information

Article

Reinforcement Learning for Efficient Network Penetration Testing

Mohamed C. Ghanem * and Thomas M. Chen

School of Mathematics Computer Science and Engineering, University of London, London EC1V 0HB, UK; tom.chen.1@city.ac.uk

* Correspondence: mohamed.ghanem@city.ac.uk

Received: 22 November 2019; Accepted: 16 December 2019; Published: 20 December 2019

Abstract: Penetration testing (also known as pentesting or PT) is a common practice for actively assessing the defenses of a computer network by planning and executing all possible attacks to discover and exploit existing vulnerabilities. Current penetration testing methods are increasingly becoming non-standard, composite and resource-consuming despite the use of evolving tools. In this paper, we propose and evaluate an AI-based pentesting system which makes use of machine learning techniques, namely reinforcement learning (RL) to learn and reproduce average and complex pentesting activities. The proposed system is named Intelligent Automated Penetration Testing System (IAPTS) consisting of a module that integrates with industrial PT frameworks to enable them to capture information, learn from experience, and reproduce tests in future similar testing cases. IAPTS aims to save human resources while producing much-enhanced results in terms of time consumption, reliability and frequency of testing. IAPTS takes the approach of modeling PT environments and tasks as a partially observed Markov decision process (POMDP) problem which is solved by POMDP-solver. Although the scope of this paper is limited to network infrastructures PT planning and not the entire practice, the obtained results support the hypothesis that RL can enhance PT beyond the capabilities of any human PT expert in terms of time consumed, covered attacking vectors, accuracy and reliability of the outputs. In addition, this work tackles the complex problem of expertise capturing and re-use by allowing the IAPTS learning module to store and re-use PT policies in the same way that a human PT expert would learn but in a more efficient way.

Keywords: penetration testing; artificial intelligence; machine learning; reinforcement learning; network security auditing; offensive cyber-security; vulnerability assessment.

1. Introduction

Computer networks are more than ever exposed to cyber threats of increasing frequency, complexity and sophistication [1]. Penetration testing (shortly known as pentesting PT) is a well-established proactive method to evaluate the security of digital assets, varying from a single computer to websites and networks, by actively searching for and exploiting the existing vulnerabilities. The practice is an emulation of the operational mode that hackers follow in real-world cyber attacks. In the current constantly evolving digital environment, PT is becoming a crucial and often mandatory component of cybersecurity auditing particularly after the introduction of the European General Data Protection Regulation (GDPR) for organizations and businesses. In addition to legal requirements, PT is considered by the cybersecurity community as the most effective method to assess the strength of security defenses against skilled adversaries as well as the adherence to security policies [2]. In practical terms, PT as illustrated in Figure 1 is a multi-stage process that often requires a high degree of competence and expertise due to the complexity of digital assets such as medium and large networks. Naturally, research has investigated the possibility of automated tools for the different PT stages (reconnaissance, identification, and exploitation) to relieve the human expert from the burden of repetitive tasks [3]. However, automation by itself does not achieve many benefits in terms of time, resources and outputs because PT is a highly dynamic and interactive process of exploring and decision making, which requires advanced and critical cognitive skills that are hard to duplicate through automation.

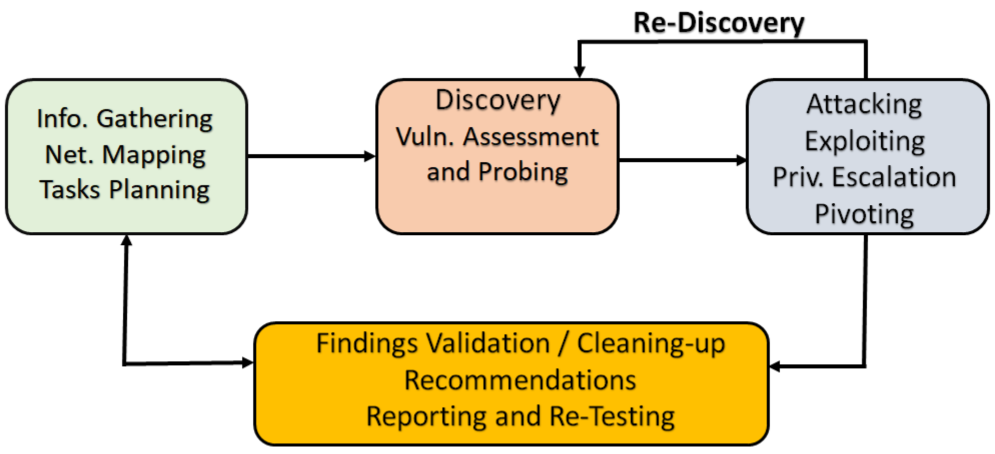

Figure 1. Pentesting (PT) is a non-standard active method for assessing network defenses by following a sequential and interactive multi-phase procedure starting by gathering information and ending by reporting the obtained results.

A natural question arises in regard to the capability of AI to provide a potential solution that goes beyond simple automation to achieve expert-like output [4]. In other research fields, AI has proven to be very helpful to not only offload work from humans but also possibly handle depths and details that humans can not tackle fast enough or accurately enough [5]. Rapid progress in AI and notably the machine learning (ML) sub-field led us to believe that an AI-based PT system utilizing well-grounded models and algorithms for making sequential decisions in uncertain environments can bridge the gap between automation and expertise that PT community experience [6]. In this perspective, the existing PT systems and framework started shifting from executing experts' tasks to become more autonomous, intelligent and optimized aiming that all existing threats are checked systematically and efficiently without or with little human expert intervention [7]. Furthermore, these systems should optimize the use of resources by eliminating time-consuming and irrelevant directions and ensuring that no threat is overlooked.

In addition to the regular use of PT, the testing results (output) should be processed and stored to serve for further use [5]. In fact, the main difference between human PT expert and automated systems is that humans learn alongside performing the tests and enrich their expertise throughout, while systems omit the re-usability of the data which is sometimes crucial especially when the testing is repeated such as regular compliance tests [1]. In practical terms, the vast majority of the assessed network configurations will not change considerably over a short period and therefore the output of previous tests could remain entirely or partly applicable for an eventual re-testing required after one or more of these following points occur:

- Network hardware, software, segments or applications were added, changed or removed
- Significant systems' upgrades or modifications are applied to infrastructure or applications
- Infrastructure changes including moving locations
- Security solutions or patches were installed or modified
- Security or users' policies were modified

Automation is the best solution to save time and resources in any domain, and PT is not an exception to this rule. Therefore, the offensive cybersecurity community gave particular attention to automation during the last decade, leading to improvements seen in saving significant time, efforts and resources in performing tasks [2]. Given the particularity of PT practice, the increasing size and complexity of the tested assets along with the significant number of vulnerabilities, exploits and attack vectors which should be covered by the tester, the blind automated system becomes powerless and often performs worse than manual practice pushing the researcher to focus on improving such systems by adopting a variety of solutions. This paper explores in depth the design and development of an ML-based PT system that carries out intelligent, optimized and efficient testing by perceiving its environment and deciding autonomously how to act in order to perform PT tasks as better as human experts, saving time and resources along with improving accuracy and testing coverage and frequency [8].

1.1. Research Context and Method

Periodically performing offensive security testing and auditing is an essential process to ensure the resilience and the compliance of the assessed asset notably the confidentiality, availability, and integrity. PT is widely perceived to be the best approach to assess the security of digital assets by identifying and exploiting its vulnerabilities [7]. Currently, dozens of commercial and freeware systems, platforms and frameworks are being used by PT experts with some offering automation features which nevertheless remain either local (specific to very limited context or tasks) or not optimized (blind automation), and therefore creating significant accuracy and performance issues notably in testing medium and large networks.

Furthermore, other issues are usually related to existing automated systems, notably the congestion created in the assessed network triggering both security and performance issues along with the associated volume of data generated from the testing outputs that are often unexploited [8]. Finally, the PT environment is characterized as fast-changing and complex, and human experts are suffering from the complexity, repetition and resemblance which in large and complex networks context such as large organizations using standard system and subsequently security protection. Performing PT in alike scenarios will create a high degree of obfuscation and make it almost impossible to cover the whole asset properly [8].

During the last decade, the use of machine learning has intensified in the cybersecurity domain and especially in defensive applications such as intrusion detection and prevention systems (IDPS), malware analysis and anti-viruses solutions [9,10]. Recently, MIT researchers developed a big data security framework named AI2 which combined security analyst expertise with machine learning to build an IDPS with active learning [5,11]. In the offensive cybersecurity domain, there have been few attempts to equip existing PT systems and frameworks with learning capabilities for the obvious reason of complexity associated with PT practice (notably in the modeling and design of an ML-led offensive security system) [7,11]. In fact, it is natural to imagine that one or more machine learning techniques can be applied to different PT phases enabling systems to perform tests by learning and reproducing tests and thus improving efficiency and accuracy over time [3] with the aim of realizing systems capable of imitating human PT experts in performing intelligent and automated penetration [12].

In practical terms, incorporating ML in any PT system will at least reduce recurrent human errors due to tiredness, omission, and stress. It will also boost system performance when performing different tests. ML-based automation will also relieve network congestion and downtime by reducing the number of tests by performing only relevant tests and doing that outside the regular business or office hours and thus avoid any type of assets' availability issues [13]. In this work, we adopted an approach that mirrors the PT expert methodology by designing and implementing a system, intelligent automated penetration testing system (IAPTS), able to self-learn, acquire and generalize PT expertise by allowing an reinforcement learning (RL) agent to interact and act within an environment that en-globe all the feature of PT domain. The extracted expertise is stored and constantly improved to reach an optimal decision policy for future use in similar situations [12]. The PT domain is represented as an RL problem to fit the particular context of learning consisting of sequential decision-making with the rewarding scheme relying on both automated function and human-expert feedback [14,15]. Finally, we pre-set an exploration limit for IAPTS in order to optimize the use of resources we call an exploration-exploitation trade-off which was set to guarantee the best possible results within reasonable use of resources. In term of training approach, IAPTS operates on both real and simulation networks, and the RL module interacts directly with the expert and an adaptable operative mode which depending on the complexity of the tests and the system maturity which leave, at early stages, PT expert full control over the expertise extraction, evaluation and storage by deciding on whether to accept or reject the expertise provided by IAPTS [12].

