Network Security Analytics on the Cloud: Public *vs.* Private Case

Vassil Vassilev, Karim Ouazzane, and Viktor Sowinski-Mydlarz
Cyber Security Research Centre
London Metropolitan University
London, UK
{v.vassilev,k.ouazzane,w.sowinskimydlarz}@londonmet.ac.uk

Herbert Maosa, Sabin Nakarmi, and Martin Hristev
School of Computing & Digital Media
London Metropolitan Universuty
London, UK
{ham104,san0693,mah1559}@my.londonmet.ac.uk

Sorin Radu
Information Governance and Cyber Security
London Borough of Hounslow
London, UK
soriniulian@gmail.com

***Abstract*— Our networks, PCs, tablets, mobile phones, and other devices are exposed to security risks and attacks executed by cybercriminals on daily bases. The detection and prevention of cyber threats are done by IDS/IPS systems but they are not flexible enough when it comes to using threat models. The threat intelligence frameworks on the other hand typically require significant computational power. All these requirements can be fulfilled by contemporary cloud technologies, but in many cases, public clouds are not acceptable due to privacy, security, and efficiency concerns. This article presents an implementation of a framework for security analytics in the area of detection of unauthorized intrusions using the technology of the private cloud. It has many of the advantages of the big public clouds but fundamentally differs from them when it comes to data management, operation interoperability, and costs. It is suitable for small and medium data centers and large companies, which prefer to keep the data on their premises or to isolate the operations within managed servers on their private clouds hosted by public data centers.**

Keywords—Intrusion Detection, Logical Analyzers, Correlation Analysis, Machine Learning, Cloud Technologies

I. Motivation

A network Intrusion Detection/Intrusion Prevention System (IDS/IPS) is a device that runs specialized software capable to capture and analyze the network traffic. After the analysis is done the device is also capable in taking a preprogramed decision (based on the previous "training" and often assisted by a human analyst) to forward or to stop the connections to the network endpoints. The IDS/IPS systems capture data from the network hardware as well as from the various system logs which register the activities within the network. The architecture of IDS includes:

Network Sensors designed to collect events related to the security of the system; sometimes they are referred also as a fully capable to run a signature detection engine too;

Traffic Analyzers designed to detect suspicious activity based on data collected by sensors; storage the log events and the results of the analysis;

Control and the reporting console intended to configure the IDS, and to extract the security statement and the status of the system.

The entire IDS/IPS system is by design to be distributed and all the above components can run as instances on different machines interconnected on the network, as well as run on a single hardware platform.

The network sensor, which is a component of an IDS/IPS can be deployed inline (capable to stop the traffic that passes through them) or can be deployed online (out-of-band) if capable to monitor/observe the traffic.

The analysis on the fly of the captured traffic is based on two main strategies, namely *signature-based* and *anomaly-based*. While the signature-based approach examines the network traffic and compares with preconfigured predetermined attack patterns (*signatures*) the anomaly-based statistics examine the network traffic and compare ot with a previously obtained baseline (*training*) that is considered "normal" expected traffic within the specific environment. Both approaches imply to compare the observed traffic with a previous sample and therefore a false positive and also a false negative relationship can be observed.

We can theoretically improve the results of anomaly-based approach if we develop a *furrier-analysis* engine that generates the baseline in real-time. In this approach we consider the network flow as an analogue wave that needs to be digitally encoded. We can sample the patterns of the network flow constantly and then compare the sampled flow with the baseline and after a short period of time, we can evaluate real-time traffic with the newly generated baseline. In this research we adopt different approach, based on *classification* and *correlation* of network packets and security events on the cloud. Our assumption is that by secondary analysis based on the methods for data analytics we can detect more intrusions and potential threats than the standard IDS/OPS systems.

II. Cloud-based security Frameworks

A. The Network Equipment Vendors: Cisco, Palo Alto, Juniper, Etc.

The IDS/IPS equipment is currently available from a number of vendors from USA, Europe and Far East. Although the industrial leaders amongst them are still trying to embed as much intelligence as possible in their IDS/IPS products in order to get higher revenue from them almost all of them recognise the importance of the cloud for providing complete and more secure solution beyond the network sensors and traffic analysers and complement their offers with cloud-based security platforms. This is evident in the recent offers from the big three – **Stealthwatch Cloud** by Cisco Systems [1], **Cortex XDR** by Palo Alto Networks [2] and **Advanced Threat Prevention** by Juniper [3] are cloud securoty platforms hosted by the vendors themselves. All of them use AI not only to enhance the ability to detect and classify the potential threats by methods of Machine Learning but also to embed threat intelligence from repositories such as MITRE [4]. However, their target market for the cloud remains the large corporate clients which limits the possibility of SMEs and public organisations to use their advanced solution.

B. The Cloud Service Providers: Microsoft, Amazon, Google, Etc.

The large public cloud service providers add another solutions for protecting customer assets for customers who subscribe to their cloud services, Security Information and Event Management specialized software (SIEM). It provides real-time analysis of security alerts generated by various application and hardware (e.g. IDS/IPS appliance). The SIEM adds another layer of analysis and correlates the events and alerts provided by IDS/IPS with the events observed and logged by other software, appliances or/and servers. Examples of this class are **Defender for Cloud** for Microsoft Azure [5], **Cloud IDS** for Google Cloud [6] and **GuardDuty** for Amazon AWS [7].

