Towards an efficient automation of network penetration testing using model-based reinforcement learning

Intelligent Automated Penetration Testing Framework

Mohamed Chahine Ghanem

Supervisor: Professor Thomas Chen

School of Science & Technology
City, University of London

This Thesis is Submitted for the Degree of
Doctor of Philosophy (Ph.D)

August 2022

Dedication

I dedicate my thesis work to my loyal family and cooperative friends. A special feeling of gratitude to S-Y without her things would be much different, my Children (Lindsey, Elian and Ayden) without them this research would be finished three years ago, to my father and my mother who still support my back and flourish my heart with her continuous precious prayers to achieve my dreams. Special thanks to my cheerful trustworthy brothers and sisters who have continuously encouraged me and lifted my spirit with their deep heartfelt laughter who have never left my side. I also dedicate this work to my companions at City, University of London who have been advising me during my tough days and sharing me their beneficial experiences throughout the journey. I will always appreciate all what they have done, words of encouragements and special thanks to my friends at City, Kroll, and MD. My colleagues at London Metropolitan University especially Prof. Karim Ouazzane and Dr Preeti Patel who stayed by my side, glaring at me with a pushing look and unforgettable inspiring words.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This thesis is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and acknowledgements.

Mohamed Chahine GHANEM
31 August 2022

Acknowledgements

I would like to express my deep sincere gratitude to my advisor whom I can proudly name as my teacher, Prof. Thomas Chen for his support, patience, trust, and encouragement in addition to his immense knowledge that has always been advantageous to complete this piece of work. His tremendous inspiration and valuable advice led me to a deeper understanding of the topic and lightened my journey to grow as a researcher at the field of offensive cyber security, artificial intelligence, and machine learning.

Abstract

Penetration Testing (PT) is an offensive method for assessing and evaluating the security of digital asset by planning, generating, and executing all or some of the possible attacks that aim to exploit its vulnerabilities. In large networks, penetration testing become repetitive, complex and resources consuming despite the use of autonomous tools. To maintain the consistency and efficiency of PT in medium and large network context. it is imperative to go through making it intelligent and optimized which will allow regular and systematic testing without having to provide a prohibitive amount of human labor in one hand and reducing the precious consumed time and tested system downtime in another hand. Reinforcement Learning (RL) led testing will unburden human experts from the heavy repetitive tasks and unveil special and complex situations such as unusual vulnerabilities or combined non-obvious combinations which are often ignored in manual testing. In this research, we are concerned with the specific context of improving current automated testing systems and making them intelligent, targeted, and efficient by embedding reinforcement learning techniques where it is relevant. The proposed Intelligent Automated Penetration Testing Framework (IAPTF) utilizes RL because of its relevance to sequential decision-making problems, it relies on a model based RL where planning and learning are combined and decomposed tasks to represent it as POMDP domain accounting for major PT features, tasks and information flowchart to realistically reflect the real-world context. The problem is then solved on an external POMDP-solver using different algorithms to identify most efficient options. As we encountered a huge scaling-up challenges in solving large POMDP which reflect the regular representation of PT on large networks, we propose thus a Hierarchical representation on which we divided large networks into security clusters and enabling IAPTF to deal with each cluster separately as small networks (intra-clusters), later we proceed to the testing of the network of clusters heads to ensure covering all possible complex and multistep attacking vectors largely adopted by nowadays hackers. The obtained results are unanimous and defeat both previous results and any human performances in term of consumed time, number tested vectors and accuracy especially in large networks. The learning is the second strength of our new model, as the generalization of the extracted knowledge become easier and allowing therefore the re-usability notably in the case of retesting the same network with few changes which is often the real-world context in PT. The performance enhancement and the knowledge extracted, and reuse confirm the efficiency, accuracy, and suitability of our proposed framework. Finally, IAPTF is designed to offload and ultimately replace human expert and to be independent, comprehensive, and versatile so it can integrate any automated PT platform or toolkit. Initially, the framework connects directly with Metasploit and Nessus APIs as both free versions coding architecture allows to perform such utilization.

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

Table of Content

List of Tables

List of Figures

Chapter 1: Introduction

In the cyber era we are living in, our lives are becoming more and more accustomed to the presence of IT equipment, devices, and systems. This emerging technology is associated with objects that, through the connection to the internet and data transmission, make everyone's life more comfortable. Nonetheless, this comfort comes at a cost, as IT networks are increasingly larger, complex, and inter-connected to ensure a wide range of tasks for the benefit of users and organizations [1]. In parallel to this evolution in networking, cyber threats are becoming more frequent, complex, and sophisticated creating more opportunities for cyber criminals to launch malicious attacks in the hopes of gaining access to sensitive data for their own gain [2-3]. The flexibility comes at a huge cost as cyber-security practitioners, experts and researchers noticed that cyber threats are becoming more frequent, complex, and sophisticated following the general rule of attack surface evolution. In engineering fields, the complexity is the worst enemy of security, and networks are not an exception to these unanimous rules. Protecting complex networks and critical assets from cyber threats pushed the network security professional into the trap of bolting on more and more security layers and policies [4]. The result was overly complex adding the multi-levels of security which is often vulnerable when faced with a high-caliber attacker because of what it contained in term of vulnerabilities due to human errors, misconfigurations, systems weaknesses. Thus, ensuring that the applied security measures are effective is the cyber-security communities' major concern, several approaches have been proposed and adopted over time. Nevertheless, using the offensive approach demonstrated that is, the best and most reliable method and the most favorably adopted by the security experts [5].

Penetration Testing, also shortly known as Pentesting or PT, is an active method for assessing and evaluating a digital assets security (network, web, server providing some service.) by trying to identify vulnerabilities and attempting to exploit them [2]. PT constitutes a central, often mandatory component of the cyber-security audit and embeds all standard auditing and testing tasks starting from information gathering and analysis, planning, generating, and executing the appropriate attacks targeting the identified vulnerabilities. Such assessments are usually seen as the best method to identify exactly how effective the existing security controls are against a skilled adversary and validating the efficacy of defensive mechanisms, as well as end-user adherence to security policies. PT practice is a methodological approach involves an active extraction, analysis, and exploitation of the assessed assets and its potential vulnerabilities [1]. PT relies on a set of classic tools that automate repetitive and complex tasks. The PT test are often initiated and carried out from the position of a potential attacker and involves active exploitation of security vulnerabilities, real-time exploring and decision making as the practice evolves is the key. The human PT expert's knowledge, decision-making, and reasoning are a cornerstone of the PT practice. PT tools, systems and frameworks were developed with the aim of making the practice efficient and allow regular and systematic testing without a prohibitive amount of human labor along with reducing the precious consumed time and networks downtime. Additionally, they are designed to offload human experts from the heavy tasks and helping him/her to focus upon more special and complex situations such as unusual vulnerabilities or combined non-obvious combinations (application flaws, improper configurations, risky end-user behaviors) which require particular attention in order to produce the best results.

One of the main branches of PT is network testing, and in the context of medium to large size networks, performing penetration testing is often lengthy, very complex and resource intensive despite the extensive use of highly developed autonomous and usually expensive systems and frameworks. Additionally, the wide variety of assets and vectors such as servers, endpoints, web applications, wireless networks, network devices, mobile devices and other potential points of exposure are also playing against the pen-tester breaking through the network firewall and evolved beyond by pivoting across networks machines, systems and applications and attempting to find a new path of attack or revealing how chains of exploitable vulnerabilities to progress further within the target network critical systems and data.

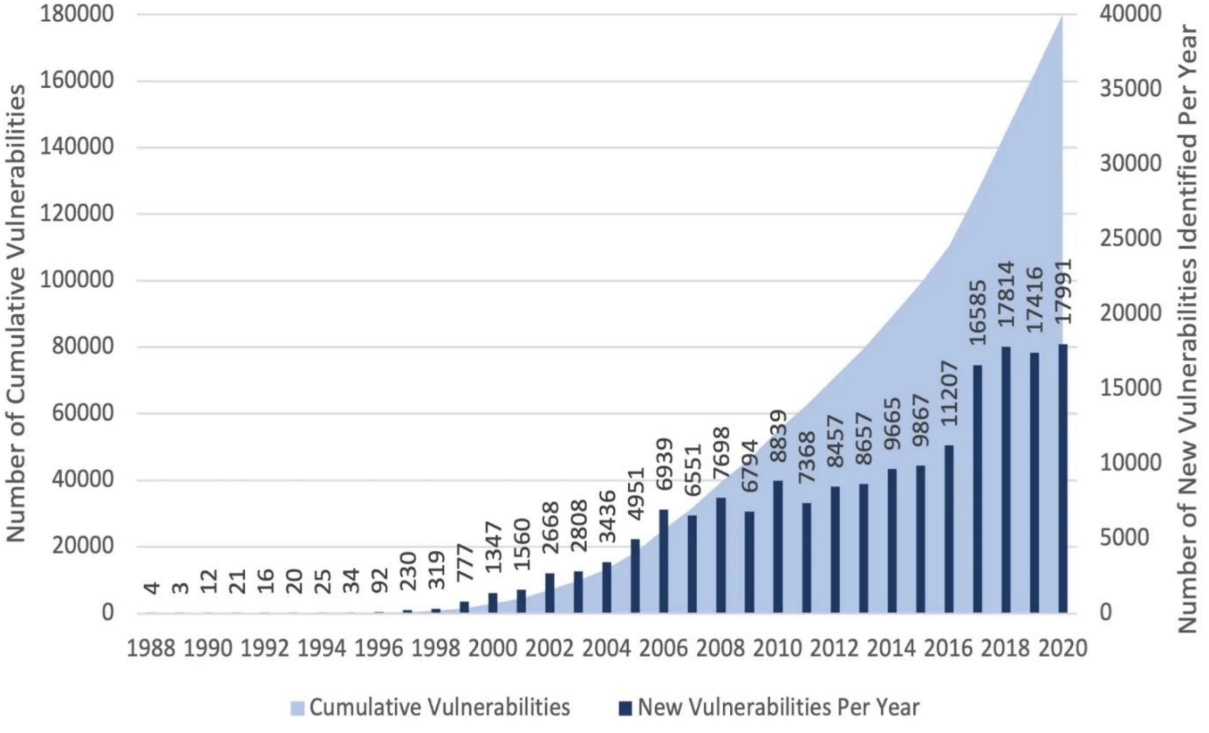


Figure 1: Cumulative and yearly number of vulnerabilities discovered from 1988 till 2020 [7].

The PT practice has significantly evolved in the last decade especially with the appearance of automated versatile tools and systems such as Metasploit, Core Impact, Nessus and other toolkits which come under Kali and Parrot OS distributions which paved the way for more automated PT when aggregated with the existing portfolio of scripts and codes commanded and orchestrated by human PT experts. To achieve more efficiency, there was several research works that focused on using AI and specifically ML techniques within PT but mainly investigated at least one activity planning and revealed more efficient and effective than automated testing notably in delivering flawless results and saving time and resources. Core Impact, a tool relying on a research project was the first to propose an AI-led automation of one activity of PT, namely vulnerability assessment, aiming to achieve a better efficiency.

In this chapter, we will cover the reasons and motivations for pursuing this research on embedding adequate machine learning techniques within current PT automated tools and systems. In addition, the problem statement and the scope of this work will be detailed along with the proposed methodology for conducting the research project. It will tie the research questions with aims and contribution expected by the end of this research. Finally, this chapter will clarify how this thesis is structured and what each chapter tries to accomplish.

1.1 Research Motivation

In this section, we discuss the motivations behind our conquest of an intelligent automation in network penetration testing practice. This research is rooted in a real-world problem that experts and technicians working in offensive cyber security field are continuously facing. In fact, the need for PT is increasing making it a central and mandatory component of the cyber-security audit and compliance with different standards and regulations worldwide. This research seeks to propose scientific solution to a real-world problem by investigating the practice automation, elect the most adequate AI approach and propose a versatile framework which produce an intelligent and optimized penetration tests on network context while remain intelligent, autonomous and self-learner.

The PT practice has significantly evolved to keep the pace with cyber advisories, and this led to the appearance of dozens of commercial and professional systems and frameworks which all aim to offer an automation of the different PT activities and tasks. Nonetheless, the existing automation remains either local (specific to one activity such as the vulnerability assessment or scanning) or not optimized (covering all cases) leaving significant issues to professionals using such systems. It has long been believed that PT is about testing the different defenses mechanisms of digital assets through ensure effectiveness in countering cyber threats and attacks and ensuring that any weaknesses or vulnerabilities in are identified and fixed before they are exposed and exploited by a real hacker. However, in practice, PT became a check-box exercise in which a set of standard systems and tools are used to test for a series of known vulnerabilities or routine tests which in fact does not adhere to the practice core aim which is mimicking hackers in their operation-modes, techniques, and methods regardless the efficiency on the technology and performance of the used system.

The current PT automated systems and framework are in fact acting as tools fully controlled by the expert to perform tasks following the human decisions because of the lack of prioritization and optimization. The expert use output to analyse, plans and request the execution of the required tasks and those systems only execute the expert instruction which in practice very similar to the time when the practice was admittedly manually oriented requiring highly skilled and experienced individuals capable of understanding that is: think, decide and act in the most complex situations. Generally, these systems and frameworks fail specifically to perform in an acceptable manner when the assessed asset size and complexity reach or exceed a certain level. In fact, a higher number of machines and online services mean larger exposition surface with attacks that can range in scale from massive state attacks to simple attacks on individuals and SMEs in the hopes of gaining credentials or financial details. In addition, other issue arises with the use of such automated systems in combination with issues raised on the manual approach notably:

- The high cost in term of human resources associated with expert systematic and periodic testing along with the impact on the assets' performances and systems downtime during the working hours.
- The high volume in term of data produced by comprehensive nontargeted testing and which is often wasted and unexploited properly.
- The nature of the PT environment where the high threats' emergence and fast changing rate along with assets continuous security protection evolution and update which require system regular testing.
- The evolving attacks complexity with more evasive threats launched in which hackers adopt complex and indirect attack routes, techniques, technologies, this results in unlikely paths being used to squeeze through the security layers which is difficult to be imitated by pentesters.
- The huge amount of repeatability as most of the performed activities and tasks are repeated with hardly any change and this is representing a significant part testers time, often repeating does not require PT human expert decision making or manual intervention which results in decreasing the performances.
- The common high degree of obfuscation in large infrastructures notably in the corporate and financial sector where organization tend to use in-house developed security systems making the coverage of the whole assets challenging.