1.2. Paper Outline

In this paper, we are mainly concerned with the network perspective of PT practice, and we will focus solely on the application of ML and specifically RL techniques to make PT practice intelligent and efficient. The proposed solution can be extended to other types of PT such as web and application testing by introducing some changes in the core program. This paper will start with a brief background on PT practice and highlight the fact that ML is crucial to today's PT frameworks and systems. The second section reviews relevant literature especially ones tackling the uses of AI and ML in the PT practice and the limits and drawbacks of current PT. The third section will briefly introduce the RL approach and justify this choice for the PT context along with presenting the first version of the proposed model and its different components. Section 4 describes the proposed system called the intelligent automated penetration testing system (IAPTS), the adopted learning approach and the modeling of PT as RL problem. Section 5 will describe IAPTS in more detail as well as the tests and results obtained within a specific context and test-bed network. Finally, we analyze and discuss the obtained results and make conclusions along with highlighting future research work.

2. Literature Review

This work is rooted in a long line of applied research works on automating and optimizing offensive cyber-security auditing processes and systems especially vulnerability assessment (VA) and PT [2,11]. Among the most significant contributions in this regard, we present here a summary of previous research with a special focus on the adopted approaches and the contributions. Initially, researchers were interested in the planning phase. Some works were implemented within the industrial PT systems and frameworks while others remained research ideas [2,4]. As PT automation and enhancement domain is situated between both cyber-security and AI research fields, several axes of research were addressed and progressed throughout different research fields and methodologies of automated planning (sub-area of AI) [6,7]. Early research focused on modeling penetration as attack graphs and decision trees reflecting the view of PT practice as sequential decision making [8]. Practically, most of the works were more relevant to vulnerabilities assessment than to PT [4]. Among the most significant contributions in this regard, we present in this literature review section a summary of the previously completed research with a special focus on the adopted approaches and the contributions. For the purpose of clarity, we start by addressing the full picture of the research in this field and proceed later into dividing the research axes by type, methodology, and approach [6].

2.1. Previous Works on PT Automation

Automation is an obvious approach to adopt for PT tasks when the objective is to produce highly efficient PT systems. Nonetheless, automating the entire process of testing including the versatile tasks and sub-tasks for each phase is challenging and often fails to reach the objective if done in an inappropriate way, e.g., the use of automated tools and systems which blindly perform all the possible and available tests without any optimization or pre-processing [6,7]. The automated systems require the permanent control of a human PT expert and often fail to produce acceptable results in medium and large assets context because of the significant number of operations required to cover the entire network [8–11]. In addition to the required time which surpasses the realistic duration of tests, more others issues are created by automation such as the generated traffic (network congestion) and the high number of false positives alerts triggered on the asset defense solution such as IDPSs and firewalls. The use of PT blind automation was limited to small networks and some medium-size networks with the use of customized scripts which are inconvenient requiring substantial effort as well [2].

Early research focused on improving PT system by optimizing the planning phase which was modeled as attack graphs or decision trees problem which reflects the nature of PT practice as sequential decision making. Most of the works were nonetheless relevant to vulnerabilities assessment (VA) rather than PT because of the static nature of the proposed approach and its limitation to the planning phase [4]. Amongst the most significant contributions, we find the modeling of VA as attack graphs in form of atomic components (actions), pre-condition and post-condition to narrow the targeted vulnerability [8] but this approach was more an application of classical planning methods in order to find the best attack graph. Further similar works were carried out on automating the planning of PT tasks but blind automation did not address the problem of enhancing performances and only covered the planning phase of PT practice [7,10].

Nevertheless, a remarkable work on optimization was introduced in [4] by modeling PT as planning domain definition language (PDDL) which for the first time accounted for attacking and post-attacking phases of PT in addition to the flexibility offered by the solution which enabled integration with some PT systems. The proposed solution generated different type of attack plans (AGs) for single and multi-paths PT scenarios which was then directly implemented within the attacking and exploiting system and executed along with interacting with information-gathering tools for transforming the information acquired during that phase into input to a planning problem to be solved separately and then used by the attacking system for the purpose of optimization. The only drawback of this approach was the scalability which was fatal as it was only limited to small and medium-size networks [6].

AI was also considered to improve PT practice in some research [6,7] but most of the proposed modeling approaches failed to deal with the persistent uncertainty in PT practice and especially the lack of accurate and complete knowledge about the assessed systems. An exception was the use of ML algorithms within a professional PT and VA system called Core-Impact in which the PT planning phase was modeled as a partially observable Markov decision process (POMDP) solved using an external POMDP solver to determine the best testing plan in form for attack vectors. However, the proposed model itself is questionable as it obviously fails to model the full PT practice and thus can not cover the remaining testing phases and tasks especially the vulnerability assessment, testing and pivoting phases known to be highly interactive, sequential and non-standard compared with the planning and information-gathering phases [12].

2.2. Drawbacks and Limits of the Current PT Practice

In this subsection, we will present an overview of the domain of PT and the automation of the practice along with highlighting the limitations of the current automated frameworks, systems, and tools. Penetration testing often involves routine and repetitive tasks which make it particularly slow on large networks. These tasks are unfortunately crucial for the practice and cannot just be dismissed although much of this routine can be automated [6]. Although the proposed solutions were in theory very relevant and seemed to solve the problem, the PT practice demonstrated that the brought improvements were not enough to solve the core issue in the practice which time and resources. Some solutions were, on the other hand, fundamentally unfit and inadequate for PT context. It is obvious that human capabilities and performance are limited when it comes to large and complex tasks compared with a machine especially with today's computing power [11].

The average penetration tester can spend days or weeks in testing a medium-size LAN (we are concerned here with comprehensive testing when the entire network is covered). In addition to the time and effort allocated, a considerable amount of systems downtime will be accounted for as a result of the performed tasks [1]. The first two points add to poor performance in term of results quality and accuracy including error and omission which could be crucial resulting from the fact that human makes mistakes, change opinion and get bored [13]. PT automation (automated systems and tools) were presented as a magic solution to the mentioned issues. A fully or even semi-automated solution was thus developed aiming to reduce human labor, save time, increase testing coverage and testing frequency, and allow broader tests. The proposed solution was very diverse in terms of adopted approach when some relied on automated planning (phase 1) by generating automatically attack plan (named attack graph) and then executing the attack in an automated manner. Other solutions were more creative and attempted to mimic the whole process and automate the system to carry out complex (chained) penetration testing tasks and use more exploits [14].

The cybersecurity research community starts questioning the limits of the existing PT systems, frameworks and tools which are expected to become more automated and perform most of PT tasks with little or without human intervention and especially during the first two phases of PT: information gathering and vulnerability discovery. Organizations with a constant need for internal security auditing are, on the other hand, interested in more efficient PT systems that are fully automated and optimized to perform basics and repetitive PT tasks without human intervention [11]. Nonetheless, researchers were struggling with automation as PT practice is a complicated process that humans barely master. Therefore designing a machine to replace PT experts is a challenge given the multi-phase nature of PT practice with high dependency between the different phases and tasks. Besides the complexity of PT practice, the information handled is another major issue as PT reconnaissance and information gathering phase usually produces an incomplete profile of the assessed system and fail to yield a complete knowledge. This issue is often dealt with by an expert by repeating some tasks, changing approach, or simply making assumptions and continuing the tests.

On top of the classic complexity associated with PT, modern attacks adopted evasive techniques and complex attacking paths that allow them to evade network and systems defenses. Skilled attackers would usually seek to achieve their goals through the exploitation of a series of vulnerabilities as part of a chain of sub-attack which enable them to can take advantage of hidden (non-obvious) and composite vulnerabilities (composed of a chain of harmless flaws when together become an exploitable vulnerability) in networks. Each part of the infrastructure or systems may be approved to be secured when considered alone, but their combination and interaction can often provide a pathway for an opportunistic attacker. The ability to detect and analyze the interaction of multiple potential vulnerabilities on a system or network of systems leads to a better understanding of the overall vulnerability of the assessed system [7]. Finally, PT output data is a crucial issue because it is currently not used properly during retesting or future tasks and simply discarded after the PT report generation. In the cybersecurity context, only a few security configurations and system architectures change over the short and medium-term. Therefore most outputs of previous tests remain applicable when a re-testing is required [12]. This particular problem is one of the key motivations of our research.

2.3. Motivation and Contribution

Generally, complexity is the worst enemy of control and therefore security; computer networks are not an exception to this rule. During the last decade, protecting and defending networks and critical digital assets from cyber threats required the security professional to consider a less classic approach (avoiding the trap of bolting on more and more security layers and policies). They turned their attention toward offensive security. Cybersecurity researchers were confronted with the need of an intelligent PT system and framework to support human experts in dealing with high-demand on PT and the associated complexity and risks by allowing systems to take over human and conduct some or all of the PT tasks notably reconnaissances, information gathering, vulnerabilities assessment and exploiting and therefore leave experts focusing on more complex issues such as post-exploiting and testing complex attacks [10,12].

Given the aforementioned facts about PT practice, ML seems to be the best answer to our problem. In fact, several AI techniques were initially considered but only machine learning through reinforcement learning was selected as the most promising option to allow an automated PT system to behave like a real tester in terms of operation and gaining skills gradually along with practice. This research comes to bridge the gap in the current PT practice and will aim to resolve the following issues:

- Reducing the cost of systematic testing and regular re-testing due to human labor cost;
- Reduce the impact on the assessed network notably the security exposure, performances, and downtime during the testing;
- Relieve human experts from boring repetitive tasks and assign them to more challenging tasks;
- Dealing more effectively with changing cyber threats by allowing flexibility and adaptability;
- Perform more broad tests by covering a wide variety of attack vectors and also consider complex and evasive attacking paths that are hard to identify and investigate by human testers.

To sum up, cyber hackers seek to achieve specific goals through the exploitation of a series of vulnerabilities as part of a chain of sub-attacks. IAPTS is intended to do the same by autonomously discover, exploit and control networks machines and networking equipment along with revealing hidden (non-obvious) and complex vulnerabilities in network infrastructure or segment. In other words, IAPTS should cover the entire PT phases and not be limited to the vulnerability assessment proposed so far by [1]. It will also allow more testing coverage and exploration to deal with commonly locally-secured systems that are approved to be secured when considered alone, but the combination of the interaction can result in opening a pathway for an attacker. Finally, the PT expertise extraction, generalization and re-utilization is the ultimate contribution of IAPTS to the current PT practice [12].