As a rule, the security solutions of the cloud service providers target the same market as the market of their services and because of this their solutions are more specialized and have lower granularity. This allows them to be more efficient in the protection of the end customers when using hosted services. At the same time, it makes it more difficult to provide complete protection and prevention of the client organizations due to the complications caused by the need to co-host multiple applications on behalf of potentially large number of customers. There are some integral solutions targeting specific types of business organizations, such as banking, eCommerce or Manufacturing industry. Amongst the big cloud service provider probably the most extensive list of solutions for that comes from **Amazon AWS**. It is worth mentioning the extended list of security services provided by AWS, which includes not only the classical security services available in most IDS/IDP systems, like AWS Network Firewall, AWS Shield and AWS Detective, but also AWS Identity and Access Management, AWS Resource Access Manager, AWS Secrets Manager and even AWS CloudHSM for generating encryption keys, securing customer-built applications deployed to the cloud and protecting their own information from services which run on the cloud. Unfortunately, this opportunity is available only to organizations which can afford hosting of their services on AWS - it is completely out of reach for SMEs and public organizations because of the high running costs.

C. The Academic Offsprings: Berkeleys, Darktrace, etc.

Some academic centers which have developed mature cyber security solutions are already on the market – either providing consultancy, or spinning out startup companies. Amongst them probably the most eminent cases are the Center for Long-Term Security of the University of California at Berkeley in the USA and the Darktrace spin-off of Cambridge University in UK [8,9]. It is not by an accident that these centers are also amongst the strongest proponents of AI in industry.

The academic organizations can offer innovative solutions to SMEs and data centers which could allow them to overcome the limitations of the big vendors and service providers with solutions based on the idea of a private cloud. Unlike the solutions based on the use of public clouds, the private clouds are focused on solving only problems within the focus of the companies and because of this their solutions are smaller and more affordable. In addition, private clouds allow to retain the ownership of the data and to increase the data protection and the data privacy which for some businesses is critical.

The Cyber Security Research Centre of London Metropolitan University was created only five years ago but from the very beginning it utilized the power of the cloud – initially on the public cloud for secure authentication when using voice control devices for transactions management and subsequently on the private cloud for fraud detection [10,11] analysis of logical vulnerability [12], intrusion detection [13] and threat intelligence [14].

III. Private Cloud-based Security Data Platform using Public Domain Software

The data platform of the Cyber Security Research Centre of London Metropolitan University runs on a small commodity hardware cluster operating under **Linux**, controlled by **Kubernetes** container management system. A simplified diagram of the architecture is shown on Fig. 1.

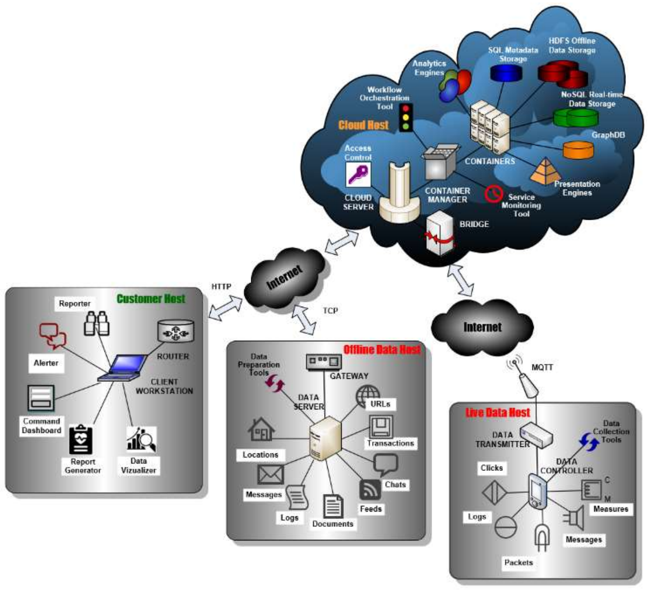


Fig. 1 Cloud-based Data Platform

A. Software Components of the Platform

The infrastructure of the platform has been built using well-proven public domain software and community edition enterprise products, which support most of the tasks for processing streamlined data in real-time and to perform offline forensic analysis of large datasets. The databases used in the prototype (**Postgres**, **MongoDB** and for future use also **Neo4J** and **Hadoop**) are deployed within **Docker** containers. Despite the limited number of off-the-shelf products used to setup the platform the pilot projects deployed to it demonstrated convincingly many of the advantages of the big public cloud-based platforms. This was achieved thanks to the careful selection of well-proven products and their full integration. At the same time, it shows that it is feasible to have an enterprise quality of services on a private cloud without substantial investment, significant resources and level of management.

The platform for processing security data is assembled using a number of software components, operating along the entire pipeline for processing the data from its source to its final destination on the Web (Fig. 2):

Client Messenger uses **Mosquitto** API to transport the network analyzer logs by sending messages over MQTT protocol.

Client Streamer collects the network packets and using **Kafka** API transports them to the cloud in a stream over binary protocol.

Message Broker server-side uses the Eclipse **Mosquitto** as MQTT server, responsible for collecting the JSON messages encapsulating the network analyzer logs.

Kafka Consumer uses **Kafka** API to broker the network packets to the final destination inside MongoDB

Data Accumulator buffers the packet data for real-time processing and maintains a queue of buffered data for storing into a permanent storage on the cloud server.

