS System

The PSDCIS model fixes the typical gaps replete in CI frameworks, especially in the aspects of automation, compliance, trust and scalability capabilities. MISP, and TAXII/STIX-enabled platforms do not have the capacity for the integration of machine learning features like real time delivery, classification, and automated compliance checks like the PSDCIS model provides as an end-to-end solution.

This business case relies on four aspects; The value edge, the cost advantage, the market difference and stakeholders' benefit. In value, the PSDCIS has an edge on time reduction for response time, precision in identifying threat, seamless compliance and more. For cost advantage, it reduces costs that are usually incurred by downtime incidences, cost of cyber breaches and even offsets cost from the workload of SOC analysts, even though it requires an initial investment in cloud subscriptions, hardware, and APIs etc. For market difference, the PSDCIS is unlike other models as it has a flexible architecture that supports various data formats, international compliances, and secure frameworks that makes it a sought-after CI sharing tool. For the benefits the PSDCIS offers stakeholders, government agencies, private companies and SOCs benefit from low cost for shared intelligence, high-speed compliance reporting, and especially its unmatched resilience as a system.

7.1.1 Deployment Strategy for PSDCIS

Deploying the PSDCIS is best in phases to reduce risks and establish scalability. First phase should be for pilot implementation to deploy the PSDCIS in a controlled SOC setting like that of a government agency or a financial institution. Integrate it with available CI tools using APIs. Conduct benchmarking to counter MISP, and TAXII/STIX frameworks. Then validate its performance by latency, throughput, precision, recall, and F1-score.

Second phase should be for enterprise integration, by extending it across multiple businesses. Enable data sharing through secure push messages, and blockchain verification methods across countries and organisations. Develop

training modules to help staff understand ML-based intelligence classification and system arrangement. Then create a model of cost-sharing that will sustain the infrastructure and fund software updates.

Third phase should be ensuring the model is adopted in sectors such as telecommunication, finance, energy, healthcare, etc. Demonstrate how the PSDCIS compliance aligns with the specific regulations of these sectors, such as HIPAA for healthcare. Also enable interoperability within the sector by implementing federated data.

Fourth phase has to be multilateral expansion across borders, which will entail collaborating with international SOCs and response teams for CI exchange. Use blockchain verification methods for wider trust. Then establish a framework for managing updates, funding and community-building by international governance standards.

REFERENCE LIST

Abu, M.S., Selamat, S.R., Ariffin, A. and Yusof, R. (2018). Cyber Threat Intelligence – Issue and Challenges. *Indonesian Journal of Electrical Engineering and Computer Science*, 10(1), p.371. doi:https://doi.org/10.11591/ijeecs.v10.i1.pp371-379.

Adam Williams and GrahamShaw (2016) *SearchSecurity,* s.l.: Threat Alert.

Administration of Barack Obama (2015). *Executive Order 13691 – Promoting Private Sector Cybersecurity Information Sharing*. [online] *Global Resilience Institute*, Northeastern University: Global Resilience Institute, pp.1–5. Available at: https://globalresilience.northeastern.edu/executive-order-13691-promoting-private-sector-cybersecurity-information-sharing-2/ [Accessed 24 Jan. 2024].

Aidman, E., Chadunow, C., Johnson, K. and Reece, J. (2015). Real-time driver drowsiness feedback improves driver alertness and self-reported driving performance. *Accident Analysis and Prevention*, 81, pp.8–13. doi:https://doi.org/10.1016/j.aap.2015.03.041.

Aliaga, M., Gunderson, B., (2002) *Interactive Statistics.* ThousandOaks: Sage.

Al-Ibrahim, O., Kamhoua, C., Kwiat, K., Njilla, L. and Mohaisen, A. (2017). Assessing Quality of Contribution in Information Sharing for Threat Intelligence. *Conference: 2017 IEEE Symposium on Privacy-Aware Computing (PAC)*, [online] pp.182–183. doi:https://doi.org/10.1109/pac.2017.39.

Almashhadani, A. O. S., (2020) *"Network-based AdvancedMalware Detection Using Multi-Classier ML",* Belfast : Queen's UniversityBelfast, UK

Al-Shamisi, Ahmed. (2014). *Active offensive cyber situational awareness: Theory and practice*. [PhD Thesis] pp.1–284. Available at: https://bura.brunel.ac.uk/bitstream/2438/13427/1/FulltextThesis.pdf [Accessed 11

Nov. 2024].

Alaeifar, P., Pal, S., Jadidi, Z., Hussain, M. and Foo, E. (2024). Current approaches and future directions for Cyber Threat Intelligence sharing: A survey. *Journal of information security and applications*, [online] 83(83), pp.103786–103786. doi:https://doi.org/10.1016/j.jisa.2024.103786.

Ali, M. (2022). *Supervised Machine Learning*. [online] www.datacamp.com. Available at: https://www.datacamp.com/blog/supervised-machine-learning [Accessed 15 Jun. 2024].

Arenas, E. (2017). Cyber Threat Intelligence Information Sharing. *The 5th International Conference on Cybercrime and Computer Forensics (ICCCF) – Technically Co-Sponsored by IEEE*. [online] Available at: https://www.researchgate.net/publication/320034441_Cyber_Threat_Intelligence_Information_Sharing [Accessed 2 Apr. 2022].

Bander Alsulami and Spiros Mancoridis (2018). *Behavioral Malware Classification using Convolutional Recurrent Neural Networks*. [online] doi:https://doi.org/10.1109/malware.2018.8659358.

Barnum, S. (2014). *Standardizing Cyber Threat Intelligence Information with the Structured Threat Information eXpression (STIX™)*. [online] *stixproject.github.io*. MITRE. Available at: https://stixproject.github.io/about/STIX_Whitepaper_v1.1.pdf [Accessed 9 Mar. 2021].

Bazrafshan, Z., Hashemi, H., Fard, S.M.H. and Hamzeh, A. (2013). A survey on heuristic malware detection techniques. In: *The 5th Conference on Information and Knowledge Technology*. [online] Institute of Electrical and Electronics Engineers (IEEE), pp.113–120. doi:https://doi.org/10.1109/ikt.2013.6620049.

Bergman, J. and Popov, O.B. (2023). Exploring Dark Web Crawlers: A systematic

literature review of dark web crawlers and their implementation. *IEEE Access*, [online] pp.1–1. doi:https://doi.org/10.1109/access.2023.3255165.

Bhunia, S., Hsiao, M.S., Banga, M. and Narasimhan, S. (2014). Hardware Trojan Attacks: Threat Analysis and Countermeasures. *Proceedings of the IEEE*, [online] 102(8), pp.1229–1247. doi:https://doi.org/10.1109/jproc.2014.2334493.

Biau, G. and Cadre, B. (2017). *Optimization by gradient boosting*. [online] arXiv.org. doi:https://doi.org/10.48550/arXiv.1707.05023.

BlueFort Security (2023). *Cyber Threat Intelligence (CTI) - BlueFort Security*. [online] Bluefort Security. Available at: https://www.bluefort.com/solutions/cti/ [Accessed 9 May 2024].

Bonan Cuan, Aliénor Damien, Claire Delaplace, and Mathieu Valois, (2018) *Malware Detection in PDF Files Using ML,* s.l.: Hal Open science

Brilingaitė, A., Bukauskas, L., Juozapavičius, A. and Kutka, E. (2022). Overcoming information-sharing challenges in cyber defence exercises. *Journal of Cybersecurity*, [online] 8(1). doi:https://doi.org/10.1093/cybsec/tyac001.

Broggi, J.J. (2013). *Building on executive order 13,636 to encourage information sharing for cybersecurity purposes*. [online] Available at: https://journals.law.harvard.edu/jlpp/wp-content/uploads/sites/90/2014/05/37_2_653_Broggi-1.pdf [Accessed 9 May 2022].

Brown, S., Gommers, J. and Serrano, O. (2015). Proceedings of the 2nd ACM Workshop on Information Sharing and Collaborative Security. In: *WISCS '15: Proceedings of the 2nd ACM Workshop on Information Sharing and Collaborative Security*. [online] New York, NY, USA: ACM, pp.43–49. doi:https://doi.org/10.1145/2808128.

Burger, E., Kampanakis,P., Zhu, K. and Goodman, M., (2014). *Taxonomy model for*

Cyber Threat Intelligence informationexchange technologies, s.l.: Proceedings of the ACM Conference on Computerand Communications Security.

Chapaneri, R. and Shah, S. (2018). A Comprehensive Survey of Machine Learning-Based Network Intrusion Detection. *Smart Intelligent Computing and Applications*, [online] 104, pp.345–356. doi:https://doi.org/10.1007/978-981-13-1921-1_35.

Chen, H., Shen, Z., Wang, Y., Ke, H. and Xu, J. (2024). Threat detection driven by artificial intelligence: Enhancing Cybersecurity with machine learning algorithms. *World Journal of Innovation and Modern Technology*, [online] 7(6), pp.58–70. doi:https://doi.org/10.53469/wjimt.2024.07(06).09.

Chen, Z., Liang, Z., Zhang, Y. and Chen, Z. (2011). Evaluating Grayware Characteristics and Risks. *Journal of Computer Networks and Communications*, 2011(569829:1-569829:28.), pp.1–28. doi:https://doi.org/10.1155/2011/569829.

Chester, J. (2015). Analysis of Password Cracking Methods and Applications. *Williams Honors College, Honors Research Projects*. [online] Available at: https://ideaexchange.uakron.edu/honors_research_projects/7/ [Accessed 10 Jun. 2020].

Chowdhury, M., Rahman,A. and Islam.R, (2018) *Malware Analysis and Detection using Data Mining andML Classification,* s.l.: International Conference on Applications andTechniques in CS and Intelligence.

CIRCL (2012). *CIRCL» MISP - Open Source Threat Intelligence Platform*. [online] www.circl.lu. Available at: https://www.circl.lu/services/misp-malware-information-sharing-platform/ [Accessed 9 Aug. 2019].

CISCO (2025). *Anatomy of an IoT attack_EN*. [online] Cisco. Available at: https://www.cisco.com/site/us/en/learn/topics/security/what-is-a-worm.html.

Climer, S. (2018). *Mindsight*. [online] Mindsight. Available at:

https://gomindsight.com/insights/blog/history-of-cyber-attacks-2018/.

Conti, M.,Dehghantanha, A. and Dargahi, T. (2018). *Cyberthreat Intelligence:Challenges and Opportunities.* s.l., Advances in Information Security.

Craig, J. (2022). All About Zero Day Attacks. *All About Zero Day Attacks*. Available at: https://phemex.com/blogs/what-is-zero-day-attack [Accessed 8 May 2024].

CREST, (2019) *What isCyber Threat Intelligence and how is it used?,* s.l.: CREST.

Creswell, J.W. (2009). *Research design : Qualitative, Quantitative and Mixed Methods Approaches.* 3rd ed. [online] Sage Publications, Inc. Available at: https://www.ucg.ac.me/skladiste/blog_609332/objava_105202/fajlovi/Creswell.pdf.

Crotty, M. (1998). *The foundations of social research: Meaning and perspective in the research process*. 1st ed. Routledge. doi:https://doi.org/10.4324/9781003115700.

Czosseck, C., Tyugu, E. and Wingfield, T (2011) Artificial Intelligence in Cyber Defense. *3rdInternational Conference on Cyber Conflict.*

Dalziel, H., (2015) *Howto Define and Build an Effective Cyber Threat Intelligence Capability.* s.l.:Elsevier Science and Technology Books.

Dandurand, L. and Serrano, O. (2013). *Towards Improved Cyber Security Information Sharing Requirements for a Cyber Security Data Exchange and Collaboration Infrastructure (CDXI)*. [online] Available at: https://ccdcoe.org/uploads/2018/10/25_d3r1s5_dandurand.pdf [Accessed 18 Apr. 2025].

Darley, T. and Schreck, T. (2018). *Why is Cyber Threat Intelligence Sharing Important?* [online] Infosecurity Magazine. Available at: https://www.infosecurity-

magazine.com/opinions/cyber-intelligence-sharing/.

Darling, D. J. (1996). Computers of the Future: Intelligent Machines and Virtual Reality. United States: Dillon Press.

Deepti, R., Singh Gill, N. and Gulia, P. (2022). CLASSIFICATION OF SECURITY ISSUES AND CYBER ATTACKS IN LAYERED INTERNET OF THINGS. *Journal of Theoretical and Applied Information Technology*, [online] 100(13). Available at: https://www.jatit.org/volumes/Vol100No13/20Vol100No13.pdf [Accessed 27 Sep. 2024].

Denning, P.J. (1991). *Computers under attack: intruders, worms, and viruses*. *ACM eBooks*. Association for Computing Machinery. doi:https://doi.org/10.1145/102616.

Dey, D., Lahiri, A. and Zhang, G., (n.d) *QualityCompetition and Market Segmentation in the Security Software Market,*

Dressler, J., 2007. *UnitedStates v. Morris". Cases and Materials on Criminal Law.* s.l.:St. Paul,MN:.

Dwyer, D. (2009). Chinese cyber-attack tools continue to evolve. *Network Security*, [online] 2009(4), pp.9–11. doi:https://doi.org/10.1016/s1353-4858(09)70039-x.

Eberz, S., Strohmeier, M., Wilhelm, M. and Martinovic, I. (2012). A practical man-in-the-middle attack on signal-based key generation protocols. *Computer Security – ESORICS 2012*, 7459, pp.235–252. doi:https://doi.org/10.1007/978-3-642-33167-1_14.

Ernst and Young Global Limited (EYG) (2024). *Cyber threat intelligence − how to get ahead of cybercrime*. [online] pp.1–16. Available at: https://www.bibliotecadeseguranca.com.br/wp-content/uploads/2015/09/EY-cyber-threat-intelligence-how-to-get-ahead-of-cybercrime.pdf.

Ertan, A., Floyd, K., Pernik, P. and Stevens, T. (2020). *Cyber Threats and NATO 2030: Horizon Scanning and Analysis*. [online] NATO CCDCOE Publications, pp.1–267. Available at: https://ccdcoe.org/uploads/2020/12/Cyber-Threats-and-NATO-2030_Horizon-Scanning-and-Analysis.pdf.

EY (2014). *Cyber threat intelligence − how to get ahead of cybercrime*. [online] EYGM Limited. Available at: https://www.bibliotecadeseguranca.com.br/wp-content/uploads/2015/09/EY-cyber-threat-intelligence-how-to-get-ahead-of-cybercrime.pdf.

Farnham, G. and Leune, G., (2013). *Tools and Standards for Cyber Threat Intelligence Projects ,* s.l.: SANS .

Fenz, S., Heurix, J.,Neubauer, T. and Pechstein, F., (2014) *Current challenges in informationsecurity risk management. Information Management and Computer Security,* s.l.: Emerald Group Publishing Limited.

F-secure (2001). *Net-Worm:W32/Nimda Description | F-Secure Labs*. [online] F-secure.com. Available at: https://www.f-secure.com/v-descs/nimda.shtml.

FlashPoint Team (2024). *Threat Intelligence*. [online] Flashpoint. Available at: https://flashpoint.io/blog/guide-to-cyber-threat-intelligence/ [Accessed 29 Dec. 2024].

Flodström , M. and Vikholm, O. (2016). *SQL-Injections: a wake-up Call for developers: A study about a major threat and issue for companies and organizations worldwide*. [online] pp.1–41. Available at: https://www.diva-portal.org/smash/get/diva2:630946/FULLTEXT01.pdf.

Franklin, O.U. and Ismail, M. (2022). THE ZERO-DAY VULNERABILITY. *International Journal of Information System and Engineering*. [online] doi:https://doi.org/10.24924/ijise/2021.04/v9.iss2/65.76.

Franz, M. (2007). Containing the Ultimate Trojan Horse. *IEEE Security and Privacy Magazine*, [online] 5(4), pp.52–56. doi:https://doi.org/10.1109/msp.2007.77.

Friedman, J. and Bouchard, M. (2015). *Definitive guide to cyber threat intelligence : using knowledge about adversaries to win the war against targeted attacks*. [online] Annapolis, Md: Cyberedge Group. Available at: https://www.crest-approved.org/wp-content/uploads/2022/04/cti-guide-2.pdf [Accessed 9 Nov. 2021].

Geers, K. (2011). *Strategic cyber security*. Tallinn: Nato Cooperative Cyber Defence Centre Of Excellence.

Gergen, K.J. (2001). Psychological science in a postmodern context. *American Psychologist*, [online] 56(10), pp.803–813. doi:https://doi.org/10.1037/0003-066x.56.10.803.

Giorgi, I. and Maksim, I., (2023) *Enhancing CICapabilities through Process Automation: Advantages and Opportunities,* Georgia:Caucasus University

GOA, (2000) *CriticalInfrastructure Protection Comments on the Proposed Cyber Security Information Act of 2000,* s.l.:United States General Accounting Office.

Gomez, F.J., Vanfretti, L. and Olsen, S.H. (2018). CIM-Compliant Power System Dynamic Model-to-Model Transformation and Modelica Simulation. *IEEE Transactions on Industrial Informatics*, 14(9), pp.3989–3996. doi:https://doi.org/10.1109/tii.2017.2785439.

Greenberg, A. (2012). *WikiLeaks Announces Massive Release With The 'Syria Files': 2.4 Million Emails From Syrian Officials And Companies*. [online] Forbes. Available at: https://www.forbes.com/sites/andygreenberg/2012/07/05/wikileaks-announces-its-largest-release-yet-in-the-syria-files-2-4-million-emails-from-syrian-officials-and-companies/#1dab291a5081 [Accessed 9 May 2024].

Guba, Egon, G. Lincoln, and Yvonna, S., (1998) *Competing Paradigms in*

Qualitative Research. s.l.:s.n.

Hackeling, G., (2014) *MasteringMachine Learning with scikit-learn.* 2nd ed. s.l.:Packt.

Hamza, A. (2012). Taxonomy of Machine Learning Algorithms to classify real-time Interactive applications. *Computer Science*.