3. Reinforcement Learning Approach

The design and analysis of intelligent decision-making system is a major research area in computer science and particularly artificial intelligence (AI). IAPTS perceives its environment and decides autonomously how to act in order to perform PT tasks as well as possible. AI-led cybersecurity solutions are often categorized under two types: expert-driven or automated systems utilizing unsupervised machine learning [16]. Expert-driven systems such AVs, FWs, IDPSs and SIEMs rely on security experts' input and usually lead to high rates of errors until RL techniques were used to guide existence to more goal-directed learning systems that provide autonomous or semi-autonomous decision making which reflect the real-world context and especially offensive security domain such as vulnerabilities assessment and PT context [11]. The main reasons for our choice of RL are:

- Effective autonomous learning and improving through constant interaction with the environment;
- Reward-based learning and existing flexible rewarding schemes which might be delayed to enable RL agent to maximize a long-term goal;
- Richness of the RL environment which helps in capturing all major characteristics of PT including uncertainty and complexity.

Reinforcement learning is a variant of machine learning and a branch of AI. It allows software agents to automatically determine the ideal behavior within a specific context, in order to maximize its performance. Only simple feedback (reward) is required for the agent to learn how to adjust its behavior. In other words, it is based on an agent capable of learning and making decisions, and repeatedly interacting with its environment [5]. Interactions between the agent and the environment proceed by the agent observing the state of the environment, selecting an action which it believes is likely to be beneficial, and then receiving feedback (reward) from the environment that provides an indication of the utility of the action [15].

As shown in Figure 2, RL allows an agent to learn from its own behavior within the RL environment by exploring it and learning how to act based on rewards received from its actions. This decision policy can be learned once and for all, or be improved or adapted if better results are encountered in the future. If the problem is appropriately modeled, some RL algorithms can converge to the global optimum which is the ideal behavior that maximizes the overall reward [5].

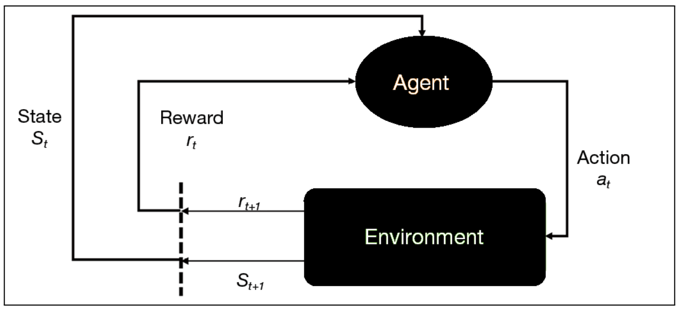

Figure 2. Reinforcement learning (RL) agent observes the state of the environment $x(t)$ at time t, selects an action $a(t)$ based on its action selection policy, and transitions to state $x(t+1)$ at time $t+1$ and receives a reward $r(t+1)$. Over time, the agent learns to pursue actions that lead to the greatest cumulative reward [15].

RL scheme excludes the need for significant intervention from a human expert. In addition, less time is allocated for the learning and customization as it is the case with ML and expert systems, respectively. In addition to what has been said about the suitability of RL for enhancing the automation of PT solutions, RL is a very active domain of research. Several new algorithms have been introduced recently along with some very efficient toolboxes and implementations with the ability to solve complex RL problems under constrained resources with great results [17,18].

3.1. Towards POMDP Modeling of PT

In PT, an attack is a set of tasks that are launched and executed, manually by a human tester or automatically by a PT platform, following a certain order in order to fulfill a goal or reach an objective. Depending on the context, the goal can be predefined or unknown and also can vary throughout the attack. The goal (objective) of the attack is known within the PT community as the target which can be either logical or physical entity. Often, the target is a computer (physical or virtual machine with an operating system and applications) or a computer network or some information hosted on a computer such as files, database servers, or web servers. The attack target can also switch during an attack if a more valuable or easily exploitable target is identified to serve as a pivot later on. Furthermore, it is also common that an attack has no specific target, e.g., a script kiddie running a set of exploits against all reachable machines [10–12]. The starting point for this research is an automated PT system which lack for efficiency and optimization which in term of number of covered tests and the consumed resources and time as any PT test should not last forever and consume an excessive amount into performing or exploring irrelevant tests along with ensuring that no threat is ignored or underestimated. Therefore, the aim is developing an RL-led autonomous PT system that utilizes RL and other techniques at different levels of the practice to improve performance, efficiency, testing coverage and reliability [12].

The starting point was by adopting POMDP [19–25] for modeling PT as RL problem. POMDP models an agent that interacts with an uncertain environment. A POMDP can be defined using the tuple M = <S, A, O, T, Ω, R, b1>. The sets S, A and O contain a finite number of states, actions and observations, respectively. The function T : S * A * S -> [0, 1] defines the stochastic state transitions. After executing action a ∈ A in state s ∈ S, the state changes stochastically to state s0 <S> with probability T(s, a, s0) = P(s0 | s, a). The function R : S * A -> R represents the reward function, such that the reward R(s, a) is received after executing action a <A> in state s <S>. The function Ω : A * S * O -> (0, 1) represents the observation function. Instead of observing the state s0 directly, the agent receives observation o <O> with probability pr(a, s0, o) = P(o | a, s0). The interaction between the agent and the environment is visualized in Figure 2. The agent executes action a, after which it receives observation o and reward R(s, a). Here it is important to note that it does not receive information about the state s itself. The modeling of PT as a POMDP problem and the justification of the choice was already detailed in [12] and some highlights will be provided in the next sub-sections.

3.2. POMDP Solving Algorithm

RL algorithms are methods for solving real-world problems modeled in the form of MDPs or POMDPs which usually involve complex and sequences of decisions in which each decision affects what opportunities are available later. In this work, we are not concerned with the development of improvement of a new RL solving algorithm or methods, but only with finding the appropriate algorithm relevant to our problem which produces acceptable results [16].

When it comes to solving a large and complex RL problem is often complicated and therefore an adequate choice of the solving algorithms and approach should be made. For solving the PT POMDP complex environment, the IAPTS should rely on different solving algorithms rather than simply one. In fact, depending on the context, IAPTS will adapt to utilize to most adequate solving approach. Furthermore, the choice of different algorithms is justified by the constraints IAPTS may face in terms of the available resources (time, memory and computational) which make the use of one solving algorithm challenging, Thus adopting a flexible approach where the accuracy is often sacrificed to acceptance. Finally, it is important to recall that large environments can also cause challenges when solving algorithms especially when dealing with a large number of transitions and observations or opting for a static reward scheme [16].

Most RL solving algorithms fall under two major categories: the reward (value) oriented solving and policy search solving [15]. The reward approach allows an RL agent evolving within the environment to select the sequences of actions that lead to maximizing the overall received reward or minimize received sanctions in the long term run and not only in the immediate future, this approach aims to dress an optimized and comprehensive rewarding function which relies on the atomic reward values associated with the RL environment to determine and an optimal (best possible) rewarding scheme (function) for each transition and observation. In terms of efficiency, this solving approach is often complex and time-consuming with several cases of an infinite horizon if the problem representation is not consistent enough and optimized [5].

The second approach, namely policy search seeks to construct a decision policy graph which is in practice done by learning the internal state/action mapping of the environment and using a direct search method for identifying policies that maximize the long term reward. The optimal policy is reached when all the states and all the actions are tried and allocated a sufficient amount of time to find the best possible associated policies. In this research, we opted for the use of both reward-optimization and policy-search approaches. For the purpose of implementing policy-search algorithms, we found it is useful to include both on-policy and off-policy implementation to allow a better evaluation in terms of policies' quality. The IAPTS POMDP solving module will use a powerful off-the-shelf POMDP-solver allowing the use of different solving approaches and state-of-the-art algorithm to allow the exclusion of all external factors when it comes to evaluating the performances of different solving algorithms [18]. Initially, the following algorithms were shortlisted.

3.3. PERSEUS Algorithm

PERSEUS is a randomized point-based Value Iteration for POMDPs proposed by [15] that performs approximate value backup stages to ensure that, in each stage, the value of each point in the belief set is improved. The strength of this algorithm is its capacity of searching through the space of stochastic finite-state by performing policy-iteration alongside to the single backup which improves the value of the belief points. Perseus backs up also a random basis by selecting a subset of points within the belief set which is enough to improve the value of each belief point in the global set. In practice, PERSEUS is reputed to be very efficient because of the approximate solving nature and is the best candidate for solving large size POMDP problems as it operates on a large belief set sampled by simulating decision sequences from the belief space leading to a significant acceleration in the solving process [19].

3.4. GIP Algorithm

GIP (generalized incremental pruning) is a variant of the POMDP exact solving algorithm family relying on incremental pruning. GIP algorithm replaces the LPs that were used in several exact POMDP solution methods to check for dominating vectors. GIP is mainly based on a Benders decomposition and uses only a small fraction of the constraints in the original LP. GIP was proven in [18] that it outperforms commonly used vector pruning algorithms for POMDPs and it reduces significantly the overall running time and memory usage especially in large POMDP environment context. The latest version of GIP is, to the best of our knowledge, the fastest optimal pruning-based POMDP [17]. In this work we introduced some non-functional changes to the current implementation of GIP notably into the belief sampling to enable the use of an external belief rather than sampling the belief from POMDP environment at the beginning of the solving process, therefore the agent belief will be uploaded directly to the RL environment for efficient use [17,18].

3.5. PEGASUS Algorithm

PEGASUS is a policy-search algorithm dedicated to solving large MDPs and POMDPs and was initially proposed by [20]. It adopts a different approach to the problem of searching a space of policies given a predefined model as any MDP or POMDP is first transformed into an equivalent POMDP in which all state transitions (given the current state and action) are deterministic and thus reducing the general problem of policy search to one in which only POMDPs with deterministic transitions are considered. Later, an estimation value of all policies is calculated making the policy search simply performed by searching for a policy with a high estimated value. This algorithm has already demonstrated huge potential as it produces a polynomial rather than exponential dependence on the horizon time making it an ideal candidate for the penetration testing POMDP solving [20].

3.6. Other Candidates

In addition to the candidates, other RL algorithms will be considered such as backwards induction and finite grid. The latter is an instance of point-based value iteration (PBVI) and will be mainly utilized in determining the shortest attack-path when more than one policy is found. Some of the proposed algorithms are already part of the POMDP-solver software and an optimized implementation is provided by the contributor and constantly improved over the versions. Nonetheless, some algorithm was implemented and integrated for the sake of benchmarking [21,22]. Initially, and as the research focus was to dress a high-quality POMDP model representation for the PT practice bridging the gap between the theoretical research and real-world situation facing the industry professional, the use of such "ready solution" was highly recommended and was hopeful in advancing the research and also for the impact of the results obtained [23]. Finally, we evaluated the algorithm Palm leaf search (PLEASE) designed to solve POMDPs problems with large observation spaces [24].