So far, both manual and regular (also referred to as blind) automation of PT fail fundamentally in addressing current challenges. In addition, recently the practice became more comprehensive and includes all standard auditing and testing tasks starting from information gathering and analysis, identifying vulnerabilities and executing the relevant attacks against the identified security flaws, it ensures the right application of security mechanisms and solutions, and it is widely trusted and adopted as industry standard. Therefore, the main motivation behind this research is to introduce a framework where automated PT practice become more efficient and effective by taking advantage of different AI techniques at different levels, the sought framework will, at this stage, cover networks and infrastructures PT and by embedding the adequate AI techniques and complementary solution will optimize the practice and enhance efficiency and effectiveness of current industrial tools and systems notably Metasploit and Nessus. In other words, the motivation is about removing (or reducing to the minimal) the PT human expert from the practice and pushing towards intelligent solutions relying on AI and notably ML to offload the human expert, making PT accessible to non-experts, and on the other hand making it more efficient and accurate in terms of consumed time, testing coverage, resource use and impact on the assessed assets.

Reinforcement learning is the designated best candidate subset of artificial intelligence which serve our problem requirements which is an entity that can learn and adapt without following explicit instructions but instead by interacting with its environment [23]. The introduction of RL in PT, especially with the current size and complexity of network infrastructure, is expected to bring a quantitative and qualitative contribution to the practice notably as PT is, more than ever, time and resource constrained practice as well as versatile in sense when it relies on real hackers' approaches and methodologies, all the aforementioned characteristics of PT are making it more difficult notably especially when aiming to comprehensively test medium and large networks. Even though repeating the practice on the same or similar assets should make it easier and faster, manual and blind automated PT brings no help or improvements in term of accuracy or time-efficiency.

1.2 Problem Statement

In this research, our overall aim is to enhance performance in term of efficiency and accuracy in current unoptimized automated networks' PT practice. In the beginning, most PT activities and tasks were performed manually with limited number of tests run against a small size computer network and thus making manual PT quite efficient (time and resources wise) and effective as experts were unlikely to dismiss or omit important tests. Then, the proliferation of large computer networks and the increase in complexity associated with the employment higher number of application and security mechanisms and the automation of processes forced the automation of PT tools and system in a hope of covering more ground in a shorter amount time [12]. Nonetheless, the automation revealed as not enough especially with current corporates and organizations complex networks with hundreds and more of IP addresses and increasing networks complexity, application and virtualization making it impossible to rely on blind automation requiring a huge amount of time and computational resources and making it more difficult for PT experts be in control of such automated tools to assess the security of every component in a reasonable amount of time. Furthermore, as PT is a repetitive process where organizations are legally and technically required to perform security testing either periodically (monthly, quarterly and yearly basis) or after each major systems upgrade occurrence (new installed software, applied system upgrades, user policy modification, security patching, addition of infrastructure) which remains a key requirements in the overall security compliances and industry standard in many fields such as PCI-DSS and RBI-ISMS in financial sector, HIPAA in healthcare institutions and ISO-27001 for other businesses information security compliance [13].

As this research focus on optimizing of the currently pseudo-automated PT practice, we anticipated that this optimization will requires a platform that covers the entire practice which enables embedding of the adequate AI techniques, Metasploit and Nessus PT systems were the only viable options. Therefore, we are going to develop an automated PT modular framework which utilize Metasploit and Nessus and which will serve as host for embedding the adequate AI techniques which is reinforcement learning in order to optimise the automation through the modelling of PT as RL problem and representing in form of POMDP environments and solved later to determine the vulnerabilities' requiring testing as well as the prioritisation in term of complex attack vectors (also called paths) to be tested and acted on their outcome in an interactive way. At later stage determined by obtaining initial results, our second research aim is shifted into tackling the scalability through better approach in dealing with medium and large networks PT generated POMDP environments which requires huge amount of time and computational power for solving. This scalability issue became problematic with larger environment as discussed in the RL chapter and thus we introduced the hierarchical representation of large problem through dividing POMDP environment which are often the results of large LANs and MANs networks, we investigated different options of enhancing the efficiency of IAPTF especially in solving medium and large size POMDP which are the logic result of representing PT in medium and large size LANs and MANs.

Furthermore, we anticipated the major issue related to embedding AI and notably RL in the PT process and specifically the sequential decision-making diagram. In practice, applying RL to solve a real-world complex problem such PT can be particularly tricky as the problem can be viewed from different angles with each having its own associated domain representation. In PT this can range from a simple planning (decision) problem allowing and research for the optimal attack plan which reduce the use of resources or ordering tasks to achieve the sought results, this point of viewing thing was specifically tackled by previous research which considered the different phases of the PT practice as entirely independent (not related or proactive) and thus modelled the PT as MDP and POMDP aiming to determine the best policy (policy search approach) which optimises the use of resources and time.

1.3 Scope of the Research:

In this section, we briefly discuss the scope and limitations of this research. The need for an intelligent automation in the PT practice is persistent, general and apply for all PT different variants notably their application domain, this is reflected by the huge versatility in PT practice which will be detailed and discussed in chapter 3. Nonetheless, in this research we are only concerned by one application of PT namely network (infrastructure) PT. This domain of application is the most common and the most technically challenging where the existing automation provided by industrial and open-source systems and tools is critically lagging in term of addressing current security challenges. Therefore, the scope of this research is limited to the domain of networks' and infrastructures' PT with a possible extension in term of application of our framework in future to cover more PT domains such strategic large grids (Internet and energy), web applications and IoT (Internet of Things) testing.

1.4 Contributions and Novelty

PT is undoubtedly the most efficient way to prove the falsity of the hypothesis that a specific network, system or software application is secure by identifying and exploiting existing vulnerabilities. In offensive cyber-security practice in general, few works engaged with the use and integration of machine learning capabilities within the existing system or developed such solution. As PT practice is mostly about knowledge, expertise, decision making (judgments) and even instinct (or guessing), An intelligent automated PT practice which relies on one or more of AI techniques in its different phases to improve efficiency and accuracy. This research is about investigating the use of AI in PT practice to offload and ultimately replace the costly and unreliable human expert. It is all about identifying where and how the AI and more specifically RL can be embedded in the current PT practice. The research resulted into developing an Intelligent Automated Penetration Testing Framework (IAPTF) which will utilize reinforcement learning and others AI techniques (intelligent planning and optimization) along with expert and cognitive techniques to perform PT efficiently and accurately in context of large and complex computer networks. Our proposed framework will be custom developed, built on a novel modeling of the PT environment and practice representing it in the best possible way which reflects the reality of the cyber environment and adversary profiles. Using techniques from AI (optimized planning), RL (partially observable Markov decision process or POMDP) and ES. In a first phase, an initial model will be developed, implemented, and tested in real world situations. After successfully validating the proposed model, the proposed solution will be integrated into an existing PT automated platform, the introduction of the pre-processing and optimization solution will, therefore, be considered and tested.

The IAPTF is intended to partially (at early stage) and fully (after maturity) substitute the human expert and act (perform penetration tests) alone automatically with efficiency and accuracy especially on the deterministic or already seen scenario. Also, the intended system will be able to understand the situation, think (find the solution) and act in due course with or without the assistance of the human expert (which will be detailed later in the RL part). It will be independently responsible for self-improving its performance by constantly updating data related to the environmental parameters, acting policies and the results (output) of the tests which will reduce considerably the consumed memory as no duplicated data or bad experiences will be uselessly stored. As with a human expert, the system will pre-eliminate the testing of previously confirmed failed tested attack (with the same input) both from the system own experience and the security data (blocked by firewall or IDS) but this will be very carefully considered as scenario with no firm confirmation or miss one or more factors contributing to the failure will not be included as experience but only as a general example configuration for the learning and experience. The system's acquired expertise will exclude all attack testing scenarios where a failure occurred, but other alternative(s) is/are available which a genuine attacker could explore by following different attack paths or approaches. In such situation, the system will rely on itself to plan and test the remaining attack plans that pivot before including the experience in its memory.

Furthermore, while an extensive academic literature exists on offensive cyber and its automation, the literature is almost entirely focusing on the planning of attacks and penetration tests independently of the real-life context. This research is among the first to approach the whole PT practice optimisation by embedding the adequate AI techniques. Therefore, after investigating the feasibility and the adequate AI approach, we proposed a modelling of PT as a RL problem, an implementation along within a novel framework named IAPTF and proceeded with testing in real-world reconstructed virtual networks created to evaluate the output of this research. The proposed framework is capable of imitating a human expert in performing an efficient automated penetration testing. In detail, this research contributions include:

1. Identifying research gaps by reviewing literature published on PT automation and different optimization approaches covered so far.
2. Identifying general principles and metrics for measuring the effectiveness and efficiency of an automated PT based on a comparison with current human-led practice.
3. Explaining the differences between the vulnerability assessment and PT in the cyber environment and the uniqueness and versatility of PT practice.
4. Investigating different AI techniques suitability for optimising the current automation of PT practice.
5. Explaining in a comprehensive manner how network PT is performed and extracting key elements relevant for the targeted AI embedding within the practice.
6. Providing a RL modelling of network PT practice and proposing a representation of the problem domain as partially observed Markov Decision process (POMDP) environment.
7. Developing an Intelligent Automated Penetration Testing Framework which covers the full PT practice and acts as replacement of the human tester in performing network PT.
8. Gathering real-world network security and configuration data and using it to reproduce a

virtual equivalent network to be used for testing and validation in this research.

9. Testing, evaluating, and validating the proposed framework in different testbeds and with different variables.
10. Demonstrating how effective is an AI-led PT compared with a human expert notably in terms of reliability and coverage measured by the number of attack vectors tested and their complexity.
11. Providing an Expert System based approach of handling repeated (regular re-testing) in PT practice and which best choices for both opponents.
12. Investigating how hierarchical RL representation of large POMDP environment address the scalability issue in the context of medium and large size networks.

Finally, it is important to highlight that this works seeks to embed RL in the whole PT process including most of activities and task currently manually or automated. The introduction if IAPTF is the cornerstone in this research as it enables the integration of AI through the interaction of the software RL agent (or eventually RL agents) with the assessed networks (environments) which actually covers the entire PT practice and not be limited to attacks' planning which does not reflect the proactivity of the practice which is more than just a controlling or planning exercise. The ultimate output of this research is a framework in which we implement the RL model of PT, and which replaces the human and is intelligent to make decisions and learn from itself. Therefore, viewing the PT as a planning problem is insufficient as the real-world situation of PT practice is quite different and every phase is closely inter-related, pro-active, and often repetitive.

1.5 Thesis Outline

This PhD thesis is organized in six chapters following this introductory chapter. A brief explanation about every chapter content is summarized follow. Chapter 2 will summarize the methodology and provides an outline of the research methodology used to identify the motivations, answer the research questions and problem, explaining research sampling and analysis methodology. In addition, the chapter will explain the limitation of the research method. It highlights the approach of research beginning by review of traditional PT practice (Manuel testing using tools and systems) methods and approaches. It investigates the differences between networks and other domain of applications for understanding the relationship aiming for producing a precise comparison presenting either similarity or differences. Furthermore, intelligent penetration testing automation section will be formed as result of embedding AI techniques and specifically Reinforcement Learning together with automated PT practice supported by the proposed framework. This chapter will also anticipate the scalability and efficiency issue which will be encountered in medium and large networks context and how we are going to deal with this major challenge. To tie this chapter, final section will explain expected research output and how test-bed networks are designed to enable framework testing, obtained results evaluation and performances validation.

Chapter 3 cover the Literature review chapter and provides an overview about main concepts of penetration testing and other related essential concepts related to offensive cyber security. Then, the chapter investigates the improvement of PT automations. This chapter will enumerate, and categories major approaches on the automation and optimization of the PT practice at different stages and activities along with presenting a critical analysis of each approach. Then, chapter will investigate the challenges of achieving an efficient PT and notably in context of medium and large size networks and will critically evaluate the suitability of major optimization approaches in term of suitability and limitations if not successfully working. Finally, the chapter will summarize, and survey previous research works and studies conducted in the field of PT and Vulnerability Assessment (VA) automation and optimization and focus how these literature will be employed as the starting point for our research and specifically, how the knowledge gap will get filled as a result of our research project.

Chapter 4 cover all the theoretical body of knowledge around AI technique to be utilized in this research for an intelligent and optimized PT namely Reinforcement Learning. The chapter introduce RL, the main component constituting RL environment and the different formalisms of RL problems with a special focus on POMDP. The chapter cover the solving approaches namely value iteration and policy search, algorithms and methods employed within the AI community and emphasis on the mathematical basics in relation with RL domain and different theorems and demonstrations relevant to our research. This chapter also covers the crucial task of modelling a problem as POMDP and the use of different approaches namely model-based and model-free as well as the major wolving POMDP algorithms.

Chapter 5 introduce the proposed RL and HRL model for network PT practice and its representation as POMDP problem. This chapter will present a comprehensive model elaboration documentation and different steps undertook towards building the proposed model and ho it will be represented in form of POMDP notably in form of States, Actions, Initial belief, Observations, Transitions and Rewards.

Chapter 6 details the different components of the proposed framework we named IAPTF. This chapter explain the design and implementation of different modules notably IAPTF-Prep, IAPTF-Core and IAPTF-memory and highlight the interactions between different modules. Also, a brief description will be allocated to each function and script developed during this research ti facilitate the work of IAPTF and ultimately test the proposed framework including the efficiency and accuracy of RL-led PT compared with human and blind automation testing. At the end of Chapter 6 we will present the global IAPTF framework and introduce the different test-bed networks designed and created in virtual environment for the sake of testing the framework.

Chapter 7 covers the actual testing of the proposed models and the IAPTF framework in general. This chapter introduce the different test carried out to validates the relevance of RL and HRL model of PT practice and also to determine the best testing parameters to be adopted in later stage such as the discount factor and the solving method, algorithms and approaches. The second part of testing deals with evaluating the performances of IAPTF in comparison with human Certified Ethical Hacker (CEH) and blind automation in term of consumed time, testing coverage, relevance, and efficiency. A deep critical comparison and evaluation is then presented along with results discussion to finally highlight the ultimate research contribution to domain of PT and offensive cyber security in general.

Finally, this thesis Conclusion summarizes research findings and shed lights mainly over the contribution. It focuses on present tasks accomplished during the research project aligned with the outcomes achieved. Conclusion chapter ties research questions, that has been listed at the beginning of the chapter with the results gained after analyses. In addition, the chapter discusses the limitation within cyber deterrence domain. Moreover, it points to the direction needed for future work.

