

Reinforcement Learning for Intelligent Penetration Testing

Mohamed C. GHANEM

Research Centre for Systems and Control
City, University of London
London, United Kingdom
mohamed.ghanem@city.ac.uk

Thomas M. CHEN

Research Centre for Systems and Control
City, University of London
London, United Kingdom
tom.chen.1@city.ac.uk

Abstract—Penetration testing (PT) is an active method for assessing and evaluating the security of digital assets by planning, generating and executing all possible attacks that can exploit existing vulnerabilities. Current PT practice is becoming repetitive, complex and resource consuming despite the use of automated tools. The goal of this paper is to design an intelligent PT approach using reinforcement learning (RL) that will allow regular and systematic testing, saving human resources. The system is modelled as a partially observed Markov decision process (POMDP), and tested using an external POMDP-solver with different algorithms. Although this paper is limited to only the planning phase and not the entire PT process, the results support the hypothesis that reinforcement learning can enhance PT beyond the capabilities of any human expert in terms of accurate and reliable outputs.

Index Terms—Penetration testing, reinforcement learning, network security, vulnerabilities assessment, cyber attack, autonomous testing, intelligent testing.

I. INTRODUCTION

Many enterprise networks are becoming more complex and open, increasing their exposure to threats that are growing in sophistication [1]. Penetration testing (PT) is an offensive approach aiming to evaluate the security of digital assets (network, website, application, database, database) by trying to actively identify its vulnerabilities and then exploit them in the same way as a genuine attacker. PT is a crucial and often mandatory component of cyber security auditing of organizations and businesses. PT is the best method to assess the security defenses against skilled adversaries as well as end user adherence to security policies [1].

It is possible to automate PT processes (such as reconnaissance, identification, and exploitation) but simple automation without optimization (mimicking human experts) will result in poor performance in terms of resources and outputs. Real-time exploring and decision making are critical, which is possible only with human experts at this time. Intelligent PT that goes beyond simple automation aims to both alleviate human resources and achieve results more quickly.

The main challenge for automated systems is optimization, i.e., ensure that all existing threats are checked systematically and efficiently. The system should not take excessive time by processing irrelevant tasks and at the same time ensure that no threat is overlooked. Our research aims to develop an

intelligent autonomous PT system which relies on machine learning techniques at different levels of the practice to improve efficiency and accuracy [2], [3].

A. Research Context

PT is the best way to prove that a specific network, system or software application is not secure by identifying and exploiting its vulnerabilities. There are dozens of commercial and amateur systems, platforms and frameworks. Automating some of the tasks improves performance but currently automation remains either local (specific to very limited context or task) or not optimized (blind automation) leaving some significant issues to practitioners. Existing systems and frameworks fail to autonomously perform a comprehensive test and assessment of large assets in the same way as human experts.

Other major issues arise with the use of such automated systems, notably: the high volume of data generated by conducting comprehensive testing; the associated waste and unexploited information; high threat emergence rate; and fast changing and complex attack paths.

The use of machine learning has intensified in the cyber security domain such as for intrusion detection. Recently MIT researchers developed a big data security system named AI2, which combined security analyst expertise with machine learning to build an intrusion detection system with active learning [2].

It is natural to imagine one or more machine learning techniques applied to different PT phases to learn and replicate testing, constantly improving efficiency and accuracy [3]. The ultimate goal is a system capable of imitating human PT experts in performing an intelligent and automated pen test. The system could and should be allowed to interact directly with the expert to deal with complex situations by offering indications and suggestions which can be accepted or rejected by the expert.


In practical terms, incorporating machine learning in an automated PT system will reduce recurrent human errors due to tiredness, omission, and pressure. It will also reduce time and resources by performing the different tests efficiently. Automation may relieve network congestion and downtime by working after normal working hours [6].

B. Motivation and Contribution

A natural question might arise amongst the cybersecurity research community that existing systems, frameworks and tools utilised in the PT practice should be able to automatically (with or without expert intervention) conduct some or all the tasks such as information gathering and vulnerabilities discovering or allow the use of external systems providing such capabilities, then the expert will be left with the duty of selecting, planning and executing of the appropriate exploits or attacks [5].

PT practice is a complicate process which human barely master, so how can a machine replace this expert in conducting those tests. The answer to this tricky question comes from the nature of the PT practice, which is often presented as multi-phases practice but in reality, phases separation is impossible and dependency between the tasks is crucial. Then, as the first two phases produce often incomplete results and fail to yield a perfect knowledge about the assessed assets, so a residual uncertainty remains which requires repeating some tasks or even changing the approach making the blind automated machine option non-feasible.