Data Analytics Engines implement the data analysis in both real-time (Correlation) and offline (Regression, NN, SVM&CNN)

Directed Acyclic Graphs specify the workflows to be executed for processing the data (ingestion, storing and analysing) and to control the iterations (initialization and updating)

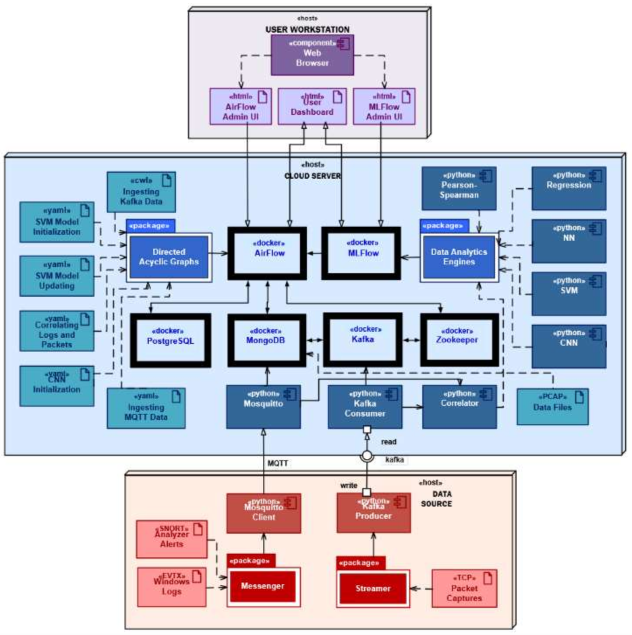
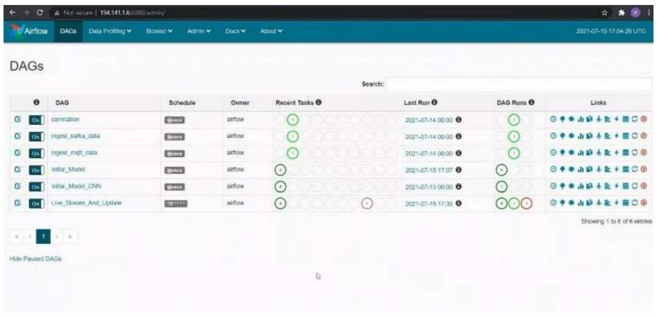


Fig. 2 Software Components of the Data Platform

B. *Orchestration and Montoring of Data Analytics*

The operation of the platform is controlled by two additional off-the-shelf products: **AirFlow** and **MLFlow**. While Airflow allows to orchestrate the workflow of data management tasks as specified in acyclic graphs within the yaml files (Fig. 3), MLFlow gathers information from the server logs during the execution of different engines which perform the data management tasks for reporting and auditing purpose. The results of the execution of separate data management operations can be inspected in MLFlow (Fig. 4).



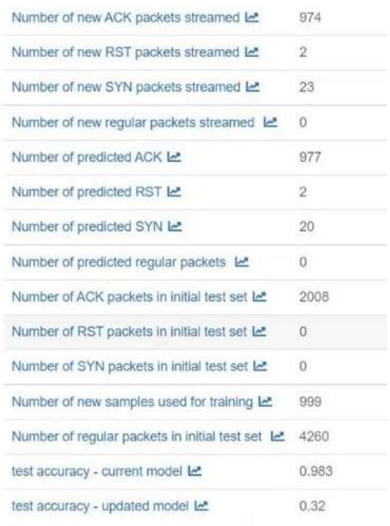
DAGs

	DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
On	correlation	[illegible]	airflow	1	2021-07-14 00:00	1	
On	ingest_kafka_data	[illegible]	airflow	1	2021-07-14 00:00	1	
On	ingest_mqtt_data	[illegible]	airflow	1	2021-07-14 00:00	1	
On	initial_Model	[illegible]	airflow	2	2021-07-15 17:07	1	
On	initial_Model_CNN	[illegible]	airflow	2	2021-07-15 00:00	1	
On	Live_Stream_And_Update	[illegible]	airflow	3 1	2021-07-15 17:30	1 1 1	

Showing 1 to 6 of 6 entries

Fig. 3 Workflow Orchestration using AirFlow

Both products provide a Web interface for convenient configuration, control of the execution, tracing and exploration of the results.



Number of new ACK packets streamed	974
Number of new RST packets streamed	2
Number of new SYN packets streamed	23
Number of new regular packets streamed	0
Number of predicted ACK	977
Number of predicted RST	2
Number of predicted SYN	20
Number of predicted regular packets	0
Number of ACK packets in initial test set	2008
Number of RST packets in initial test set	0
Number of SYN packets in initial test set	0
Number of new samples used for training	999
Number of regular packets in initial test set	4260
test accuracy - current model	0.983
test accuracy - updated model	0.32

Fig. 4 Inspecting the Results of the Engine Execution in MLFlow

IV. Security Analytics on the Private Cloud

During the process of development and in order to test the security platform we were planning to use data generated from one data centre in London. Unfortunately, this period coincided with the total lockdown during the pandemics and we were unable to setup the environment. Because of this and in order to generate suitable data for processing on our cloud we moved to completely simulated data source. Despite this small inconvenience we managed to complete the prototype and successfully to test it on the private server we built. Our goal was only to validate the framework without focusing on the quality of the analytic although the experimental results are also exciting because they demonstrate the methods of data analysis for detection, classification and correlation in a totally convincing way.

A. *Forensic Data Generation*

The data needed for our security analytics framework was generated within a sandbox, shown on Fig. 5.