3.7. POMDP Solving Choices

PT is a complex practice in which the targets can be known or unknown, global or local, simple or composite and each phase is a sequence of non-standard tasks in which the order is a crucial factor. Therefore, the IAPTS should reflect to the best the real-world domain of PT and RL approach here is meant to address the kind of learning and decision-making problems that allow the PT system to capture, reproduce and store expertise in the whole PT tasks and sub-tasks relying on well established RL solving algorithms elected to be the fit to PT context and produces acceptable results [25,26]. The PT practice is thus represented as POMDP environment and serve as an input to the off-the-shelf solver in which a decision-making agent will be exploring its environment to aiming to maximize the cumulative reward it receives or finding the optimal policies graph (PGs) through the RL agent which perceives the environment and solve the problem by estimating the value function to dress the best decision policies or rewarding function [18].

To summarize, we presented a general background of the RL domain and justified the choice of such approach for IAPTS along with a brief introduction of the considered candidate algorithms and their advantages and in consideration of the specific context of PT in complex and large RL environments [27]. Finally, we discussed the solving approaches notably the value-function and policy search, and we decided to opt for the policy search algorithms given IAPTS' aims for sequential decision making under uncertainty in an environment when the RL agent will be assigned objectives but with no guidance on how to achieve such objective which indeed reflect PT expert behavior [28,29].

4. IAPTS Design and Functioning

The proposed intelligent automated penetration testing system (IAPTS) functioning diagram is detailed in Figures 3 and 4. Python scripts were developed to perform the pre-processing from the raw data and then the produced results are used into optimizing the representation of the PT domain in form of POMDP problem. The IAPTS knowledge base (memory) will be initially handled manually and a human PT expert will decide on the storage of the obtained results (policies extracted after applying the generalization) along with the management of tasks related to expertise extracting and storing. In other words, the extracted expertise will be performed manually until the IAPTS reach a pseudo-maturity state in which it will be in charge of capturing, assessing and storing the expertise will be implemented and embedded within the IAPTS expertise memory. The projected IAPTS will be an independent module that can be embedded with the industrial PT framework. The current version of IAPTS is associated with metasploit framework (MSF) as an external module communicating via customized python scripts with Metasploit "MSFRPC" API. The purpose of such configuration is to avoid modifying the core component of the MSF and allowing us, for research purposes, to measure the IAPTS performances away from the PT framework.

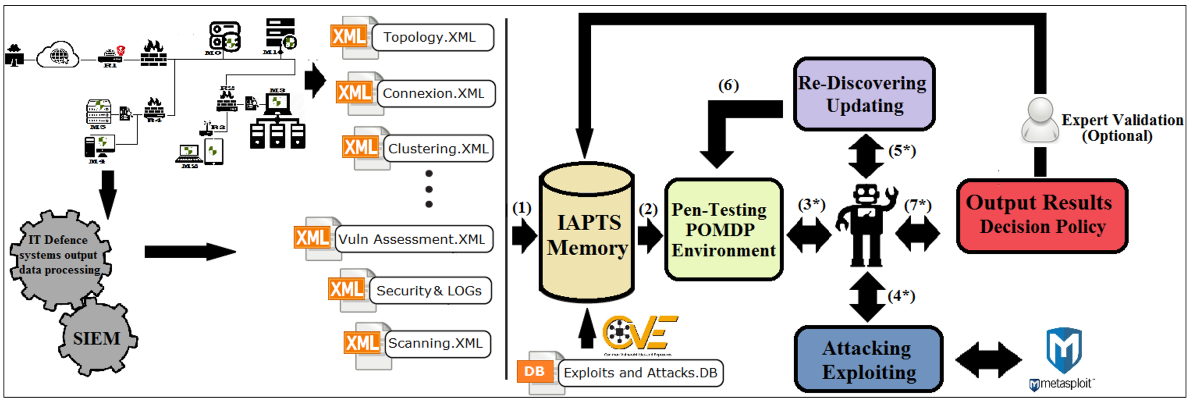

Figure 3. IAPTS functional diagram.

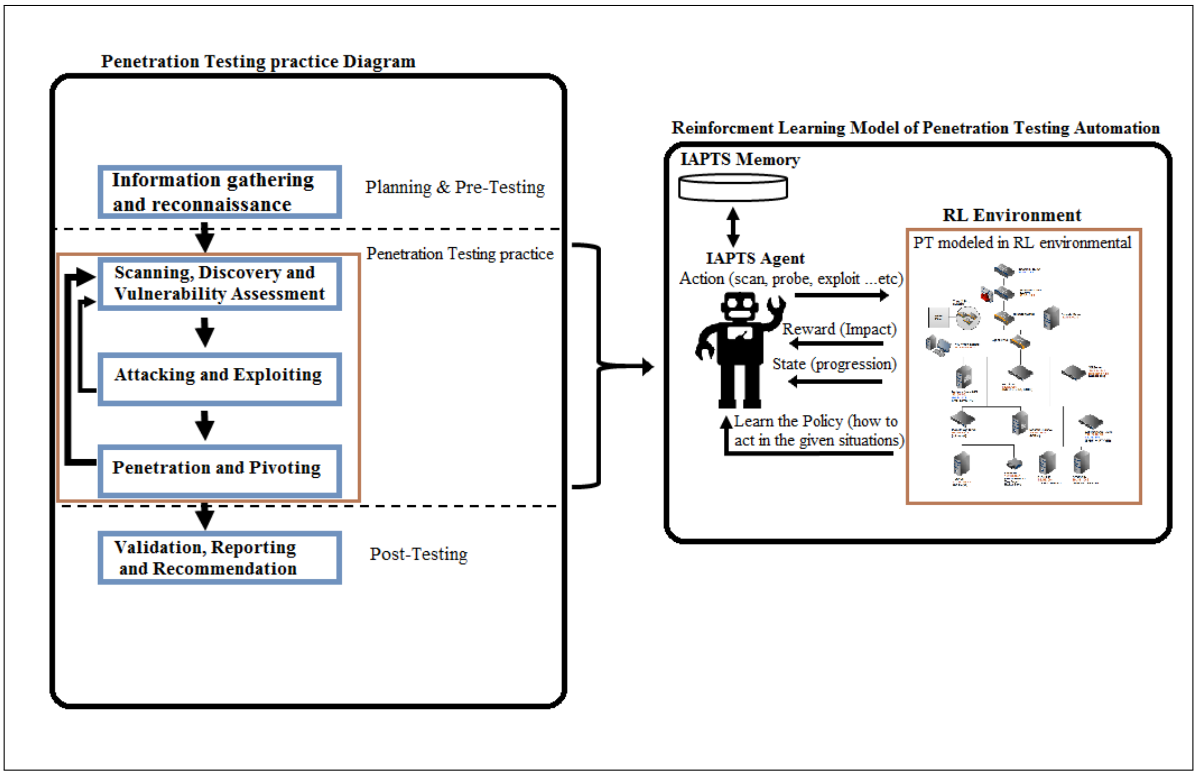
Penetration Testing practice Diagram
Information gathering and reconnaissance
Planning & Pre-Testing
Penetration Testing practice
Scanning, Discovery and Vulnerability Assessment
Attacking and Exploiting
Penetration and Pivoting
Validation, Reporting and Recommendation
Post-Testing
Reinforcment Learning Model of Penetration Testing Automation
IAPTS Memory
RL Environment
PT modeled in RL environmental
IAPTS Agent
Action (scan, probe, exploit ...etc)
Reward (Impact)
State (progression)
Learn the Policy (how to act in the given situations)

Figure 4. IAPTS modelling of PT as RL problem diagram.

4.1. IAPTS Operative Modes

IAPTS will evolve through different levels of automation and intelligence to reach the pseudo-maturity level in which it should be able to perform an entire PT on networks. Overall, IAPTS can operate in four different levels which are dictated by the development of the system knowledge base in term of captured and generalized expertise as follow:

1. Fully autonomous; IAPTS entirely in control of testing after achieving maturity so it can perform PT tasks in the same way that human experts will do with some minor issues that will be reported for expert review.
2. Partially autonomous; the most common mode of IAPTS and reflect the first weeks or months of professional use when IAPTS will be performing tests under the constant and continuous supervision of a high-caliber PT expert.
3. Decision-making assistant; IAPTS will shadow human experts and assist him/her by providing a pinpoint decision on scenarios identical to those saved into the expertise base and thus alight tester from repetitive tasks.
4. Expertise building; IAPTS running in the background while human tester performs tests and capture the decisions made in form of expertise and proceed to the generalization and of the extracted experience and build the expertise base for future use.

4.2. From PT to A Reinforcement Learning Problem

We present here an improved version of the modeling of PT practice as a POMDP problem which constitutes the core module of IAPTS. For simplicity purposes, we use an illustrative example to introduce the different steps towards the representation of a PT domain in form of POMDP problems. In the context of PT, we believe that there is no need to represent the entire network topology and security configurations in the RL environment but only representing specific data judged relevant from the PT point of view and thus lightening the RL environment [12]. The RL representation will capture the following information about the assessed network: machines and networking equipment architecture, connectivity and reachability, network defense and security configurations. The aforementioned information will be used to create a PT-style view of the network without encumbering the RL environment and impact the performance. In addition, we used pre-processing output relevant data to be included within the environment or to serve in enhancing RL learning algorithms to acquire such as proxy server logs, web-server logs, database logs, routing device logs, apps and other security logs [12].