The first two phase of PT are the most opportune places where blind automation could be most helpful to reduce the generated amounts of data (big data problem) and costs in terms of running time and network traffic. We propose machine learning, specifically reinforcement learning, to automate PT to behave like a real attacker gathering information, assessing and exploiting in an intelligent manner. This research aims to improve current PT practice which raises several issues that an AI led system will address notably:



The associated High-Cost of human-led systematic and periodic testing (re-testing) along with the impact on the assets performances and downtime,
The huge repeatability in the performed tasks which often account for the large part of the practice and and thus human expert time,
The catching-up with cyber threats high emergence and fast changing rate (Short Lived Patterns) along with the highly targeted and adaptive nature,
The evasive composite attacks in which hackers adopt complexes and non-obvious attacking routes, techniques and following some unlikely paths which are hard to be imitated and test by experts,

PT results (output) should be stored and reused for future tasks. Existing systems tend to dismiss the re-usability of the data which is sometimes crucial especially when the testing is repetitive. The majority of network parameters and configuration will not change in a small interval, so the output of previous testing will remain relevant and useful for further testing. PT should be repeated and performed on a regular basis to ensure continuous security. An intelligent and autonomous PT system can be a great aid for repeated tests, which is the main motivation of our research.

C. Paper Outline

Section 2 is a review of relevant literature highlighting attempts to automate PT. Section 3 introduces our approach using reinforcement learning leading to a partially observed Markov decision process (POMDP) model. Section 4 describes our proposed system called Intelligent Automated Penetration Testing System (IAPTS). Section 5 describes IAPTS in more detail. It is evaluated through simulations in Section 6.

II. LITERATURE REVIEW

Initially, researchers were interested in the planning phase. Some works were implemented in professional PT tools and frameworks while others remained research ideas [4], [7], [9].

A. Previous works on automating PT

Simple and blind automation is an obvious approach, automating all or some of the tasks and subtasks for each phase of PT practice. Many organizations use automated tools and systems which blindly perform tests without any exception or pre-elimination [3]. These systems should be imperatively controlled by a skilled human tester. However, these systems fail to produce acceptable results in medium and large network because a huge number of operations is required to cover the whole asset. The consumed time, resources, and generated traffic (network congestion) become too high. Thus, the use of this approach is limited to a small network or only effective with the use of customized scripts which are inconvenient requiring substantial effort [8], [10].

Early research focused on modelling PT planning as attack graphs and decision trees which reflected the view of PT practice as sequential decision making. Practically, most of the works were more relevant to vulnerabilities assessment than to PT. Amongst the most significant contributions, attack graphs were used to represent atomic components (attack actions) by a conjunctive pre-condition and post-condition over relevant properties of the system under attack [11]. This approach was more related to classical planning and attempted to find the optimized attack graph.

Further works were carried out on automating planning of PT tasks but most lacked efficiency. Blind automation did not address the problem of optimising PT practice as a whole but only focused on elaborating and executing the attack resulting from static planning [3], [12], [16].

A pioneering work introduced a complete PDDL representation of the attacking problem along with a practical implementation that integrates a planner into a PT tool [4]. The work automatically generated attack plans (single and multi-paths) for real world PT scenarios which could be directly used by exploitation tools to execute the corresponding exploits. The main idea was an algorithm for transforming the information acquired during the information gathering phase into planning domain and solve separately the problem before getting back to the actual execution. However, the work showed scalability issues, being limited to small and medium size networks. It also failed to cope with dynamic environments that are typical of PT practice.

Artificial intelligence was considered as a way to improve PT practice [7], [9]. The classic modelling approach fails to deal with the uncertainty in PT practice, especially the lack of accurate and complete knowledge about the assessed systems. ML algorithms were integrated into an industrial tool, and the PT process was modelled as a partially observable Markov decision process (POMDP). Nonetheless, the proposed model itself is questioned as it obviously fails to model the full picture of PT and focusing rather on the separate entities.

B. Limits of previous approaches

After a comprehensive survey of the previous literature and the popular solutions available for professional PT (Core-Impact, Nexpose, Nessus, Qualys, Tenable, Immunity Canvas and Metasploit), we were able to understand that existing tools and frameworks are essentially vulnerability scanners rather than PT solutions. They do not cover the whole process from information gathering to exploitation, and rarely offer the promised simplicity and flexibility of use (automation of certain tasks, visualization, reporting). Automation is limited to the planning of the practice, organization of tasks, optimization/visualization, and reporting.

We believe the heart of PT practice (the third phase) remains neglected or unexploited. The most challenging part is finding the exploitable vulnerabilities and launching the relevant exploits, and when needed, pivoting to create a new vector of attack. The difficulty is partly due to current PT systems which have radically changed into more complex systems encompassing new attack vectors and increasing numbers of exploits and information gathering modules. Thus, the problem of efficiency has emerged [6].