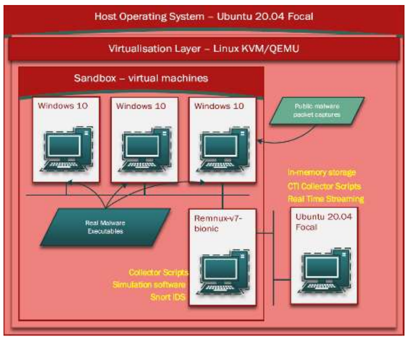


Fig. 5 Data Collection Sandbox

The sandbox set up is based on isolating the infected environment from the remaining processes running on the cloud using virtualisation technology. Its components are

Host System: The host system is a dedicated cloud server configured with 4TB of HDD storage, 64GB RAM and 16 cpu cores.

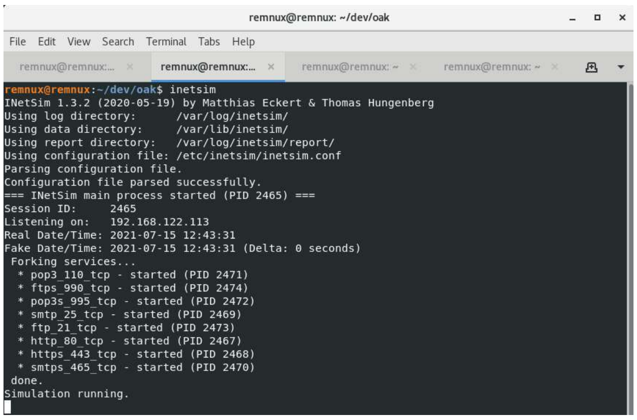
Host Operating System: Ubuntu Linux 20.04 Focal, with hardware acceleration enabled for virtualisation support

Virtualisation Layer: Implemented using the Linux native Kernel Virtual Machine Manager (KVM) with Qemu.

VIRTUAL MACHINES: Four Virtual Machines have been set up in the sandbox. Three are running **Microsoft Windows 10 Home**, configured with 60GB HDD and 4GB RAM. The fourth virtual machine is **Remnux** system, which is based on **Ubuntu 18.04 bionic**. The **Remnux** system is a purpose-built Linux system comprising a curated list of applications, tools, and utilities for malware analysis.

In order to generate security data suitable for the analysis live malware samples of viruses, worms and trojans have been downloaded and introduced into the windows host machines. As the malware execute inside the windows hosts, the windows event management system logs open file handles, processes, files, command execution and records all these events as log files in the windows **evtx** format. Further, the malware may attempt network connections as they scan for more targets or to contact their command-and-control centre. If the network connections match the snort IDS rules the **Remnux** machine then captures them. This setup is repeated for all the sandbox machines.

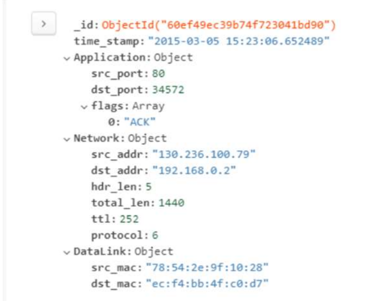
To capture suitable malware activity in this virtual environment, we needed to create an environment which appears real to the malware. To make sure that typical network application services are running in our environment we used the tool **inetsim** in our **Remnux** system. This tool starts several typical network applications (mail server, ftp server, webserver, etc) and listens for connections. The screenshot in Fig. 6 shows **inetsim** in action.



```
remnux@remnux:~/dev/oak$ inetsim
INetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg
Using log directory:      /var/log/inetsim/
Using data directory:     /var/lib/inetsim/
Using report directory:   /var/log/inetsim/report/
Using configuration file: /etc/inetsim/inetsim.conf
Parsing configuration file.
Configuration file parsed successfully.
=== INetSim main process started (PID 2465) ===
Session ID:     2465
Listening on:   192.168.122.113
Real Date/Time: 2021-07-15 12:43:31
Fake Date/Time: 2021-07-15 12:43:31 (Delta: 0 seconds)
 Forking services...
  * pop3_110_tcp - started (PID 2471)
  * ftps_990_tcp - started (PID 2474)
  * pop3s_995_tcp - started (PID 2472)
  * smtp_25_tcp - started (PID 2469)
  * ftp_21_tcp - started (PID 2473)
  * http_80_tcp - started (PID 2467)
  * https_443_tcp - started (PID 2468)
  * smtps_465_tcp - started (PID 2470)
 done.
Simulation running.
```

Fig. 6 Monitoring the sandbox using **inetsim**

The packet captures, log files and alerts are collected by the client collector script running on the **Remnux** machine and sent to the central collector running on the host system. After parsing and filtering as described earlier, these data are streamed into MQTT and Kafka brokers in the cloud.



```
_id: ObjectId("60ef49ec39b74f723041bd90")
time_stamp: "2015-03-05 15:23:06.652489"
Application: Object
  src_port: 80
  dst_port: 34572
  flags: Array
    0: "ACK"
Network: Object
  src_addr: "130.236.100.79"
  dst_addr: "192.168.0.2"
  hdr_len: 5
  total_len: 1440
  ttl: 252
  protocol: 6
DataLink: Object
  src_mac: "78:54:2e:9f:10:28"
  dst_mac: "ec:f4:bb:4f:c0:d7"
```