4.3. Representing Network PT As RL Environment

We describe here the process of elaborating an RL environment starting from a given PT example. The overall extraction and elaboration process is explained, in a mirror with the PT diagram, in Figure 4 in which we build upon the IAPTS logic into converting the PT domain into POMDP representation along with respecting real world PT and adopting the same approach into the elaboration of the POMDP environment sections. noting that all the sections are dynamic and allow high frequency-changes as the PT progress and information are updated or upgraded. The following are the different components of the RL POMDP environment:

State space: contains all relevant information, from the PT expert view, about the assessed network. It will include information about any software or hardware machine including virtual and networking equipment that runs an OS. The information is OS parameters, port, services and applications, patches in addition to relevant security and connectivity information. This information is represented in POMDP language using a special notation that aims to minimize the size of the file but remain concise, clear and precise. In practice, most of the Action space is dressed at an early level as modern PT relies on initial information gathered during the first phases. nonetheless, some information will remain missing or not accurate enough and thus represented in a probabilistic way after being enhanced by information coming from the pre-processed output to avoid redundant or useless representations [20]. Any machine or device within the network will be assigned a number "i" and will be represented as Mi or Ri and the remaining associated information is represented in, but not limited to, the following way Mi-OS1-Port80-ServiceXXX or Ri-OS2-Port443-ServiceYYY. This information will be continuously updated as the discovering and scanning tasks progress to confirm previous probabilistic information or to add a new one. Furthermore, modern network routers are more than simple transmission equipment, in fact, they can run operating systems and embed one or more security isolation and protection mechanisms notably FWs, AVs, IDPSs, VLANs and others. Following this logic, network and firewalls can be considered as machines (running OS and thus having vulnerabilities) or just security isolation border for clustering purposes detailed later.

In addition to the machine and devices information, state-space will include information about the networking and security configuration of the assessed network such as connectivity, security isolation (sub-net, virtual LAN) and defense restrictions. The purpose of such representation is to enhance and optimize the input for the POMDP solving algorithms so a better RL environment is represented. The following example summarises the information captured about two machines Mi and Mj as Mi-Mj-TCP-SSH-0". Only relevant security and networking configurations information is considered and machines that belong to the same segment and have the same protection should be represented together. Then we represented other segments' machines.

Action space: POMDP model action is an exact reflection of the PT actions performed by testers and thus en-globe all PT tasks and sub-tasks following a certain notation. As with any RL problem, the number of action is known, static and limited and PT does not fall out of this logic and we include in this space as variety of PT related actions such as Probe, Detect, Connect, Scan, Fingerprint, VulnAssess, Exploit, PrivEscal, Pivot in addition to some generic action that will be used for control purpose by RL agent. The number of actions that the expert can perform is huge and cannot be totally represented within the RL action space such as Terminate, Repeat and Give-Up and others as detailed previously in [20]. Furthermore, as in PT domain successful or failed action might require further or repeating actions we defined some additional actions in order to differentiate between the original action and the others action. In practice, the purpose of such representation is to deal with the special and complex scenarios notably:

- a failed action to fully (root) control a machine that leads to further action attempting user session or escalates privileges or switching to other attack paths;
- dealing with action relying on uncertain information and fail because of the assumption made and require further actions when additional information becomes available and might be successful;
- actions prevented or stopped by security defense (FWs or IDPSs) which may be re-attempted under different circumstances.

4.4. POMDP Transitions and Observations Probabilities

In the first phase of this work, transitions and observations were uniformly sampled. After multiple attempts, we found out that in the particular context of PT, it is far more efficient and reasonable to use real-world data built from IAPTS past tests and enhanced by the human expert initially meant to passively supervise the IAPTS. The data used to artificially simulate testing environment is captured and stored by IAPTS during the regular testing but is carefully inspected by the authors who will rely on their expertise to only include the adequate data and discard the rest of the data. In addition to the regular output of the past experiences, failed or incomplete testing scenarios will be of crucial use during the retesting process. In fact, as IAPTS aims to gradually replace human expert in PT, the system should act as human in dealing with failure into performing some PT tasks or successfully carrying-out tests. Similar to humans, IAPTS will use an evaluation procedure to recognize that what has been done could be useful in another context or with minor amendments for a similar context. In IAPTS, we rely initially on human expert interaction to provide feedback on the failed and incomplete testing to select and store the highly prominent ones for future use even if they ultimately failed. In terms of data, IAPTS will be mainly dealing with the policies stored into the PG file which constitute the outcome of the POMDP problem solver [12].

In this research, the probabilistic output of PT action (scanning, fingerprinting, exploiting) was a crucial factor we considered doing allocation the adequate probabilities for Transitions and Observations in order to mirror the real-world PT practice. Therefore, we opted for a cross-validated method using two well-established and standard sources, respectively, NIST National Vulnerability Database (CVSS) [30] and Common Vulnerability and Exploits (CVE) [31] which constitute a reliable online catalog for all known proven vulnerabilities associated with a different type of operating systems, software and applications. The use of such sources is motivated by the rich content, easy accessibility, regular update and the available scoring function and mechanism such as CVSSv3 and the calculation of the probabilities associated with each transition or observation are detailed in [30,31].

4.5. Rewarding Schema

On the other hand, IAPTS rewarding will be two-fold depending on the system maturity. In the early stage, IAPTS will rely solely on the rewards allocated by the PT expert supervising it along with some default rewarding values. Rewarding the performed actions will be predefined by a human expert who will have to decide the adequate reward for each action performed depending on his/her overall insight he got on the practice, experience and testing achievements. Afterward, IAPTS will relieve the human expert from the rewarding task and only request a human decision on the global PG (attack policies). IAPTS reward function will be utilized, and thus the reward for the performed actions will be calculated following well-established criteria such as: reaching a terminal state; achieving a final (global) target or local goal (controlling intermediates machines); or failing to reach any goal. The criteria for the choice of rewards will mainly be: the estimated value of the achievement, the time consumed; and the associated risk of detection as detailed in [12].

4.6. IAPTS Memory, Expertise Management and Pre-Processing

This research is all about applying RL learning to medium and large LANs which subsequently mean that the projected system IAPTS will need to deal with a big amount of intimation described as complex and redundant among the cybersecurity community. Modeling and representing the PT as POMDP environment is particularly complex and will result in producing a huge POMDP environment, and thus make it impossible to solve given the restriction in time and computing power (memory). Therefore, smart use of resources is required. The system memory as shown in Figure 5 is used for dynamically storing the data handled by the system such as the environment attributes (states, action, observation, transition, reward) and agent's memory (data regarding the policy and acquired knowledge and experiences that an agent gains by acting within the environment). In fact, the first part of this research will focus exclusively on searching the policy as an agent acts within the environment in a particular state and receives a reward from the environment. Initially, for the purpose of research facilitation, the reward value will be pre-defined by a human expert so no reward function will be used. Moreover, the knowledge gained in similar situations to equip the penetration testing system with "expert knowledge" will be completely done by this module in the future. In practical terms, IAPTS will solve the RL problem, extract PGs and instruct MSFRPC API which will execute the testing plan and keep updating the IAPTS of the outcome on a real-time base especially at vulnerability assessment and exploiting phases. This will enable IAPTS to adjust and adapt the tests as well as the post-exploitation tasks such as pivoting or privilege escalation [1,12].

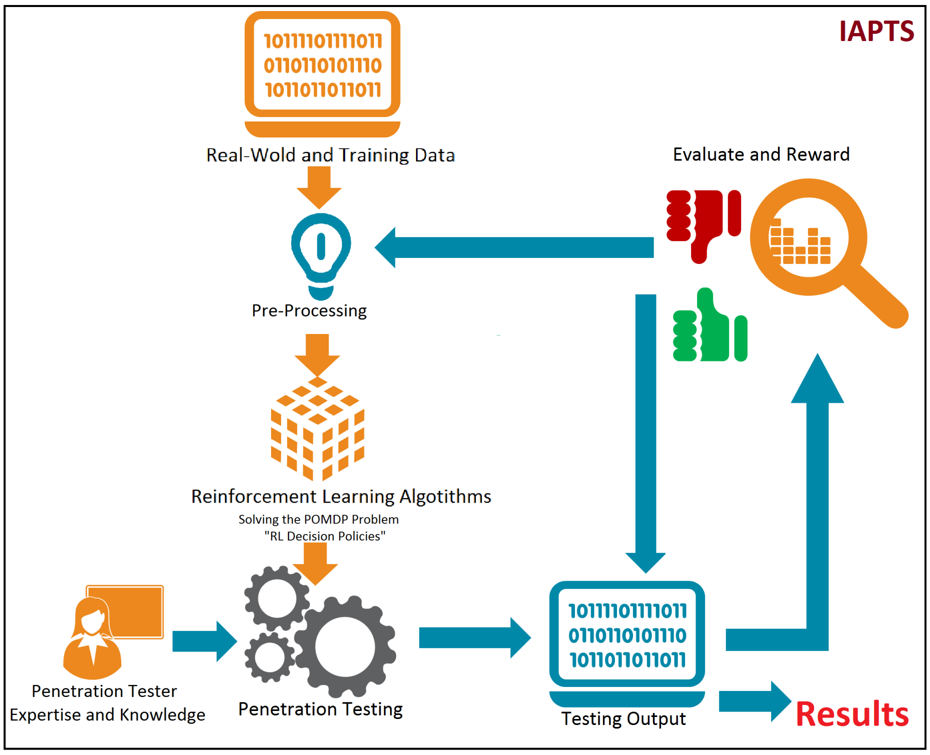

Figure 5. IAPTS learning, expertise extraction and validation procedure

In addition to the RL framework on which the system will operate, a parallel knowledge-based expert system will be implemented and constantly (with every practice) enhanced and fed. This pseudo-system will serve as RL initial belief. This system will capture details of the performed (manually) action by the human tester and also extract knowledge from the output of the information-gathering phase and security system data (Firewalls, AVs, IDPSs and SIEMs) and structure the relevant details. One can say that such a system will be useless alongside the RL system which is a legitimate interrogation. The answer will be that giving the known limits of RL in multi-dimensional state context along with the important size of the RL components, including initial belief detail will only slow down the system performance. Furthermore, the crucial information extracted from the security data will otherwise be omitted [12].

The only remaining issue is the human intuition (the ability to acquire knowledge without inference and/or the use of reason) which a system will not be able to substitute. Intuition provides penetration tester experts with beliefs that cannot be justified in every case and humans can sometimes solve some brilliant problems without the use of any reasoning. Artificial decision making is the ultimate aim of the use of AI but still cannot model the intuition. As a result, this issue will be sorted out by allowing the controlling human to interact with the system. In other words, a mechanism to obtain feedback from the expert tester (security analysts) should be utilized to overcome this issue. The feedback (along with the surrounding context) will be stored in the system memory for future use. The system memory will incorporate policy assessment and generalization features and experience-replay from the previous test where the human expertise is extracted and defined as a policy automatically by the system (direct learning) along with the management of the input and output data such as the initial belief and reporting [12].