Limits of the proposed solutions can be summarised as:

- Computer-generated plans and attack graphs rely on static data and isolate tests from the dynamical nature of the context;
- Suitability of the proposed models and representation in dealing with versatile network topologies, configurations and security architectures, depend on a number of assumptions;
- The need for a human expert supervising or controlling the system and making the crucial decisions;
- The eventual compatibility to be incorporated or embedded within industrial PT solutions;
- Blindly automated PT systems solve a problem but create an even bigger problem of efficiency as the required resources and time increase sharply;
- The increasing exposure of security detection, and the saturation of the security mechanisms which expose assets to further risk.

III. REINFORCEMENT LEARNING FOR INTELLIGENT PT

An obvious way to produce a fully automated system that can replace human expert into performing PT is by automating all or some of the tasks and subtask for each phase of the practice [5]. The idea itself is not novel as many large organizations rely on self-developed automated systems which

were created specifically to meet their needs. They often aim to blindly perform all the possible tests with no exception or pre-elimination. These kind of systems are typically controlled and guided by a skilled human tester who will deal with more complex scenario and make the crucial decisions [12].

The disadvantage of such approach is mostly noticed in medium and large networks where the time, resources, generated traffic and network congestion is too high. Thus, the use of such approach is limited to small networks.

Our work was initiated to overcome the following issues in current PT practice:

- The associated high cost of human-led systematic and periodic testing along with the impact on performance and downtime;
- The repeatability of tasks which account for a large part of PT practice;
- Emerging cyber threats that are fast changing, targeted and adaptive to new security measures;
- Evasive and composite attacks where adversaries adopt complex and non-obvious attack routes and techniques that are hard for penetration testers to imitate;
- Obfuscation in large and complex networks (e.g., in the finances sector) making it almost impossible to cover the entire network properly.

A. Reinforcement Learning

Cyber-security solutions fall mostly into two categories: analyst-driven or unsupervised machine learning. Analyst-driven solutions such IDS rely on rules determined by security experts and usually lead to high rates of uncovered and untested attack vectors [5]. On the other hand, reinforcement learning (RL) is concerned with goal-directed learning and decision making which accurately reflect the PT context [12].

The following points summarise the choices of RL:

- RL allows the agent to learn from previous experiences by interacting with the environment.
- RL rewards are often delayed in time, and the agent tries to maximize a long-term goal.
- RL agent actively interacts with the environment via states, actions, and rewards.

As shown in Fig. 1, RL allows a machine (the software agent) to learn from its own behaviour based on feedback received from performing actions within its environment. This behaviour can be learned once and for all, or be improved or adapted as time goes by. If the problem is modelled with care, some RL algorithms can converge to the global optimum which is the ideal behaviour that maximises the reward.

This automated learning scheme implies that there is little need for a human expert who knows about the domain of application. Much less time will be spent designing a solution since there is no need for handcrafting complex sets of rules as with expert systems. RL algorithms have become popular in recent years because of their ability to solve complex tasks with minimal feedback. Both genetic algorithms (GA) and temporal difference (TD) methods have proven effective

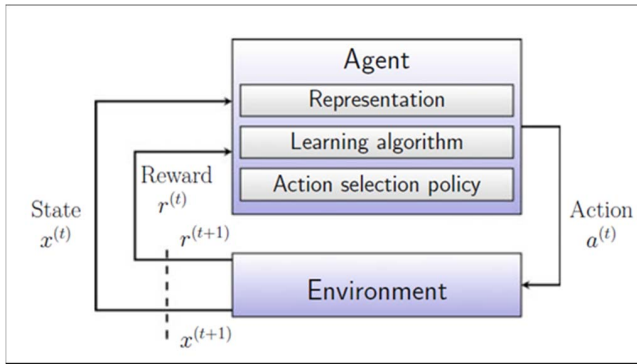


Fig. 1. RL agent observes the state of the environment $x(t)$ at time t , selects an action $a(t)$ based on its action selection policy, and transitions to state $x(t+1)$ at time $t+1$ and receives a reward $r(t+1)$. Over time, the agent learns to pursue actions that lead to the greatest cumulative reward [5].

at solving difficult RL problems especially in cyber security contexts [7].

B. POMDP solving

After modelling and representing the real-world PT problems as POMDP, which was selected as the best feasible way to represent the context, the next step is to solve the modelled problem. In this case the problem is a complex decision making sequential process. It allows the system to achieve the best possible solution which will be adopted as the decision policy for future similar cases. The ensemble of policies is called a policies graph (PG) that is stored in IAPTS memory for future use. It can be either updated or enhanced when a better solution is achieved or when a human tester decides to teach the system manually [13].

As PT is more about global target than local, each action made by the system will be evaluated by the expert on its local effect and also on the global opportunities it make available later. The effects need not be deterministic, and in which there are long-term goals [14].