Chapter 2: Methodology

Our research methodology examines the optimisation of PT automation in the context of large networks and how it should work tosolve the problem of deterring cyber threats. In summary, the proposed methodology is expected to address in a scientific manner the real-world problem of efficiency and effectiveness related to the current PT automation. Furthermore, it answers research questions raised and address the succinctly research problem and thus achieving the overall aims of this research. This chapter provides an outline of the research methodology followed to gradually answer research questions, explaining research methodology and chosen approaches. The methodology chapter will cover main milestones of our journey towards an intelligent network penetration testing. This will start by reviewing the state of the art in the domain of PT automation and optimization, identify key elements of the current practice requiring an optimization, survey and critically evaluate the suitability of many AI techniques to our problem domain and later developing, testing and evaluating the proposed IAPTF [23].

2.1 Research questions:

In this section, we attempt to formalise our research aims and objectives into tangible and scientifically sound research questions. As already discussed in chapter 1, the size and complexity of today's corporate and organizations networks are making the regular and systematic testing problematic and adding further issues to the current PT practice. As both manual and automated testing fail to achieve the intended objectives within the allocated time and resources, we are concerned in this research with embedding AI within the process and addressing the efficiency and accuracy issues raised in the motivations. As one of the aims in this research is developing an intelligent automated PT framework which will rely on especially reinforcement learning (RL), other classic AI techniques (intelligent planning and optimization) along with expert and cognitive techniques (expert system). This enables the framework to replace the human expert and efficiently and effectively reasoning, prioritising, and acting in the face of complex PT problem which contain a vast amount of uncertainty and versatile testing scenarios. Therefore, the key questions that arise for this research are:

- Which activities, tasks and subtasks of the current network PT are blindly automated and making the practice slow and heavy in term of resources and generated traffic?
- Which AI technique will fit the purpose of optimising the sequential decision making made by the human expert?
- How to apply or embed the elected AI techniques to the current automated systems used for PT and making it intelligent and thus efficient and effective?
- How to reduce the human expert intervention and ultimately remove it from the different phases of PT practice. In other words, can the human expert in the field of PT be replaced

by an intelligent system or framework which relies on AI?

- Can an intelligent machine assisted-PT enable non-expert PT practicians to perform complex tests by assisting users in form of indications and suggestions which can be accepted or rejected by the expert?
- Should the AI-led PT framework be allowed to operate independently as stand-alone PT framework or enable direct interaction and control for a supervising human expert to enable benchmarking and auditing capabilities?
- How efficient and effective is the proposed AI-led penetration testing in comparison with human experts such as certified ethical hacker or fully automated machine?
- Which metrics should we use to measure efficiency and effectiveness of AI-led testing?

In other words, this research aims to answer the big general question, "How to replace the human expert by an intelligent framework which employs AI along with fully automated PT systems to produce an intelligent and automated framework?". Answering this question needs a deep understanding and analysis of the fundamentals of PT practice, defining the adequate model for the practice as RL problem, representing the RL problem and solving it to finally extract the expertise and store it for future use. Furthermore, considering the scaling up issue in large networks, the research needs to answer the additional question of how to deal efficiently with large networks. This research attempts to split these challenges and tackle each individual question.

2.2 Research Methodology:

In this section we present our research methodology which is our systematic approach adopted toward our aim of replacing human expert pentester from current manual and automated PT practice and enhance performances to reach a new conclusion on the contribution and relevance of the use of reinforcement learning in PT automation and optimization. The methodology used in this research begins with reviewing the offensive and few closely related defensive cyber security domains to digest different approaches and methods used for optimization and automation notably in vulnerability assessment and the wide penetration testing domain. Then we moved into traditional PT automation theory, principles, approaches and challenges in order to have a full grasp of the practice and understand the theory that could lead us into intelligent PT automation domain understanding by analysing research problem and electing the most suitable and adequate ML technique. This system should be fully intelligent and optimised enough to reason as a human expert but with better performances and take advantage of the particularities of the PT practice to enhance efficiency and accuracy.

There are different methodology elements adopted in different phases of this research. First, we will start the research by doing a domain understanding the PT domain and its different components notably the interaction between the environment and the human expert. In this step, we aim to grasp the manual (human) PT process and notably, the reasoning and approach and the decision-making process by scrutinizing certified ethical hacker way of thinking and executing the task in context of a large network (analyze the methods, thinking, tools and techniques that experts use to execute advanced security testing). Additionally, we aim to identify what human expertise is made from in context of PT and what kind of prior knowledge PT experts rely on in terms of data, information, guidelines, and recommendation gathered from previous similar experiences. This will be followed by a full mapping of PT data workflow which includes, but is not limited to, the target network topology, subnets, machines' configuration, security architectures, protection updates and patches. In addition, we cover data output from vulnerabilities scanners, available threat intelligence data, intrusion detection and security incidents data imported from Security Information and Event Management (SIEM).

The second element, we will proceed into reviewing related PT automation literature and full grasp the principles, functioning, approaches of PT practice followed by a critical study of the current automation in PT in general and in network PT practice specifically, we will scrutinize different automation and optimisation propositions and analyse them. As we elaborated in chapter 1 research motivations, large organizations' network PT is problematic where both manual and regular automation fail to achieve the intended testing objectives within the allocated time and resources. Therefore, we considered embedding adequate state-of-the-art AI techniques within automated systems and tools used for PT and therefore making the full practice activities fully automated, intelligent, and optimised. The aim is to offload (reduce the intervention) the human expert and ultimately replace them by a fully autonomous and functional PT framework which covers each phase, activity, and task of the practice. Here, we will decorticate the PT practice and identify all activities, tasks and sub-tasks which will enable us to elaborate a measurable definition of the term expertise from an automated framework/system and how this can be extracted, implemented and used within the elaborated processes and data workflow in PT.

The third element will be investigating different AI techniques that fit the aim of our research with a special focus on reinforcement learning approaches. This review includes also rule-based expert systems, attack graphs, neural networks, decision trees, clustering, and association rules. At this step, we select the adequate technique which is RL, and we will deepen our research by proposing an initial model PT practice as RL problem and propose a novel representation the global network PT problem as set of POMDPs.

Finally, we will complete this research by enhancing the model to address the medium and large networks issue and we will proceed into the development, implementation of IAPTF which will embed the RL module and expert system as well. In addition, a different hierarchical RL model will be implemented to address the scaling-up issue in the proposed RL model, enhancing performances, and tackling operational issues such as expertise capturing and generalization. The research methodology six steps are summarized as follow:

- ➢ Grasping the PT domain and main areas around environments and component along with the interaction between the different entities and expert.

- Reviewing the current state of the art of the current methods of PT automation at different phases of the practice namely, information gathering, discovery, vulnerabilities assessment and exploiting to fully digest and analyze the functioning mechanisms of each and the reason why they fail to meet the PT expectation in term of efficiency and accuracy.
- Studying the PT experts (Certified Ethical Hackers) and criminal hackers operating approaches and decision-making process when performing tests. This includes detailed understanding of what, why and how of every task and activity that expert performs from the initial reconnaissance and data gathering to the exploiting and post-exploitation tasks.
- Investigating how AI (more specifically ML) can replace or reduce human intervention in sequential decision process and notably PT and which approach is more suitable and likely to produce results.
- Producing an initial model of network PT as RL problem and ensured that all PT tasks and variables are captured and represented. The representation as RL problem will be implemented and solved using an external state-of-the-art POMDP solver which constitute alongside with the IAPTF environments generator the framework core module.
- Enhancing the proposed model notably by addressing performances and efficiency issue through the introduction of a new Hierarchical RL model and development of dedicated Pre-processing, Security Clustering, Expert System and Memory management modules.
- Testing the proposed solution and evaluating its contribution in term of efficiency and accuracy in real-world complex scenarios and subsequently introducing the appropriate changes in due course.

This adopted methodology aims to achieve the research final output which is a novel RL model of PT and a framework IAPTF that will offload and eventually replace the human expert in performing all PT phases in context of computer networks. The projected framework will automate with optimisation all aspects of the PT practice including a wide variety of activities and tasks and will take advantage of use of different AI and machine learning techniques in different approaches to answer the research questions.

2.3 Employed Research Method

This section aims to describe methods used in our research to collect data for virtual networks (testbeds) re-creation, elaborating the RL model of PT and finally to develop, test and validate IAPTF framework. The method employed will take advantage of use of RL algorithms. Optimized planning and rule-based expert systems to produce an intelligent framework which aims to assist and eventually replace the human expert in carrying network PT activities and tasks. In this section, we will detail methods used in our research to design, implement and test IAPTF with a special emphasis on virtual test-bed networks' construction out of data collected from real-world corporate networks. This research will produce a proof-of-concept (PoC) framework along with its practical implementation which will replace the human expert in performing PT in an intelligent (efficient and effective) manner.

2.4 Research Data Input

The input of this research are different size virtual networks which mimic the real world and constructed from importing and using real networks data. In term of collecting data, as this was the first phase, we extracted imported data from real-world small, medium and large corporate networks. The collected data include networking, functioning and security data which was used to recreate virtual equivalent of these networks in a virtual box platform. Computer machines and servers were included in the virtual networks by directly downloading virtual equivalent from a specialized open-source website vulnshub.com which serves as repository and provides materials that allow ethical hackers to experience in digital security, computer software and network administration using virtual appliances. Security mechanism including firewalls, routers and IDPSs were also imported along with the associated configurations (implemented security policy) and included in the virtual networks by adopting a specific approach of considering them as machines and forcing the traffic to transit through them in a specific way to reflect the real-world scenarios, thus approach was unavoidable as the virtual environment is restricted in term of networking. To sum up, we constructed 53 different networks with size varying from 2 to 250 machines and were categorised as follow: 2-50 small LAN, 55-100 medium LAN and 105-250 large LAN. Even though our research focuses on medium and large networks, we were obliged to start from small LAN to test the framework modules, the proposed RL model and the POMDP representation solving. Finally, it is worth to mention that the 200-250 machine limitation is just for testing purposes and larger networks can be also accommodated with the adequate hardware.

2.5 IAPTF modular choice

In this research we opted at early stage for a modular framework that covers the entire PT practice. The choice is justified by the nature of PT itself and the sought-after framework. Moreover, we were not sure to what extent this research will be pushed in term of implementation. Therefore, we started the development of the first module which used input data from information gathering, discovery and vulnerability assessment phases to represent it as POMDP environments. Then we launch the RL solving process where the software agent will determine which attack vector to follow then instruct Metasploit MSF to execute attacks and act on the obtained feedback. This module is the heart of our research and is named IAPTF-Core. The second module is named IAPTF-Prep and groups all data acquisition, collection, transfer, and formatting, pre-processing and feature extraction functions which work together as independent scripts. As with a human PT expert, this module aims to pre-eliminate previously confirmed failed attacks (while target configuration remains unchanged) and attacks which will certainly fail because of the security mechanism (blocked by firewall or IDS). The third module of the framework is IAPTF-memory which serve as the main memory for the framework and the expert system in charge of expertise capturing, generalization, storing and replaying. Lastly, the Metasploit (MSF) is considered as an entire module of IAPTF and consists of interfaces, libraries, MSF modules, tools and plugins which all will be controlled by the IAPTF-core through Ruby scripts.

2.6 Optimization evaluation and criteria

Currently, penetration testing efficiency is measured and assessed following several quantitative metrics which are widely adopted and standardized as performances measurement criteria. We elaborated a list of five which represent the best possible way measured metrics to assess a PT cycle. Fist metric is the average running time which is a straightforward metric to measure and reflect the time required to complete a testing cycle (which is also translated into cost as experts are often hourly paid) constraint in current PT practice, thus this metric will perfectly reflect the efficiency part of the measurement. The second metric is the testing coverage measured by the number of performed tests which are in our research measured by the number of valid attack vectors executed by IAPTF, this metric reflects the effectiveness part of the measurement. The third metric is slightly subjective and deal with the overall attacks success rate in term of high-value target compromising and executed post-exploitation activities such as rootkits deployment. This metric is highly probabilistic as the success rate of any exploit execution varies significantly and depends on other variables such as the execution order, the used configurations and existing security defenses and attacks' detection capacities. Lastly, the fourth metric is the amount of network traffic generated which is crucial in corporates as the downtime should be maintained to the minimal and thus systematic and comprehensive testing imply more tests and thus network traffic is generated, in addition to increase in detection probability by the existing IDSs. In fact, skilled hacker makes the least noise as possible especially during the information gathering and discovery phase as any suspected traffic will result in prevention action by IDSs or attract the attention of the SOC operator.

2.7 Intelligent Penetration Testing and Learning Choices

In this section we briefly discuss the choice of RL for our proposed IAPTF. All ML approaches respectively supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning were initially considered in this research with the aim of identifying the most suitable and adequate one for our context [23]. The choice of RL is justified by the sequential decision-making process characterizing the PT which was highlighted as result of our deep study manual PT practice.