Haque, Md.F. and Krishnan, R. (2021). Toward automated cyber defense with secure sharing of structured cyber threat intelligence. *Information Systems Frontiers*, [online] 23(4). doi:https://doi.org/10.1007/s10796-020-10103-7.

Harish Padmanaban (2024). Machine learning algorithms scaling on large-scale data infrastructure. *Deleted Journal*, [online] 3(1), pp.1–26. doi:https://doi.org/10.60087/jaigs.vol03.issue01.p26.

Hart, S., Margheri, A., Paci, F. and Sassone, V. (2020). Riskio: A Serious Game for Cyber Security Awareness and Education. *Computers and Security*, [online] 95(1), p.101827. doi:https://doi.org/10.1016/j.cose.2020.101827.

Haustein, M., Sighart,H. and Titze, D. S. P., 2013. Collaboratively exchanging warning messagesbetween peers while under attack. *2013 International Conference onAvailability, Reliability and Security.*

Hevner, A. (2007). Issue 2 Article 4 2007 Recommended Citation Hevner. *Scandinavian Journal of Information Systems*, [online] 19(2), pp.1–7. Available at: https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1017&context=sjis [Accessed 4 Feb. 2022].

Irei, A. (2025). *How to spot and expose fraudulent North Korean IT workers*. [online] Search Security. Available at: https://www.techtarget.com/searchsecurity/feature/How-to-spot-and-expose-fraudulent-North-Korean-IT-workers [Accessed 8 May 2025].

Javatpoint (2022). *Unsupervised Machine learning - Javatpoint*. [online] www.javatpoint.com. Available at: https://www.javatpoint.com/unsupervised-machine-learning.

Johnson, B. and Christensen, L.B. (2008). *Educational research : quantitative, qualitative, and mixed approaches*. Los Angeles: Sage Publications.

Johnson, C.S., Badger, M.L., Waltermire, D.A., Snyder, J. and Skorupka, C. (2016). Guide to Cyber Threat Information Sharing. *Guide to Cyber Threat Information Sharing*, [online] SP(800-150). doi:https://doi.org/10.6028/nist.sp.800-150.

Julian, T. (2014). *Defining Moments in the History of Cybersecurity*. [online] Infosecurity Magazine. Available at: https://www.infosecurity-magazine.com/opinions/the-history-of-cybersecurity/ [Accessed 2019].

Jurecek, M., (2021) *AutomaticMalware Detection,* s.l.: s.n.

Kabay, M. (2005). *INFOSEC YEAR IN REVIEW 2004*. [online] Norwich University. Available at: https://www.mekabay.com/iyir/2004.pdf [Accessed 10 Aug. 2022].

Kampanakis, P. (2014). Security Automation and Threat Information-Sharing Options. *IEEE Security and Privacy*, [online] 12(5), pp.42–51. doi:https://doi.org/10.1109/msp.2014.99.

Kara, I. (2021). Cyber-Espionage Malware Attacks Detection and Analysis: A Case Study. *Journal of Computer Information Systems*, [online] 62(6), pp.1–18. doi:https://doi.org/10.1080/08874417.2021.2004566.

Kaspersky (2021). *Kaspersky for Business Machine Learning for Malware Detection*. [online] Kaspersky, pp.1–15. Available at: https://media.kaspersky.com/en/enterprise-security/Kaspersky-Lab-Whitepaper-Machine-Learning.pdf [Accessed 23 May 2023].

Keita, Z. (2022). *Classification in Machine Learning: A Guide for Beginners*. [online] Datacamp. Available at: https://www.datacamp.com/blog/classification-machine-learning.

Kim, D. and Kim, H., (2018) *Automated Dataset Generation System for Collaborative Research ofCyber Threat Analysis,* s.l.: Hindawi.

Koepke, P., (2017). *Cyber Security Interdisciplinary Systems Laboratory (CISL): CS Information Sharing Incentivesand Barriers,* Massachusetts Institute of Technology: Sloan School ofManagement, MIT.

Kokkonen, T., Hautamäki, J., Siltanen, J. and Hämäläinen,T., 2016. Model for Sharing the Information of CS Situation Awareness betweenOrganizations. *23rd International Conference on Telecommunications (ICT)*

Korte, K. (2021). *Measuring the quality of Open Source Cyber Threat Intelligence Feeds*. [Thesis [MSc]] pp.1–62. Available at: https://www.theseus.fi/bitstream/handle/10024/500534/Korte_Keijo.pdf [Accessed 9 May 2025].

Krishnan, R., Sandhu,R. and Ranganathan, K., (2007). *PEI Models towards Scalable, Usable andHigh-Assurance Information Sharing,* s.l.: ACM.

Kurtz, G. (2019). *2019 Global Threat Report Shows It Takes Innovation and Speed to Win Against Adversaries*. [online] Crowdstrike.com. Available at: https://www.crowdstrike.com/en-us/blog/2019-global-threat-report-shows-it-takes-innovation-and-speed-to-win-against-adversaries/.

Landage, J. and Wankhade, M.P. (2013). Malware and Malware Detection Techniques: A Survey. *International Journal of Engineering Research and Technology (IJERT)*, [online] 2(12). Available at: https://www.ijert.org/research/malware-and-malware-detection-techniques-a-survey-IJERTV2IS120163.pdf [Accessed 24 Aug. 2020].

Lantz, B. (2013). Machine Learning with R: Learn how to Use R to Apply Powerful Machine Learning Methods and Gain an Insight Into Real-world Applications. United Kingdom: Packt Publishing.

Lapiedra, J. (2000). *Global Information Assurance Certification Paper*. [online] pp.1–8. Available at: https://www.giac.org/paper/gsec/501/information-security-process-prevention-detection-response/101197.

Leedy, P., Ormrod, J., (2001) *Practical research: Planning and design ed.).*. Upper Saddle River,NJ:: s.n.

Lenchner, O., (2020) *Darkreading.*[Online] Available at: https://www.darkreading.com/threat-intelligence/threat-intelligence-through-web-scraping[Accessed 15 03 2022].

Lenchner, O. (2022). *Threat Intelligence Through Web Scraping*. [online] Darkreading.com. Available at: https://www.darkreading.com/threat-intelligence/threat-intelligence-through-web-scraping.

Leyden, J. (2008). *RBS worldpay breach exposes 1.5 million*. [online] Theregister.com. Available at: https://www.theregister.com/2008/12/29/rbs_worldpay_breach/ [Accessed 19 Jun. 2024].

Lipson, H. (2002). *Tracking and Tracing Cyber-Attacks: Technical Challenges and Global Policy Issues*. [online] Pittsburgh: CERT ® Coordination Center. Available at: https://insights.sei.cmu.edu/documents/1827/2002_003_001_13928.pdf.

Lu, D., Fei, J. and Liu, L. (2023). A semantic learning-based SQL injection attack detection technology. *Electronics*, 12(6), p.1344. doi:https://doi.org/10.3390/electronics12061344.

Luzwick, P. (2000). Situational Awareness and OODA Loops: Coherent

Knowledge-based Operations Applied. *Computer Fraud and Security*, [online] 2000(4), pp.15–17. doi:https://doi.org/10.1016/s1361-3723(00)04017-3.

Martin, R. (2008). *MAKING SECURITY MEASURABLE AND MANAGEABLE*. [online] pp.1–11. Available at: https://makingsecuritymeasurable.mitre.org/about/Making_Security_Measurable_and_Manageable.pdf [Accessed 11 Jan. 2023].

Martins, C. and Medeiros, I., 2022. Generating Quality Threat Intelligence Leveraging OSINT anda Cyber Threat Unified Taxonomy. *ACM Transactions on Privacy and Security.*

Martínez Torres, J., Iglesias Comesaña, C. and García-Nieto, P.J. (2019). Review: Machine learning techniques applied to Cybersecurity. *International Journal of Machine Learning and Cybernetics*, [online] 10(10), pp.2823–2836. doi:https://doi.org/10.1007/s13042-018-00906-1.

Mavroeidis, V. and Bromander, S. (2017). *Cyber Threat Intelligence Model: An Evaluation of Taxonomies, Sharing Standards, and Ontologies within Cyber Threat Intelligence*. [online] IEEE Xplore. doi:https://doi.org/10.1109/EISIC.2017.20.

McAfee, (2016) McAfeeLabs Threats Report. *McAfee Labs Threats Report.*

Meng, G., Liu, Y., Zhang, J., Pokluda, A. and Boutaba, R. (2015). Collaborative Security. *ACM Computing Surveys*, [online] 48(1), pp.1–42. doi:https://doi.org/10.1145/2785733.

Mezquita, T. (2020). *Brute force attack*. [online] CyberHoot. Available at: https://cyberhoot.com/cybrary/brute-force-attack/.

Mill, J.S. (1843). *A system of logic, ratiocunative and inductive: Being a connected view of the principles of evidence, and methods of scientific investigation*. [online] *Google.com.ng*, Google Books. Available at:

https://www.google.com.ng/books/edition/A_System_of_Logic_Ratiocinative_and_Indu/y4MEAAAAQAAJ?hl=en&gbpv=1&dq=A+system+of+Logic&printsec=frontcover [Accessed 5 May 2024].

Miller, M. (2024). Password cracking 101: Attacks and defenses explained | beyondtrust. *www.beyondtrust.com*. Available at: https://www.beyondtrust.com/blog/entry/password-cracking-101-attacks-defenses-explained.

Mitchell, T. M., (1997).*M. Learning. Mcgraw-hill science.* s.l.: Engineering.

Mohaisen, A., Al-Ibrahim, O., Kamhoua, C., Kwiat, K. and Njilla, L. (2017). Rethinking information sharing for threat intelligence. In: *Proceedings of the fifth ACM/IEEE Workshop on Hot Topics in Web Systems and Technologies - HotWeb '17*. [online] ACM Digital library, pp.1–7. doi:https://doi.org/10.1145/3132465.3132468.

Motlhabi, M. (2022) *Context-Aware Cyber Threat Intelligence Exchange Platform,* s.l.: Proceedings of the 17thInternational Conference on Information Warfare and Security.

Möller, D.P.F. (2020). *Cybersecurity in Digital Transformation*. *SpringerBriefs on Cyber Security Systems and Networks*. Cham: Springer International Publishing. doi:https://doi.org/10.1007/978-3-030-60570-4.

Murphey, D. (2019). *A history of information security - IFSEC Global | Security and Fire News and Resources*. [online] IFSEC Global | Security and Fire News and Resources. Available at: https://www.ifsecglobal.com/cyber-security/a-history-of-information-security/ [Accessed 21 Nov. 2019].

NATO (2013) *NewsRoom.* [Online] Available at: https://www.NATO.int/cps/en/NATOlive/news_105485.htm[Accessed 23 01 2022].

Navetta, D. and Mathur, U. (2015). *Sharing Cyber Threat Information: A Legal Perspective (ISSA Journal Article)*. [online] Data Protection Report. Available at: https://www.dataprotectionreport.com/2015/01/sharing-cyber-threat-information-a-legal-perspective-issa-journal-article/?__cf_chl_tk=dWmgRQFs9xrhwffTw_Yv1Yr0p9fQSNxJe7o2boQZpSo-1745120253-1.0.1.1-TDaDysCslD8LiVSJOvWPd.ofmqZ52_1sGR2gIE4C.SQ [Accessed 20 Apr. 2025].

NCIAgency (2017) *MalwareInformation Sharing Platform..* [Online] Available at: https://www.ncia.NATO.int/Documents/Agency%20publications/Malware%20Information%20Sharing%20Platform%20(MISP).pdf[Accessed 24 09 2019].

Neuman, W.L. (2013). *Social Research methods: Qualitative and Quantitative Approaches*. 7th ed. [online] Harlow, Essex: Pearson, pp.1–599. Available at: https://letrunghieutvu.yolasite.com/resources/w-lawrence-neuman-social-research-methods_-qualitative-and-quantitative-approaches-pearson-education-limited-2013.pdf [Accessed 30 Sep. 2021].

Nobanee, H., Alodat, A.Y., Bajodah, R., Al-Ali, M. and Al Darmaki, A. (2023). Bibliometric analysis of cybercrime and cybersecurity risks literature. *Journal of Financial Crime*, [online] 30(6). doi:https://doi.org/10.1108/jfc-11-2022-0287.

Nolan, A. (2015). *Cybersecurity and Information Sharing: Legal Challenges and Solutions*. [online] pp.1–62. Available at: https://sgp.fas.org/crs/intel/R43941.pdf [Accessed 5 Jul. 2023].

Nweke, L. and Wolthusen, S., (2020) *Legal Issues Relatedto Cyber Threat Information Sharing Among Private Entities for CriticalInfrastructure Protection,* s.l.: NATO CCDCOE

O'Dowd, A. (2017). *Major global cyber-attack hits NHS and delays treatment*. *BMJ*, p.j2357. doi:https://doi.org/10.1136/bmj.j2357.

Odat, E., Alazzam, B. and Yaseen, Q.M. (2022). Detecting Malware Families and Subfamilies using Machine Learning Algorithms: An Empirical Study. *International Journal of Advanced Computer Science and Applications*, [online] 13(2). doi:https://doi.org/10.14569/ijacsa.2022.0130288.

Official Journal of the European Union (2016). Directive (EU) 2016/1148 of the European Parliament and of the Council. *Europa.eu*. [online] Available at: https://eur-lex.europa.eu/eli/dir/2016/1148/oj/eng.

Olaniran, B.A., Potter, A., Ross, K.A. and Johnson, B. (2014). A Gamer's Nightmare: An Analysis of the Sony PlayStation Hacking Crisis. *Journal of Risk Analysis and Crisis Response*, [online] 4(3), p.151. doi:https://doi.org/10.2991/jrarc.2014.4.3.4.

Panhalkar, T. (2019). *Threat Intelligence Lifecycle*. [online] Infosavvy Security and IT Management Training. Available at: https://info-savvy.com/threat-intelligence-lifecycle/.

Papaspiliopoulos, O. and Roberts, G. (2007). *Stability of the Gibbs Sampler for Bayesian Hierarchical Models*. [online] arXiv.org. Available at: https://arxiv.org/abs/0710.4234 [Accessed 7 May 2022].

Park, S. Y., Lars, and. Anthony, R., (2020). Positivism paradigm research. *Health scienseresearch commons.*

Park, Y.S., Konge, L. and Artino, A.R. (2020). The positivism paradigm of research. *Academic Medicine*, [online] 95(5), pp.690–694. doi:https://doi.org/10.1097/ACM.0000000000003093.

Pathak, P.B. (2016). Malware a growing cybercrime threat: Understanding and combating malvertising attacks. *International Journal of Advanced Research in Computer Science*, [online] 7(2). Available at: www.ijarcs.info [Accessed 22 Apr. 2025].

Patil, D.P. (2015). Challenges and Analysis in Cyber Crime. *International Journal of Computer Science and Mobile Applications*, [online] 3(11), pp.47–50. Available at: https://www.ijcsma.com/abstract/challenges-and-analysis-in-cyber-crime-95492.html [Accessed 6 May 2025].

Pawlinski, P.. (2014). *Actionable information forsecurity incident response,* s.l.: European Union Agency for Network andInformation Security (ENISA),

Peretti, K., (2014) *Cyber Threat Intelligence:To Share or Not to Share—What Are the Real Concerns?,* s.l.: Bloomberg BNA.

Poputa-Clean, P. (2015). *Automated Defense Using Threat Intelligence to Augment Security*. [online] *giac.org*, Global Information Assurance Certification Paper, pp.1–38. Available at: https://www.giac.org/paper/gcih/13998/automated-defense-threat-intelligence-augment/121748 [Accessed 24 Aug. 2024].

Project 2049 Institute (2015). *SPECIAL: Sun Tzu Simplified: An Approach to Analyzing China's Regional Military Strategies*. [online] Project2049.net. Available at: https://project2049.net/2015/04/10/special-sun-tzu-simplified-an-approach-to-analyzing-chinas-regional-military-strategies/.

Racz, N., Weippl, E. and Seufert, A., (2011) *Governance, risk and compliance (GRC) Software-anexploratory study of software vendor and market research perspectives. In 44thHawaii International Conference on Information Sciences (HICSS),* s.l.:IEEE, Los Almitos.

Rafaeli, S. and Ravid, G. (2003). Information sharing as enabler for the virtual team: An experimental approach to assessing the role of electronic mail in disintermediation. *Information Systems Journal*, 13(2), pp.191–206. doi:https://doi.org/10.1046/j.1365-2575.2003.00149.x.

Reneke, P., Grant, C., Bryner, N.P., Jones, A.W. and Koepke, G.H. (2015). *Research Roadmap for Smart Fire Fighting*. [online] *nvlpubs.nist.gov*, National

Institute of Standards and Technology , pp.1–247. doi:https://doi.org/10.6028/nist.sp.1191.

Repschlaeger, J., (2013) *Transparency in cloud business: Cluster analysis of software as a servicecharacteristics. In: International Conference on Grid and Pervasive Computing,,*Berlin Heidelberg: Springer, .

Ronald, C. and Lokaiah, P., (2023). Malware Detection andClassification Using ML Algorithms. *International Journal for Research inApplied Science and Engineering Technology (IJRASET),* Volume 11 IssueVIII (2) Department of Computer Science and Engineering, Faculty of Engineeringand Technology JAIN (Deemed-to-be University),

Rookard, C. and Khojandi, A. (2024). RRIoT: Recurrent reinforcement learning for cyber threat detection on iot devices. *Computers and Security*, [online] 140, p.103786. doi:https://doi.org/10.1016/j.cose.2024.103786.

SailPoint (2023). *What is data accuracy? Definition, importance, and best practices*. [online] SailPoint. Available at: https://www.sailpoint.com/identity-library/data-accuracy.

Sander, T. and Hailpern, J. (2015). UX Aspects of Threat Information Sharing Platforms. *Proceedings of the 2nd ACM Workshop on Information Sharing and Collaborative Security*, [online] pp.51–59. doi:https://doi.org/10.1145/2808128.2808136.