The RL approach here is meant to address the kind of learning and decision-making problems that allow the PT system to be autonomous and intelligent in performing the whole PT tasks and sub-tasks. In this, we are not concerned with the improvement of the RL solving algorithm, but only with electing the appropriate algorithm that best fits our context and produces acceptable results.

The PT problem previously represented as POMDP will be utilised as input to an off-the-shield solver in which a decision-making agent will be interacting with its environment to maximize the cumulative reward it receives over time. The agent perceives aspects of the environments state and selects actions. The agent will estimate the value function and use it to construct better and better decision-making policies over time [11].

To solve the PT POMDP complex environment, the IAPTS should incorporate a set of solving algorithms rather than just one. In fact, different situations and scenarios have different particularities and require a different solving approach. Fur-

thermore, the limitation in available resources (time, memory and computational) makes solving some problems with large state and action spaces challenging. A softer approach is required where some level of accuracy is often sacrificed.

It is important to highlight the fact that large environments can also cause challenges to solving, especially when it comes to enumerating explicit and implicit transition and observation probabilities and relying on static rewarding schemes [9].

There are many different approaches to solving an RL problem. The two most popular are the reward-oriented approach and the policy search. The **reward-oriented approach** allows an RL agent evolving within the environment to select the sequences of actions that leads to maximizing the overall reward in the long term. This approach seeks to elaborate an efficient and comprehensive reward function which utilize and compute the atomic rewards value associated with the transitions and observation to determine and an optimal reward function.

The **policy search approach** seeks to address an optimal decision policy which is practically done by learning to estimate the value of a particular state. This estimate is adjusted over time by propagating part of the next state's reward. If all the states and all the actions are tried a sufficient amount of times, this will allow an optimal policy to be defined; the action which maximizes the value of the next state is chosen [9].

As a first step, we decided to test the policy search approach defined briefly as follows:

On policy: the same policy is used to act and evaluate the quality of actions.

Off policy: a different policy is used to act and evaluate the quality of actions.

Initially, we used and off-the-shelf POMDP solver allowing the choice of different solving approaches and state-of-the-art algorithm. The following two solving algorithms were initially adopted:

PERSEUS algorithm: randomized point-based value iteration for POMDPs proposed by [11]. The algorithm performs approximate value backup stages, ensuring that in each backup stage the value of each point in the belief set is improved. The strength of this algorithm is its capacity to search through the space of stochastic finite-state by performing policy-iteration and the single backup can improve the value of many belief points. Previous experimental results showed that PERSEUS is perfect for large scale POMDP problems as it operates on a large belief set sampled by simulating random trajectories through belief space leading to a significant speed-up in the solving process [12].

PEGASUS algorithm: a policy search algorithm dedicated to solving large MDPs and POMDPs initially proposed by [13]. This algorithm adopts a different approach to the problem of searching a space of policies given a predefined model. Any MDP or POMDP is first transformed into an equivalent POMDP in which all

state transitions (given the current state and action) are deterministic. 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 demonstrated huge potential as it produces a polynomial rather than exponential dependence on the horizon time making it an ideal candidate for solving the PT problem [3].

In addition to the two algorithms above, other RL algorithms were considered such as backwards induction and finite grid. Some of the proposed algorithms are already part of the POMDP-Solver software. Some algorithms were implemented and integrated for the sake of the research benchmarking [17].

IV. IAPTS DESIGN AND FUNCTIONS

Our proposed system is named **Intelligent Automated Pen-etration Testing System (IAPTS)**. Figure 2 illustrates the functional design. Initially, the first phases (clustering and data processing) will be performed manually to allow producing the model, implementing, and testing it. After testing the proposed model resilience and validating it, the remaining modules will be added to complete the system.

The system memory will be initially handled manually and store the results (policies extracted after applying the generalization) along with the management of tasks related to expertise extracting and storing from the human expert decision. In other words, the learning will be initially done manually and the modules in charge of capturing, assessing and storing the expertise will be implemented and embedded within the IAPTS in due course.

The projected system can be either a module or an independent system. It can be implemented by modifying the core component of an existing automated PT framework (namely Metasploit) which such modification. Otherwise, a module (software system) will be developed separately and then integrated/adapted to work with the existing industry automated PT systems and frameworks.

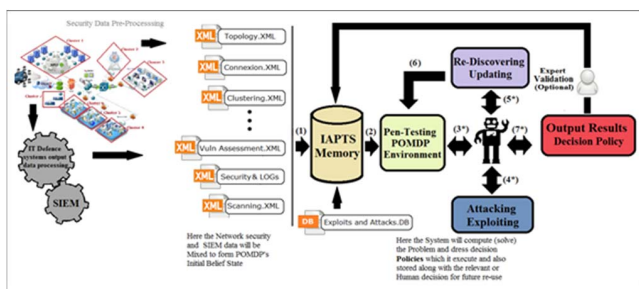


Fig. 2. Proposed IAPTS design and functional diagram.