Prioritized experiences replay is an effective approach to improve the learning and thus efficiency in RL algorithms. In this work, we adopted this approach, but introduced some modifications for technical reasons, in order to enable RL algorithms to prioritize the use of certain sequences of transitions over others in order to enhance the learning of the IAPTS RL agent. In addition to selecting the most plausible and relevant policies (state-action decision sequences), we injected some other artificially constructed transition sequences using information gathered from previous tests which were validated by a human PT expert. These sequences, when replayed, allow value function information to trickle down to a smaller selection of POMDP transitions and observations, thereby improving solving algorithm efficiency in terms of consumed time and memory. All the proposed customization was implemented within a modified version of the standard POMDP solving GIP LPSolve algorithm we called "with initial belief" [22].

Finally, it is important to introduce our modified GIP LPSolve algorithm which was meant to improve the performance of IAPTS and also allow the IAPTS to capture the appropriate expertise in form of decision policy) process it to make it general decision rule and store it within IAPTS memory for future use. The simplest way to illustrate the importance of learning on the long-term PT practice by adopting a test scenario inspired by the real-world situation of re-testing the same network after some updates or upgrades. In the retesting process, one or more machine configuration will be changed but not all of the machines and therefore IAPTS will re-use already acquired PG when it comes to repeat PT on the partially modified network with the use as an initial belief the output of the previous tests [18,29,32].

5. Testing IAPTS and Results

5.1. Simulation Platform

To test IAPTS, we designed and implemented several test-bed networks of different sizes. In the first phases of this research, we aimed to assess the effectiveness of the proposed POMDP modeling of PT and evaluating our choices in terms of learning approaches, used algorithms, and capturing and managing the expertise as we discussed in detail in [12]. The adopted test-beds are, to the best of our knowledge, an illustration of the real-world networks used widely by different types of organizations which include: Internet-connected side, DMZ, intranet and internal sensitive segments where crucial data is kept securely. In this work, we tested IAPTS performance on different size experimental networks composed of a number of machine (computer) and networking routers varying from 2 to 100 machines. Networking equipment is considered as machines as well as any network equipment that runs an OS and applications. The only excluded machine is the hacker(s) computer(s) which will be represented as one entity along with the Internet. Figure 6 shows a sample test-bed network of 10 machines (seven computers and three routers).

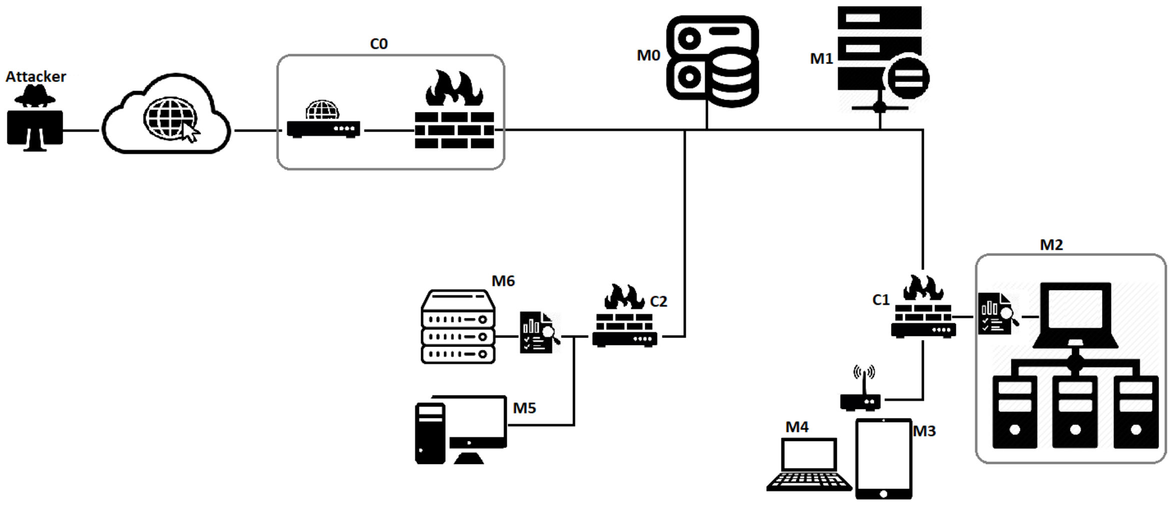

Figure 6. 10 Machines test-bed network.

5.2. IAPTS Results

Evaluating IAPTS is a multi-step operation starting by validating the RL approach, then examining the obtained results in solving real-world PT associated POMDP problems, and finally analyzing the relevance and accuracy of the obtained results from PT point of view. In practice, the output of the RL solving is acting policies graphs (PGs) which undergo additional processing to convert the results into a more understandable format. In addition to the consumed time for solving the POMDP problem, other factors will be considered notably the time required to perform different PT tasks by the Metasploit MSF and other variables which are either calculated or approximated in order to define the overall consumed time that IAPTS will take to perform a full testing on the test-bed networks. The obtained results shown in the Figures 7–9 illustrate an initial comparison of different RL solving algorithms performances on different LANs which were also compared with manual PT consumed time basing on author experience as PT consultant and also the overall time required to perform an automated comprehensive PT with no optimization (refer to results obtained by authors in [12]) and it is clearly obvious that IAPTS outperforms both manual and automated PT. In addition, different discount rates were considered in the optic of finding a suitable balance between performance enhancements and preserving the realistic nature of our IAPTS. The discount rate of "0.5" was selected following multiple testing and simulations.

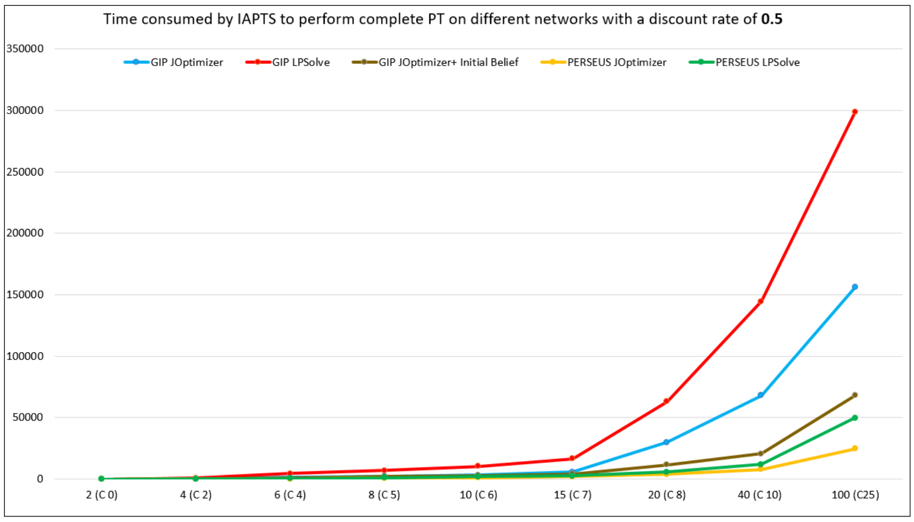

Figure 7. Time in seconds required by IAPTS to complete PT tasks on different LANs' sizes with a Discount rate of 0.5

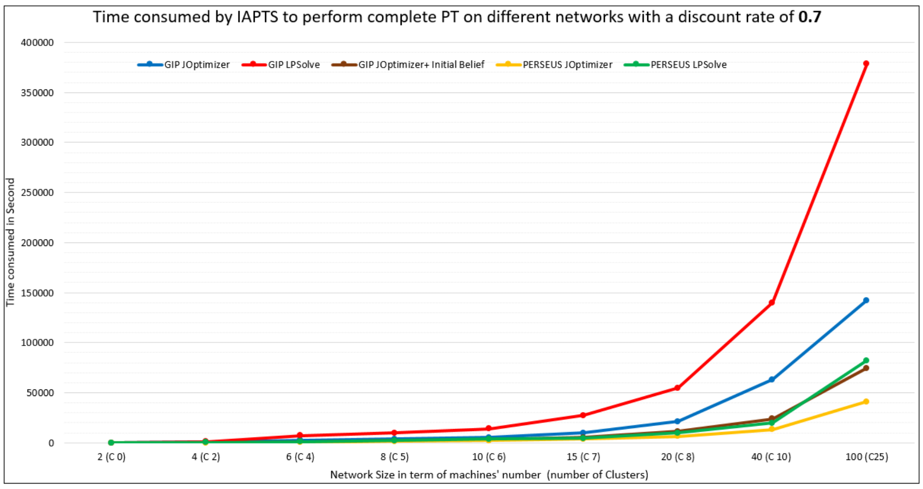

Figure 8. Time in seconds required by IAPTS to complete PT tasks on different LANs' sizes with a Discount rate of 0.7

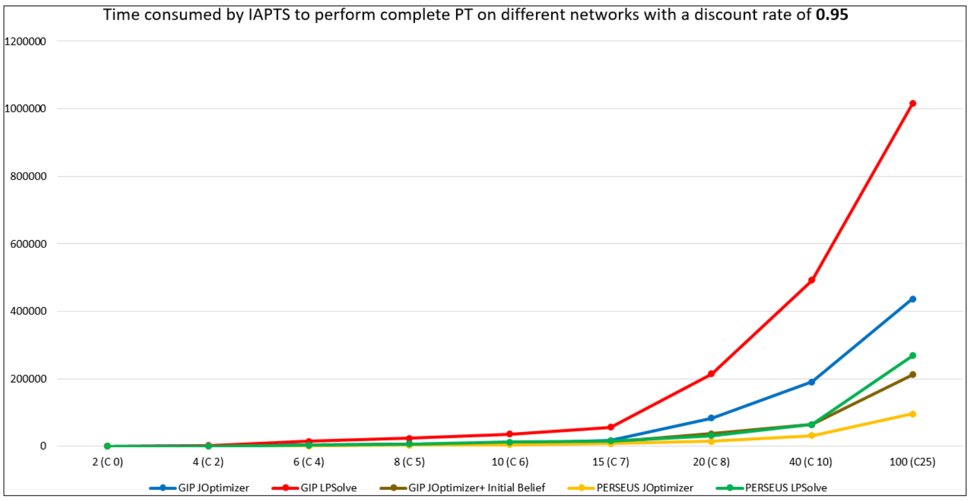

Figure 9. Time in seconds required by IAPTS to complete PT tasks on different LANs' sizes with a Discount rate of 0.95

Following the obtained results, we decided to introduce some changes within the solving algorithm and notably GIP aiming a better performance from IAPTS on a short term basis. We opted for prioritized transitions and observations through the manipulation of the associated probabilities along with introducing some customization into the initial beliefs sampling. The obtained results were surprisingly good, and the new variant of GIP which we named GIP-LPSolve with Initial Belief performed much better than the classic GIP in both time consumed and PG accuracy as shown in Figures 7–9. Furthermore, in order to assess IAPTS performance expertise extraction and storing capabilities and the impact of performance enhancement, we proceeded to re-test the same network with or without introducing minor or major changes to a different number of machine configurations. The obtained results in the context of a 10-machine LAN were nearly perfect as the performance enhancement was huge especially when re-testing the very same network as shown in Figure 10.