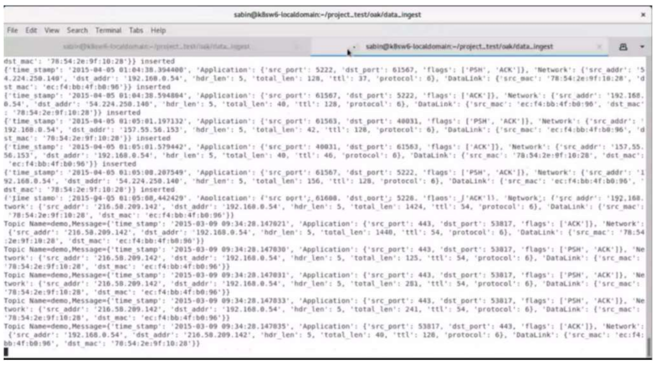
Fig. 7 Data sample of network packet in JSON format

B. Ingesting, pre-rpocessing and storing the data

The final destination of the security data arriving from the sandbox during the simulation is MongoDB. The two components dealing with the ingestion of network packets and security events are **Mosquitto** MQTT broker and **Kafka** consumer and both of them are written in Python.

`ingest_kafka_data.py` performs the ingestion of log data from the sandbox using **`kafka,`** **`json`** and **`pymongo`** libraries. It specifies the parameters for connection with the broker and the topic in order to receive data. Using these parameters, the script creates the Kafka consumer. It receives the data sent by the producer and stores it directly in the **MongoDB** database. A sample of data in JSON format as stored in MongoDB is shown on Fig. 7.

`ingest_mqtt_data.py` script performs the ingestion of network packets using **`paho,`** **`json`** and **`pymongo`** libraries. It specifies the parameters for connection with the broker and the topic to subscribe data. A subscriber is then created using these parameters. It also receives the data sent by the publisher and stores it in the **MongoDB** database. Fig. 8 shows a data sample containing one network packet in JSON format.


sabin@k8sw6-localdomain:~/project_test/oak/data_ingest
File Edit View Search Terminal Tabs Help
sabin@k8sw6-localdomain:~/project_test/oak/data_ingest
sabin@k8sw6-localdomain:~/project_test/oak/data_ingest

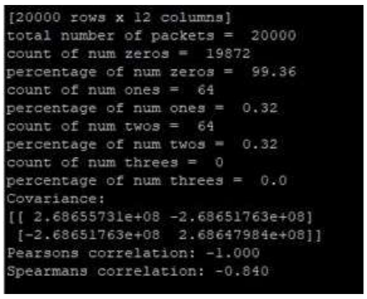
```
dst_mac': '78:54:2e:9f:10:28'}} inserted
{'time_stamp': '2015-04-05 01:04:38.394400', 'Application': {'src_port': 5222, 'dst_port': 61567, 'flags': ['PSH', 'ACK']}, 'Network': {'src_addr': '5
4.224.250.140', 'dst_addr': '192.168.0.54', 'hdr_len': 5, 'total_len': 128, 'ttl': 37, 'protocol': 6}, 'DataLink': {'src_mac': '78:54:2e:9f:10:28', 'd
st_mac': 'ec:f4:bb:4f:b0:96'}} inserted
{'time_stamp': '2015-04-05 01:04:38.594804', 'Application': {'src_port': 61567, 'dst_port': 5222, 'flags': ['ACK']}, 'Network': {'src_addr': '192.168.
0.54', 'dst_addr': '54.224.250.140', 'hdr_len': 5, 'total_len': 40, 'ttl': 128, 'protocol': 6}, 'DataLink': {'src_mac': 'ec:f4:bb:4f:b0:96', 'dst_mac'
: '78:54:2e:9f:10:28'}} inserted
{'time_stamp': '2015-04-05 01:05:01.197132', 'Application': {'src_port': 61563, 'dst_port': 40031, 'flags': ['PSH', 'ACK']}, 'Network': {'src_addr': '
192.168.0.54', 'dst_addr': '157.55.56.153', 'hdr_len': 5, 'total_len': 42, 'ttl': 128, 'protocol': 6}, 'DataLink': {'src_mac': 'ec:f4:bb:4f:b0:96', 'd
st_mac': '78:54:2e:9f:10:28'}} inserted
{'time_stamp': '2015-04-05 01:05:01.579442', 'Application': {'src_port': 40031, 'dst_port': 61563, 'flags': ['ACK']}, 'Network': {'src_addr': '157.55.
56.153', 'dst_addr': '192.168.0.54', 'hdr_len': 5, 'total_len': 40, 'ttl': 46, 'protocol': 6}, 'DataLink': {'src_mac': '78:54:2e:9f:10:28', 'dst_mac':
 'ec:f4:bb:4f:b0:96'}} inserted
{'time_stamp': '2015-04-05 01:05:08.207549', 'Application': {'src_port': 61567, 'dst_port': 5222, 'flags': ['PSH', 'ACK']}, 'Network': {'src_addr': '1
92.168.0.54', 'dst_addr': '54.224.250.140', 'hdr_len': 5, 'total_len': 156, 'ttl': 128, 'protocol': 6}, 'DataLink': {'src_mac': 'ec:f4:bb:4f:b0:96', '
dst_mac': '78:54:2e:9f:10:28'}} inserted
{'time_stamp': '2015-04-05 01:05:08.442429', 'Application': {'src_port': 61608, 'dst_port': 5228, 'flags': ['ACK']}, 'Network': {'src_addr': '192.168.
twork': {'src_addr': '216.58.209.142', 'dst_addr': '192.168.0.54', 'hdr_len': 5, 'total_len': 1424, 'ttl': 54, 'protocol': 6}, 'DataLink': {'src_mac':
 '78:54:2e:9f:10:28', 'dst_mac': 'ec:f4:bb:4f:b0:96'}}
Topic Name=demo,Message={'time_stamp': '2015-03-09 09:34:28.147021', 'Application': {'src_port': 443, 'dst_port': 53817, 'flags': ['ACK']}, 'Network':
 {'src_addr': '216.58.209.142', 'dst_addr': '192.168.0.54', 'hdr_len': 5, 'total_len': 1440, 'ttl': 54, 'protocol': 6}, 'DataLink': {'src_mac': '78:54
:2e:9f:10:28', 'dst_mac': 'ec:f4:bb:4f:b0:96'}}
Topic Name=demo,Message={'time_stamp': '2015-03-09 09:34:28.147030', 'Application': {'src_port': 443, 'dst_port': 53817, 'flags': ['PSH', 'ACK']}, 'Ne
twork': {'src_addr': '216.58.209.142', 'dst_addr': '192.168.0.54', 'hdr_len': 5, 'total_len': 125, 'ttl': 54, 'protocol': 6}, 'DataLink': {'src_mac':
'78:54:2e:9f:10:28', 'dst_mac': 'ec:f4:bb:4f:b0:96'}}
Topic Name=demo,Message={'time_stamp': '2015-03-09 09:34:28.147031', 'Application': {'src_port': 443, 'dst_port': 53817, 'flags': ['PSH', 'ACK']}, 'Ne
twork': {'src_addr': '216.58.209.142', 'dst_addr': '192.168.0.54', 'hdr_len': 5, 'total_len': 281, 'ttl': 54, 'protocol': 6}, 'DataLink': {'src_mac':
'78:54:2e:9f:10:28', 'dst_mac': 'ec:f4:bb:4f:b0:96'}}
Topic Name=demo,Message={'time_stamp': '2015-03-09 09:34:28.147033', 'Application': {'src_port': 443, 'dst_port': 53817, 'flags': ['PSH', 'ACK']}, 'Ne
twork': {'src_addr': '216.58.209.142', 'dst_addr': '192.168.0.54', 'hdr_len': 5, 'total_len': 241, 'ttl': 54, 'protocol': 6}, 'DataLink': {'src_mac':
'78:54:2e:9f:10:28', 'dst_mac': 'ec:f4:bb:4f:b0:96'}}
Topic Name=demo,Message={'time_stamp': '2015-03-09 09:34:28.147035', 'Application': {'src_port': 53817, 'dst_port': 443, 'flags': ['ACK']}, 'Network':
 {'src_addr': '192.168.0.54', 'dst_addr': '216.58.209.142', 'hdr_len': 5, 'total_len': 40, 'ttl': 128, 'protocol': 6}, 'DataLink': {'src_mac': 'ec:f4:
bb:4f:b0:96', 'dst_mac': '78:54:2e:9f:10:28'}}
```