A. IAPTS operative modes

Given the aforementioned challenges and aims, the PT system should be able to intelligently perform network PT by operating alone, under supervision or along with a human expert. The system can have four different operating modes

depending on the context and level of maturity (learning) as follow:

Fully autonomous (level 4): the system is fully autonomous in performing all PT tasks after reaching the pre-set level of expertise learning enabling it to behave at least in the same way as a human expert and achieve at worst an acceptable result.

Partially autonomous (level 3): the likely mode that the system will adopt during the first time of “commercial use”. In this case the system will be under constant and continuous supervision (locally or remotely) of a high-calibre human expert.

Decision-making assisting mode (level 2): the system will be operating along with the expert handling the PT and assist him/her in the decision making by computing and processing the data on a real-time basis and suggesting better alternatives.

Learning mode (level 1): the system will be running in the background when the human tester is performing PT and will only extract data, perform the required processing and use this data for training. In this situation, the system will have the opportunity to learn from the decisions made by the expert and extract the relevant knowledge from new situations (or even repeated ones) to build its own memory.

V. PT MODELLING AND REPRESENTATION AS REINFORCEMENT LEARNING PROBLEM

In this section, we present an initial “basic” version of the proposed system IAPTS which the core module relies on modelling the PT environment as a POMDP problem, the system is introduced by using an illustrative example to allow smooth introduction of the different concepts. In the context of PT, there is no need to fully represent all network information in the RL environment but only represent specific information relevant from the PT point of view. Thus, an expanded representation of the whole network topology and security architecture is not required and will only encumber the RL environment [16]. Nonetheless, the environment will focus on connectivity, security configuration and status of each machine on the network which will be fully represented in order to capture the whole picture and mirror the true full PT practice despite the fact that additional information will heavily impact the problem solving performances. The proposed model was therefore able to cover all the aspects including:

Understanding the PT human expert behaviour by analysing the different methods, approaches, tools and techniques that ethical hackers adopt to perform different types of security testing;
Investigating how prior knowledge is acquired to determine the best approach to be integrated within IAPTS;
Identifying the relevant data processing and the learning algorithms to acquire such data (proxy server logs, web server logs, database logs, routing device logs, apps and other security device logs);

Modelling the network PT environment as an RL problem and defining each component, then solving the RL problem using an off-the-shield solver (POMDP) by adopting “policy search” approach.

A. From Network PT data to RL problem

We briefly describe here the process of elaborating the RL environment out of the PT information previously extracted and processed by a dedicated IAPTS module as illustrated in Fig. 3. The following is the logic adopted to move from computer network data to build the file constituting the PT RL environment. The network is seen as a set of digital systems called machines which can be anything that runs an operating system, e.g., computers, servers, networking devices and security systems. All the mentioned components could be either physical (hardware), logical (software) or a combination of both.

The following are the different components of the RL POMDP environment that will illustrate the problem:

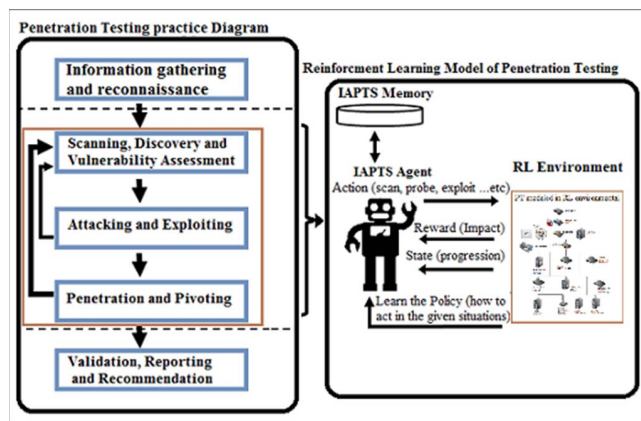


Fig. 3. IAPTS functional diagram showing the transition from network to POMDP model.

State space: incorporating, for each system, relevant information from a security point of view. It will include but not be limited to operating system, port, services and applications as well as generic network connectivity and defence information. The information will be represented in a special notation designed to be concise, clear (easy to understand) and precise. As in modern PT, the majority of the information will be known or guessed from the initial data collection phase. The connectivity information will be pre-processed against the security information to only represent open connection paths and thus avoid useless information [7].

In the state space, we represent machine number “i” by the annotation “Mi” and thus each machine (for example number i) will be represented under different state such as “Mi-OS1-Port80-ServiceABC” or a part of it “Mi-OS1”. These information will be continuously updated as the discovering and scanning tasks progress to confirm previous probabilistic information or to add a new one.