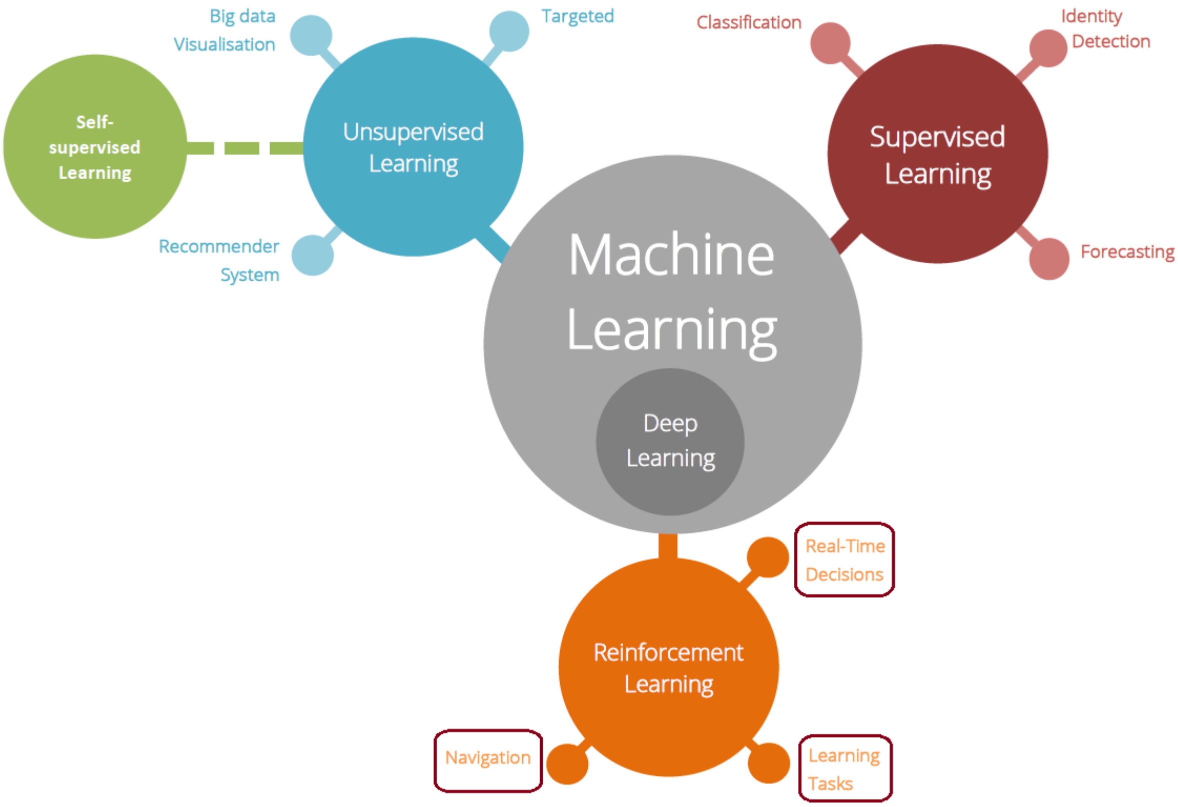


Figure 2: Machine Learning families and their main application domains [21]

Therefore, we conclude that the only relevant approach for the sought-after full PT practice automation and optimization framework is RL. Furthermore, RL reflects the interactivity and dynamicity in the PT practice which is fundamentally different from previous use of AI in PT and even proposed RL approaches [32] which was limited to attack planning and often resulted into a high level of inaccuracy as the generated plans for attacking were isolated from the real PT environment notably in discovery and exploitation activities and tasks. We also investigated the relevance of other complementary AI techniques to be used alongside the main RL approach and considered decision trees, attack graphs optimisation, and rule-based expert systems.

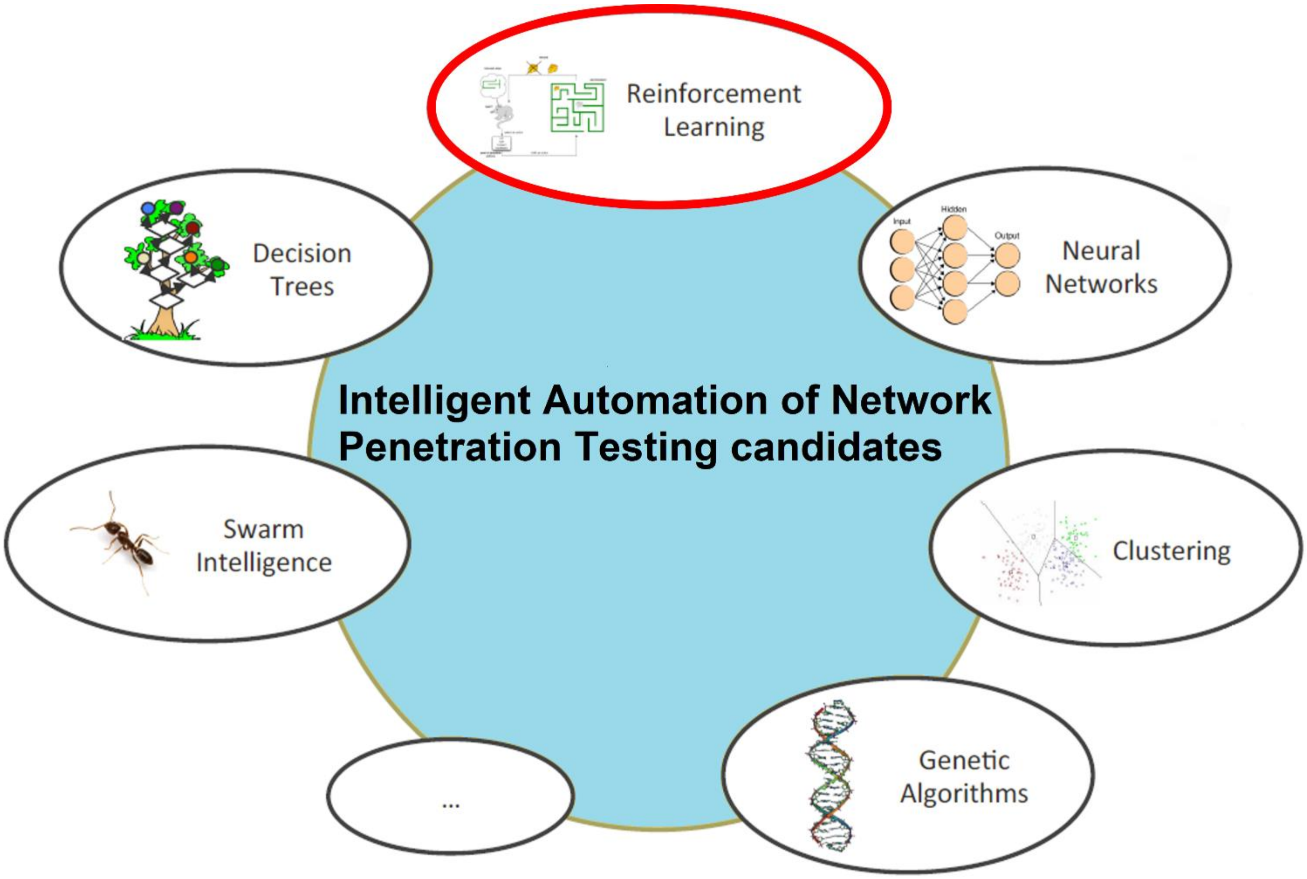


Figure 3: AI techniques candidates for an optimised PT practice [4]

The expert system choice is backed by the lack of knowledge extraction, re-usability and improvement as it is the case during manual penetration which is the main reason behind expert PT high efficiency. Doing this allowed IAPTF to perform better and continuously learn, memorize and reply expertise gained from previous experiences.

Finally, we briefly describe the RL model, representation and solving choices made in throughout this research. The figure 4 summarize the choices made which will be critically discussed in Chapter 6.

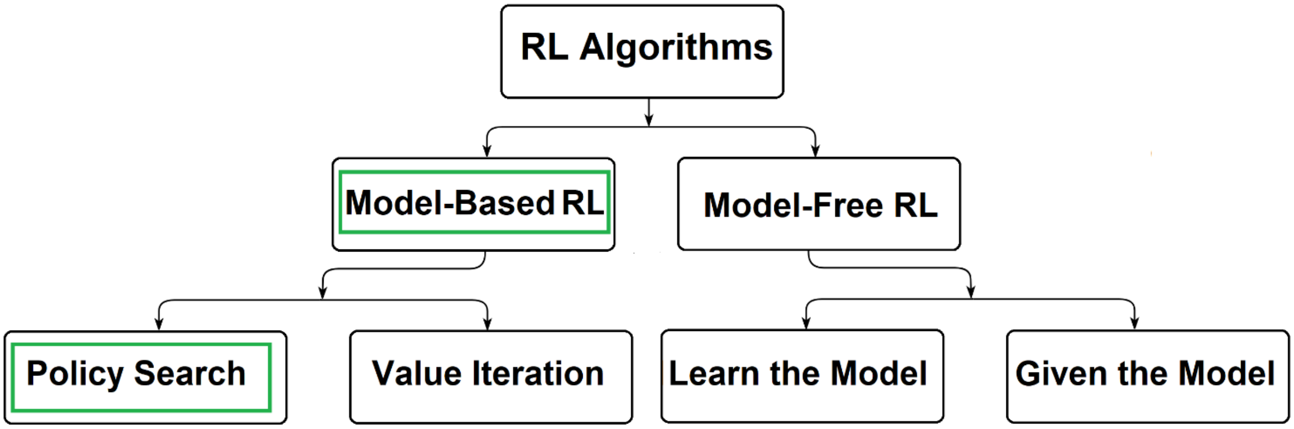


Figure 4: choices made in this research for an RL-led Penetration testing [24].

As we described earlier, after settling on modeling network PT as a RL problem as result of the reasoning approach followed is more of a deductive approach than inductive. The RL suitability for an automated and optimized PT framework that replace the human expert came after a comprehensive review of all candidates following the deep PT domain understanding and following a deductive reasoning approach. The model-based RL option is then selected to be the more suitable and reflective of the PT practice as we propose an accurate model as part of the RL problem definition which includes state transition and observation probabilities and rewarding distributions to calculate optimal actions. Finally, model-based solving algorithms utilise dynamic programming for solving following two main approaches namely Policy Iteration and Value Iteration. In term of solving POMDP choices, and as IAPTF aims to replace the human expert in decision making, the solving choice will be value iteration following by direct policy search.

2.8 Expertise extraction, generalization, and replay

IAPTF is designed to learn and adapt without following explicit instructions but instead by interacting with its environment and achieve expert-level competence in solving problems or extracting knowledge from human experts' operative modes in form of performing tasks and taking decisions. To convert this knowledge into useable acting policy, we opted to develop a separate module named IAPTF-Memory which includes the processing engine, expertise extraction and validation scripts, expertise generalisation scripts and the CLIPS expert system. This module development is challenging because of its complexity and arduous (labor-intensive) script coding but remains a crucial component of IAPTF. In this research, we will prioritize the option of enabling the framework to be supervised by a certified ethical hacker. On the top of CEH supervision, we allow IAPTF to receive some high-quality initial knowledge to get it started, and then leave it to learn the rest for itself or from observing the human expert handling it. We anticipated that IAPTF, after a period of time, will perform better than a human expert in both time and coverage along with pinpointing new complex attacking paths which human expert would not discover. IAPTF acquired expertise will then be used directly on similar cases and scenarios. In addition failed attacks will also be excluded when tested asset configurations remain unchanged, but when other exploit alternatives are available (variants which a genuine attacker could exploit) the attack is repeated. In such situations, IAPTF will rely on its core RL module to plan and execute the relevant attack plans as well as the associated post-exploitation and pivoting before including the acquired expertise in its memory.

2.9 Representing penetration testing RL problem as POMDP

Each network PT exercise will be represented in a set of POMDP environments, we provide here a brief description of the method adopted in elaborating POMDP. Basically, the IAPTF-prep output is the raw date used for elaborating the first POMDP, but further data is generated throughout the practice progression and will follow several processes that reflect the complexity of the PT practice activities. In a few occasions, we include data provided initially when testing approach is white-box or when re-testing the same network which is an important factor to be considered. In fact, information gathering, probing, and scanning data generated as part of early vulnerability assessment by automated systems and tools (which are part of the Metasploit professional including InsightVM and Nexpose) will be served be processed and double-checked in comparison with this provided data. The second phase is the security clustering which will be detailed in Chapter 6 which aims to classify each machine under a given security cluster. At the end of these phases, a python script will import and pare the raw data files (.text, .xml and .log) and input for IAPTF-Core. The data will be then translated into POMDP files with Actions, States, Initial-belief, Observations, Transitions and Rewards. The only exception is the Rewards which will be allocated using a pre-defined grid. All other POMDPs components are purely the product of the real-world extracted data including some processing notably in the transitions and observations probabilities calculation detailed in Chapter 5 and data from the cyclic information gathering, discovery and vulnerability assessment phases will be fed to IAPTF-Core during the continuous update process as part of the Initial Believe (starting point) for RL.

2.10 Addressing the scalability problem in medium and large LANs

In this research we encountered a famous curse of dimensionality problem which stands for the exponential growth in computational power demand as result of the increase of the size of POMDP environment [38]. The scalability problem itself was expected giving the complexity of the PT and therefore its representation as RL problem which often leading to very large POMDP environments in term of number of states, transitions, and observations. Therefore, we considered different approaches to model PT as Hierarchical RL problem instead of regular RL modelling in context of medium and large size networks. Different representation has been shortlisted, investigated, implemented, and tested to find the most suitable Hierarchical RL (HRL) in the context of PT. The selected hierarchical approach namely security clustering proved to be the most adequate and take advantage of HRL reducing dimensionality through decomposing the RL problem into several sub-problems which ultimately overcome the scalability problem in regular modelling. This point will be detailed further in Chapter 5.

2.11 Virtual test-bed networks, design, and testing

To achieve the research goals, we opted for real-world networks to be used as testbed and evaluate the proposed RL model in one hand, and on the other hand test IAPTF efficiency and effectiveness. As a matter of fact, we needed to set limits in term of network size and to make some decision regarding the nature of networks and we decided to import real networks data, setups and security configuration and reconstruct a virtual equivalent that mimic the real ones. This choice is also forced by the General Data Protection Regulation (GDPR) [33], and other privacy and security restrictions related to performing tests on real networks for research purposes and the associated implication in term of requesting authorizations [6], mitigating risks and post-testing checklists which make our research lasting longer in addition to the associated costs. Figure 5 illustrates a sample corporate medium size LAN of 50 machines as one of 53 different networks with size varying from 2 to 250 machines created for this research.