Sanka, S., Gochhayat, Dr.S.P., Torremoch, V. and Kethar, J. (2024). Accuracy of AI in Cyber Security: Navigating the Dual-Edged Dynamics of an Evolving Security Frontier. *Journal of Student Research*, 13(2). doi:https://doi.org/10.47611/jsrhs.v13i2.6773.

SANS (2024). *Top 25 software errors | SANS institute*. [online] www.sans.org. Available at: https://www.sans.org/top25-software-errors/.

Sarkar, D., Bali, R. and and Sharma, T., (2018) *PracticalML with Python.*. New York: : Apress

Sarker, I.H., Kayes, A.S.M., Badsha, S., Alqahtani, H., Watters, P. and Ng, A. (2020). Cybersecurity data science: An overview from machine learning perspective. *Journal of Big Data*, [online] 7(1). doi:https://doi.org/10.1186/s40537-020-00318-5..

Sauerwein, C., Sillaber, C., Mussmann, A. and Breu, R. (2017). Threat Intelligence Sharing Platforms: An Exploratory Study of Software Vendors and Research Perspectives. In: *Computer Science*. [online] https://link.springer.com/journal/11576: Wirtschaftsinformatik. Available at: https://www.wi2017.ch/images/wi2017-0188.pdf.

Saunders, M.N.K., Lewis, P. and Thornhill, A. (2007). *Research methods for business students*. Essex: Pearson Education.

Sciarra, D. (1999). The Role of the Qualitative Researcher. In: M. Kopala and L.A. Suzuki, eds., *Using Qualitative Methods in Psychology*. pp.37–48. doi:https://doi.org/10.4135/9781452225487.n4.

SecurityGroup (2013). *Department of Computer Science and Technology – Security Group: Banking security*. [online] Cam.ac.uk. Available at: http://www.cl.cam.ac.uk/research/security/banking/ [Accessed 7 May 2019].

Sehrawat, S. and Singh, D. D., (2022) Malware and Malware Detection Techniques: A survey.May.Volume 10 (May 2022).

Shahzad, R. M. K.,(2024) *Automated Malware Detection and Classification Using SupervisedLearning,* s.l.: Department of Computer Science Blekinge Institute ofTechnology.

Shalev-Shwartz andS. Ben-David, (2014) *Understanding Machine Learning: From*

theory to algorithms., NewYork: Cambridge university press.

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding machine learning : from foundations to algorithms*. [online] Cambridge Etc: Cambridge University Press. Available at: https://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/understanding-machine-learning-theory-algorithms.pdf [Accessed 2 Jan. 2025].

Sharp, R. (2007). *An Introduction to Malware*. [online] Available at: https://backend.orbit.dtu.dk/ws/portalfiles/portal/4918204/malware.pdf.

Sigholm, J. and Bang, M. (2013). Towards Offensive Cyber Counterintelligence: Adopting a Target-Centric View on Advanced Persistent Threats. In: *2013 European Intelligence and Security Informatics Conference*. [online] United States: IEEE Computer Society, pp.166–171. doi:https://doi.org/10.1109/eisic.2013.37.

Sillaber, C., Sauerwein, C., Mussmann, A. and Breu, R. (2016). Data Quality Challenges and Future Research Directions in Threat Intelligence Sharing Practice. In: *WISCS '16: Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security*. [online] New York, NY, USA: Association for Computing Machinery. Available at: https://doi.org/10.1145/2994539.2994546.

Simon, H.A. (1996). *The sciences of the artificial*. 3rd ed. Cambridge, Mass.: Mit Press.

Siri Bromander, Swimmer, M., Eian, M., Geir Skjotskift and van (2020). Modeling Cyber Threat Intelligence. *Duo Research Archive (University of Oslo)*, [online] pp.273–280. doi:https://doi.org/10.5220/0008875302730280.

Skoudis, E. and Zeltser, L. (2003). *Malware fighting malicious code*. [online] Upper Saddle River, Nj Prentice Hall Ptr, p.672. Available at: https://elhacker.info/manuales/An%C3%A1lisis%20de%20malware/Malware.Fighting.Malicious.Code.pdf.

SlideTeam (2020). *What is supervised machine learning input raw ppt powerpoint presentation ideas designs*. [online] Slideteam.net. Available at: https://www.slideteam.net/what-is-supervised-machine-learning-input-raw-ppt-powerpoint-presentation-ideas-designs.html [Accessed 1 Mar. 2022].

Smithu, B.S., Leela, C.P. and Nagashree, N. (2021). SQL injection vulnerability analysis. *World Journal of Advanced Research and Reviews*, 9(1), pp.312–318. doi:https://doi.org/10.30574/wjarr.2021.9.1.0018.

Song, W., Jiang, M., Yan, H., Xiang, Y., Chen, Y., Luo, Y., He, K. and Peng, G. (2020). Android data-clone attack via operating system customization. *IEEE Access*, [online] 8(199733-199746), pp.199733–199746. doi:https://doi.org/10.1109/access.2020.3035089.

Sood, A.K. (2014). *Targeted cyber attacks : multi-staged attacks driven by exploits and malware*. [online] Syngress Uuuu-Uuuu. Available at: https://shop.elsevier.com/books/targeted-cyber-attacks/sood/978-0-12-800604-7.

Spiegel, Y. (2013). Commercial software, adware, and consumer privacy. *International Journal of Industrial Organization*, 31(6), pp.702–713. doi:https://doi.org/10.1016/j.ijindorg.2013.03.001.

Steinberger, J., Sperotto, A., Baier, H. and Pras, A. (2020). Distributed DDoS Defense:A collaborative Approach at Internet Scale. *Zenodo (CERN European Organization for Nuclear Research)*. [online] doi:https://doi.org/10.1109/noms47738.2020.9110300.

Stillions, R. (2014). *The DML model*. [online] Ryan Stillions. Available at: https://ryanstillions.blogspot.com/2014/04/the-dml-model_21.html.

Sukhabogia, Sandhya.;Anushab, Dr. M., 2021. A Theoretical review on the importance of ThreatIntelligence Sharing and The challenges intricated. *Turkish Journal ofComputer and Mathematics Education.*

Sullivan, B. (2016). *Exchanging Cyber Threat Intelligence: There Has to Be a Better Way | Ponemon-Sullivan Privacy Report*. [online] Ponemonsullivanreport.com. Available at: https://ponemonsullivanreport.com/2016/01/exchanging-cyber-threat-intelligence-there-has-to-be-a-better-way/ [Accessed 9 May 2022].

Sweeny, L. (2002). k-ANONYMITY: a MODEL FOR PROTECTING PRIVACY. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, [online] 10(05), pp.557–570. doi:https://doi.org/10.1142/s0218488502001648.

Tagliaferri, L., (2017) An Introduction to Machine Learning *DigitalOcean.*

Terekhov, A. (2017). *History of the Antivirus*. [online] Hotspot Shield VPN. Available at: https://www.hotspotshield.com/blog/history-of-the-antivirus/ [Accessed 9 Nov. 2023].

Thomasian, N.M. and Adashi, E.Y. (2021). Cybersecurity in the Internet of Medical Things. *Health Policy and Technology*, [online] 10(3), p.100549. doi:https://doi.org/10.1016/j.hlpt.2021.100549.

Tosh, D. M., Molloy, S. S., Kamhoua, C. and Kwiat, K., 2015.Cyber-Investment and Cyber-Information Exchange Decision Modeling. *2015 IEEE17th International Conference on High Performance Computing and Communications,2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and2015 IEEE 12th International Conference on Embedded Software and Systems :24-26 August* .

Townsend, K. (2019). *Analyzing 2018 Attacks to Prepare for Those in 2019*. [online] SecurityWeek. Available at: https://www.securityweek.com/analyzing-2018-attacks-prepare-those-2019/ [Accessed 6 May 2025].

Trabelsi, S., Mahmoud, S.B. and Zouaoui, A. (2016). Predictive model for exploit kit based attacks. In: *Proceedings of the 13th International Joint Conference on e-Business and Telecommunications.* [online] Proceedings of the 13th International

Joint Conference on e-Business and Telecommunications. Lisbon, Portugal: SciTePress, pp.477–482. Available at: https://www.scitepress.org/Link.aspx?doi=10.5220/0005999904770482 [Accessed 21 Apr. 2025].

Tyler, W. (2022). *Industrial Cybersecurity Archives - Control Engineering*. [online] Control Engineering. Available at: https://www.controleng.com/industrial-cyber-security/.

Ullman, G. (2016). *Report: Industry eager to receive, hesitant to share cyber threat data*. [online] FedScoop. Available at: https://fedscoop.com/report-industry-eager-to-receive-hesitant-to-share-cyber-threat-intelligence/ [Accessed 5 May 2025].

UNODC (2010). *Cybercrime*. [online] *United Nation Office of Drug Crime*. United Nations. Available at: https://www.unodc.org/documents/data-and-analysis/tocta/10.Cybercrime.pdf [Accessed 11 Oct. 2024].

Vandure, A., (2017) *Malware Information Sharing Platform,* s.l.:NATO information and telecommunication agency

Vázquez, DF, Acosta, OP, Spirito, C, Brown, S and Reid, E 2012, Conceptual framework for cyber defense information sharing within trust relationships. in *2012 4th International Conference on Cyber Conflict (CYCON 2012).*, 6243990, IEEE, pp. 1-17, 2012 4th International Conference on Cyber Conflict (CYCON 2012), 06/5/12. <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6243990>

Wagner, T.D., Mahbub, K., Palomar, E. and Abdallah, A.E. (2019). Cyber threat intelligence sharing: Survey and research directions. *Computers and Security*, [online] 87(87), p.101589. doi:https://doi.org/10.1016/j.cose.2019.101589.

Wang , Z., Adeyemo , O.D. and Akinrayo, A.E. (2023). Summary of cyber threat intelligence. *International Journal for Innovative Research in Multidisciplinary Field*, [online] 8(3), pp.32–42. doi:https://doi.org/10.2015/IJIRMF/202203006.

Watson, J.A. and Holmes, C.C. (2020). Machine learning analysis plans for randomised controlled trials: Detecting treatment effect heterogeneity with strict control of type I error. *Trials*, [online] 21(1). doi:https://doi.org/10.1186/s13063-020-4076-y.

Wickr (2021) *AWS.* [Online]Available at: https://wickr.com/the-pros-and-cons-of-threat-information-sharing/

Williams, C., (2011) Research methods. Journal of Business and Economics Research. p. 5(3).

Woolf, N. (2016). *DDoS attack that disrupted internet was largest of its kind in history, experts say*. [online] The Guardian. Available at: https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet.

Xue, D., Li, J., Lv, T., Wu, W. and Wang, J. (2019). Malware Classification Using Probability Scoring and Machine Learning. *IEEE Access*, [online] 7, pp.91641–91656. doi:https://doi.org/10.1109/access.2019.2927552.

Yeboah-Boateng, E. O., (2018) Cybersecurity Intelligence Gathering: Issues With Knowledge Management.In: o*rganizational learning.* | *Knowledge management.* | *Digital media..* s.l.:s.n.,pp. Library of Congress Cataloging-in-Publication Data.

Zahra, B., Hashem, H.,Fard, S. M. H. and Ali, H., (2013) *A Survey on Heuristic Malware Detection Techniques,* s.l.: 5th Conference on Information and Knowledge Technology(IKT).

Zheng, D.E. and Lewis, J.A. (2015). *Cyber Threat Information Sharing Recommendations for Congress and the Administration A Report of the CSIS Strategic Technologies Program*. [online] Washington, DC 20036: Center for Strategic and International Studies (CSIS), pp.1–18. Available at: https://csis-website-prod.s3.amazonaws.com/s3fs-

public/legacy_files/files/publication/150310_cyberthreatinfosharing.pdf [Accessed 14 Mar. 2022].

Zrahia, A., (2018) *Threat intelligence sharing between Cyber Security vendors: Network, dyadic, and agent views,* TelAviv: Journal of Cyber Security

LIST OF APPENDICIES

MODES PSDCIS DEVELOPMENT CODES

Appendix 1: Dataset 1 Model Development Training and Testing Code

```
import pandas as pd

import time

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.tree import DecisionTreeClassifier

from sklearn.linear_model import LogisticRegression, SGDClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive_bayes import GaussianNB

from sklearn.ensemble import RandomForestClassifier

from lightgbm import LGBMClassifier

from xgboost import XGBClassifier

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
roc_auc_score, confusion_matrix

import seaborn as sns

import matplotlib.pyplot as plt

# Load and preprocess the dataset

df = pd.read_csv('data_011.csv')  # Replace with your actual dataset path

df['classification'] = df['classification'].map({'benign': 0, 'malware': 1})

df = df.drop(columns=['hash'], errors='ignore')

df = df.select_dtypes(include=[float, int])

X = df.drop(columns=['classification'], errors='ignore')

y = df['classification']

# Split the data into 80% training and 20% testing

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature scaling for models

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)

# Defined the models to evaluate
```

```
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "Logistic Regression": LogisticRegression(random_state=42),
    "K-Nearest Neighbors": KNeighborsClassifier(),
    "Naive Bayes": GaussianNB(),
    "Stochastic Gradient Descent": SGDClassifier(loss='log_loss', random_state=42),
    "Random Forest": RandomForestClassifier(n_estimators=100, random_state=42),
    "LightGBM": LGBMClassifier(n_estimators=100, random_state=42),
    "XGBoost": XGBClassifier(use_label_encoder=False, eval_metric='logloss',
n_estimators=100, random_state=42)
}
# Function to evaluate a model and print results
def evaluate_model(name, model, X_train, X_test, y_train, y_test, scaled=True):
    print(f"\n--- {name} ---")
    if scaled:
        X_train_input, X_test_input = X_train_scaled, X_test_scaled
    else:
        X_train_input, X_test_input = X_train, X_test
    # Training
    model.fit(X_train_input, y_train)
    # Training metrics
    y_train_pred = model.predict(X_train_input)
    train_accuracy = accuracy_score(y_train, y_train_pred)
    train_precision = precision_score(y_train, y_train_pred)
    train_recall = recall_score(y_train, y_train_pred)
    train_f1 = f1_score(y_train, y_train_pred)
    train_roc_auc = roc_auc_score(y_train, y_train_pred)
```

```
    print("\n--- Training Results ---")
    print(f"Accuracy: {train_accuracy:.4f}, Precision: {train_precision:.4f}, Recall: {train_recall:.4f}, F1 Score: {train_f1:.4f}, ROC-AUC: {train_roc_auc:.4f}")

    # Training confusion matrix
    cm_train = confusion_matrix(y_train, y_train_pred)
    plt.figure(figsize=(6, 5))
    sns.heatmap(cm_train, annot=True, fmt='d', cmap='Blues', cbar=False,
                xticklabels=['Predicted Benign', 'Predicted Malware'],
                yticklabels=['Actual Benign', 'Actual Malware'])
    plt.title(f'Confusion Matrix - {name} (Training Data)')
    plt.xlabel('Predicted Labels')
    plt.ylabel('True Labels')
    plt.show()

    # Testing metrics
    start_time = time.time()
    y_test_pred = model.predict(X_test_input)
    end_time = time.time()
    prediction_time = end_time - start_time

    test_accuracy = accuracy_score(y_test, y_test_pred)
    test_precision = precision_score(y_test, y_test_pred)
    test_recall = recall_score(y_test, y_test_pred)
    test_f1 = f1_score(y_test, y_test_pred)
    test_roc_auc = roc_auc_score(y_test, y_test_pred)

    print("\n--- Testing Results ---")
```