In addition, the state space will include information about the connectivity and networking isolation (sub-net, virtual

LAN). As an example, the connection M0 to M1 using a TCP connection and SSH service directly is represented as “M0-M1-TCP-SSH-0”. Only the security and networking mechanism and system is considered as the separation between the machine. The machines belonging to the same cluster should have a direct connection which is reflected in the model by the number “0” meaning zero hops. The clustering approach basing on separation will be detailed in later works.

Action space: to capture processes and represent PT tasks and sub-tasks in a logic that reflects the PT. As with any PT practice, the number of available tests (attack, scan, and exploit) that the expert can perform is huge and cannot be totally represented within the RL action space.

Nevertheless, the atomic components of any exploitation or scanning action are limited. Thus we attempted to represent the complex task by dividing them into simple action. As the time allowed for each test is limited, information about the cost of each action in term of execution time, success probability and associated cost and possible gain will be carefully represented.

Finally, some special actions will be introduced to allow IAPTS to achieve a realistic and acceptable result such as Terminate, Repeat and Give-Up actions which will prevent the system from running infinitely. Termination occurs either manually or automatically and can be triggered by the tester or the system in one of the following situations:

- when the sought objective is achieved which can be either local or global;
- failed or detected attacks and no other alternatives are possible;
- stopped/dropped test locally or globally as no gain or further penetration is possible;
- exceeding the allowed time limit.

Some successful or failed attack outcomes will still trigger further action and not be considered as reaching the terminal state. For example, the following scenarios:

- an attack which failed to gain full control over a machine could be followed by other attacks (user session) or attempt to escalate privileges or exploit other attack paths; when a launched attack based on a significant amount of uncertain information and thus assumption fails, this situation should not constitute a terminal state as further information can be collected and other attacks can be

- successful;
- when some event occurs such as attack detection by IDPS based on behaviour, or when the additional access rights that could yet be gained (from the finite number of access rights) do not outweigh the associated costs.

Observation and transition probabilities: the probabilistic nature of the PT tasks and sub-tasks make it complicated to define the transitions and observations which are based on probabilistic actions and states. Therefore, we utilised a simple equation to calculate the different probabilities based on previously acquired information and other data defined by the human expert (based on his/her expertise). Thus we were able

to assign the transition and observation probabilities which reflect the real world conditions.

We proposed a metric to be used in assigning transition and observation values based on the potential success of each action undertaken on the specific state while taking into account the pre-and post- state of the given action. Initially, we make use of the National Vulnerability Database (NVD), a comprehensive cyber security vulnerability database sponsored by the Department of Homeland Security (DHS) and the National Institute of Standards and Technology (NIST) [18].

Reward scheme: After reinforcement learning was selected as the approach for IAPTS, the research aim and context was chosen to be the policy search. No reward function will be initially utilised and rewarding the performed actions will be predefined by a human expert who will have to decide on the adequate reward for each action performed. The reward values are then set for different scenarios such as: reaching a terminal state; achieving a final (global) target or local goal (controlling an 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.

VI. TESTS AND EVALUATION

A. Simulation platform

The first test aimed to assess the effectiveness of the proposed PT POMDP model and make adequate choices in terms of approaches, algorithms, and scaling. The initially tested representation is simple and features are often very general. The aim is simply a better understanding of the behaviour of the RL agent on the defined environment and identifying potential weaknesses which should be addressed straight away to be tested again. Later, the IAPTS system will be gradually upgraded, the core component fully developed and tested along with the remaining modules.

In the experiment, the assessed network N was composed of seven machines (M0 . . . M6) excluding the Attacker which is connected to the Internet and viewed as an independent cluster in the modelling as shown in Fig. 4.

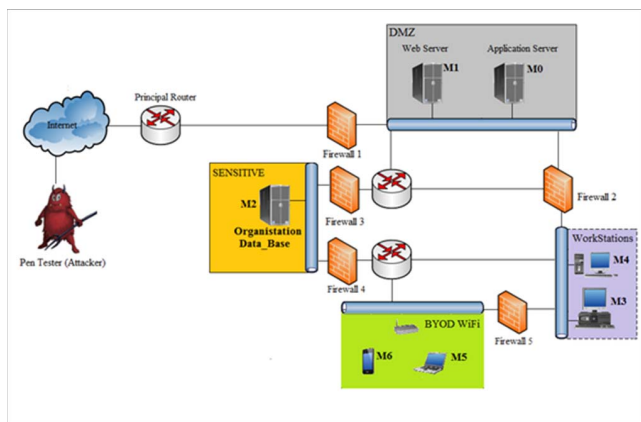


Fig. 4. Simulation network.