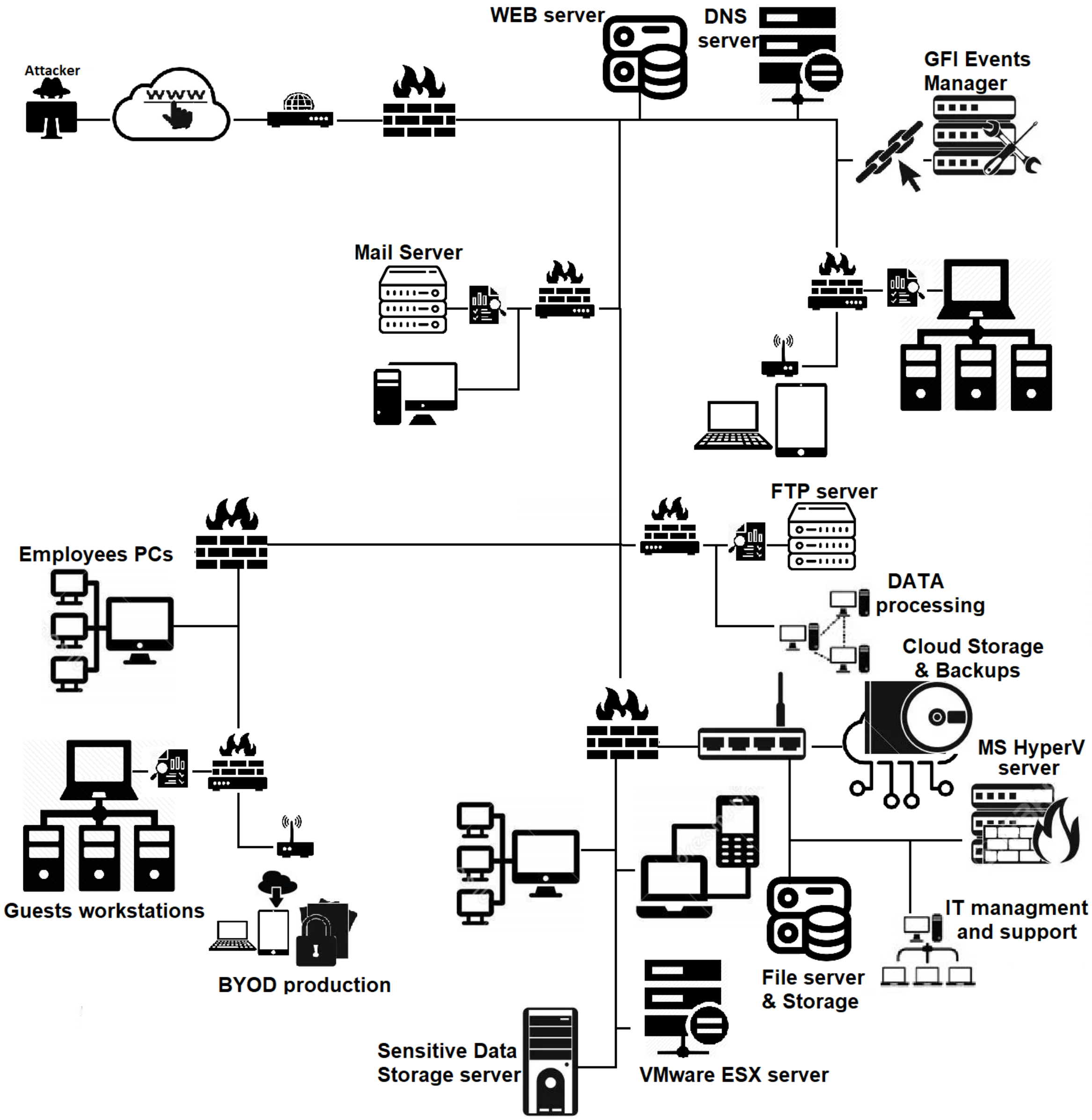


Figure 5: typical corporate medium size LAN equivalent testbed.

Chapter 3: Literature Review

The literature review subjects are structured as per the research methodology. This research review and categorisation is built to serve research approach and to help to provide answers for the research questions. The chapter has been organised as follows: Section 2 introduces the offensive cybersecurity and the wide PT practice along with the rationale and reasons behind the increasing need for PT practice, and we will present an anatomy of a typical cyber-attack to consolidate the presented topic. Section 3 discusses the actual PA practice and the various testing types, approaches and tools notably Metasploit Community and Pro, and details of Network PT methods. Section 4 reviews the activities and tasks constituting the network PT and the development of the theory and its involvement in enhancing security and preventing cyber-attacks and how organisations can benefit from this extra security along with new compliance requirements. Section 5 pinpoint the uniqueness of PT practice compared with other cybersecurity activities and domains, and highlight the crucial role human expert intervention in the practice. Section 6 goes through current approach and methods used in PT automation and challenges associated with dealing with large assets and the repeatability. We will review related works and categorises them and will present a research tree summarising who, what, how and when PT automation and optimisation techniques have been proposed and implemented. Section 7 reviews the literature that mainly discussing machine learning use in PT optimisation and provide a brief description of the relevant AI techniques. Section 8 goes through what are the challenges of penetration testing. Finally, Section 9 summarises what could make an AI-led network PT work successfully and limitations if not working as expected. Finally, section 10 summarises the literature review chapter and specifies a research road map.

3.1 Preface

The aim of this literature review chapter is to establish a theoretical base for our research work in term of offensive cybersecurity, PT and vulnerability assessment, the current state-of-the-art in terms of PT automation, human expert role in the practice, the use of AI in improving and optimising PT tools and systems, and research gaps. Towards the goal of the literature review, there is a need to align it with research requirements such as research problem, research questions, research methodology and approach followed. This alignment will achieve the expected advantages from the following systematic literature review.

3.2 Literature Review Methodology

The approach followed in selecting and classifying the literature is focused on three parameters: traditional deterrence, game theory and cyber deterrence. The work has been executed via three main phases which can be simply explained as:

3.2.1 Planning:

The main objective of this phase is to locate and specify the literature we are looking for and the parameters we should follow aligned with the research project problem and research questions. Focusing the literature over the research problem “optimising automated PT” will drive research project and gain up to date understanding about penetration testing, AI and ML topics. Specifically, literature is carefully selected that are linked to the main research question and utilized to bridge AI techniques applied for cybersecurity problem to the benefit of PT and vulnerability assessment.

3.2.2 Executing:

As the fundamental problem of this research project is “replace or at least limit the human expert intervention in PT practice?” inspired by other cybersecurity fields such as intrusion detection and prevention and security auditing. Based on this mission the selection, filtration and prioritisation of the literature was conducted. For that, offensive cybersecurity, network PT, vulnerability assessment, applied reinforcement learning for sequential decision making, attack graphs, were parameters for selecting the literature. Other issues were considered like linkage to the research problem, quality of publications, credibility of publisher and specialty of the authors.

3.2.3 Reporting:

Reporting is the final phase of reviewing the literature task. The structure of reporting begins by introducing main concepts, approaches and contributions of PT automation and optimization. Then explain different strengths and weaknesses of the proposed solution related directly to network PT. Finally, we elaborated a research graph where different research works are categorised under the adequate rubrics.

3.3 The need for offensive cyber-security

Cyber-attacks and exploits have constituted the cyber risks and exposure during the last decades. A key element about cyber-attacks is that they remain non-standard in term of methods and approaches and involve the use of a variety of tools, systems, and scripts to accomplish different activities and tasks [9-11]. In the heart of this cyber advisories we find exploits which are pieces of software, data or sequence of commands that take advantage of a vulnerability to cause unintended behavior or to gain unauthorized access to sensitive data [3-4]. Figure 6 illustrate the number of weekly reported cyber-attacks during 2020 per organisation per industry [4].

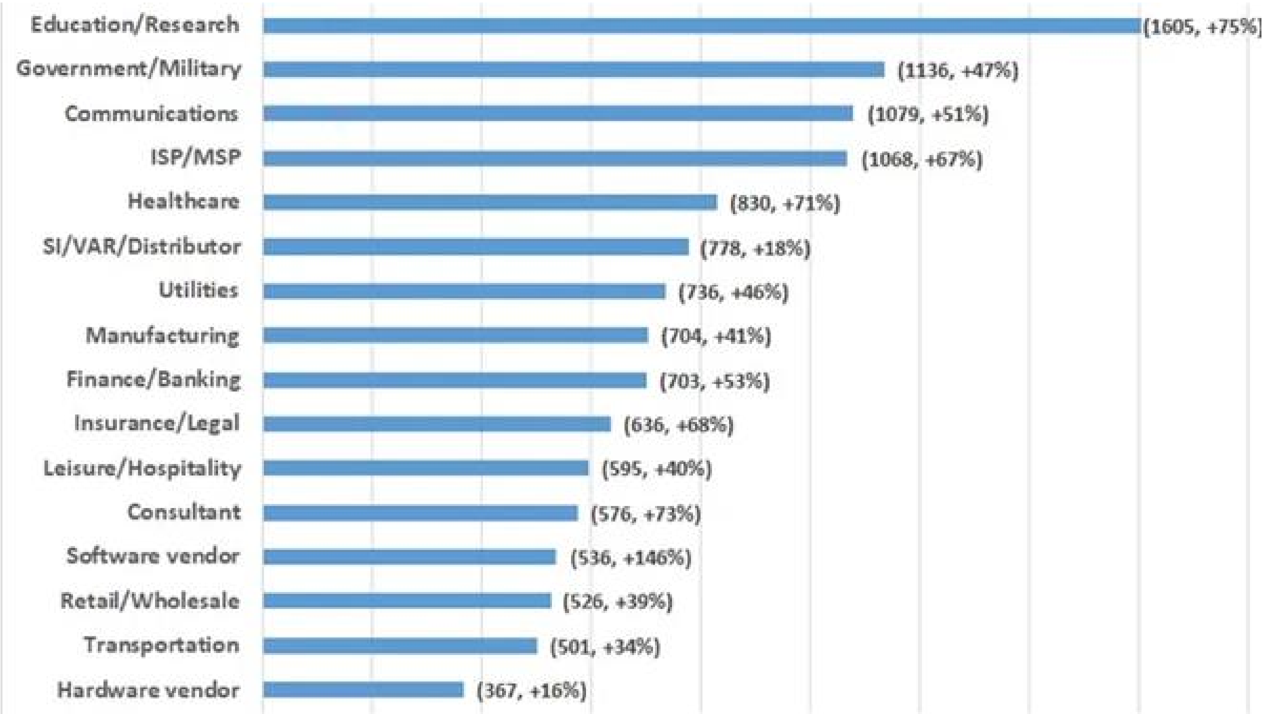


Figure 6: Average weekly attacks per organisation by industry in 2020 [11].

Cyber attacking and hacking is a full-scale business across the world with global cybercrime costs expected to grow by 15 percent per year over the next five years, reaching $10.5 trillion USD annually by 2025, up from $3 trillion USD in 2015 [5-6]. This evolution is illustrative and reflect an explosion in number of exploits, attacking techniques and technologies. Figure 7 illustrated average zero-day exploit price in dark-web which reflect time and effort invested into developing and weaponising exploits which the observation enable researcher and managers to grasp the current and forecast for the cyber advisories [13].

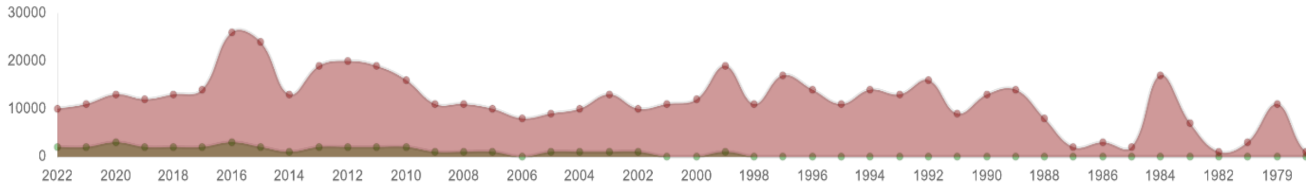


Figure 7: average exploit price in the dark and legitimate market [14].

3.4 Anatomy of a Cyber Attack

PT practice is a standard and well-established security auditing method designed to imitating hackers' footsteps and approaches by imitating the same behavior that real attacker will eventually adopt when attacking the digital asset. To achieve this aim, PT practice was divided into activities and steps following a certain logic. The division itself is just a way to help the penetration tester better cope with the pattern of steps that we describe below. Of course, this division in steps is not arbitrary and corresponds to a well-established practice in the field of offensive cyber-security domain. In this part, we attempt to summarise the major constituent of PT. An example of real-world attack starts with a Reconnaissance phase where hacker gather network information and build a knowledge about system profiles. Second phase named Discovery in which the hacker will be attempting to identify and perform vulnerabilities analyses the available information such as OS, device, service, and application and available exploit. The third phase named Exploitation where hackers attempt to launch programs and scripts to compromise the target, access and manipulate data, often in this phase hackers operate many breaks to seek further information or re-adapt the configuration of exploits. The hacker fourth phase is often to proceed with leverage compromised systems as beachheads to attack other network resources through proxy pivots and test defensive technologies' ability to identify and stop attacks client-side and testing of end users and endpoints. Finally, the hacker exfiltrates important data and installs backdoors and rootkits which enable him to gain sustainable and permanent access to the controlled system for future use. Figure 8 summarises the hacker typical workflow in network environment [18-20].

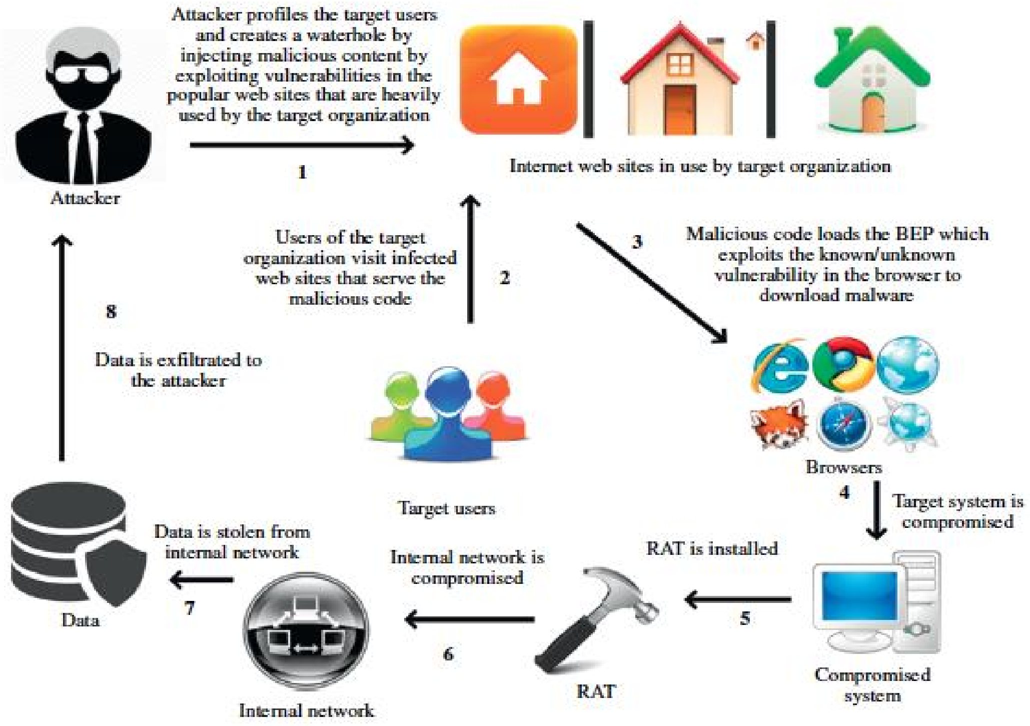

Attacker profiles the target users and creates a waterhole by injecting malicious content by exploiting vulnerabilities in the popular web sites that are heavily used by the target organization
1
Attacker
Internet web sites in use by target organization
Users of the target organization visit infected web sites that serve the malicious code
2
3
Malicious code loads the BEP which exploits the known/unknown vulnerability in the browser to download malware
8
Data is exfiltrated to the attacker
Browsers
4
Target system is compromised
Target users
Data is stolen from internal network
Internal network is compromised
RAT is installed
Data
7
5
6
RAT
Compromised system
Internal network

Figure 8: The Offensive and Active security standard workflow [18].