Appendix 2: Dataset 2 Models Development Training Testing Code

```
import pandas as pd
import time
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression, SGDClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from lightgbm import LGBMClassifier
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
roc_auc_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
# Load and preprocess the dataset
df = pd.read_csv('Malware_dataset.csv')  #
df['classification'] = df['classification'].map({'benign': 0, 'malware': 1})
df = df.drop(columns=['hash'], errors='ignore')
df = df.select_dtypes(include=[float, int])
X = df.drop(columns=['classification'], errors='ignore')
y = df['classification']
# Split the data into 80% training and 20% testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Feature scaling for models that benefit from scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
# Defined  models to evaluate
models = {
   "Decision Tree": DecisionTreeClassifier(random_state=42),
   "Logistic Regression": LogisticRegression(random_state=42),
   "K-Nearest Neighbors": KNeighborsClassifier(),
   "Naive Bayes": GaussianNB(),
   "Stochastic Gradient Descent": SGDClassifier(loss='log_loss', random_state=42),
   "Random Forest": RandomForestClassifier(n_estimators=100, random_state=42),
   "LightGBM": LGBMClassifier(n_estimators=100, random_state=42),
   "XGBoost": XGBClassifier(use_label_encoder=False,  eval_metric='logloss',
n_estimators=100, random_state=42)
}
# Functionality to evaluate a model and print results
def evaluate_model(name, model, X_train, X_test, y_train, y_test, scaled=True):
   print(f"\n--- {name} ---")
   if scaled:
      X_train_input, X_test_input = X_train_scaled, X_test_scaled
   else:
      X_train_input, X_test_input = X_train, X_test
   # Training
   model.fit(X_train_input, y_train)
   # Training metrics
   y_train_pred = model.predict(X_train_input)
   train_accuracy = accuracy_score(y_train, y_train_pred)
   train_precision = precision_score(y_train, y_train_pred)
   train_recall = recall_score(y_train, y_train_pred)
   train_f1 = f1_score(y_train, y_train_pred)
   train_roc_auc = roc_auc_score(y_train, y_train_pred)
   print("\n--- Training Results ---")
```

```
    # Training confusion matrix
    cm_train = confusion_matrix(y_train, y_train_pred)
    plt.figure(figsize=(6, 5))
    sns.heatmap(cm_train, annot=True, fmt='d', cmap='Blues', cbar=False,
                xticklabels=['Predicted Benign', 'Predicted Malware'],
                yticklabels=['Actual Benign', 'Actual Malware'])
    plt.title(f'Confusion Matrix - {name} (Training Data)')
    plt.xlabel('Predicted Labels')
    plt.ylabel('True Labels')
    plt.show()
    # Testing metrics
    start_time = time.time()
    y_test_pred = model.predict(X_test_input)
    end_time = time.time()
    prediction_time = end_time - start_time
    test_accuracy = accuracy_score(y_test, y_test_pred)
    test_precision = precision_score(y_test, y_test_pred)
    test_recall = recall_score(y_test, y_test_pred)
    test_f1 = f1_score(y_test, y_test_pred)
    test_roc_auc = roc_auc_score(y_test, y_test_pred)
    print("\n--- Testing Results ---")
    print(f"Accuracy: {test_accuracy:.4f}, Precision: {test_precision:.4f}, Recall:
{test_recall:.4f}, F1 Score: {test_f1:.4f}, ROC-AUC: {test_roc_auc:.4f}")
    print(f"Prediction Time: {prediction_time:.4f} seconds (or {prediction_time * 1000:.2f}
milliseconds)")
```

```
    # Testing confusion matrix
    cm_test = confusion_matrix(y_test, y_test_pred)
    plt.figure(figsize=(6, 5))
    sns.heatmap(cm_test, annot=True, fmt='d', cmap='Blues', cbar=False,
                xticklabels=['Predicted Benign', 'Predicted Malware'],
                yticklabels=['Actual Benign', 'Actual Malware'])
    plt.title(f'Confusion Matrix - {name} (Testing Data)')
    plt.xlabel('Predicted Labels')
    plt.ylabel('True Labels')
    plt.show()
    # Print TP, TN, FP, FN values for Testing Data
    tn, fp, fn, tp = cm_test.ravel()
    print(f"\nTesting Confusion Matrix Details:")
    print(f"True Negatives (TN): {tn}")
    print(f"False Positives (FP): {fp}")
    print(f"False Negatives (FN): {fn}")
    print(f"True Positives (TP): {tp}")
# Models valuate
for model_name, model in models.items():
    # Used scaled  specific models
    if model_name in ['K-Nearest Neighbors', 'Logistic Regression', 'Stochastic Gradient
Descent', 'LightGBM', 'XGBoost']:
        evaluate_model(model_name, model, X_train, X_test, y_train, y_test, scaled=True)
    else:
        evaluate_model(model_name, model, X_train, X_test, y_train, y_test, scaled=False)
```

Appendix 3: Initiation State of PSDCIS Code

```
import itertools

import flask

from flask import Response, escape, abort,render_template, make_response, Markup, request

from subprocess import Popen, PIPE, STDOUT

import subprocess

import sys

import os

import threading

from pathlib import Path

import glob

import pandas as pd

from flask_paginate import Pagination, get_page_args

app = flask.Flask(__name__)

SENTINEL = '------------SPLIT----------HERE---------'

VALID_ACTIONS = ('what', 'ever')

def logview(logdata):

   """Render the template used for viewing logs."""

   # Probably a lot of other parameters here; this is simplified

   return render_template('logview.html', logdata=logdata)

def stream(first, generator, last):

   """Preprocess output prior to streaming."""

   yield first

   for line in generator:

      yield Markup(f'<pre class="scroll">{line.decode("utf-8")}</pre>')  # Don't let subproc break our HTML

   yield last

@app.route('/subprocess', methods=['GET','POST'])
```

```
def perform_action():
   #if reqeust method is post - means button is clicked and we want to run the statemachine
   #other wise just display logview.html
   if request.method == "POST":
      """Call subprocess and stream output directly to clients."""
      first, _, last = logview(SENTINEL).partition(SENTINEL)
      import pathlib
      file_path = pathlib.Path(__file__).parent.resolve()
      path = os.path.join(file_path,'main.py')
      proc = Popen([sys.executable, path], stdout=PIPE, stderr=STDOUT)
      generator = stream(first, iter(proc.stdout.readline, b''), last)
      gen1, gen2 = itertools.tee(generator)
      # last_logs = threading.Thread(target=create_last_logs, name="writer", args=generator2)
      # last_logs.start()
      try:
         return  make_response(gen1)
      finally:
         create_last_logs(gen2)
   else:
      return render_template("last_logs.html")
   #return render_template("index.html", data=generator)
@app.route('/subprocess', methods=['PUT'])
def show_log():
   """Show one full log."""
   # print(action)
   # if action not in VALID_ACTIONS:
   #    abort(400)
```

```
@app.route('/')
def home():
    return render_template("index.html")
@app.route('/results')
def results():
    return render_template("last_results.html")
@app.route('/output')
def output():
   path = Path(__file__).parent.absolute()
   list_of_files = glob.glob(os.path.join(path, 'downloads/*')) # * means all if need specific format then *.csv
   latest_file = max(list_of_files, key=os.path.getctime)
   print(latest_file)
   data = pd.read_csv(latest_file)
   page = request.args.get('page', 1, type=int)
   per_page = 25
   start = (page - 1)* per_page
   end = start + per_page
   total_pages =(len(data) + per_page - 1) // per_page
   items_per_page = data[start:end]
   print(items_per_page)
   print(start)
   print(end)
   # row_data=list(data.head().values.tolist())
   data = pd.DataFrame.from_records(items_per_page)
   return render_template('table.html', headers=data.columns.values.tolist(), items_per_page=items_per_page.values.tolist(), total_pages=total_pages, page=page )
def create_last_logs(generator):
   # make last log
```

```
    with open('./templates/last_logs.html', 'a') as out:
        out.write("{% extends 'base.html' %} {% block content %}\n")
    for i in generator:
        with open('./templates/last_logs.html', 'a') as out:
            out.write(f"{i}\n")
    with open('./templates/last_logs.html', 'a') as out:
        out.write("{% endblock %}\n")
if __name__ == '__main__':
    app.run(debug=True, port=5001, host='0.0.0.0')
```

Appendix 4 Initiation State

```
import logging
from .state import State
from utils import database as db
from utils.database import Sql
logger = logging.getLogger()
TABLES = ["int_data", "batch"]
class InitialState(State):
   def on_event(self, event):
      if event == "Initial":
         try:
            #main database
            tables = [
               table
               for table, in db.select(
                  "SELECT name FROM sqlite_master WHERE type='table'ORDER BY name;"
               )
            ]
            for name in TABLES:
               if name not in tables:
                  logger.debug(
                     f"table missing run statement [....{Sql.create(name)}.....]"
```

```
                )
                db.execute(Sql.create(name))
            else:
                logger.debug(f"table already exist {name}")
        # secondary database
        tables = [
            table
            for table, in db.select(
                "SELECT name FROM sqlite_master WHERE type='table'ORDER BY name;",
                "database2"
            )
        ]
        if name not in tables:
            logger.debug(
                f"table missing run statement [....{Sql.create(name)}.....]"
            )
            db.execute(Sql.create(name))
        else:
            logger.debug(f"table already exist {name}")
        return self.on_next("LoadVirusShareState", "Process virusshare")
    except Exception as exp:
        logger.error(
            f"Error occuried in state: {self.__str__} error: {exp}",
            exc_info=True,
        )
        return self.on_next("ErrorState", "InitialError")
```

Appendix 4: Stage 1 Load Virus State code

```
import logging
from .state import State
from utils import database as db
from utils.database import Sql
import requests as req
from bs4 import BeautifulSoup as bs
import urllib.parse
logger = logging.getLogger()
class LoadVirusShareState(State):
    def on_event(self, event):
        db.execute("INSERT INTO batch (status) values('Inprogress')")
        batch_id = db.select("SELECT * FROM batch WHERE status=='Inprogress'")[0][0]
        black_list = ["hashes/unpacked_hashes.md5"]
        try:
            url = "https://virusshare.com/hashes.4n6"
            soup = self.get_soup(url)
            pass
            for link in soup.findAll("a", href=True):
                link = urllib.parse.urljoin(url, link["href"])
                if link.endswith("00000.md5"):
```

```
        page = self.get_soup(link)
            md5s = page.body.p.text
            for md5 in md5s.splitlines()[6:-1]:
              if len(md5) == 64:
                logger.info(f"{link} - {md5}")
                db.execute(
                  f"INSERT OR IGNORE INTO int_data (md5sum, batch_id) VALUES('{md5}',
'{batch_id}')"
                )
        db.execute(f"UPDATE batch SET status='Completed' WHERE id={batch_id}")
        return self.on_next("KaggleDownloadState", "Download file from kaggle")
      except Exception as exp:
        db.execute(f"UPDATE batch SET status='Failed' WHERE id={batch_id}")
        db.execute(f"DELETE FROM int_data WHERE batch_id={batch_id}")
        logger.error(
          f"Error occuried in state: {self.__str__} error: {exp}", exc_info=True
        )
        return self.on_next("ErrorState", "virusShareError")
    def get_soup(self, url):
      response = req.get(url)
      return bs(response.text, "lxml")
```

Appendix 5: Stage 1 Kaggle Download State code

```
import logging
import os
from .state import State
from utils import database as db
from utils.database import Sql
import requests as req
from requests.auth import HTTPBasicAuth
from bs4 import BeautifulSoup as bs
import urllib.parse
```

```
from kaggle.api.kaggle_api_extended import KaggleApi
from os.path import expanduser
import zipfile
logger = logging.getLogger()
class KaggleDownloadState(State):
  def on_event(self, event):
    try:
      logger.info("download malware-detection csv")
      path =
os.path.abspath(os.path.join(os.path.dirname(os.path.abspath(__file__)),"downloads"))
      api = KaggleApi()
      api.authenticate()
      api.dataset_download_files("nsaravana/malware-detection/code", path=path)
      logger.info(f"unzipping downloaded file")
      with zipfile.ZipFile(os.path.join(path,'malware-detection.zip'), "r") as zip_ref:
        zip_ref.extractall(path)

      return self.on_next("RunAlgorithmState", "algorithm execution")
    except Exception as exp:
      logger.error(
        f"Error occuried in state: {self.__str__} error: {exp}", exc_info=True
      )
      return self.on_next("ErrorState", "Kaggle Download Error")
```

Appendix 6: Stage 2 RunAlgorithm State

```
import logging
import os
import time
from .state import State
from utils import database as db
from utils.database import Sql
import pandas as pd
```

```
import csv
import re
from datetime import datetime
from io import BytesIO
import base64
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
roc_auc_score, confusion_matrix, matthews_corrcoef, cohen_kappa_score, log_loss,
balanced_accuracy_score
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from sklearn.model_selection import cross_val_score, StratifiedKFold, LeaveOneOut
from sklearn.inspection import permutation_importance
from sklearn.utils import resample
from scipy.stats import ks_2samp
import shap
import matplotlib.pyplot as plt
import seaborn as sns
logger = logging.getLogger()
class RunAlgorithmState(State):
   def on_event(self, event):
      self.write_header()
      self.create_to_page("<h2>------------------------Results-------------------------</h2>")
      try:
         # Fetch and replace dataset with MD5 hashes
         db_data = [i[0] for i in db.select("SELECT md5sum FROM int_data limit 50000")]
         path = os.path.abspath(
            os.path.join(
               f"{os.path.dirname(__file__)}",
```

```
        out_put = os.path.abspath(
            os.path.join(
                f"{os.path.dirname(__file__)}",
                r"..",
                r"downloads",
                r"test_out.csv",
            )
        )
        j = 0
        with open(path, 'r', encoding='utf-8') as in_file, open(out_put, 'w', encoding='utf-8') as out_file:
            data = [row for row in csv.reader(in_file)]
            for i in range(len(data)):
                if re.findall(r"([a-fA-F\d]{32})", data[i][0]):
                    data[i][0] = db_data[j]
                    j += 1
            writer = csv.writer(out_file)
            writer.writerows(data)
        # Load and preprocess dataset
        logger.info(f"Reading CSV from {out_put} for event {event}")
        df = pd.read_csv(r"{}".format(out_put), encoding='utf-8')
        df["classification"].replace(to_replace=["benign", "malware"], value=[0, 1], inplace=True)
        logger.info("Dataset successfully loaded and preprocessed.")
        print(f"Dataset shape: {df.shape}")
        print(f"Null values:\n{df.isnull().sum()}")
        print(f"Target value distribution:\n{df['classification'].value_counts()}")
        # Visualize dataset distribution
        plt.figure(figsize=(8, 6))
```

```
plt.savefig("./templates/dataset_distribution.png")
plt.show()


# Prepare features and labels
x = df.drop(
   ["hash", "classification", "vm_truncate_count", "shared_vm", "exec_vm",
"nvcsw", "maj_flt", "utime"],
   axis=1,
)
y = df["classification"]
# Helper function to evaluate models
def evaluate_model(model_name, y_true, y_pred, y_proba, duration):
   metrics = {
      "Model": model_name,
      "Accuracy": accuracy_score(y_true, y_pred),
      "Precision": precision_score(y_true, y_pred, zero_division=0),
      "Recall": recall_score(y_true, y_pred),
      "F1-Score": f1_score(y_true, y_pred),
      "ROC-AUC": roc_auc_score(y_true, y_proba),
      "MCC": matthews_corrcoef(y_true, y_pred),
      "Cohen's Kappa": cohen_kappa_score(y_true, y_pred),
      "LogLoss": log_loss(y_true, y_proba),
      "Balanced Accuracy": balanced_accuracy_score(y_true, y_pred),
      "Duration (s)": duration,
   }
   # Plot confusion matrix
   cm = confusion_matrix(y_true, y_pred)
   plt.figure(figsize=(6, 5))
```

```
    plt.savefig(f"./templates/confusion_matrix_{model_name}.png")
    plt.show()
    return metrics
def cross_validation_analysis(model, x, y, model_name):
    kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
    scores = cross_val_score(model, x, y, cv=kfold, scoring='accuracy')
    print(f"{model_name} Cross-Validation Scores: {scores}")
    print(f"{model_name} Mean Accuracy: {scores.mean():.4f}")
    print(f"{model_name} Standard Deviation: {scores.std():.4f}")
    return {"CV_Mean_Accuracy": scores.mean(), "CV_Std_Dev": scores.std()}
def ks_stat_analysis(y_true, y_proba):
    ks_stat, ks_pvalue = ks_2samp(y_proba[y_true == 0], y_proba[y_true == 1])
    print(f"KS Statistic: {ks_stat:.4f}, p-value: {ks_pvalue:.4f}")
    return {"KS_Statistic": ks_stat, "KS_p-value": ks_pvalue}
def shap_analysis(model, x):
    explainer = shap.TreeExplainer(model)
    shap_values = explainer.shap_values(x)
    shap.summary_plot(shap_values, x, show=False)
    plt.savefig(f"./templates/shap_summary_{type(model).__name__}.png")
    plt.close()
# Initialize results storage
results = []
# Decision Tree Classifier (Training on all data)
start_time = time.time()
dt_model = DecisionTreeClassifier()
dt_model.fit(x, y)
y_pred_dt = dt_model.predict(x)
y_proba_dt = dt_model.predict_proba(x)[:, 1]
```

```
# Decision Tree Classifier (K-Fold Cross-Validation)
        dt_cv_results = cross_validation_analysis(dt_model, x, y, "Decision Tree (K-Fold)")
        dt_metrics.update(dt_cv_results)
        results.append(dt_metrics)
        # XGBoost Classifier (Training on all data)
        start_time = time.time()
        xgb_model = XGBClassifier()
        xgb_model.fit(x, y)
        y_pred_xgb = xgb_model.predict(x)
        y_proba_xgb = xgb_model.predict_proba(x)[:, 1]
        xgb_metrics = evaluate_model("XGBoost (All Data)", y, y_pred_xgb, y_proba_xgb,
time.time() - start_time)
        xgb_metrics.update(ks_stat_analysis(y, y_proba_xgb))
        shap_analysis(xgb_model, x)
        results.append(xgb_metrics)
        # XGBoost Classifier (K-Fold Cross-Validation)
        xgb_cv_results = cross_validation_analysis(xgb_model, x, y, "XGBoost (K-Fold)")
        xgb_metrics.update(xgb_cv_results)
        results.append(xgb_metrics)
        # LightGBM Classifier (Training on all data)
        start_time = time.time()
        lgbm_model = LGBMClassifier()
        lgbm_model.fit(x, y)
        y_pred_lgbm = lgbm_model.predict(x)
        y_proba_lgbm = lgbm_model.predict_proba(x)[:, 1]
        lgbm_metrics = evaluate_model("LightGBM (All Data)", y, y_pred_lgbm,
y_proba_lgbm, time.time() - start_time)
        lgbm_metrics.update(ks_stat_analysis(y, y_proba_lgbm))
        shap_analysis(lgbm_model, x)
```

```
        # LightGBM Classifier (K-Fold Cross-Validation)
        lgbm_cv_results = cross_validation_analysis(lgbm_model, x, y, "LightGBM (K-Fold)")
        lgbm_metrics.update(lgbm_cv_results)
        results.append(lgbm_metrics)
        # Display and log results
        results_df = pd.DataFrame(results)
        print(results_df)
        logger.info("Model evaluation completed successfully.")
        # Save results as an HTML table
        metrics_table = (
            f'<table class="stripes"><thead><tr>'
            f'<th>Model</th><th>Accuracy</th><th>Precision</th><th>Recall</th>'
            f'<th>F1-Score</th><th>ROC-AUC</th><th>MCC</th><th>Cohen’s Kappa</th><th>LogLoss</th><th>Balanced Accuracy</th>'
            f'<th>Duration (s)</th><th>CV_Mean_Accuracy</th><th>CV_Std_Dev</th><th>KS_Statistic</th><th>KS_p-value</th>'
            f'</tr></thead><tbody>'
            + ''.join(
                f'<tr><td>{row["Model"]}</td>'
                f'<td>{row["Accuracy"]:.4f}</td>'
                f'<td>{row["Precision"]:.4f}</td>'
                f'<td>{row["Recall"]:.4f}</td>'
                f'<td>{row["F1-Score"]:.4f}</td>'
                f'<td>{row["ROC-AUC"]:.4f}</td>'
                f'<td>{row["MCC"]:.4f}</td>'
                f'<td>{row["Cohen’s Kappa"]:.4f}</td>'
                f'<td>{row["LogLoss"]:.4f}</td>'
                f'<td>{row["Balanced Accuracy"]:.4f}</td>'
```

```
            f'<td>{row["KS_p-value"]:.4f}</td></tr>'
            for _, row in results_df.iterrows()
        )
        + '</tbody></table>'
    )
    self.create_to_page(metrics_table)

    # Save file with malware hashes
    date = datetime.today().strftime("%d-%m-%Y")
    malware_file = df[df["classification"] == 1]
    malware_file.to_csv(
        os.path.join(os.path.dirname(path), f"{date}.output.csv"),
        columns=['hash'],
        index=False,
    )
    self.write_footer()
    return self.on_next("CheckVirusState", "Check")
  except Exception as exp:
    logger.error(f"Error occurred in state: {self.__str__} error: {exp}", exc_info=True)
    return self.on_next("ErrorState", "CheckVirusState")

@staticmethod
def write_header():
  filename = './templates/last_results.html'
  if os.path.exists(filename):
    os.remove(filename)
  with open('./templates/last_results.html', 'a', encoding='utf-8') as out:
    out.write("{% extends 'base.html' %} {% block content %}\n")
@staticmethod
```

```
            out.write(f"{input_data}\n")


    @staticmethod
    def write_footer():
        with open('./templates/last_results.html', 'a', encoding='utf-8') as out:
            out.write("{% endblock %}\n")
```