B. Initial results

Based on the obtained results (solving the POMDP problem), we then performed additional calculations and estimations to convert the results into a more understandable format. Execution time for solving the POMDP along with times required to perform the other PT tasks and other variables were used to calculate an approximate PT time which could be consumed by the OAPTS if it was fully developed and functional and have a full control over the PT system.

The results shown in Fig. 5 were compared with a manually estimated consumed time based on the researcher's experience in the field and also a possible blind automation of all tasks. The result from IAPTS shows clearly the performance enhancements translated into a significant reduction of the running time consumed to finish the testing.

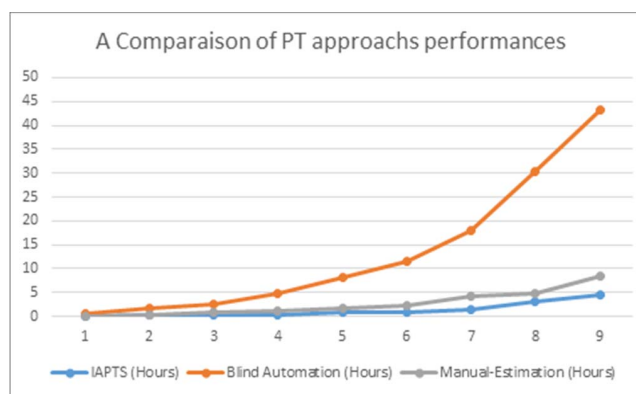


Fig. 5. Comparison of different PT approach performance in term of consumed time (hours) by the size of the network (number of machines).

C. Experience replay

This initial testing of IAPTS was not meant to assess the performance of the system but only demonstrate that RL works well for PT automation and then assess the acquired learning in terms of learned policies. Next, more advanced tests were carried out to illustrate the importance of the learning on the long-term PT practice by adopting a test scenario inspired from the real-world situation of re-testing the same network after some updates and upgrades.

The results presented in Fig. 6 confirmed that previously acquired learning stored as acting policies within the system memory were reapplied in most of the cases. Thus, a considerable amount of time was saved, exceeding our initial expectations especially in the cases of minor changes in which the performance enhancement of IAPTS were considerable.

D. Results discussion and future works

The obtained results clearly prove that RL can enhance the performances of automated PT systems and frameworks which traditionally are either human controlled or blindly automated. In addition to the performances enhancement, we evaluated, relying on a personal professional experience as Penetration Tester, the pertinence of the produces result (policies) and quality in term of relevance, coverage and accuracy. under

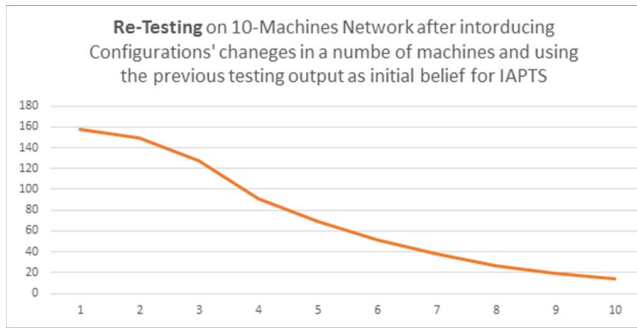


Fig. 6. IAPTS experience replay performance enhancement

a specific context, IAPTS produce a very relevant attacking policies when targeting the Machine M2 suspected to contain sensitive information and defined as the most secured machine within the network. Indeed, the produced policy is from an attacker point of view the most likely to adopt. Despite the fact that the attack path itself is not minimal, the success rate is much higher than another alternative path which support the initial assumption that an RL led PT system can mirror human-expert. Furthermore, the IAPTS produced other suggestion which many experts could neglect in spite of being very relevant and constitute a possible attack path which a real hacker can chance it. Finally, despite the fact that pivoting was not the main concerns during this early phases of IAPTS development, the proposed IAPTS architecture was meant to make the future integration of such feature very smooth by introducing the concept of security clustering (isolation) which will be detailed in future works along with IAPTS pre-processing, and memory management modules that will gradually integrate IAPTS.