3.5 PT Background

PT has been a cornerstone in cyber security practice during the last decade. It implies planning and performing a real and controlled attack on a digital asset (computer machine, Web server, IoT, software or network) with the aim of evaluating its security [3]. The process of PT is divided into a sequence of tasks in order to methodically and comprehensively assess the security of the system and often include actively identifying vulnerabilities and perform a set of actions to test if the target could be compromised by running exploits against those vulnerabilities [6].

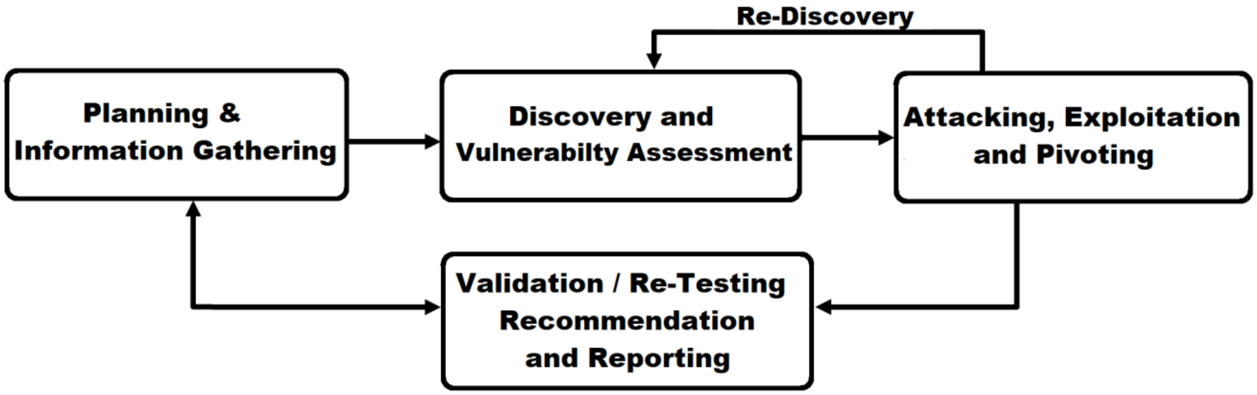

Re-Discovery
Planning & Information Gathering
Discovery and Vulnerabilty Assessment
Attacking, Exploitation and Pivoting
Validation / Re-Testing Recommendation and Reporting

Figure 9: Penetration testing standard phases [24].

In practice, PT activities and tasks shown in Figure 9 vary from case to case but generally include four main phases as follows:

3.5.1 Phase 1: Information Gathering, Planning, and Preparation

As with real world cyber-attacks, the PT practice relies heavily upon what the tester and/or the system knowledge of the target asset, and thus the ability of gathering, structuring, and processing the relevant information prior to starting the exploitation is a key point for the global practice. The information itself can be very wide and vary from one asset to another. In a network context, these properties often include general knowledge about the assessed asset, previous incident, previous PT attempts, active domains, and machines (network devices and hosts). This first crucial phase is widely known among the pen-tester (and hackers) as information gathering or reconnaissance. This phase includes performing tasks (set of actions, observations, and transitions) which will be detailed below [14-15].

3.5.2 Phase 2: Network Discovery

During this phase, the tester performs a deep and comprehensive reconnaissance against the targeted asset to gather as much information as possible, from which the next phase effectiveness will heavily depend upon. In practice this is a complex, often incomplete and resource consuming phase. There are many ways to gather this data and it depends upon the target (network, Web, or client). In this phase we differentiate Network Discovery which attempts to discover additional systems, servers, and devices; Host Discovery to determine open ports on these devices; and Service Interrogation which investigates identifying actual services running. Overall, tasks related to discovering, dressing and determining the network content, topology and configurations are performed using systems relying upon mechanisms such as ARP, TCP SYN packets, ICMP echo request, TCP connect and passive discovery. Following an initial discovery, a more comprehensive and deep discovery is also performed to get a clear idea about the network functioning and enumerating in the best possible way the network components, communication, used configuration. Tasks such as port scanning, OS identification, running applications and services identification are performed. The output of this phase is crucial for the later phases and usually, incomplete or partial results are initially obtained (lack of information or reliability of the results) and thus this phase will be re-iterated depending upon the required information [8].

This phase usually requires heavy involvement from the human tester despite the use of semi-automated or fully automated tools and systems such as Nmap and Nessus. The intervention of the human is often necessary. For instance, OS detection utilizing the running service produce results with a certain degree of uncertainty (probabilistic) and the use of further techniques by the pen-tester such as social engineering and Google hacking (using publicly available information to gain insight into the target organization) although these techniques are difficult (or impossible) to automate [23].

3.5.3 Phase 3: Attacking, Exploiting, and Pivoting

During the attack phase, the tester will attempt to identify (select) and launch the matching (giving the outcome of the precedent phase) exploits. Exploits are pieces of systems (code) that allows taking advantage of the systems’ vulnerabilities (such as injecting malicious code in the system's memory to modify the execution of the original code. One way that benefits the attacker to retrieve information, install an agent or gain access) in order to compromise or control the targeted asset. Once a system is compromised it can serve as a starting point (also known as vantage point) for launching other attacks against the connected or reachable systems so new attack vectors and paths can be revealed and tested. This practice of mimicking attacker behavior is widely known among the offensive cyber security community to be the most efficient in term of revealing system hidden weaknesses or open new attack vectors. A perfect example to illustrate the pivoting concept is a successful attack against a machine belonging to a subnetwork which after installing the agent and gaining access (controlling). Then, the privilege escalation process is performed to attempt a deeper penetration into the compromised system such as local exploits in order to gain “admin” privileges which will be used to compromise other machines belonging to the sub-network which are linked (via trusted connection) to the compromised machine [22]. The compromised machine could also be used to launch a new Local Information Gathering campaign to collect additional information about reachable systems or capture sensitive data transiting within the sub-network. This phase ends usually by performing a cleaning to erase the attacker footprints in order to avoid the reusability by an unauthorized party [17].

3.5.4 Phase 4: Reporting, Analysis, and Mitigation

Recommendations for PT are often prioritised by severity or impact. They are not necessarily rated according to their overall contribution to the organization's cyber risk. This reduces the value of a PT report and means that organisations do not necessarily make the right decisions about where to focus their remedial efforts across the spectrum of defensive, detective and operational security controls [17].

3.6 Penetration Testing Types and Levels

This section provides a brief summary of the different types and levels of PT (Figure 10) employed to assess and exploit potential network vulnerabilities. This categorization is often theoretical as the real-world situation is quite different and may require combinations of the below types and levels of tests depending upon the specificity of the situation and the client requirements. Moreover, the type and the scope of testing are generally established and agreed prior the beginning and may be changed or extended during the work [21-23].

3.6.1 White Box

White box testing is a PT method (also apply for software context) where the tester has complete knowledge of the internal structure, content, and configuration of the assessed asset. It is also called glass box testing, open box testing, transparent box testing, structural testing, and clear box testing. All these terms indicate that internal mechanisms are visible to testers. In white box testing, it is all about testing internal security in place (coding, infrastructure of software) and used often as methodology that validates the internal security. It also used to carry out vulnerabilities assessment such as highlight threats from inside the network that was using knowledge of your network, such as IP addresses, router access, active ports, web servers, FTP, and even passwords [2-3].

3.6.2 Black Box

This means that testers are given very little or no information prior to the penetration test. It is also referred to as "blind testing" because the tester will start by finding an open route to access the network which is often through the Internet. In a black box penetration test, the attacker will be unfamiliar with the assessed asset, which mirrors the everyday penetration attacks or "hacks". This will stimulate more accurate results, as they will not be privy to any additional information and would produce a relatively accurate indication of potential threats to your network [6].

3.6.3 Grey Box

Grey box is meant to illustrate the partial disclosure of information about the assessed asset. In between black box and white box testing, gray box testing is without any doubt the most realistic and accurate representation of the hacking activities as hackers will rarely attack an asset which they have no knowledge about and tend to gather a partial knowledge about the target from different sources in advance. Therefore, in this approach, the penetration tester will be given partial details about the network infrastructure and left with the obligation to gather enough information to initiate the testing [7-9].

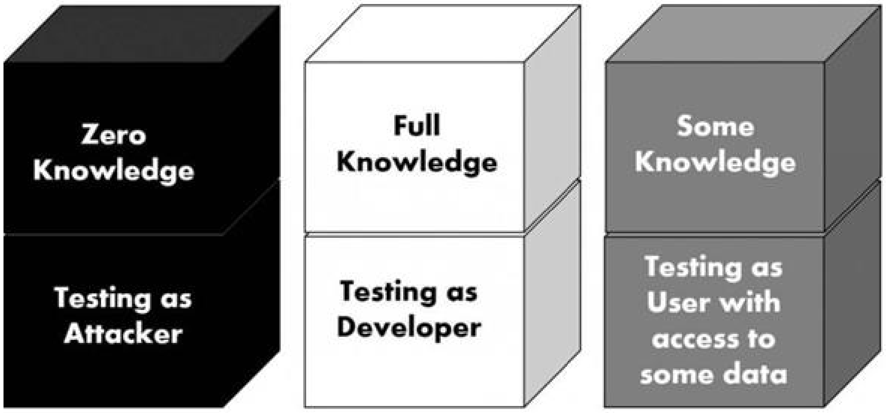


Figure 10: Difference between the three PT approaches; black-box, white-box and grey-box in term of input and ethical hacker starting position [17].

3.7 Testing Intensity and coverage

We distinguish two main testing intensity practice which is dictated by the amount of test to cover, namely: comprehensive testing also known as type 1 PT, and targeted testing referred to as type 2 PT.

3.7.1 Comprehensive penetration test

Comprehensive testing aims to identify and testing specific (not all) vulnerabilities and that the assessed asset (networks, systems, websites and web applications, or wireless networks) may be exposed to. This is often a medium load of work which requires a moderate effort and use of resources and combines both manual assessments (where systematic and logical thought processes, analytical thinking and skillful decision-making are required) with automated scans to assess the true extent of the assets vulnerabilities [10]. This type of testing is obligatory conducted by highly skilled penetration tester and should include a detailed report providing recommendations for fixing any spotted security breach and addressing each of the identified issues. Such testing provides with a good overview of an organisation's security posture and in most cases, is a faster and more cost-effective solution than the lengthier Level 2 Penetration Test. IT security governance bodies (CREST, NIST and SANS) recommend a Level 1 Penetration Testing that will identify exploitable vulnerabilities before they can be uncovered by an indiscriminate cyber-attack [12].

3.7.2 Targeted penetration test

Targeted testing requires an extensive effort and involves looking in details at all the potential vulnerabilities and explicitly attempt the relevantly associated exploits one by one to determine whether a successful exploitation is possible, and if so what kind of damage to the asset the attacker can do such as obtaining access to the sensitive resources. A targeted penetration test is a painstakingly detailed process of identifying security holes and vulnerabilities in your hardware and software (including printers, fax machines, workstations), systems or web applications and then attempting to exploit them. Due to the extent of these tests, Level 2 Penetration Tests often take several weeks to complete and are usually only recommended to clients who require a complex cyber-attack simulation [13].

3.8 PT versus Vulnerability Assessment

A vulnerability assessment (VA) is a series of automated scans and tests that provide a very high-level overview of the potential vulnerabilities and exposures that exist in the assessed system or network. Often, VA consist of providing overall and general insights of the system security and potential point of exposure which can be used by the attacker. On the other hand, PT plays a critical part in maintaining the security of a network by actively probing it for weakness and vulnerabilities. It mainly focuses on identifying, assessing the existing vulnerability, and attempting to exploit them by executing the adequate attack and exploits. Nowadays, PT involves, in addition, the comprehensive assessment and testing of the entire state of a network by emulating real adversaries (mimicking the attacker approaches), including the use of the attacker techniques, tactics, procedures, and the way they define the potential goals. PT practice requires a high level of expertise and in-depth knowledge of the systems and securities measure functioning along with the adequate resources [17].

3.9 The versatility of PT

The PT is characterized by its many domains of practicing and uniqueness in term of versatility of testing contexts and the level of expertise required for ensuring reliable results. For that, when we consider an intelligent automation if PT practice, we intend to cover the entire practice which present many challenges and is far different from other cyber security fields in term of operating modes and practicing environments where each of the contexts has its own uniqueness [15]. The PT is very versatile in term of testing vectors such as networks, databases, cloud infrastructure, web applications, wireless and IoT., and in term of testing approaches and methods. Figure 11, illustrates the versatility in the PT practice as discussed in out previous work [23].

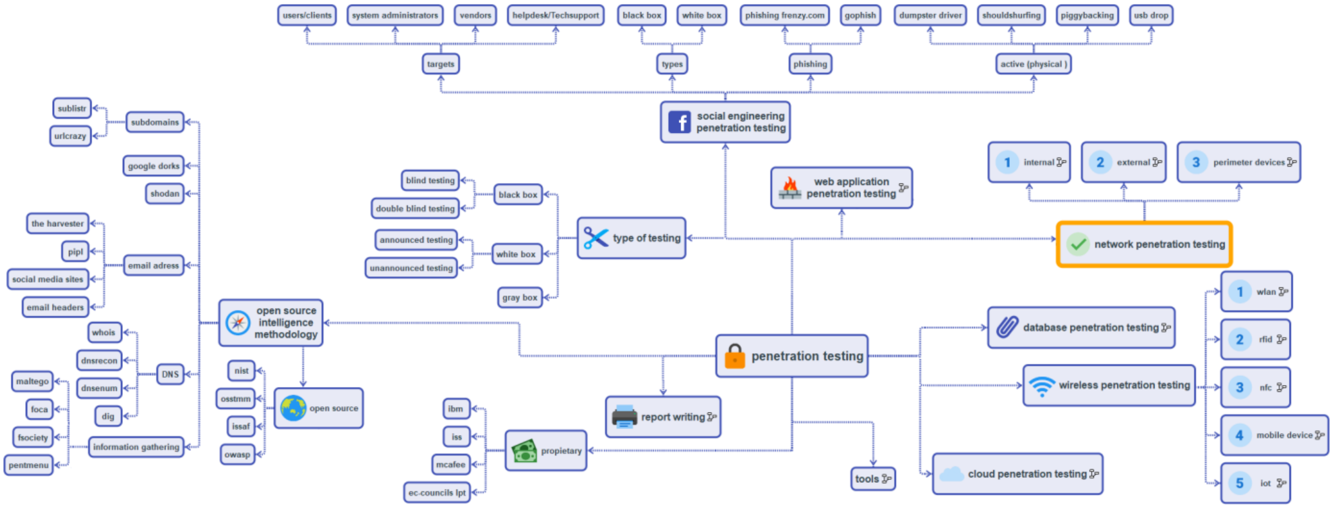


Figure 11: versatility in PT practice and richness of testing tasks, approaches and methods [23].