Appendix 7: Stage 2 CheckVirusState Code

```
import os
import glob
import logging
from .state import State
from utils import helper
import pandas as pd
import requests as req
from time import sleep
logger = logging.getLogger()
class CheckVirusState(State):
    def fetch_with_retries(self, url, headers, retries=3, delay=15):
        for attempt in range(retries):
            try:
                response = req.get(url, headers=headers)
                if response.ok:
                    return response
                logger.warning(f"Attempt {attempt + 1} failed for URL: {url}. Retrying in {delay}
seconds...")
                sleep(delay)
            except req.exceptions.RequestException as e:
                logger.error(f"RequestException on attempt {attempt + 1}: {e}")
                sleep(delay)
        return None
    def on_event(self, event):
        try:
```

```
        headers = {
            "accept": "application/json",
            "x-apikey": os.getenv("VT_API_KEY",
"cc9cf3df91afdc4ca6bb521d822649faf5861561ee5de878968c2fb6ffeddae4")
        }
        # Fetch the most recent 20 files
        list_of_files = sorted(glob.glob(os.path.join('downloads', '*.output.csv')),
key=os.path.getctime, reverse=True)
        files_to_process = list_of_files[:2]
        all_data = []
        for file in files_to_process:
            logger.info(f"Processing file: {file}")
            df = pd.read_csv(file)
            # Process up to 20 hashes from the file
            for idx, row in enumerate(df.iterrows()):
                if idx >= 40:
                    break
                file_hash = row[1]['hash']
                file_url = f"https://www.virustotal.com/api/v3/files/{file_hash}"
                ips_url = f'https://www.virustotal.com/api/v3/files/{file_hash}/contacted_ips'
                # API Requests with retry mechanism
                file_response = self.fetch_with_retries(file_url, headers)
                ips_response = self.fetch_with_retries(ips_url, headers)
                if file_response is None:
                    logger.error(f"Failed to fetch data for hash: {file_hash} after retries")
                    continue
                if ips_response is None:
                    logger.error(f"Failed to fetch IP data for hash: {file_hash} after retries")
                    continue
```

```
            file_data = file_response.json().get('data', {}).get('attributes', {})
            # Extract contacted IPs
            contacted_ips = ', '.join([ip['id'] for ip in ips_response.json().get('data', [])]) if
'data' in ips_response.json() else 'No IPs Found'
            # Log contacted IPs
            logger.info(f"Contacted IPs for hash {file_hash}: {contacted_ips}")
            # Extract family labels
            family_labels = file_data.get('popular_threat_classification',
{}).get('threat_family_name', 'No Family Label Found')
            # Extract file type description
            type_description = file_data.get('type_description', 'N/A')
            # Log file type
            logger.info(f"File type for hash {file_hash}: {type_description}")
            # Extract security vendor analysis
            vendors_analysis = file_data.get('last_analysis_results', {})
            vendor_results = {
                vendor: 'Yes' if result.get('category', 'undetected') == 'malicious' else 'No'
                for vendor, result in vendors_analysis.items()
            }
            # Append the data
            record = {
                "File Hash": file_hash,
                "Tags": ', '.join(file_data.get('tags', [])),
                "Type Description": type_description,
                "Contacted IPs": contacted_ips,
                "Family Labels": family_labels
            }
            # Add vendor analysis
            record.update(vendor_results)
```

```
    # Create a DataFrame to display the extracted data
    results_df = pd.DataFrame(all_data)
    if not results_df.empty:
        # Display the DataFrame and save it to a CSV file
        print(results_df)
        results_df.to_csv('output.csv', index=False)
        logger.info("Detailed results saved to 'output.csv'")
    else:
        logger.info("No valid data extracted.")
    return self.on_next("EmailState", "Email")
except Exception as exp:
    logger.error(
        f"Error occurred in state: {self.__str__} error: {exp}", exc_info=True
    )
    return self.on_next("ErrorState", "CheckVirusStateError")
```

Appendix 8: Stage 3 EmailState Code

```
import logging
from .state import State
from utils import helper
from datetime import datetime
import os
logger = logging.getLogger()
class EmailState(State):
    def on_event(self, event):
        date = datetime.today().strftime('%d-%m-%Y')
        path  = os.path.abspath(
              os.path.join(
                 f"{os.path.dirname(__file__)}",
                 r"..",
                 r"downloads",
                 rf"{date}.output.csv",
```

```
            )
        )
    if event == "Email":
        try:
            helper.send_mail('intelligencetest3@gmail.com', f'New Intelligence: {date}', 'Good morning \n I hope your day starts well. This is a message from the intelligence system notifying you of \n Please file attachment on this email which regards new malware discovered by the PSDCIS model.\n We have attached all the malicious file signatures to this email.\n Please use this file and attempt to prevent any harm to our system.', path)
            helper.send_mail("dmnsmlmanager@gmail.com",f"New Intelligence: {date}", 'Good morning \n I hope your day starts well. This is a message from the intelligence system notifying you of \n Please file attachment on this email which regards new malware discovered by the PSDCIS model.\n We have attached all the malicious file signatures to this email.\n Please use this file and attempt to prevent any harm to our system.', path)
            return self.on_next("SuccessState", "Success")
        except Exception as exp:
            logger.error(
                f"Error occuried in state: {self.__str__} error: {exp}",
                exc_info=True,
            )
            return self.on_next("ErrorState", "InitialError")
```

Appendix 9: Stage 3 Update AntiVirus State:

```
import sys
import logging
import os
import requests as req
import csv
from utils import database as db
from .state import State
logger = logging.getLogger()
class UpdateAntiVirus(State):
    def on_event(self, event):
        try:
            logger.info("Starting UpdateAntiVirus state...")
            file_path = os.path.abspath(os.path.join('downloads', 'malware_signatures.csv'))
```

```
 if not os.path.exists(file_path):
            logger.error(f"File not found: {file_path}")
            return self.on_next("ErrorState", "FileNotFoundError")
        logger.info(f"Reading malware signatures from {file_path}")
        signatures = self.read_csv(file_path)
        if not signatures:
            logger.error("No signatures found in the file.")
            return self.on_next("ErrorState", "EmptyFileError")
        logger.info(f"Total signatures found: {len(signatures)}")
        url = "https://antivirusapi.example.com/updateSignatures"
        headers = {
            "Content-Type": "application/json",
            "Authorization": "Bearer YOUR_API_KEY"
        }
        payload = {"signatures": signatures}
        logger.info("Sending signatures to the antivirus system...")
        response = req.post(url, json=payload, headers=headers)
        if response.status_code == 200:
            logger.info("Successfully updated antivirus signatures.")
            return self.on_next("SuccessState", "AntivirusUpdated")
        else:
            logger.error(f"Failed to update antivirus. Status: {response.status_code},
Response: {response.text}")
            return self.on_next("ErrorState", "AntivirusUpdateError")
    except Exception as exp:
        logger.error(f"Error occurred in UpdateAntiVirus: {exp}", exc_info=True)
        return self.on_next("ErrorState", "UnexpectedError")
  def read_csv(self, file_path):
    """Reads a CSV file and extracts malware signatures."""
```

```
with open(file_path, 'r') as file:
                reader = csv.reader(file)
                signatures = [row[0] for row in reader if len(row) > 0]
            return signatures
        except Exception as e:
            logger.error(f"Failed to read CSV file: {e}")
            return []
```

Appendix 10: Stage 3 Firewall Update Code

```
import sys
import logging
import os
import requests as req
import csv
from utils import database as db
from .state import State
logger = logging.getLogger()
class UpdateFirewalls(State):
    def on_event(self, event):
        try:
            logger.info("Starting UpdateFirewalls state...")
            file_path = os.path.abspath(os.path.join('downloads', 'firewall_rules.csv'))
            if not os.path.exists(file_path):
                logger.error(f"File not found: {file_path}")
                return self.on_next("ErrorState", "FileNotFoundError")
            logger.info(f"Reading firewall rules from {file_path}")
            rules = self.read_csv(file_path)

            if not rules:
                logger.error("No rules found in the file.")
                return self.on_next("ErrorState", "EmptyFileError")
```

```
            logger.info(f"Total rules found: {len(rules)}")
            url = "https://firewallapi.example.com/updateRules"
            headers = {
                "Content-Type": "application/json",
                "Authorization": "Bearer YOUR_API_KEY"
            }
            payload = {"rules": rules}
            logger.info("Sending rules to the firewall system...")
            response = req.post(url, json=payload, headers=headers)
            if response.status_code == 200:
                logger.info("Successfully updated firewall rules.")
                return self.on_next("SuccessState", "FirewallRulesUpdated")
            else:
                logger.error(f"Failed to update firewall. Status: {response.status_code}, Response:
{response.text}")
                return self.on_next("ErrorState", "FirewallUpdateError")
        except Exception as exp:
            logger.error(f"Error occurred in UpdateFirewalls: {exp}", exc_info=True)
            return self.on_next("ErrorState", "UnexpectedError")
    def read_csv(self, file_path):
        """Reads a CSV file and extracts firewall rules."""
        try:
            with open(file_path, 'r') as file:
                reader = csv.reader(file)
                rules = [row[0] for row in reader if len(row) > 0]
            return rules
        except Exception as e:
            logger.error(f"Failed to read CSV file: {e}")
            return []
```

Appendix 11: Error State

```
import sys
import logging
from utils import database as db
from .state import State
logger = logging.getLogger()
class ErrorState(State):
    def on_event(self, event):
        if event == 'InitialStateError':
            logger.error(f"Initial state has exited into ErrorState", exc_info=True)
        elif event == 'virusShareError':
            logger.error(f"virusShare state has exited into ErrorState", exc_info=True)
        else:
            logger.error(f"Unknown event [.....{event}.....]", exc_info=True)
        sys.exit(1)
```

Appendix 12: Success State

```
import logging
from .state import State

logger = logging.getLogger()

class SuccessState(State):
    def on_event(self, event):
        if event == 'Success':
            logger.info(f"Finished succesfully")
```

Appendix 13: State Machine State

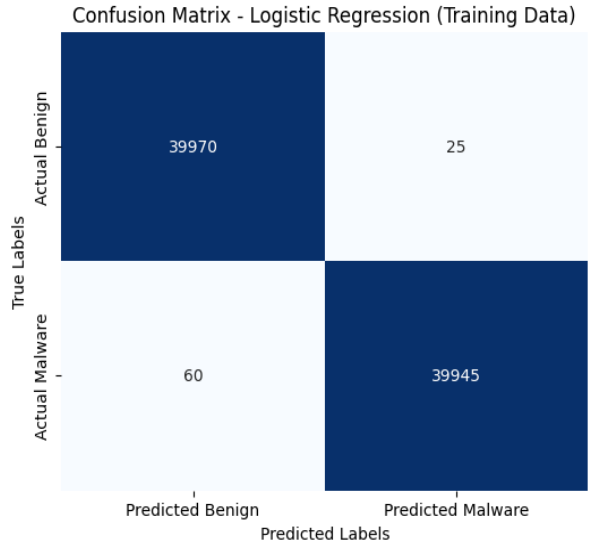
```
import os
import logging
from logging.config import fileConfig
from statemachine import StateMachine
from states import (
    ErrorState,
    InitialState,
    LoadVirusShareState,
    SuccessState,
    KaggleDownloadState,
    RunAlgorithmState,
    EmailState,
    UpdateFirewalls,
    CheckVirusState
)
LOGGING_CONFIG =
os.path.join(os.path.dirname(os.path.abspath(__file__)),"config","logging_config.ini")
print(LOGGING_CONFIG)
fileConfig(LOGGING_CONFIG)
logger = logging.getLogger()
def main():
    logger.info('Starting StateMachine')
    statemachine = StateMachine()
    #states
    statemachine.add_state('InitialState', InitialState)
    statemachine.add_state('EmailState', EmailState)
    statemachine.add_state('LoadVirusShareState', LoadVirusShareState)
    statemachine.add_state('KaggleDownloadState', KaggleDownloadState)
    statemachine.add_state('RunAlgorithmState', RunAlgorithmState)
```

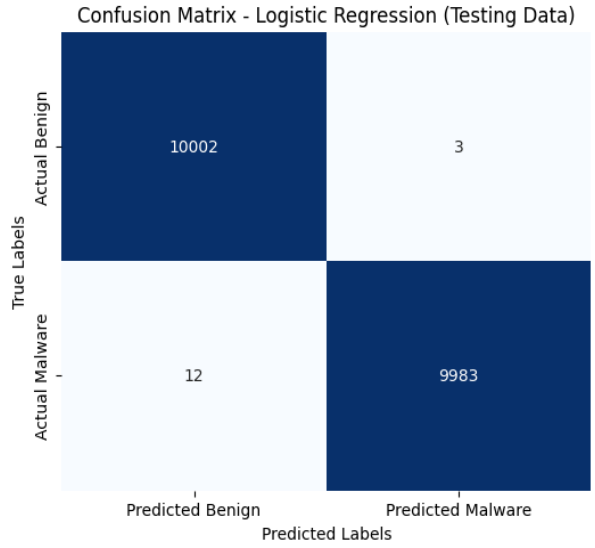
```
    statemachine.add_state('ErrorState', ErrorState, end_state=True)
    statemachine.add_state("SuccessState", SuccessState, end_state=True)
    statemachine.set_start('InitialState')
    os.environ["KAGGLE_username"] = 'condevnull'
    os.environ["KAGGLE_key"] = "e50ec7056acefaa13a0a7a48ae98f4d0"
    statemachine.run('Initial')

if __name__ == '__main__':
    main()
```

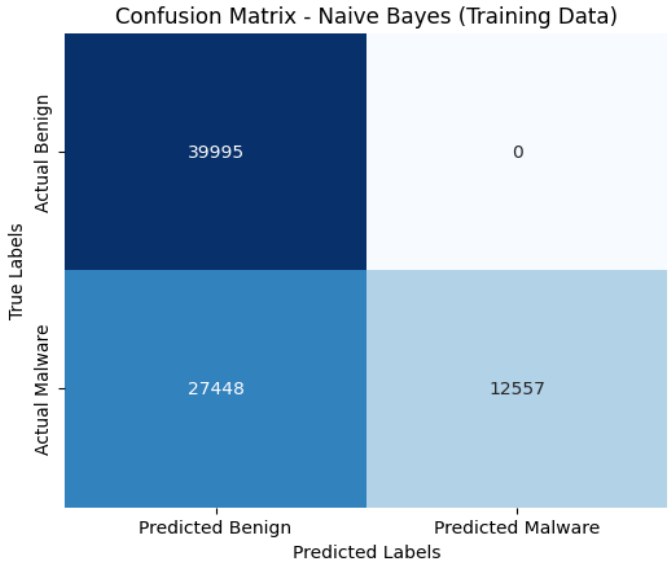
DATASET 1 (D1) CONFUSION MATRIX

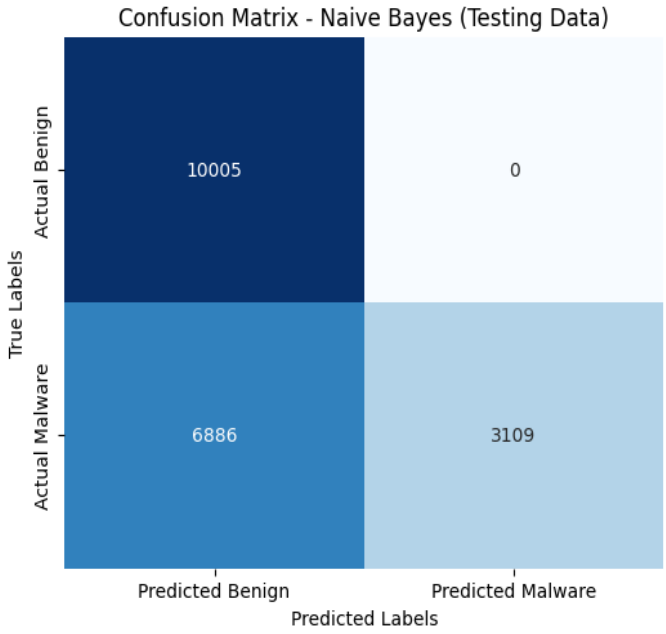
Appendix 14: Confusion Matrix Logistic Regression Train and Test D1