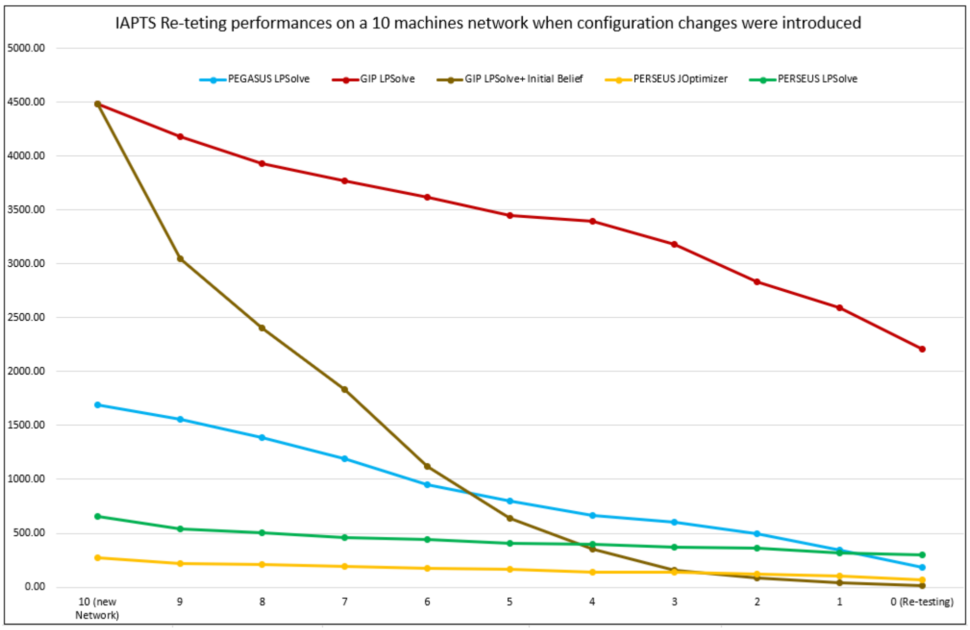

Figure 10. IAPTS re-testing performances' enhancement by algorithm on 10 Machines LAN

Finally, on the top of the overall performance enhancement and notably, when using GIP LPSolve with initial belief algorithm, the quality of the produced decision policies was beyond human expertise especially in the case of the 10 machines network when IAPTS highlighted two additional attack vectors which an average human PT expert would easily omit and illustrated in Figure 11.

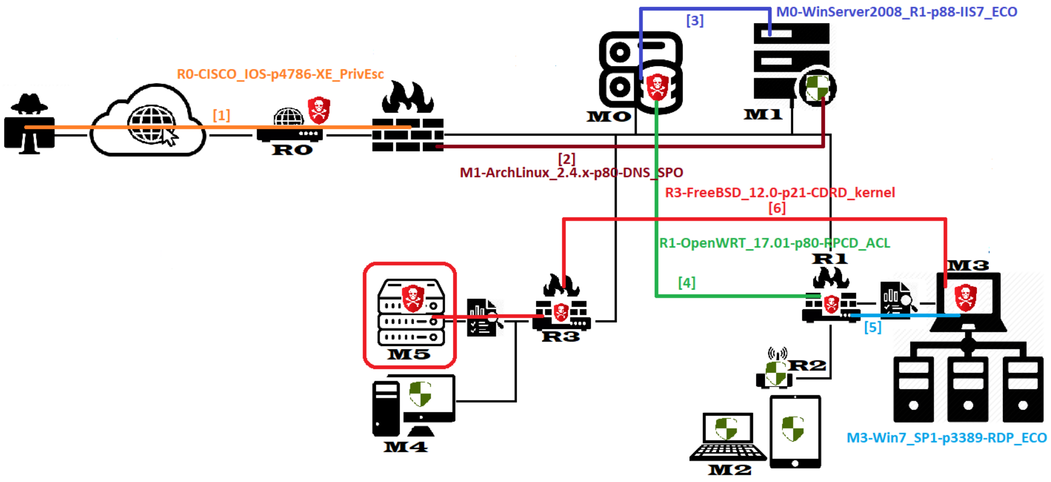
R0-CISCO_IOS-p4786-XE_PrivEsc
[1]
R0
M0-WinServer2008_R1-p88-IIS7_ECO
[3]
M0
M1
[2]
M1-ArchLinux_2.4.x-p80-DNS_SPO
R3-FreeBSD_12.0-p21-CDRD_kernel
[6]
R1-OpenWRT_17.01-p80-RPCD_ACL
R1
M3
[4]
[5]
M5
R3
R2
M4
M2
M3-Win7_SP1-p3389-RDP_ECO

Figure 11. Example of IAPTS output PT policy translated into attacking vectors

5.3. Discussion

The obtained results consolidate prior thoughts on the role of ML and specifically RL in the performance enhancement and optimization of resources use in PT. Commercial and open-source PT systems and frameworks were designed initially to work either under human instructions or in a blindly automated manner, but both approaches fail to address the current environment in which PT practice is evolving notably the increasing size and complexity of the networks, the high number of vulnerabilities and the composite testing scenarios which mimic modern hackers operating approaches. RL was very efficient when used properly, and IAPTS results are additional evidence of the drastic performance enhancement comparing to an average human tester. Several other positive points were noticed, notably the pertinence of the produced result (acting policies) in terms of relevance, coverage and accuracy. In practical terms, using the adequate RL algorithm and adopting a new learning scheme enabled IAPTS to produce a very optimized attacking policies when targeting the Machine M5 suspected to contain sensitive information and defined as the most secured machine within the test-bed network as illustrated in Figure 11. Indeed, the produced policy is from an attacker point of view obvious but getting an automated system to opt for such attacking vectors despite being not minimal in terms of cost of the exploits and consumed time is the novelty in IAPTS which is able to sacrifice simplicity for a higher objective. IAPTS exploring and large coverage capabilities were able to find a very complex and non-obvious attacking path in medium-size networks where, relying on authors experience, no human tester will be tempted to adopt and possibly neglect in-spite of being very relevant and constitute a possible attack path which a real hacker can exploit it.

Furthermore, the proposed enhanced GIP-LPSolve which utilizes a new mechanism in creating and managing POMDP initial belief was proved very efficient especially in small and medium-size LANs. In fact, GIP LPSolve is a variant of an exact solving RL algorithm which are often labeled as good in results quality but bad in performance, but the introduced changes in initial belief sampling and managing along with prioritizing some decision sequence over others enabled the new variant to perform much better and even outperform other RL approximate solving algorithms. On the other hand, re-testing of the same network after the introduction of minor changes in few machine permitted to appreciate the full contribution of RL to PT practice by cutting drastically the consumed time and thus allowing a fast and reliable re-testing which is often the case in PT when periodic re-testing is compulsory despite the lack of any significant configuration changes within the networks systems.

The explanation of the obtained results with different algorithms roots deep on the functioning of the algorithms and how they use the data represented in the POMDP environment along with the initial belief sampling. In fact, policy search algorithms adopt a direct but approximate technique to solve the RL problem and thus sacrifice accuracy for the sake of performances and this applies for the two versions of PERSEUS and PEGASUS with different levels depending on the solving approach and optimization. On the other side, the exact solving algorithm and notably the GIP are famous by their bad performances but good results and utilizing this algorithm within IAPTS were not meant for finding the exception but for comparison and evaluation purposes. The obtained results with the GIP were expected and confirmed that relying on the basic version of GIP would not produce any positive output, and therefore we implemented a customized version (mainly on belief sampling and sequences prioritization) which indeed produced far better results on re-testing scenario and even outperformed the approximate approaches in case of exactly or nearly similar network re-testing where the adopted customization brought the sought impact on the IAPTS performance. Finally, it is important to highlight the fact that this work is not intended to evaluate the RL algorithms but to pick and use the best ones for the penetration testing context, and thus no details of algorithms functioning were provided and discussed in line with works [27,32].

To highlight the performance enhancement brought by IAPTS especially on small and medium network contexts, we compared it with manual and blind automated approaches in terms of time consumed to perform a comprehensive (all possible attack vector exploration) testing of different network sizes. Figure 12 illustrates the performance enhancement which is more obvious compared with an automated testing system. In fact, IAPTS performs better than the manual expert tester with all the associated expertise and cost in small networks and falls short on medium LANs. This was expected as solving large POMDP problems and time wasted in interactions will slow down the IAPTS. Nevertheless, the weakness in the performances is covered by IAPTS testing coverage (validated PGs) which exceed by at least double human covered attack paths, a human tester is often pushed to pre-eliminate some testing vectors or omitting some complex attacking vectors which in some cases revealed catastrophic for the asset security. The coverage and exploration metrics will be measured in detail and discussed in future works. Finally, we noticed that IAPTS performances on large size LANs decrease sharply and this is mainly due to the complexity which impacts the size of the POMDP environments along with usage of memory during the solving of the problem. This major issue is currently being dealt with by proposing a hierarchical PT POMDP model relying on grouping several machines under the same cluster which will be detailed in future works along with improving IAPTS pre-processing.