Fig. 8 Ingesting the logs

This MongoDB database has been chosen as a storage for twofold reason – as a NoSQL database it is suitable for real-time data because its memory allocation mechanism matches the real-time data, which makes the storing very fast. It also supports JSON format, which additionally allows storing the data without any additional pre-processing.

C. Correlation of Network Packets and Analyzer Logs

Correlation analysis can be used to reveal dependence between different attributes within one dataset and between live data streams. The correlated data can be synchronized structurally or temporarily which requires preliminary indexing of the data.



```
[20000 rows x 12 columns]
total number of packets =  20000
count of num zeros =  19872
percentage of num zeros =  99.36
count of num ones =  64
percentage of num ones =  0.32
count of num twos =  64
percentage of num twos =  0.32
count of num threes =  0
percentage of num threes =  0.0
Covariance:
[[ 2.68655731e+08 -2.68651763e+08]
 [-2.68651763e+08  2.68647984e+08]]
Pearsons correlation: -1.000
Spearmans correlation: -0.840
```

Fig. 8 Packet Correlation using Pearson Algorithm

We applied correlation analysis to find the dependence between the network packet stream and the flow of security events, captured by the network analyzer, which is based on temporary synchronization. Such a secondary analysis would allow us to detect missed intrusions. For this purpose we implemented the standard Pearson and Spearman methods of correlation using **`scipy`** library of Python with the additional libraries for synchronization **`concurrency`**, **`asyncio`** and

asyncio-mqtt. Fig. 8 shows the correlation metrics for a portion of the network packets stream consisting of 20000 packets and 12 security events. The high degree of correlation in this case shows that there are no missed intrusions. Alternatively, lowering the correlation would indicate potential intrusion which has been missed.

This particular result proves that the simulation is successful and the network analyzer has done a good job to register the malicious behavior. With more realistic experiments, based on real security data this may change. In such a case the secondary analysis would detect the missed intrusions.

D. Analyzing the Traffic using Regression, NN and SVM

Another secondary analysis of the network traffic which can be used to predict the potential security threats is based on classification of the packets within the live according to their type [15]. In this case we needed much bigger datasets. For training we used directly the PCAP files while the algorithms were then applied to the stored data in MongoDB.

We implemented four methods for supervised machine learning – linear and logistic regression, classification using neural networks (NN) and support vector machines (SVM), For this we used the Python libraries for machine learning **scipy** and **scikit-learn**. The results show that the regression methods are not suitable for such tasks, while although the precision of SVM in classification is higher than NN its predictive power is lower (Tab. 1).