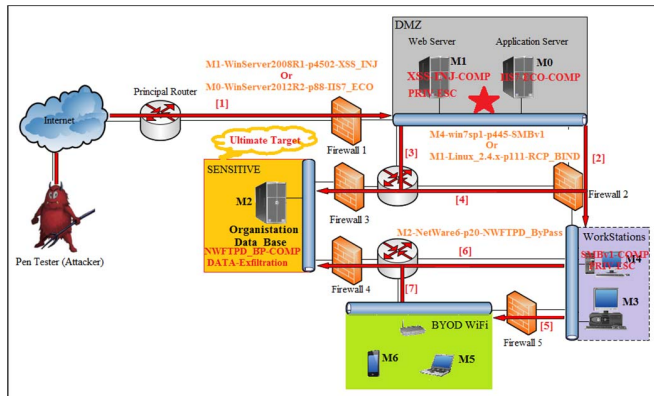


Fig. 7. example of an IAPTS output attack policies

VII. CONCLUSIONS

This paper explores a novel approach which allows to intelligent and autonomous penetration testing and re-testing on medium and large networks. The proposed reinforcement learning approach will allow, if properly implemented and embedded, automated PT systems and frameworks notably Metasploit to perform testing without the need for constant

human expert intervention. As with any machine learning, the system will require a significant amount of training which will be mainly focused on rewarding the RL agent action. This will require a high calibre PT human expert.

The first major contribution of this approach will be improved performance (in terms of consumed time and resources) and enhanced quality of results and persistence which will lead optimistically to a PT system free from human error.

The second major contribution of the system will be capturing the expertise of the human expert without instructing him/her. The system will rely on the expert feedback to adapt its actions and thus achieving better results in the future.

Finally, it is important to highlight that the presented work constitutes an initial output of a comprehensive research work currently in progress aiming to produce a prototype PT system which will perform all the different tasks constituting the normal PT that the human expert is doing manually or in a blindly automated manner.

REFERENCES

- [1] CREST, A guide for running an effective Penetration Testing program, <http://www.crest-approved.org>. CREST Publication, 2017.
- [2] N. Alzubairik, G. Wills, Automated penetration testing based on a threat model. 11th International Conference for Internet Technologies and Secured Transactions, ICITST, 2016.
- [3] A. Applebaum, D. Miller, B. Strom, C. Korban, and R. Wol, Intelligent, automated red team emulation. 32nd Annual Conference on Computer Security Applications (ACSAC '16), 2016, pp. 363-373.
- [4] J. Obes, G. Richarte, and C. Sarraute, Attack planning in the real world. Journal CoRR Article, 2013, abs/1306.4044.
- [5] M. Spaan, Partially Observable Markov Decision Processes, Reinforce-ment Learning: State of the Art, Springer Verlag, 2012.
- [6] J. Hoffmann, Simulated penetration testing: From "Dijkstra" to aaTur-ing Test++. 25th Int. Conf. on Automated Planning and Scheduling (ICAPS15). AAAI Press, 2015.
- [7] C. Sarraute, Automated attack planning. Instituto Tecnológico de Buenos-aires, Ph.D. Thesis, 2012, Argentina.
- [8] X. Qiu, Q. Jia, S. Wang, C.Xia and L. Shuang, Automatic generation algorithm of penetration graph in penetration testing, 19th Int. Conf. on P2P, Parallel, Grid, Cloud and Internet Computing, 2014.
- [9] C. Sarraute, O. Buffet and J. Hoffmann, POMDPs make better hackers: Accounting for uncertainty in penetration testing. 26th AAAI Conf. on Artificial Intel. (AAAI12), pp. 18161824, July 2012.
- [10] C. Heini, Artificial (intelligent) agents and active cyber defence: policy implications. 6th Int. Confe. on Cyber Conflict. NATO CCD COE Publications, 2016, Tallinn.
- [11] M. Backes, J. Hoffmann, R. Kunnemann, P. Speicher and M. Steinmetz, Simulated penetration testing and mitigation analysis. <http://arxiv.org/abs/1705.05088>, 2017.
- [12] S. Jimenez, T. De-la-rosa, S. Fernandez, F. Fernandez and D. Borrajo, A review of machine learning for automated planning. The Knowledge Engineering Review, Vol. 00:0, pp. 124. 2009.
- [13] Y. Andrew and M. Jordan, PEGASUS: A policy search method for large MDPs and POMDPs. 16th Conf. on Uncertainty in Artificial Intel., 2013.
- [14] T. Schaul, J. Quan, I. Antonoglou and D. Silver, Prioritized experience replay, Google DeepMind. ICLR 2016.
- [15] K. Veeramachaneni, I. Arnaldo, A. Cuesta-Infante, V. Korrapati, C. Bassias and K. Li, AI2: Training a big data machine to defend. CSAI, MIT Cambridge, 2016.
- [16] K. Durkota, V. Lisy, B. Bosansk and C. Kiekintveld, Optimal network security hardening using attack graph games. 24th Int. Joint Conf. on Artificial Intelligence (IJCAI-2015), 2015.
- [17] N. Meuleau, K. Kim, L. Kaelbling and A. Cassandra, Solving POMDPs by searching the space of finite policies. 15th Conf. on Uncertainty in Artificial Intel., 2013.
- [18] NIST, Computer Security Resource Center - National Vulnerability Database, <https://nvd.nist.gov>, 2017.