3.10 PT automation state of the art

During the last decade, cyber-security firms and security systems developers have been heavily focusing on producing automated and semi-automated PT frameworks and systems aiming to facilitate the work of network penetration testers and make the assessment of network security more accessible to non-experts. Multiple systems are available for public use varying from free and open source to more costly products. Popular products used in PT communities include Core-

Impact, Nexpose, Nessus, Qualys, Tenable, Immunity Canvas and Metasploit [18]. The main contribution offered by these systems compared with the traditional security and vulnerabilities scanners such as Nessus, is their functionalities (planning, scanning, and exploiting) along with simplicity and flexibility of use (automation of certain tasks, visualization, reporting). Yet, the offered automation (mainly related to the planning phase of the practice) remains limited to the planning of the practice, the organization of the tasks and the optimization/visualization (usually phase 1 and 2) and the automated reporting (phase 4). Nevertheless, the heart of the PT practice (phase 3) was often neglected or poorly exploited. In fact, determining the exploitable vulnerabilities and launching the relevant exploits, digging inside, and pivoting to create a new vector of attack is undoubtedly the most challenging part. The difficulty itself lies on the current PT systems which have radically changed and evolved and have become more complex, covering new attack vectors, and shipping increasing numbers of exploits and information gathering modules. Thus, the problem of efficiency has emerged and controlling alike framework successfully along with maintaining efficiency, is indeed the most important challenge [19].

3.11 Review of related research works

This research is rooted in a long line of research and development on network PT, starting with the planning phase for which work was undertaken and propositions were made with some already in existence in the industry, whilst others remain research ideas. As the penetration testing, automation, and enhancement (usually means optimisation and efficiency) domain is situated between both cyber-security and AI research fields, several axes of research were addressed starting with the consideration of attack graphs and progressed throughout different research fields and methodologies of Automated Planning and consequently the sub-area of AI. Early research, focused on the modeling penetration as attack graphs, planning attacks, and decision trees which all reflected the insight that researcher had relating to the PT practice as standard sequential decision making. In theory, these researchers were very encouraging and produced tremendous results although (real world implementation) most of the proposed work was more relevant to vulnerabilities assessment practice than to PT. This section of the literature review considers the most significant contributions in this regard and will summarise previously completed research with a special focus on the adopted approaches and the contributions. For clarity, we start by addressing the full picture of the research in this field (Figure bellow) and will proceed into dividing the research axes by type, methodology, and approach. Attack graphs are the most known security model for representing the chains of vulnerability exploits in a network first introduced by the work of [19] which proposed an attack graph that break down the environment of network vulnerabilities assessment tasks by modelling the space of possible attacks into atomic components called attack actions. Each action is described by a conjunctive pre-condition and post-condition over relevant properties of the system under attack [20]. This approach is closely related to the syntax of classical planning formalisms and thus later research used the classical planner to attempt to solve the problem by finding the optimized solution. An attack graph represents all known sequences of actions that compromise a system in form of a graph.

3.11.1 Regular and Blind Automation

An obvious way to produce a system or tool which autonomously performs PT is by automating all or some of the tasks and subtasks for each phase of the penetration testing practice. While automated PT tools are great for discovering low-hanging fruit, automatic vulnerability scanners should not be confused with advanced human expert testing which include a deep understanding of an organisation’s current security configuration and attempting targeted testing against specific high-profile areas of the application that will require more time and attention. Human testers rely on their expertise to find flaws that require logic and comprehension. Many organizations use automated tools and systems which blindly performs all the tests without any exception or pre-elimination. These systems are often guided by a skilled human tester, but many steps are automated such as vulnerability scanners to test multiple systems for the presence of vulnerabilities. The disadvantage of such an approach is mostly noticed in the medium and large network context where time, resources, generated traffic, and network congestion is too high, thus making the use of such approach limited to a small network or by writing scripts which are itself very inconvenient for expert point of view.

3.11.2 Graph-based approaches

An attack graph is undoubtedly the oldest method for planning security tests especially in a network context and was for a while used in penetration testing. In cyber-security, the attack graphs are particularly efficient in analysing security flaws in computer networks. In addition to the use in the defense part, several researchers tackled the problem of enhancing the PT practice by generating and optimising the attack graphs and met a lot of challenges. Some methods were only theoretical and not applicable for practical scenarios where other methods produced some consistent results by reducing the number of scenarios, attack vectors or even finding new attacks paths which a human may not notice or just omit.

An algorithm to automatically generate a penetration graph was proposed by (Qiu et al., 2014) which optimises the network topology before generating the penetration graph which helps to reduce the amount of redundant information and provides a clear overview of the networks' possible security flaws by exploring all possible attack paths from the raw information gathering output, optimising the reachability matrix (network inter-connections) and highlighting the most critical attack paths. The researchers pretend to be able to generate multi-path correctly and effectively in all situations.

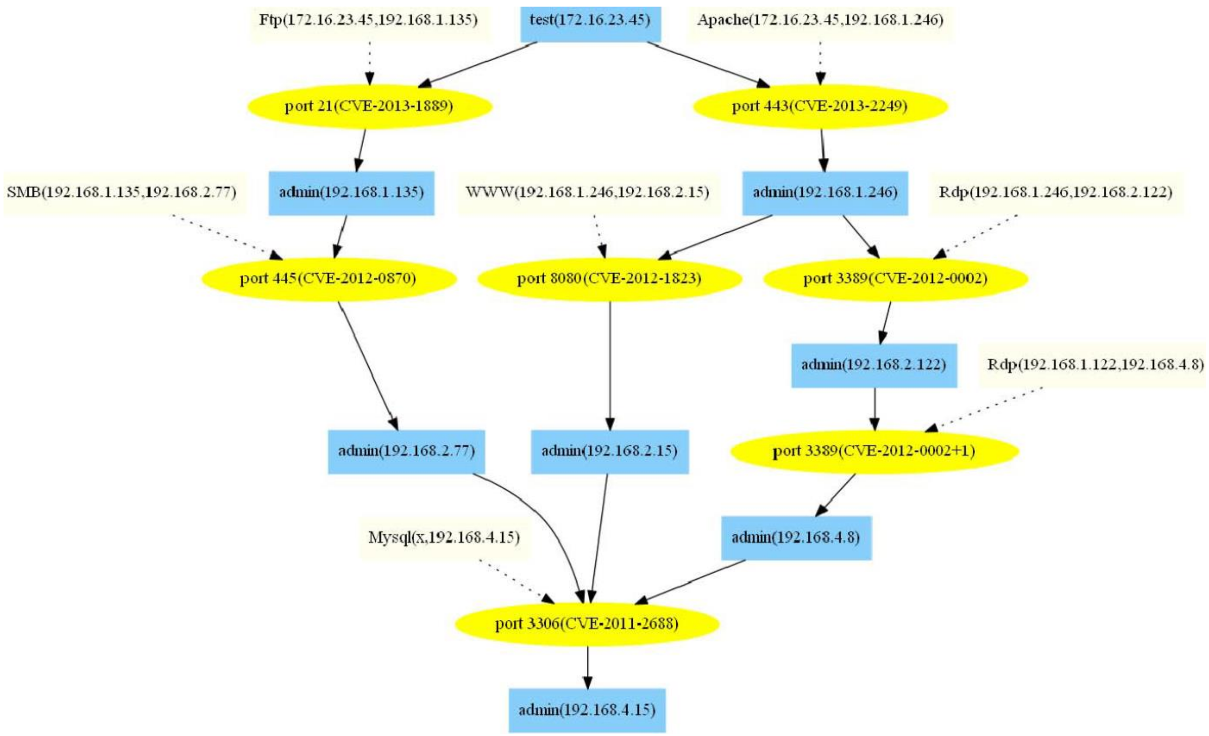


Figure 12: An example of an automated generated penetration (attack) graph

In a previous research work entitled "An Intelligent Technique for Generating Minimal Attack Graph" [24], an approach based on planner has been proposed for time-efficient scalable representation of the attack graphs. The planner itself relies on a special purpose search algorithm from an AI domain to determine the best solutions within a large state space without suffering state space explosion. In addition, another research performed by [25] where he assumed the action in the attack graph by assigning two values: action cost and success-probability. He then proposed an algorithm for computing an action selection policy which minimises the expected cost of performing an attack. Finally, he modeled the problem as a finite-horizon MDP problem and used forward search, transposition tables and pruning techniques (enhancing the structure of the attack graph). As a result, he compared the obtained results using the proposed approach to a generic MDP solver along with a solver transforming the problem to an unconstrained influence diagram and demonstrated the performance enhancement (reducing the runtime) [26].

In a PT context, attack representation is not a hierarchy (tree) but often a directed acyclic graph (as the practice diagram is in reality) due to the repetitively and interrelation between nodes which means performing an action can be beneficial in several possible branches at once if the action has more than one root-node path in the attack graph. This results in the probability of the action being counted multiple times and the overall probability of success is increased. This causes the expected cost (computed as first probability) or penalty to decrease but remains admissible so can be added on later [22]. Since a minimised value is compared with the expected cost this issue keeps the heuristics still admissible.

3.11.3 Classic and Intelligent Planning Approaches

The automation of the planning phase in PT practice was the first approach to be tackled by researchers for the simple fact that it was very close to the attack graph for security auditing and vulnerabilities assessment. This approach encloses different planning mechanisms that automatically finds a plan when given as input [24]. A high-level description of the relevant world properties (the state variables), the initial state, a goal condition, and a set of actions where each action is described in terms of a pre-condition and a post-condition over state variable values. In classical planning, the initial state is completely known, and the actions are deterministic, so the underlying state model is a directed graph (the state space), and the plan is a path from the initial state to a goal state in that graph [25].

This grounds PT in a well-researched formalism, highlighting important aspects relating to the nature of this problem. This approach was initially followed by [24-25] attempting to make strong independence assumptions for the sake of scaling and lacks a clear formal concept of what the attack planning problem actually is. The founding motivation for applying automated planning mechanisms in a PT context is facilitating and assisting the decision-making process within manual or autonomous systems [26]. The AI planning techniques' generality of concepts and models make them applicable to the diverse context where a control problem emerges and network security assessment and PT are undoubtedly a natural application [23]. The latter investigates the idea of

AI and optimised attack planning as an approach for automated security assessment starting with [27-28] works.

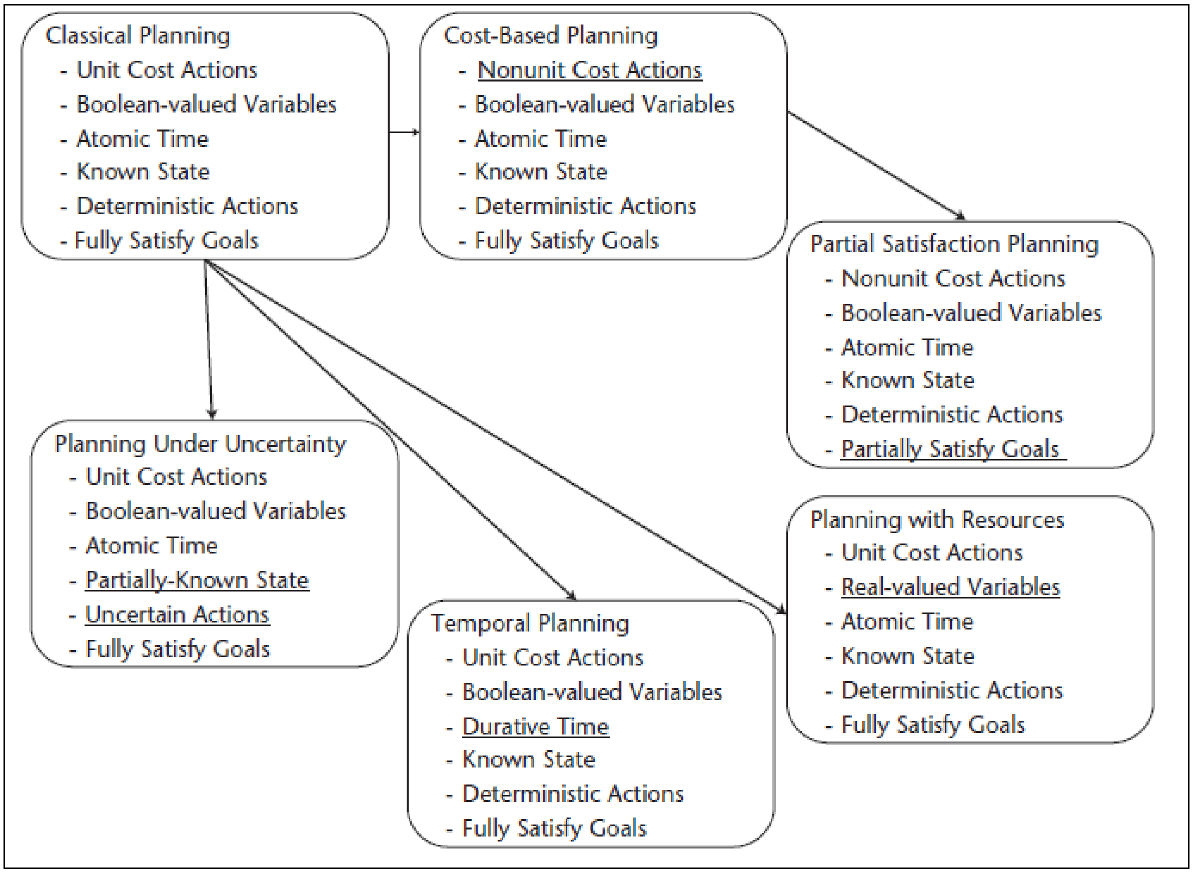


Figure 13: Different planning approach previously investigated for PT context [28].