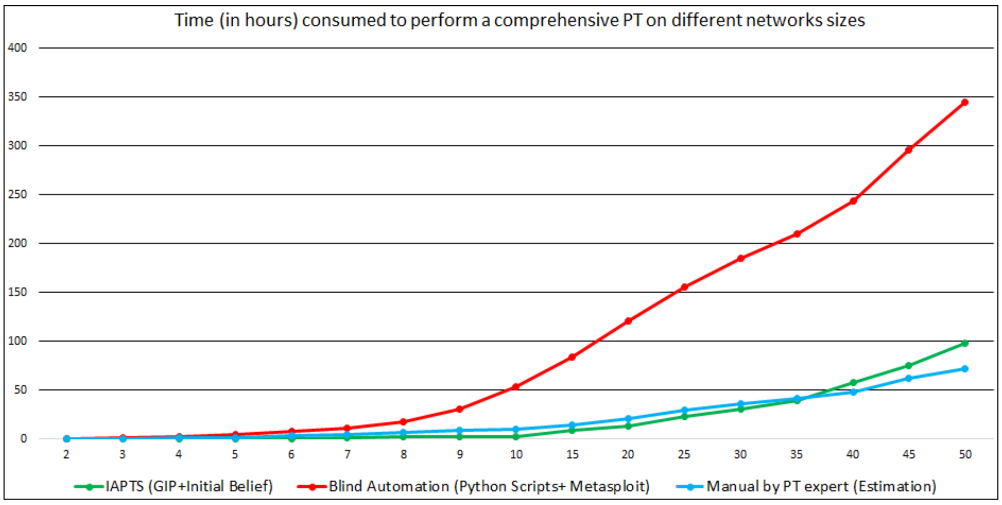

Figure 12. Comparing IAPTS performances with manual and automated testing

6. Conclusions and Future Works

This paper explores a novel application of RL techniques to the offensive cybersecurity domain which allows PT systems and frameworks to become intelligent and autonomous and thus perform most of testing and re-testing tasks with no or little human intervention. The proposed system named IAPTS can act as a module and integrate with most of the industrial PT frameworks to improve significantly the efficiency and accuracy of medium and large networks context. The proposed modeling of PT in the form of the RL problem allowed the coverage of the entire PT practice and thus producing a system fit for the real-world context. The current implementation of IAPTS is integrated into the most commonly used PT frameworks such as Metasploit and permitted highly efficient testing in terms of consumed time, allocated resources, covered tests and accuracy of the produced results. The main drawback of IAPTS is the need of high-caliber human expert supervision during early learning phases where a human trainer will perform PT along with IAPTS and adjust the learning and veto the output of the system to ensure a good quality training by acting as a rewarding provider for the RL agent actions.

The major contribution of this approach is to apply RL techniques to a real-world problem of automating and optimizing PT practice and resulted into a net improvements of PT framework performances notably in terms of consumed time and covered attack-vectors as well as enhancing the produced results reliability and persistence which will lead optimistically to a PT system free from human error. The second major contribution of the system is the ability to capture the expertise of human experts without instructing it as IAPTS will rely initially upon the expert feedback in form of rewarding values until it reaches a certain maturity. Thirdly, IAPTS will increase testing coverage by attempting tests that a human expert will not be able to explore because of the frequent lack of time. Finally, IAPTS permits the re-usability of the testing output by either learning and/or capturing the expertise during the test and storing it with the system memory for future use. It was proved to be very efficient in re-testing scenario (very common in PT) and nearly similar cases when the testing time and accuracy of the produced results were exceptional.

Finally, IAPTS performance on relatively large networks remains superior to the acceptable time usually allocated to PT expert, and therefore we plan to improve the current version by adopting a hierarchical POMDP model of PT practice in which the large networks are initially divided into segments (clusters) following a security-oriented approach and the overall POMDP environment will contain the representation of the clusters rather than all machines within the network. This approach is expected to solve two major issues faced during the IAPTS testing: the performance enhancement as the system will be solving several small POMDP problems rather than dealing with one large and complex environment. On the other hand, the hierarchical model will simplify and optimize the process of expertise capturing and handling as attack vectors at two levels clusters and machines for future use which will depend on the changes introduced in the assessed network.

Author Contributions: Data curation, M.C.G.; Formal analysis, M.C.G. and T.M.C.; Funding acquisition, T.M.C.; Investigation, M.C.G.; Methodology, M.C.G. and T.M.C.; Project administration, T.M.C.; Supervision, T.M.C.; Validation, M.C.G.; Writing—original draft, M.C.G.; Writing—review & editing, M.C.G. and T.M.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Creasey, J.; Glover, I. *A guide for Running an Effective Penetration Testing Program*; CREST Publication: Slough, UK, 2017. Available online: http://www.crest-approved.org (accessed on 18 December 2019).
2. Almubairik, N.; Wills, G. Automated penetration testing based on a threat model. In Proceedings of the 11th International Conference for Internet Technologies and Secured Transactions, ICITST, Barcelona, Spain, 5–7 December 2016 .
3. Applebaum, A.; Miller, D.; Strom, B.; Korban, C.; Wol, R. Intelligent, automated red team emulation. In Proceedings of the 32nd Annual Conference on Computer Security Applications (ACSAC '16), Los Angeles, CA, USA, 5–8 December 2016; pp. 363–373.
4. Obes, J.; Richarte, G.; Sarraute, C. Attack planning in the real world. *arXiv* **2013**, arXiv:1306.4044.
5. Spaan, M. *Partially Observable Markov Decision Processes, Reinforcement Learning: State of the Art*; Springer: Berlin/Heidelberg, Germany, 2012.
6. Hoffmann, J. Simulated penetration testing: From Dijkstra to aaTuring Test++. In Proceedings of the 25th International Conference on Automated Planning and Scheduling, Israel, 7–11 June 2015.
7. Sarraute, C. Automated attack planning. Available online: https://arxiv.org/abs/1307.7808 (accessed on 20 December 2019).
8. Qiu, X.; Jia, Q.; Wang, S.; Xia, C.; Shuang, L. Automatic generation algorithm of penetration graph in penetration testing. In Proceedings of the Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Guangdong, China, 8–10 November 2014.
9. Heinl, C. Artificial (intelligent) agents and active cyber defence: Policy implications. In Proceedings of the 6th International Conference On Cyber Conflict (CyCon 2014), Tallinn, Estonia, 3–6 June 2014.
10. Sarraute, C.; Buffet, O.; Hoffmann, J. POMDPs make better hackers: Accounting for uncertainty in penetration testing. Available online: https://arxiv.org/abs/1307.8182 (accessed on 20 December 2019).
11. Backes, M.; Hoffmann, J.; Kunnemann, R.; Speicher, P.; Steinmetz, M. Simulated Penetration Testing and Mitigation Analysis. *arXiv* **2017**, arXiv:1705.05088.
12. Ghanem, M.; Chen, T. Reinforcement Learning for Intelligent Penetration Testing. In Proceedings of the WS4 the World Conference on Smart Trends in Systems, Security and Sustainability, London, UK, 30–31 October 2018.

13. Durkota, K.; Lisy, V.; Bosansk, B.; Kiekintveld, C. Optimal network security hardening using attack graph games. In Proceedings of the 24th International Joint Conference on on Artificial Intelligence (IJCAI-2015), Buenos Aires, Argentina, 25–31 July 2015.
14. Veeramachaneni, K.; Arnaldo, I.; Cuesta-Infante, A.; Korrapati, V.; Bassias, C.; Li, K. *AI2: Training a Big Data Machine to Defend*; CSAIL, MIT Cambridge: Cambridge, MA, USA, 2016.
15. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
16. Jimenez, S.; De-la-rosa, T.; Fernandez, S.; Fernandez, F.; Borrajo, D. A review of machine learning for automated planning. *Knowl. Eng. Rev.* **2009**, *27*, 433–467.
17. Walraven, E.; Spaan, M. Point-Based Value Iteration for Finite-Horizon POMDPs. *J. Artif. Intell. Res.* **2019**, *65*, 307–341. [CrossRef]
18. Walraven, E.; Spaan, M. Planning under Uncertainty in Constrained and Partially Observable Environments. Ph.D. Thesis, Delft University of Technology, Delft, The Netherlands, 2019.
19. Spaan, M.; Vlassis, N. PERSEUS: Randomized point-based value iteration for POMDPs. *J. Artif. Intell. Res.* **2005**, *24*, 195–220. [CrossRef]
20. Andrew, Y.; Jordan, M. PEGASUS: A policy search method for large MDPs and POMDPs. In Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence, Stanford, CA, USA, 30 June–3 July 2000 .
21. Meuleau, N.; Kim, K.; Kaelbling, L.; Cassandra, A. Solving POMDPs by searching the space of finite policies. In Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence, Bellevue, WA, USA, 11–15 July 2013.
22. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized experience replay, Google DeepMind. *arXiv* **2015**, arXiv:1511.05952.
23. Dimitrakakis, C.; Ortner, R. Decision Making Under Uncertainty and Reinforcement Learning. Available online: http://www.cse.chalmers.se/~chrdimi/downloads/book.pdf (accessed on 20 December 2019).
24. Osband, I.; Russo, D.; van Roy, B. Efficient reinforcement learning via posterior sampling. Available online: https://papers.nips.cc/paper/5185-more-efficient-reinforcement-learning-via-posterior-sampling.pdf (accessed on 20 December 2019).
25. Grande, R.; Walsh, T.; How, J. Sample efficient reinforcement learning with gaussian processes. In Proceedings of the International Conference on Machine Learning, Beijing, China, 21–26 June 2014; pp. 1332–1340.
26. Agrawal, S.; Jia, R. Optimistic posterior sampling for reinforcement learning: Worst-case regret bounds. In Proceedings of the Annual Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 1184–1194.
27. Ngiam, J.; Khosla, A.; Kim, M.; Nam, J.; Lee, H.; ; Ng, A.Y. Multimodal deep learning. In Proceedings of the 28th International Conference on Machine Learning (ICML-11), Washington, DC, USA, 28 June–2 July 2011; pp. 689–696.
28. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv* **2018**, arXiv:1801.01290.
29. Zhang, Z.; Hsu, D.; Lee, W.; Lim, Z.; Bai, A. PLEASE: Palm Leaf Search for POMDPs with Large Observation Spaces. In Proceedings of the International Conference on Automated Planning and Scheduling, Israel, 7–11 June 2015; pp. 249–257.
30. NIST. Computer Security Resource Center—NATIONAL VULNERABILITY DATABASE (CVSS). 2019. Available online: https://nvd.nist.gov (accessed on 18 December 2019).
31. MITRE. The MITRE Corporation—Common Vulnerabilities and Exposures (CVE) Database. 2019. Available online: https://cve.mitre.org (accessed on 18 December 2019).
32. Lyu, D. Knowledge-Based Sequential Decision-Making Under Uncertainty. In Proceedings of the 15th International Conference on Logic Programming and Non-monotonic Reasoning, LPNMR, Philadelphia, PA, USA, 3–7 June 2019.