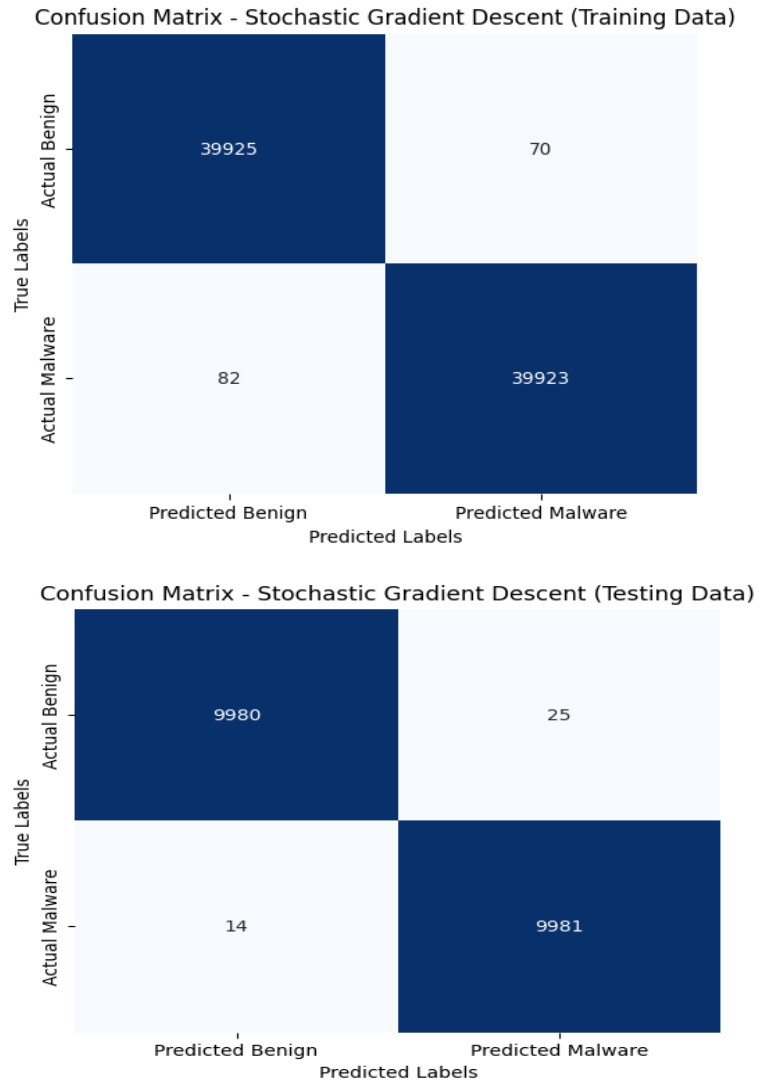


Appendix 15: Confusion Matrix Naive Bayes Train and Test D1

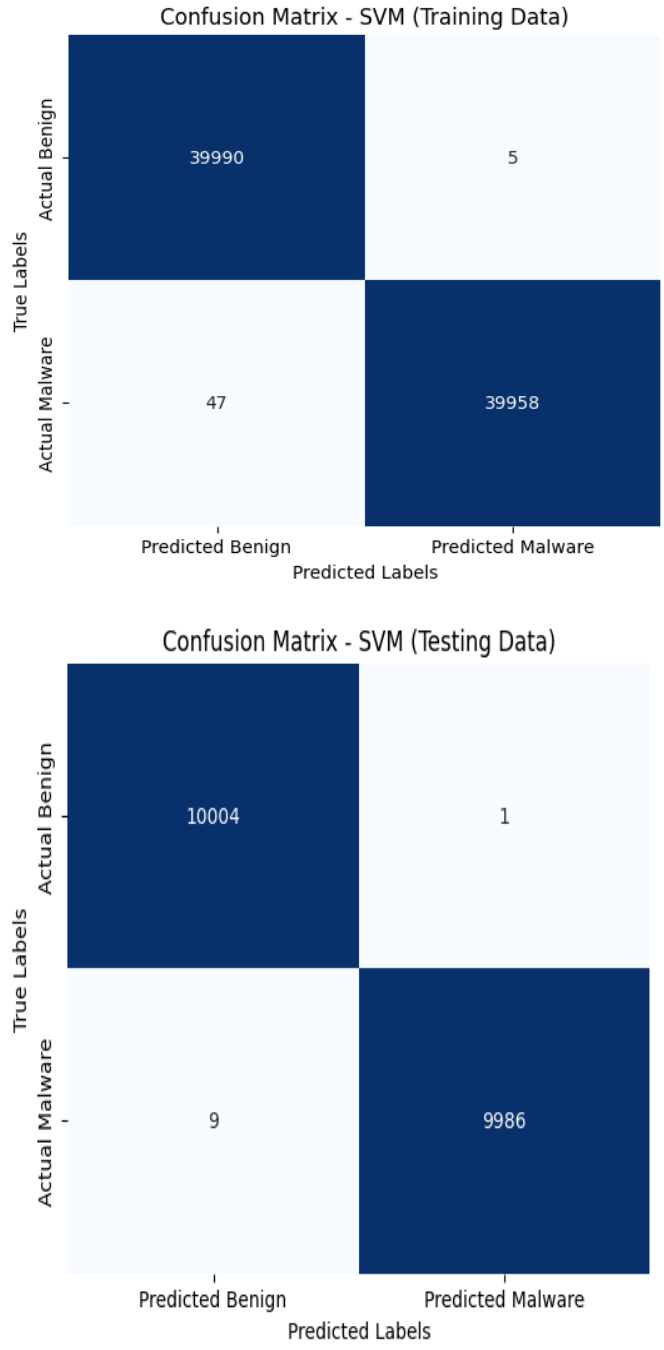




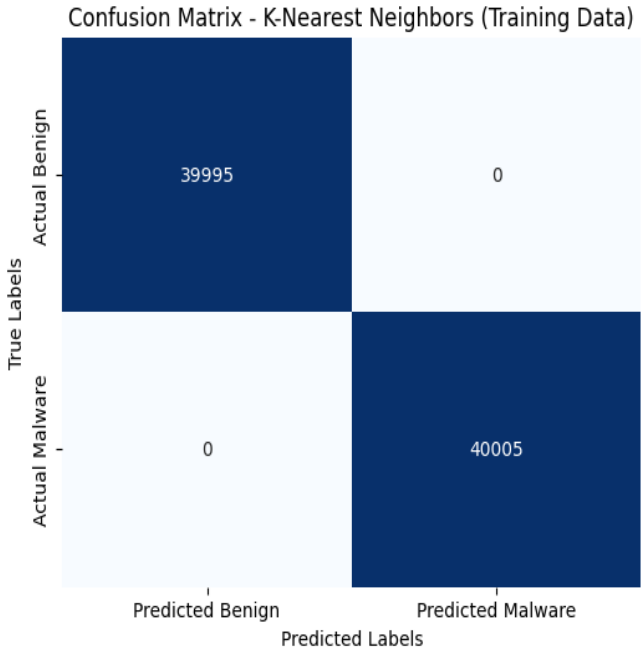
Appendix 16: Confusion Matrix Stochastic Gradient Descent (SGD) Train and Test D1

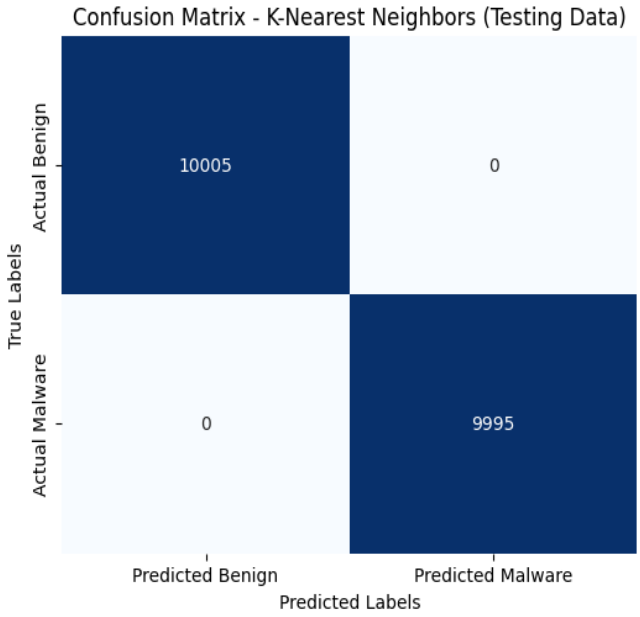


Appendix 17: Confusion Matrix SVM Train and Test D1

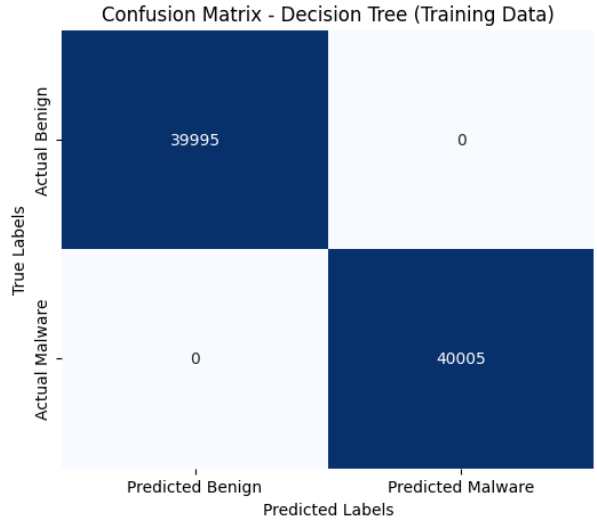


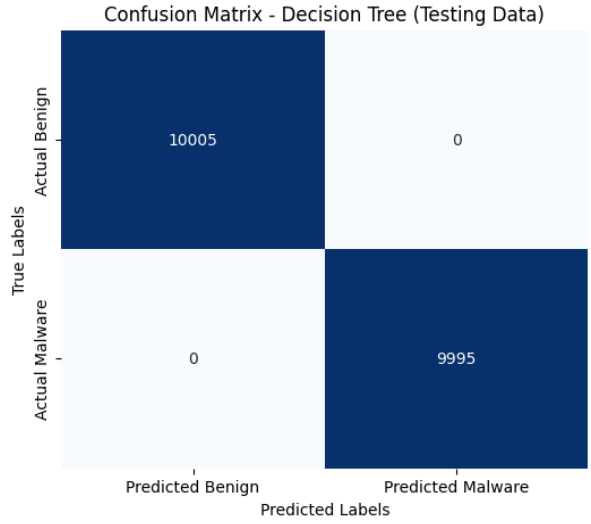
Appendix 18: Confusion Matrix KNN Train and Test D1



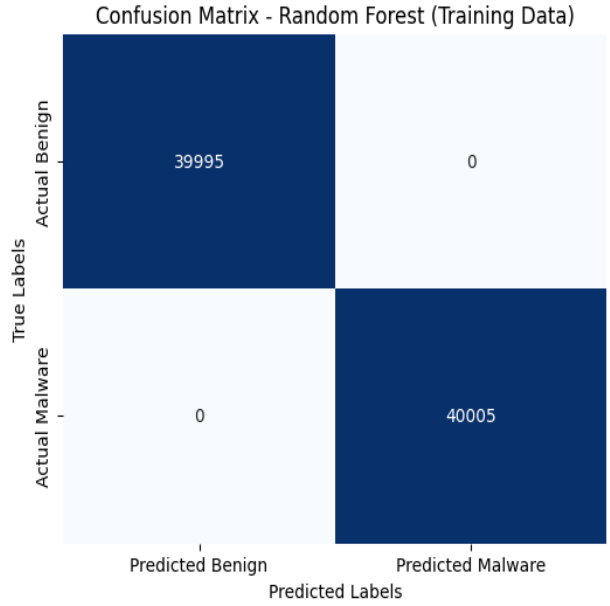


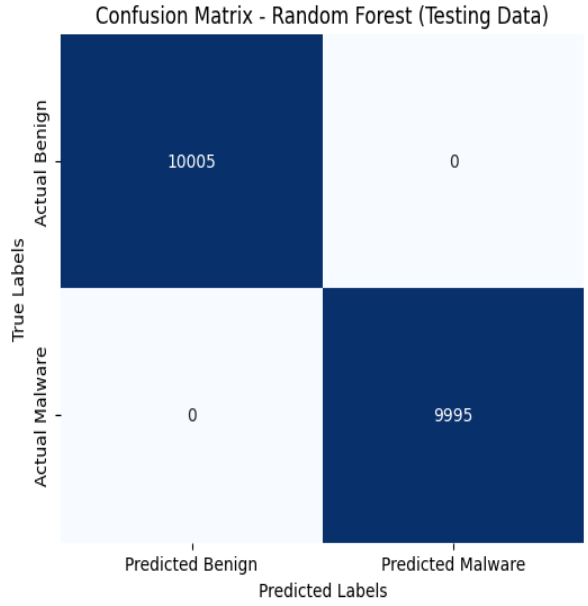
Appendix 19: Confusion Matrix Decision tree Train and Test D1



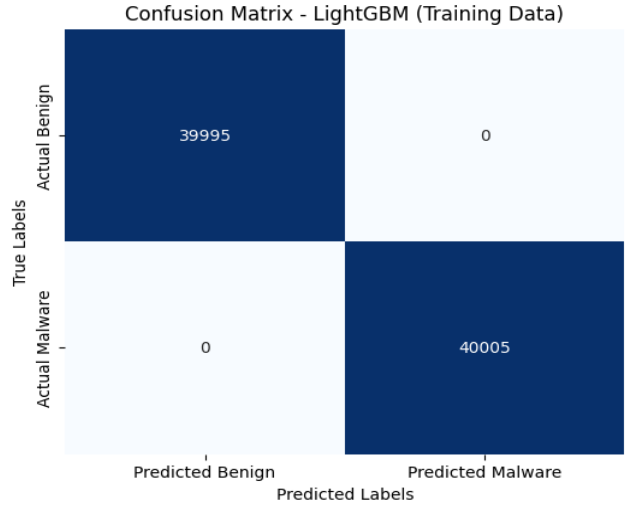


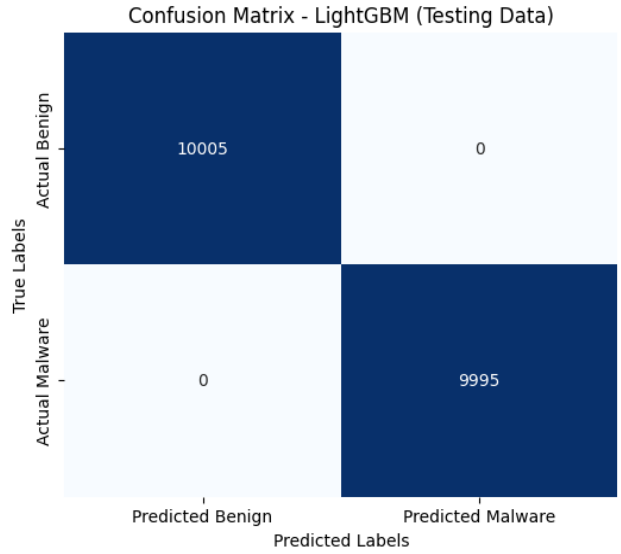
Appendix 20: Confusion Matrix Random Forest Train and Test D1



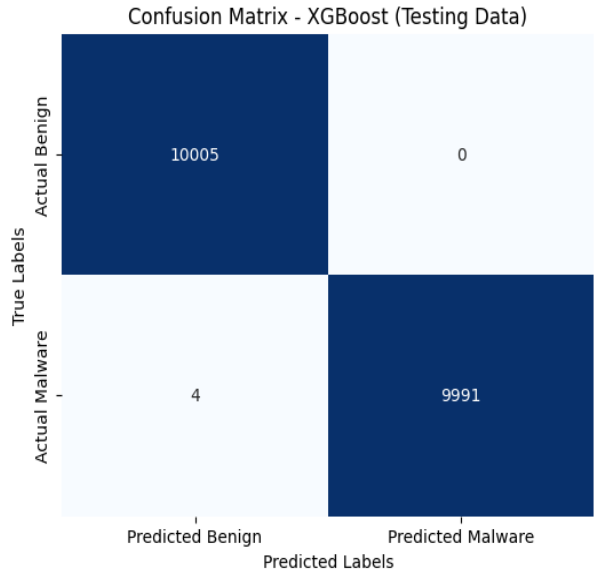


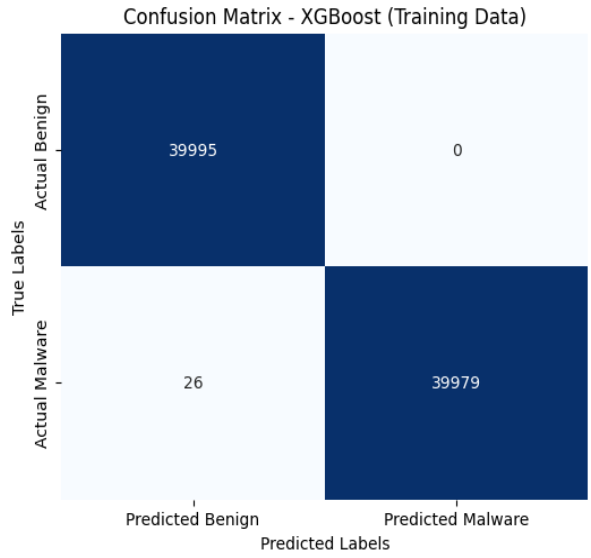
Appendix 21: Confusion Matrix LightGBM Train and Test D1