Tab. 1 Comparison of four methods for packet classification

Model	Predicted regular packets:	Regular packets in test set:	Predicted ACK packets:	ACK packets in test set:	Predicted SYN packets:	SYN packets in test set:	Accuracy:
Neural Network	129	303	2023	1851	8	6	79%
Support Vector Machine	107	276	2050	1877	3	7	90%
Logistic Regression	417	258	1743	1893	0	9	71%
Linear Regression	791	255	1369	1897	0	8	60%

Legend

ACK	- acknowledgement flag confirming normal exchange of packets between two sites
SYN	- synchronization flag signaling initiation of normal communication between two sides

E. Prediction of Security Threats using CNN

Convolution Neural Networks (CNN) are more powerful method for supervised machine learning which accounts for much deep regularities and dependencies within the data thanks to the use of multiple layers. They are typically used for more complex analytical tasks, such as object recognition in machine vision or naltural language processing. Due to their higher complexity CNN require longer training and much greater computational power than the standard algorithms for machine learning. For prediction of potential threats within the stream of network packets we have implemented an CNN algorithm using **scikit-learn** and **tensorflow** libraries of Python which is based on 7 network layers (Fig. 9).

The model consists of 3 connected convolution layers, 2 post-processing Layers, 2 fully connected layers, and an Activation Layer. Since our analysis addresses a multiclass classification, we are using **SoftMax** activation function in the output layer. The model distinguishes between four types of packets and predicts normal and suspicious flags. In our case, the suspicious flags are RST, SYN, and FIN packets. It has been trained over 30 epochs with a batch size of 64 before updating the model parameters.

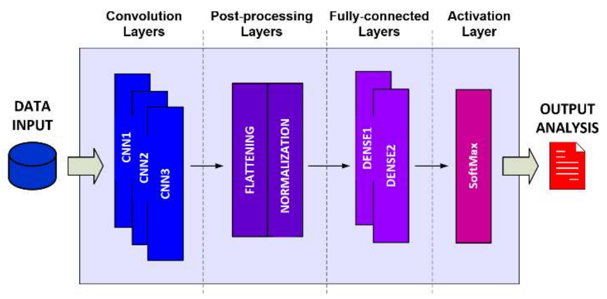


Fig. 9 Convolution Neural Network Architecture

From the original packet data stored in MongoDB database in JSON format we have extracted a portion of 100,000 packets and split it into two sets - training set (75%) and testing set (25%). For convenience of the implementation, they have been transformed into a heterogeneous tabular data structure with labeled axes as (pandas data frames). During the data preparation phase the data has been checked for missing records, incorrect values and duplicate information, and has been cleaned.

Tab. 2 Use of CNN for prediction of suspicious flags

Actual flags within the dataset		Predicted Suspicious Flags	
ACK packets in test set:	39913	Predicted ACK packets:	43647
SYN packets in test set:	13	Predicted SYN packets:	12
RST packets in test set:	165	Predicted RST packets:	164

Legend

ACK	- acknowledgement flag confirming normal exchange of packets between two sites
SYN	- synchronization flag signaling initiation of normal communication between two sides
RST	- warning flag sent after anomaly has been detected in the previous communication

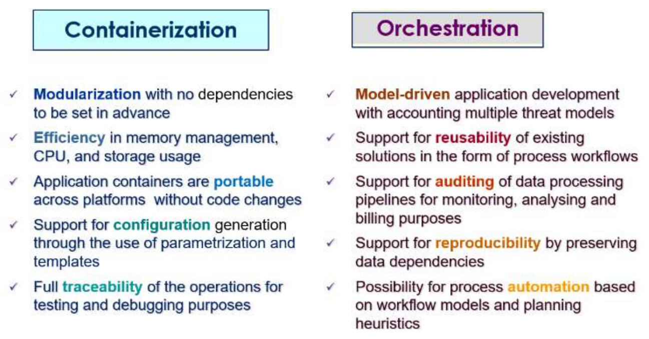
The results of using CNN for predicting suspicious flags are shown in Tab. 2. They are quite good (92-99%) but the data in our case is imbalanced since most of the flags are normal, which is a consequence of the use of simulated data.

V. Take-offs and a Sight at the Future

Completing this project was a challenge because the lockdown during the pandemic interrupted the normal operation of the Cyber Securoty Research Centre of London Metropolitan University, where the project was hosted, and the company closed operations. As a consequence, most of the work has been done remotely using simulated data. Despite these complications, the project completed successfully with a fully functional prototype of a platform for data analysis on private cloud which proved the feasibility of the private cloud data platform for security analytics. Tab. 3 bellow summarizes the advantages of cloud-based solutions for data analysis from software engineering point of view which are all applicable to this project. In addition, the use of private cloud based on public domain software allows to implement lighter solutions which are best tunned to the specific tasks at a lower cost.

Currently we are working on replacing the data generated within the security sandbox used for simulation with proper security data, generated in enterprise environment. The Cyber Security Research Centre of London Metropolitan University have received a large quantity of network traffic from its partner, Palo Alto Networks, which is considering for this purpose.