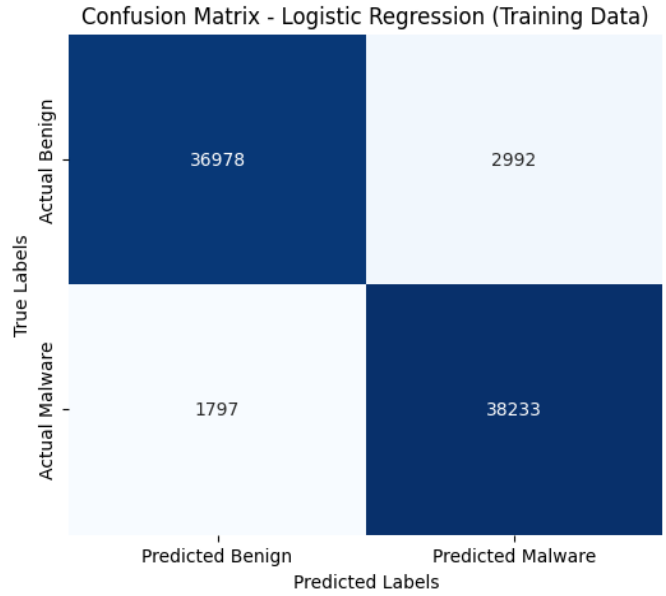
Appendix 22: Confusion Matrix XGBoost Train and Test D1

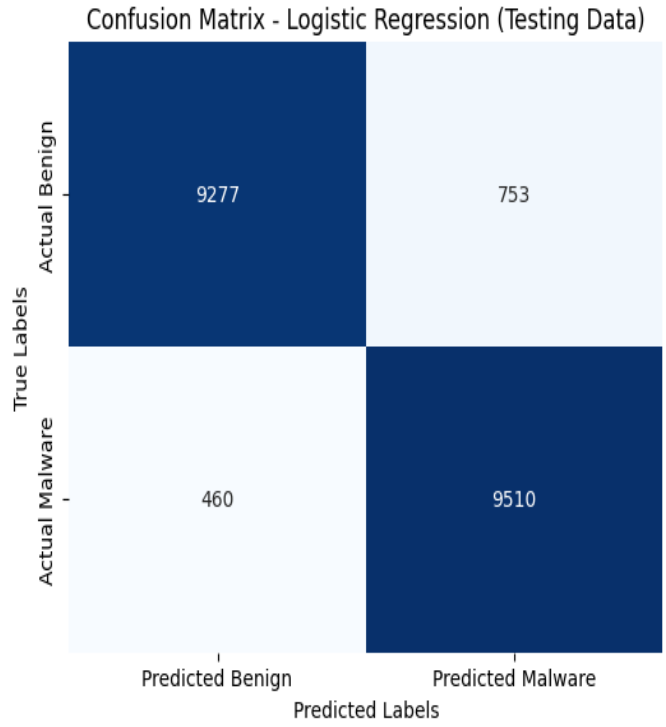




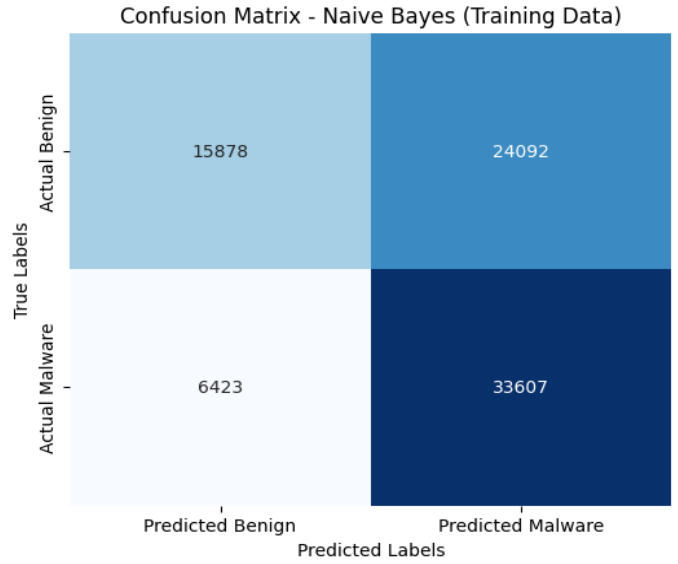
DATASET 2 (D2) CONFUSION MATRIX

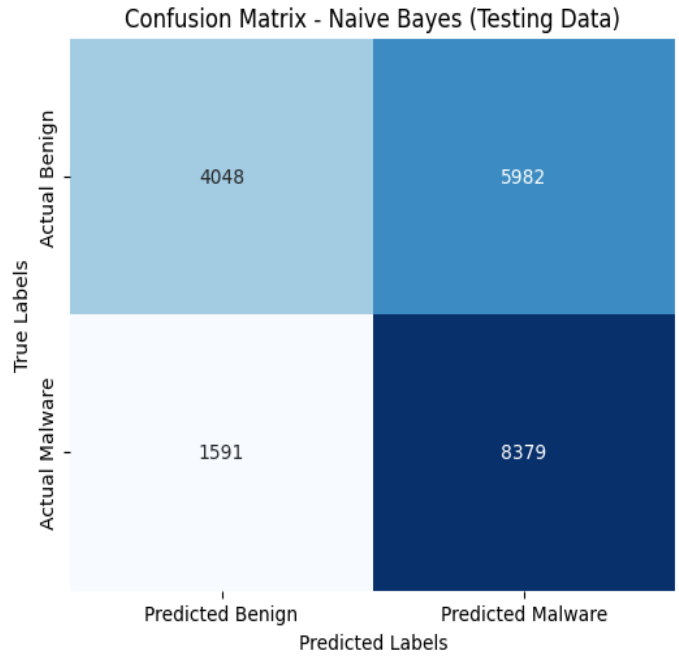
Appendix 23: Confusion Matrix Logistic Regression Train and Test D2



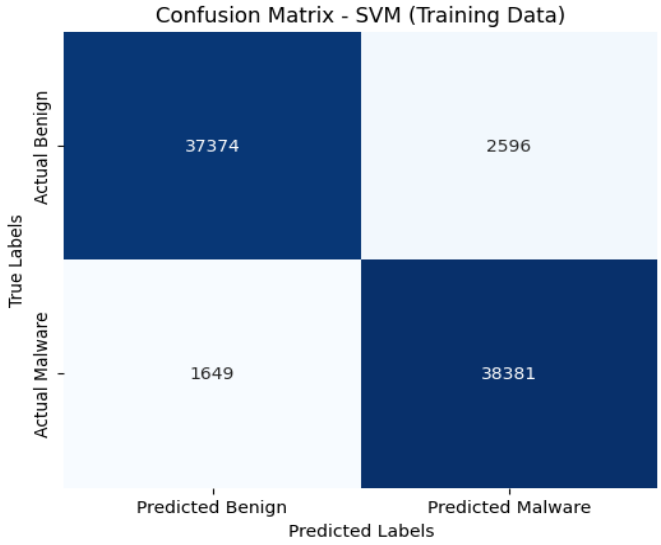


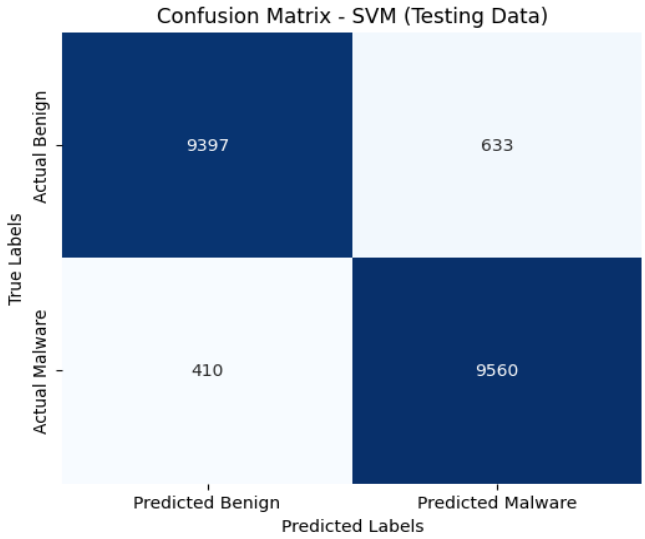
Appendix 24: Confusion Matrix Naive Bayes Train and Test D2



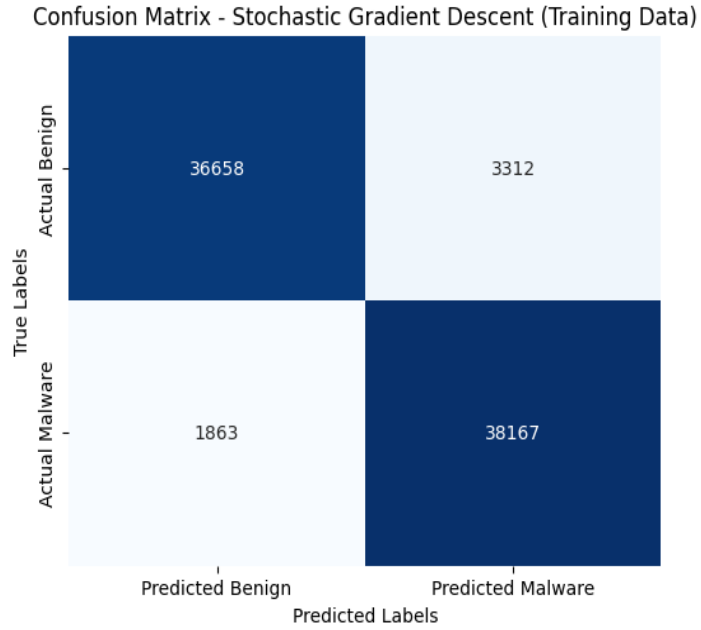


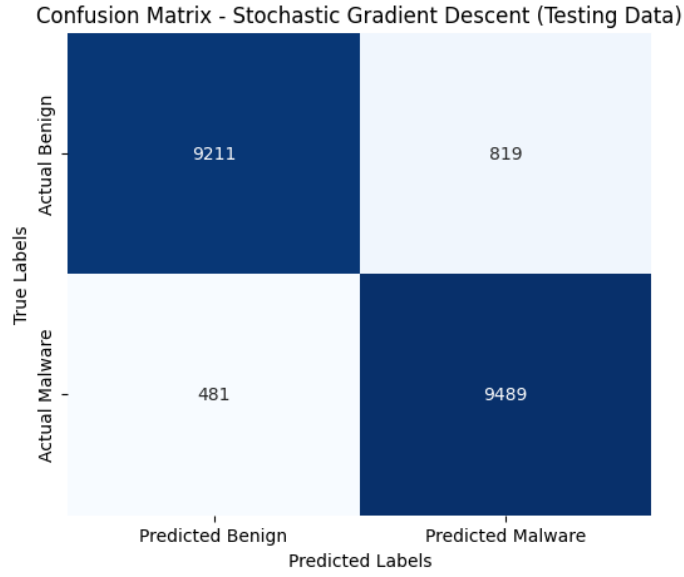
Appendix 25: Confusion Matrix SVM Train and Test D2



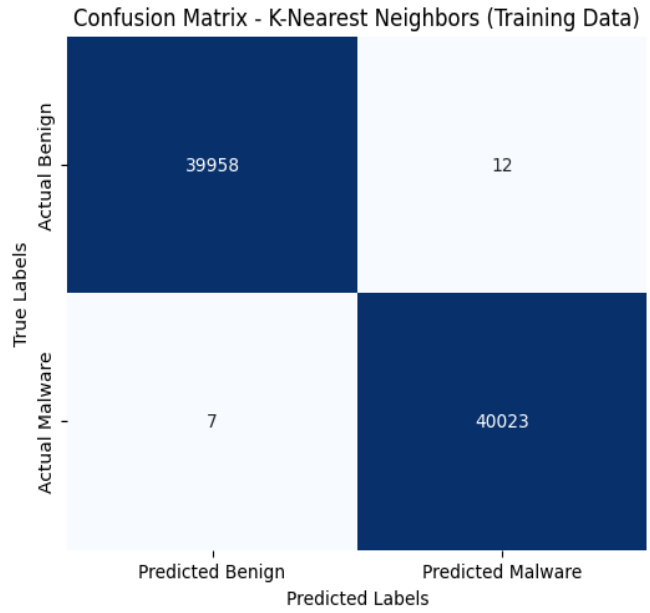


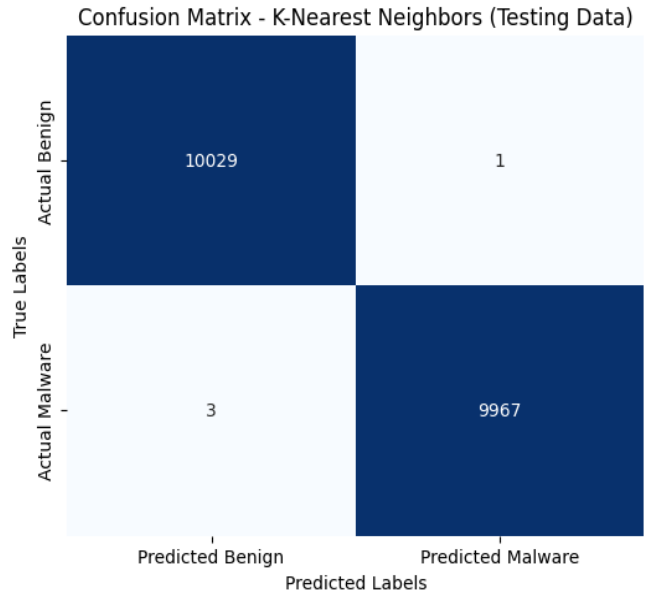
Appendix 26: Confusion Matrix SGD Training and Test D2



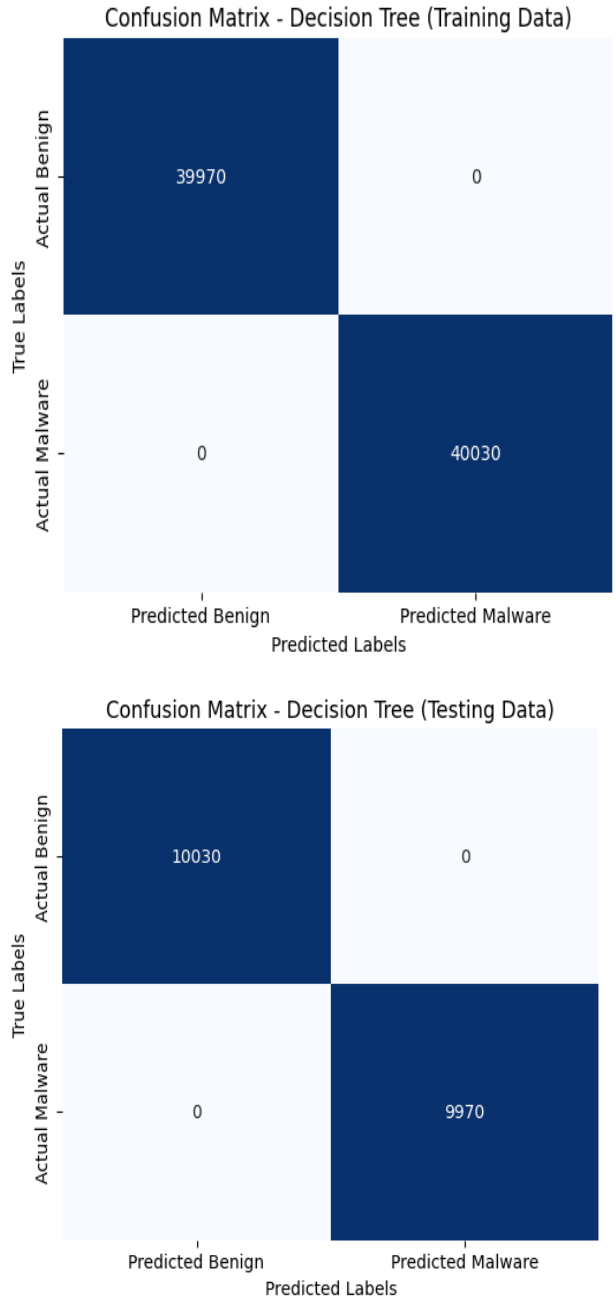


Appendix 27: Confusion Matrix KNN Train and Test D2

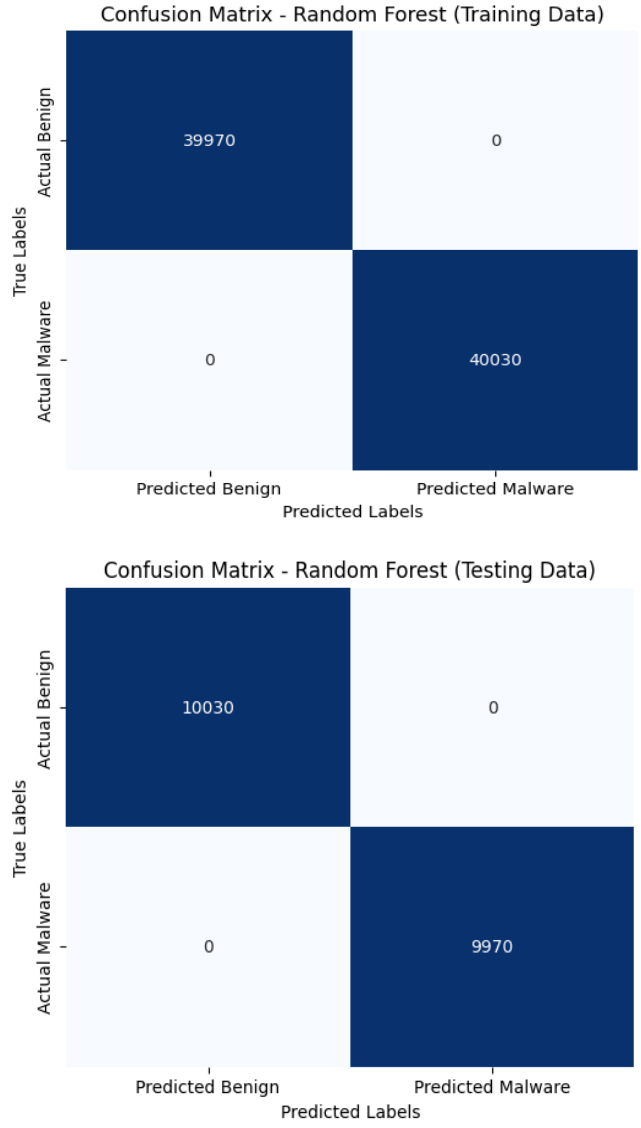




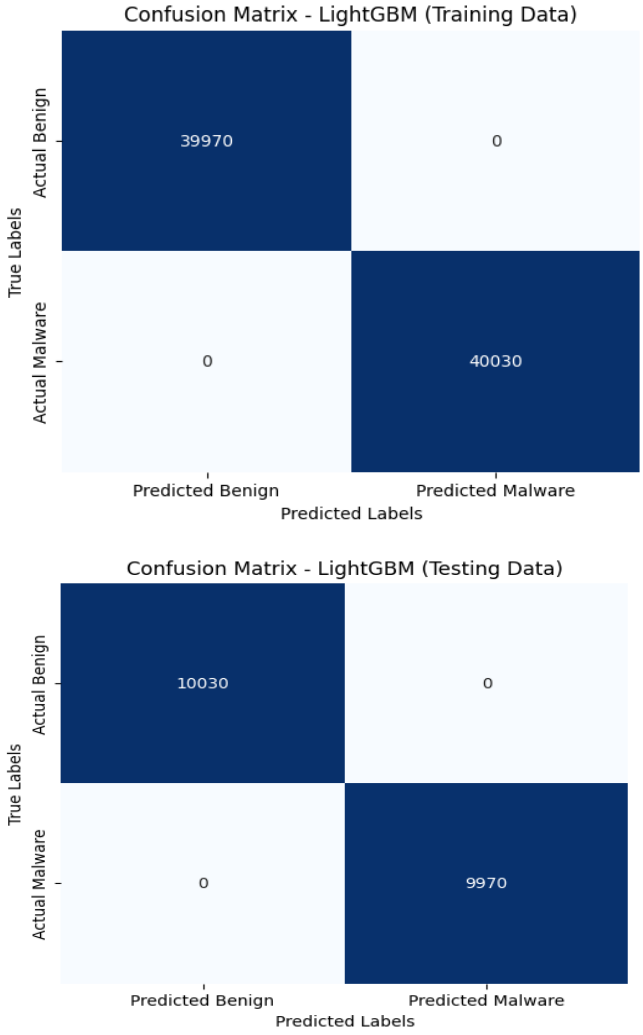
Appendix 28: Confusion Matrix Decision Tree Train and Test D2