Tab. 3 Advantages of Cloud-based Security Analytics



Containerization	Orchestration
✓ **Modularization** with no dependencies to be set in advance	✓ **Model-driven** application development with accounting multiple threat models
✓ **Efficiency** in memory management, CPU, and storage usage	✓ Support for **reusability** of existing solutions in the form of process workflows
✓ Application containers are **portable** across platforms without code changes	✓ Support for **auditing** of data processing pipelines for monitoring, analysing and billing purposes
✓ Support for **configuration** generation through the use of parametrization and templates	✓ Support for **reproducibility** by preserving data dependencies
✓ Full **traceability** of the operations for testing and debugging purposes	✓ Possibility for process **automation** based on workflow models and planning heuristics

Of course, there remain some challenging problems which need to be addressed properly. One of the biggest problems is the security of the data processing on both client-side and server-side. If we know the private key used we can implement decryption wherever needed in principle, but the remaining issue is the computational resource of the decrypting device.

We can use an SSUTLS termination proxy server (reverse-proxy), that enables to employ the IDS/IPS inside the trusted network area. Although currently there is no available pre-processor for Snort IDS/IPS to perform the decryption process, it is theoretically possible to develop such a pre-processor or plug-in. The decryption feature is available, for example, in some propriety IDS devices like Juniper IDS.

The traffic can be decrypted on the server instead. We have experimented with a simple private key security for MQTT protocol we use to transport the data, which showed that it is feasible and does not add too big overhead on both sides due to the relatively small amount of data which is transmitted.

Another known issue is the fact that an IDS/IPS system is having problems with the encrypted traffic. A specialised component named SSL Dynamic Pre-processor (SSLPP) enables the IDS/IPS to inspect SSUTLS handshakes of each connection but without data inspection (default disabled). It inspects the unencrypted portion of the connection (headers) for faulty encrypted traffic. To inspect the encrypted traffic without decryption we can use elliptic algorithms instead. This is a modern direction but the research is still in its infancy and it is a long way before it becomes practical.

Acknowledgment

The research reported here was initiated with funding provided by Lloyds Banking Group, focused on the methods for security analytics for detecting fraud and unauthorized intrusions in data processing of financial transactions. Subsequent funding from Innovate UK made possible to continue this research by utilizing the concept of private cloud. During the pandemics the project team received additional support from GATE Institute of Sofia University where the final implementation took place. The team is grateful to all these organisations for the support.

References

[1] Cisco Systems, Inc., Cisco Secure Cloud Analytics [Online: https://www.cisco.com/c/en/us/products/collateral/security/stealthwatch/at-a-glance-c45-739611.pdf; Accessed: 30 Oct 2022]

[2] Palo Alto Networks (2022), Understanding XDR and Modern Threats [Online: https://www.paloaltonetworks.com/cortex/cortex-xdr-resource-center; Accessed: 30 Oct 2022]

[3] Juniper Networks, Advanced Threat Prevention Cloud [Online: https://www.juniper.net/documentation/us/en/day-one-plus/atp-cloud/id-step-1-begin.html; Accessed: 30 Oct 2022]

[4] The MITRE Corporation, MITRE ATT&CK [Online: https://attack.mitre.org/; Accessed: 30 Oct 2022]

[5] Microsoft, Microsoft Defender for Cloud [Online: https://learn.microsoft.com/en-us/azure/defender-for-cloud/; Accessed: 30 Oct 2022]

[6] Google, Cloud IDS [Online: https://cloud.google.com/intrusion -detection-system; Accessed: 30 Oct 2022]

[7] Amazon Web Services, Inc., Amazon GuardDuty [Online: https://aws.amazon.com/guardduty/; Accessed: 30 Oct 2022]

[8] UC Berkeley Center for Long-Term Cybersecurity, Amplifying the upside of the digital revolution [Online: https://cltc.berkeley.edu/; Accessed: 30 Oct 2022]

[9] Darktrace Holdings Ltd, Autonomous Response: Streamlining Cyber Security and Business Operations [Online: https://darktrace.com/resources/autonomous-response-everywhere; Accessed: 29 Oct 2022]

[10] V. Vassilev, A. Phipps, M. Lane, et al. (2020), Two-factor authentication for voice assistance in digital banking using public cloud services, In: Proc. 10th Int. Conf. on Cloud Computing, Data Science & Engineering (CONFLUENCE2020), Noida, IEEE (2020), pp. 404-409.

[11] A. Phipps, K. Ouazzane, and V. Vassilev, Securing Voice Communications Using Audio Steganography (2022), Int. J. Computer Network and Information Security, 14 (3), pp.1-18.

[12] V. Vassilev, V. Sowinski-Mydlarz, P. Gasiorowski, et al., Intelligence Graphs for Threat Intelligence and Security Policy Validation of Cyber Systems (2020), In: P. Bansal et al. (eds.), Advances in Intelligent Systems and Computing, 2020, Vol. 1164, Springer, pp. 125-140.

[13] Y. Patel, K. Ouazzane, V. Vassilev, and J.Li (2019). Remote banking fraid detection framework using sequence learners, J. of Internet Banking and Commerce, 24 (1), pp. 1-31.

[14] V. Sowinski-Mydlarz, V. Vassilev, K. Ouazzane, and A. Phipps (2022). Security Analytics Framework Validation based on Threat Intelligence, in: Int. Conf. on Computational Science and Computational Intelligence (CSCI'22), Dec 14-16, 2022, Las Vegas, USA, IEEE (To appear).

[15] V. Sowinsky-Mydlarz, J. Li, K. Ouazzane and V. Vassilev (2021), Threat Intelligence Using Machine Learning Packet Dissection, In: Proc. 20th Int. Conf. on Security & Management (SAM'21), 26-29 July 2021, Las Vegas, USA, Springer (In Press).