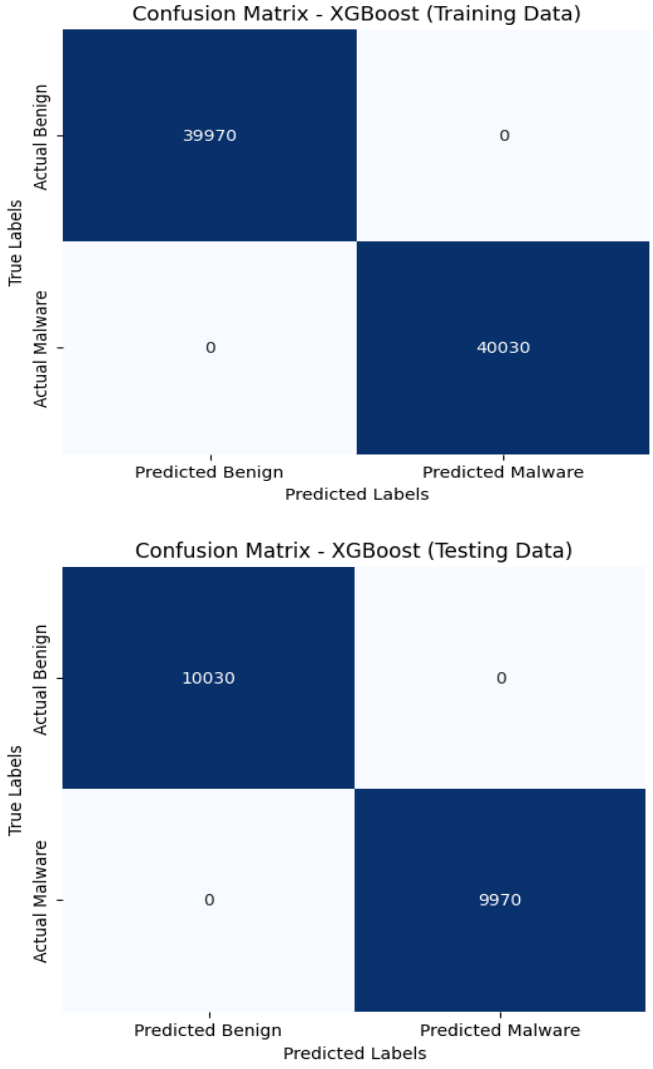
Appendix 29: Confusion Matrix Random Forest Train and Test D2



Appendix 30: Confusion Matrix LightGBM Train and Test D2



Appendix 31: Confusion Matrix XGBoost Train and Test D2



Appendix 32: Steps and description of State machine initiation state

Step	Description
InitialState	Start the process and initialize necessary components.
LoadVirusShareState	Start processing virus hash data from VirusShare.
KaggleDownloadState	Download additional data required for processing from Kaggle.
RunAlgorithmState	Execute the malware classification algorithm.
CheckVirusState	Check if the processed data contains any virus signatures.
EmailState	Validate user email before proceeding.
UpdateFirewalls/Anti-Virus	Update firewalls with the latest virus definitions.
SuccessState	Mark the process as successful if no errors occur.
ErrorState	Handle any errors that arise during execution and log them.

Appendix 33: Steps and description of LoadVirusshare State

Step	Description
Start (LoadVirusShareState)	Begins the process of extracting hashes from VirusShare.
Insert New Batch (status=Inprogress)	Adds a new tracking entry in the database.
Get VirusShare Page	Downloads the VirusShare webpage.
Extract MD5 Hash Links	Finds and extracts all hash file links.
Process Each MD5 File	Reads the content of each MD5 file.
Validate & Store Hashes in Database	Filters and saves valid hashes to the database.
Processing Complete?	Checks if all hashes were stored successfully.
Update Batch Status to Completed	Updates batch status to 'Completed'.
Transition to KaggleDownloadState	Moves to the next phase of processing.
Error Occurred?	Handles any errors in the process.
Update Batch Status to Failed	Updates batch status to 'Failed'.
Delete Stored Hashes	Cleans up any incomplete data.
Transition to ErrorState	Moves to error handling.
End	The process terminates successfully or with an error.

Appendix 34: Steps and descriptions of RunAlgorithms of the inside state machine process

Steps	Description
Start (RunAlgorithmState)	Begin the execution of the RunAlgorithmState.
Write Header for Results Page	Create a new HTML file for storing results.
Fetch MD5 Hash Dataset from Database	Retrieve MD5 hash data from the database.
Read and Modify Malware Dataset	Read the malware dataset and replace hash values.
Save Modified Dataset	Save the modified dataset for further processing.
Load Dataset into Pandas	Load the dataset into a Pandas DataFrame.
Preprocess Data (Replace Labels, Check Distribution)	Replace string labels (benign/malware), check for null values, and display data statistics.
Visualize Dataset Distribution	Plot a scatter plot to visualize the dataset classification distribution.
Prepare Features and Labels	Drop unnecessary features and separate features (X) and target labels (Y).
Train Decision Tree Model	Train a Decision Tree Classifier on the full dataset.
Evaluate Decision Tree Model	Calculate performance metrics and generate a confusion matrix.
Perform K-Fold Cross-Validation (Decision Tree)	Perform Stratified K-Fold cross-validation on the Decision Tree model.
Train XGBoost Model	Train an XGBoost Classifier on the full dataset.
Evaluate XGBoost Model	Calculate performance metrics and generate a confusion matrix.
Perform K-Fold Cross-Validation (XGBoost)	Perform Stratified K-Fold cross-validation on the XGBoost model.
Train LightGBM Model	Train a LightGBM Classifier on the full dataset.
Evaluate LightGBM Model	Calculate performance metrics and generate a confusion matrix.
Perform K-Fold Cross-Validation (LightGBM)	Perform Stratified K-Fold cross-validation on the LightGBM model.
Perform KS Statistic Analysis	Calculate the KS Statistic to measure model discrimination.
Perform SHAP Analysis	Perform SHAP analysis to explain feature importance.
Store and Display Model Evaluation Results	Store model evaluation results and generate an HTML table for display.
Save Malware Hashes to CSV	Extract malware hashes and save them as a CSV file.
Write Footer for Results Page	Finalize the results HTML page.
Transition to CheckVirusState	Move to the CheckVirusState for further processing.
Handle Errors (Transition to ErrorState)	If any errors occur, log the error and transition to the ErrorState.

Appendix 35: CheckState steps and description

Steps	Description
Start (CheckVirusState)	Begin the execution of the CheckVirusState.
Fetch API Key and Set Headers	Retrieve API key from environment variables and define HTTP headers.
Get Most Recent Malware Hash Files	Identify the most recent malware hash output files.
Read and Process CSV File	Read the malware hashes stored in CSV format.
Iterate Through Up to 20 Hashes	Process only the first 20 hashes for analysis.
Construct VirusTotal API URLs	Generate VirusTotal API endpoints for file and IP data.
Fetch File Data from VirusTotal	Attempt to fetch file attributes from VirusTotal with retries if necessary.
Extract File Attributes	Extract key file attributes such as tags, type, and vendors.
Extract Contacted IPs	Extract contacted IP addresses associated with the malware file.
Extract Threat Family Labels	Extract family labels that classify the malware.
Extract File Type Description	Extract file type description from the VirusTotal response.
Analyze Security Vendor Results	Check security vendor analysis results for malicious classification.
Compile and Store Extracted Data	Compile all extracted information into a structured format.
Save Extracted Data to CSV	Save extracted data into a CSV file for further analysis.
Transition to EmailState	Move to the EmailState to send results.
Handle Errors (Transition to ErrorState)	If any errors occur, log the error and transition to ErrorState.

Appendix 36: Result of automatic check state

File Hash	malware type	Description	Contacted Ips
000021ce9241b56a22923f51ec5895ab	peexe, spreader	Win32 EXE	
000046c82c02ebe2633268f4ca8080ea	overlay, assembly, peexe	Win32 EXE	
0000adda489b33d5f2d22d76e3bd8907	nsis, direct-cpu-clock-access, checks-network-adapters, via-tor, peexe, runtime-modules, overlay, nxdomain	Win32 EXE	13.107.4.50, 192.229.211.108, 20.99.133.109, 20.99.185.48, 20.99.186.246, 23.216.147.64, 23.216.147.76
0001617ffcd2415814904556ba2252d8	runtime-modules, peexe, armadillo	Win32 EXE	114.114.114.114, 192.229.211.108, 20.99.184.37, 218.85.157.99, 23.216.147.76
00019322819480aec09da95c3321d864	peexe, armadillo, spreader	Win32 EXE	
0001bbaeafc6d232705939a859eda8dc	corrupt, peexe, overlay	Win32 EXE	
000211b61fca8d27adde9fc5d4c6baf1	overlay, execryptor, upx, peexe	Win32 EXE	
0002422da43f24e30470fa08f294acbe	bobsoft, peexe	Win32 EXE	
000251962a5d01c0d8bd79408744da13	spreader, peexe	Win32 EXE	
0002c9617dbaa0291cfb67c5f7204159	peexe, overlay, checks-user-input, armadillo, cve-2017-0144, cve-2005-1206, cve-2017-0146, cve-2008-4835, cve-2017-0143, exploit	Win32 EXE	114.114.114.114, 192.168.0.50, 192.168.0.8, 20.99.133.109, 20.99.186.246, 218.85.157.99, 23.192.210.9, 23.216.81.152, 52.154.209.174, 64.233.181.94

)

Appendix 37: Sandboxes Results

CMC	No	No	No	No	No	No	No	No	No	No
Trapmine	Yes	No	No	No	Yes	No	Yes	No	Yes	No
McAfeeD	Yes	Yes	Yes	Yes	Yes		Yes	Yes	Yes	Yes
TrendMicro	Yes	Yes	Yes	No	Yes	No	Yes	Yes	No	Yes
VIPRE	Yes	Yes	Yes	No	Yes	No	Yes	Yes	Yes	Yes
DrWeb	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
F-Secure	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Google	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Emsisoft	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No
Tencent	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Avast	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
SUPERAntiSpyware	Yes	Yes	No	Yes	No	No	No	No	No	No
NANO-Antivirus	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
BitDefender	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Kaspersky	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No
ClamAV	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No
Paloalto	Yes	No	Yes	No	Yes	No	Yes	Yes	No	Yes
TrendMicro-HouseCall	Yes	Yes	Yes	No	Yes	No	Yes	Yes	No	Yes
APEX	Yes	Yes	No	No	Yes	No	Yes	Yes	Yes	No
ESET-NOD32	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Elastic	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No
Symantec	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
VirIT	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Baidu	Yes	No	No	No	No	No	Yes	No	No	No
huorong	Yes	Yes	Yes	No	No		Yes	Yes	Yes	No
Cybereason	Yes					No	Yes			
K7GW	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Alibaba	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
K7AntiVirus	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Sangfor	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Zillya	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	No	No
Malwarebytes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	No	No
ALYac	Yes	No	Yes	No	Yes	No	Yes	Yes	Yes	Yes
Skyhigh	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
CAT-QuickHeal	Yes	Yes	No	Yes	Yes	No	Yes	Yes	Yes	Yes
FireEye	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
MicroWorld-eScan	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
tehtris	No	No	No	No	No	No	No	No	No	No
Lionic	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Bkav	Yes	Yes	Yes	No	Yes	No	Yes	Yes	Yes	Yes

Appendix 38: Sandboxes results CheckState

CTX		Yes	Yes	Yes	Yes			Yes	Yes	Yes
Avast-Mobile	No	No	No	No	No	No	No	No	No	No
BitDefenderFalx	No	No	No	No	No	No	No	No	No	No
SymantecMobileInsigh	No	No	No	No	No	No	No	No	No	No
Trustlook	No	No	No	No	No	No	No	No	No	No
alibabacloud	Yes	No	Yes	No	No		Yes	No	No	No
CrowdStrike	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
DeepInstinct	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
AVG	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
BitDefenderTheta	Yes					No	Yes			
Fortinet	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
MaxSecure	Yes	Yes	No	Yes	Yes	No	Yes	Yes	Yes	Yes
SentinelOne	Yes	Yes	No	Yes	Yes	No	Yes	Yes	Yes	No
Yandex	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Rising	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No
Zoner	No	No	No	No	No	No	No	No	No	No
Panda	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Cylance	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
VBA32	Yes	Yes	Yes	No	Yes	No	Yes	Yes	Yes	Yes
TACHYON	Yes	No	Yes	Yes	Yes	No	No	No	Yes	No
McAfee	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Acronis	Yes	No	No	No	No	No	Yes	No	No	No
AhnLab-V3	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No
Cynet	Yes	No	Yes	Yes	No	No	Yes	Yes	Yes	No
Microsoft	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
ZoneAlarm	Yes					No	Yes			
ViRobot	Yes	No	No	Yes	No	No	No	Yes	Yes	Yes
Arcabit	Yes	Yes	Yes	No	Yes	No	Yes	Yes	Yes	Yes
Xcitium	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Gridinsoft	Yes	Yes	No	No	Yes	No	Yes	No	No	No
Kingsoft	Yes	Yes	No	Yes	Yes	No	Yes	Yes	Yes	Yes
Antiy-AVL	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
MAX	Yes					No	Yes			
Avira	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Varist	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Webroot	Yes	No	No	No	No	Yes	Yes	No	Yes	No
Jiangmin	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No
GData	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No
Ikarus	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No
Sophos	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes

Appendix 39: EmailState steps and description of the process inside the state machine

Steps	Description
Import Modules	Import necessary modules: logging for logging events, State as the base class, helper for utility functions, datetime for date operations, and os for file path manipulations.
Initialize Logger	Initialize a logger instance to record error messages and other log entries.
Define EmailState Class	Define the EmailState class inheriting from the State base class, representing a specific state
Define on_event	Define the on_event method to handle incoming events and determine state transitions.
Get Current Date	Retrieve the current date and format it as 'day-month-year' (e.g., '26-02-2025').
Construct File Path	Build the absolute path to the CSV file named with the current date, located in the 'downloads' directory.
Check Event Type	Evaluate if the received event is of type 'Email'.
Attempt to Send Emails	Within a try block, use the send_mail function from the helper module to send emails to specified recipients with the constructed file as an attachment.
Handle Success	If emails are sent successfully, transition to the 'SuccessState'.
Handle Exceptions	If an exception occurs during the email-sending process, log the error with details and transition to the 'ErrorState'.

Appendix 40: Firewall update steps and description

Steps	Description
Initialize Logging and Dependencies	Imports necessary modules (`sys`, `logging`, `os`, `requests`, `csv`). Imports database utilities and `State` class. Initializes logger.
Define the `UpdateFirewalls` Class	Inherits from `State` class. Contains `on_event` method for handling firewall update events.
Locate Firewall Rules File	Sets expected file path (`downloads/firewall_rules.csv`). Logs error and transitions to `ErrorState` if file is missing (`FileNotFoundError`).
Read Firewall Rules from CSV	Calls `read_csv(file_path)` to extract rules. Logs error and transitions to `ErrorState` if the file is empty (`EmptyFileError`).
Prepare API Request to Update Firewall R	Constructs API request to `https://firewallapi.example.com/updateRules`. Sets headers (`Content-Type`, `Authorization`). Prepares JSON payload with extracted rules.
Send API Request to Firewall System	Sends `POST` request using `requests`. Logs success and transitions to `SuccessState` if response is `200 OK`. Logs error and transitions to `ErrorState` if request fails (`FirewallUpdateError`).
Handle Exceptions	Catches unexpected errors, logs them, and transitions to `ErrorState` (`UnexpectedError`).
Define `read_csv` Method	Reads CSV file containing firewall rules. Extracts rules while ignoring empty rows. Logs error and returns an empty list if reading fails.