Subsequent planning for PT using attack graphs techniques was proposed and will be detailed in this section. However, most of the proposed solutions lacked efficiency as blind automation did not tackle the problem related to optimising the penetration itself by validating and executing the attack paths resulting from the analysis phase. A brilliant piece of research [29] introduced a complete PDDL representation of the attack problem along with a practical implementation that integrates a planner into a PT tool. Their research allowed an automated generation of attack plans (single and multi-paths) for real world PT scenarios, and to validate these attacks by executing the corresponding exploits against the real target network [28]. The idea was executed using an algorithm for transforming the information acquired during the information gathering phase of the PT into the planning domain, and they showed how the scalability issues of attack graphs can be solved using the proposed model for small and medium sized networks. Nevertheless, the proposed approach is very limited in a large network context and fails to cope with dynamic environment characterising the cyber security domain [30]. Related works carried out by [30-32] highlighted the failure of the classic modeling approach in dealing with the uncertainty in the domain of PT, especially the lack of accurate and complete knowledge about the assessed system which the classical planned requires producing a plausible result. The researcher worked on an industrial system considered to be one of the market leaders. In the vulnerabilities assessment (Core-Impact) modelled the problem as partially observable Markov decision processes (POMDP) and integrating for the first part the first phase (information gathering) of the practice within the remaining mechanisms by utilising the particularity of POMDP in terms of the open and flexible "Initial Belief State". This work provided a first intelligent PT system with a mix of scanning, planning, assessing and exploiting. There were two major flaws in this reasoning; the information gathering (scans and enumeration) does not yield perfect knowledge so a residual uncertainty remains, and the significant impact of the scanning on the assessed network as it often generates large amount of traffic susceptible to be detected by security system and requires a long running time. They wanted a technique that (like a real hacker) can deal with uncertainty by intelligently inserting scanning actions where they are used for scheduling the best exploits. The proposed model itself is questioned as it obviously fails to model the full picture of PT and focusing rather on the separate entities [31].

Similar research work carried out by [32] adopted an intelligent planning approach to generate a minimal attack graph described as a time-efficient scalable representation of the attack graphs. A planner is a special purpose search algorithm from an artificial intelligence domain, used for identifying solutions within a large state space without suffering state space explosion [33]. A case study has also been presented and the proposed methodology is found to be more efficient than some of the earlier reported works [34]. In another related research, introduced a game-theoretic model for anticipating and preventing attacks in a computer network by representing the network security components and the associated interactions [35]. This approach remains very limited in covering the entire scope as a huge gap will always exist between a network administrator who uses the limited resource to secure the network and an attacker who is flexible and can adopt hard to prevent complex and multi-stage attacks which are aconsidered to be NP-hard problem than the solving rquire huge amount of time when it is large graphs [36]. Later, [37] presented another approach which first translates an attack graph into an MDP and solves it using policy search with a set of pruning techniques. This solution fails too in complex real-world scenarios where the search for the policies is more expensive in terms of time rather than establishing it.

Furthermore, the attack graph techniques have also been used for the analysis of threats that arise through the possible combinations of cyber-attacks and related actions which then modeled as planning optimization in [33-36]. Other previous work [36-38] used classical planning and hence ignores all the incomplete knowledge that characterises the information gathering and vulnerability assessments. More recent work [32] proposed an independent point of view PDDL language modeling of the attack planning by considering each asset (machine or system) separately. These works were further enhanced by [33-35] and introduced models that considered the uncertainty in the PT practice at the action level (machine alone context) was introduced in form of POMDP planning problem. In fact, the last work the uncertainty was introduced into a POMDP model both at the initial belief state and the probabilistic action outcomes as in [37-38]. The first contributes to considering the real dynamics of the system when the second enhances the choice of the relevant exploits to test. Unlike the MDPs approach, POMDPs allow to model information gathering as an integral part of the problem, thus providing for the first time a means to intelligently mix scanning actions with actual exploits.

In applied research works conducted by [39-41], researchers tackled the problem of automatically designing an efficient plan for remote PT with no prior knowledge of the target network's machines, they proposed a model for generating and executing remote testing plans that considers the uncertainty of using remote tools both to gain knowledge of the system and to provide the PT actions. Our solution provides automated generation of multi-step penetration test plans that are robust to uncertainty during execution. We tackle this problem by making use of modeling techniques from POMDPs. The researcher attempted to automate this process by taking advantage of efficient solutions for solving POMDPs, and further, automatically derive these models through automated access to vulnerability databases such as the national vulnerabilities database (NVD). We demonstrate our implemented solution on a series of example problems.

Recently, important research was carried-out by [44-46] which tackled the attack mitigation issue as part of the PT practice by adopting a what-if approach for conducting a comprehensive analysis of the different mitigation strategies on a simulated PT platform. The work used automated attack generation based on the adversary network model which analyses the different mitigation actions such as changes to the network topology, system updates, configuration changes, etc. They aimed to determine optimal combinations that minimise the maximal attacker success and proved the efficiency of the proposed what-if analysis approach which can be derived from network scan, public vulnerability databases and manual inspection with various degrees of automation and detail. They also used a simulated PT approach allowing automated attack-finding when different network changes were applied (topology, system updates, configuration changes, etc.). As a result, they noticed that using this technique they were able to determine the optimal combinations that minimise the maximal attacker success and therefore proposing the best mitigation strategy. This work covers mainly the mitigation phase, which is out of the scope of our research, nevertheless the simulation part will be considered as an option later in our research.

3.11.4 Expert-System and Knowledge-Based Approaches

In this section, we will provide an extensive description of expert system (ES) and knowledge-based expert system (KBES) as part of this research will rely on these technologies. ES or KBES is a system which solves specific types of problems by codifying human experts' knowledge in a knowledge base, and by mimicking the human problem-solving process. Technologies use qualitative knowledge rather than mathematical models provides the needed support. In fact, it is a computer system that applies reasoning methodologies to knowledge in a specific domain to render advice or recommendations, much like a human expert. A computer system that achieves a high level of performance in task areas that, for human beings, require years of special education and training. The basic concepts of ES include:

- How to determine who experts are (level of experience and knowledge)?
- How expertise can be transferred from a person to a computer?
- How the system works in term of reasoning?

As in any other domain, an expert is a human being who has developed a high level of proficiency in making judgments in a specific, usually narrow, domain. The penetration testing was not considered as a good candidate to apply these approaches but similar work on defensive cyber-security demonstrated the feasibility and the efficiency of these approaches in solving the complex and large problem along with capturing the human expertise and modeling it for future use.

3.11.5 Statistical and heuristic approaches

Purely mathematic techniques, such as statistical and heuristics were the first to be proposed for improving the PT system performance and accuracy. They rely on simplistic, easily evaded data processing which utilises several variants of algorithms to attempt identifying a mathematical solution to the problem. Research work in this regard was earlier presented by [55-56]. The researchers proposed the use of certain criteria to evaluate individual testing steps and the non-deterministic behavior of the tested system. They considered six heuristic algorithms based on these ideas and implemented four of them having a game-like approach to imitate the black-box PT [57]. The algorithms were compared by measuring the number of testing steps required for finding or spotting a vulnerability in the assessed system [58].

Academic work by [59] surveyed the use of statistic and heuristic-based approach to enhance the automated PT system performances. They were concerned with the planning and the scale-up challenges in the previously proposed approaches and their impact on performance. The research focused on the development of a powerful reachability extraction system based on heuristics along with a planning graph, which is an interesting means to obtain informative look-ahead heuristics for search and has become ubiquitous in state-of-the-art heuristic search planners. They proposed heuristics classical planner allowing a flexibility to adapt to more expressive scenarios that consider action costs, goal utility, numeric resources, time, and uncertainty.

The heuristic approach was again used, although differently, by [60] to find optimal policies to execute when using attack graphs. The work is based on two major criteria: action failures and costs. They designed a heuristic approach to compute the lower bound of the expected cost of an attack graph by setting costs of the actions to zero and considering only the actions' probabilities, enabling more freedom in action ordering as any (valid) ordering will produce the same probability of success of the policy and thus the overall expected cost. They used the produced heuristics to order the action in which compute remaining actions' expected costs by prioritizing the most promising action. Although the principle of ordering action used is scientifically correct, the context of attack graph (PT) creates an issue since attack representation in form of AG is not always in form of hierarchy (tree) but instead often a directed acyclic graph. In addition to this, the possibility of attack or exploit failure makes finding the optimal sequence of attack action non-trivial. These particularities cause expected cost (probability) or penalty to decrease but remains admissible, therefore the research concluded that performing an action can still be beneficial in several possible branches at once if action has more than one root-node path in the attack graph. Ultimately, this will result in the probability of the action being count multiple times into the overall probability of success is increased.

3.11.6 Artificial Intelligence and intelligent planning approaches

Very few research works tackled the potential intervention of AI in the offensive cyber-security domain in general and PT in specific. In fact, ML was widely used within the cyber-defense community in designing security systems able to learn and act alone on a real-time basis without referring or waiting for human (such as an incident responder) decisions or approval. To the best of our knowledge, ML capabilities with advanced algorithms that can adapt and learn along with probabilistic mathematics for learning the patterns was never considered for the domain of PT. These approaches were widely adopted in defensive cyber-security such as intrusion detection and anti-viruses in which systems inspired from the human immune system were developed to identify and responds to cyber threats autonomously without referring to a cyber defense expert or professional to make decisions.

Excluding the work carried out by [14-15] and [17] which were mostly oriented on planning for PT rather than intelligent practice (full process), the AI intervention was considered neither by industry developers nor by researchers. A system named DIRP was proposed in which a standardized automated cyber-attacks emulation was imagined possible based on an intelligence automation model and data interoperability by fusing information from multiple freeware programs that can be thought of as cyber sensors into an interoperable, robust system in a manner that can tailor itself and learn over time [29-30].

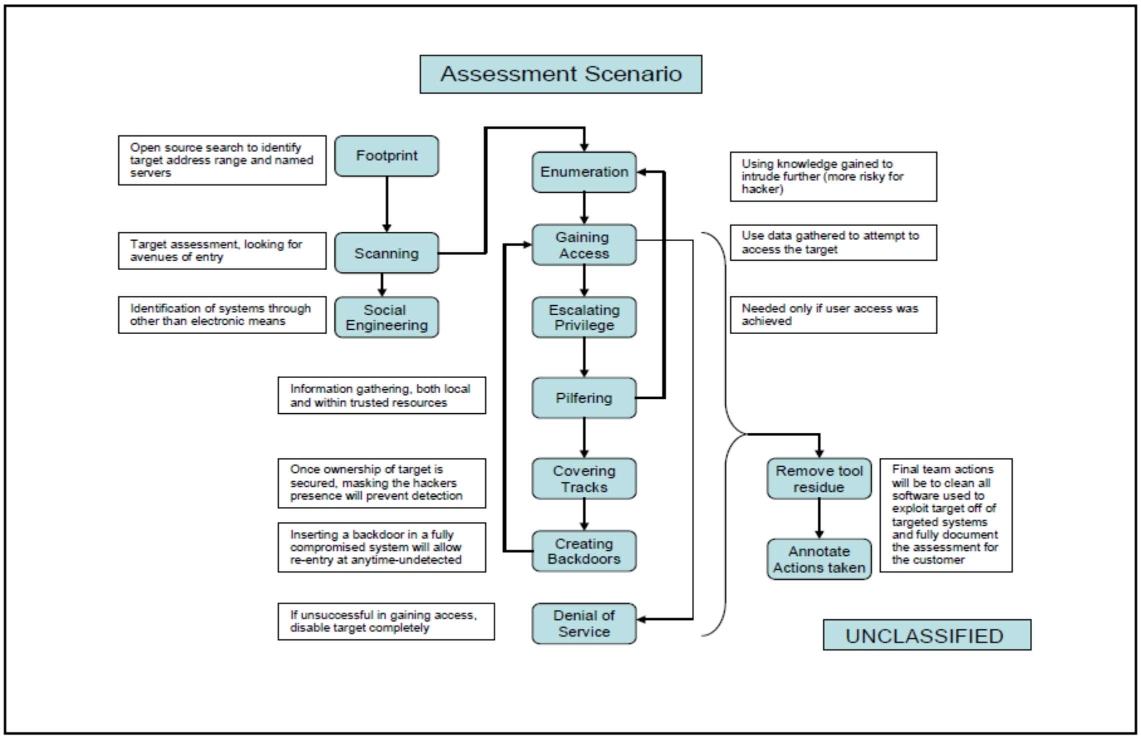

Assessment Scenario
Open source search to identify target address range and named servers
Footprint
Target assessment, looking for avenues of entry
Scanning
Identification of systems through other than electronic means
Social Engineering
Enumeration
Gaining Access
Escalating Privilege
Pilfering
Covering Tracks
Creating Backdoors
Denial of Service
Information gathering, both local and within trusted resources
Once ownership of target is secured, masking the hackers presence will prevent detection
Inserting a backdoor in a fully compromised system will allow re-entry at anytime-undetected
If unsuccessful in gaining access, disable target completely
Using knowledge gained to intrude further (more risky for hacker)
Use data gathered to attempt to access the target
Needed only if user access was achieved
Remove tool residue
Annotate Actions taken
Final team actions will be to clean all software used to exploit target off of targeted systems and fully document the assessment for the customer
UNCLASSIFIED

Figure 14: related work which propose intelligent DIRP system working diagram [61].

During latest work carried by MITRE Corporation (Applebaum et al., 2016) which tackled the problem of making the network security red teamwork autonomous and intelligent, which would play a critical part in assessing the security of a network by comprehensively and accurately probing it for weakness and vulnerabilities. Unlike vulnerability assessment, which is typically focused on identifying vulnerabilities, PT automation aims to effectively assess the entire state of a network by emulating and launching real adversaries, including their techniques, tactics, procedures, and goals. Unfortunately, executing a comprehensive testing is prohibitive: cost (time and exploits' fees) and impact on the network making it useless in a real-world situation. We seek to solve this problem by creating a framework for an intelligent automation of the PT by focusing on every part of the practice along with the post-compromise scenarios (after the perimeter has been breached or a machine has been controlled).

Additional work proposed solution which act autonomously and self-learn as a human tester will do, the machine learning enabled the system to learn independently, execute the learned policies and actively move through the target network using the acquired results to look for further weaknesses and not only limited to an automated planner designed to accurately reason about future plans in the face of the vast amount of uncertainty in red teaming scenarios. Our solution is custom developed, built on a logical encoding of the cyber environment and adversary profiles, using techniques from classical planning, Markov decision processes, and Monte Carlo simulations [54]. In related work, researcher proposed a framework which focus on planning and reported that they have been able to successfully validated our planner against other techniques via a custom simulation. The tool in question has been deployed to identify vulnerabilities and is currently used to train defending blue teams [66].

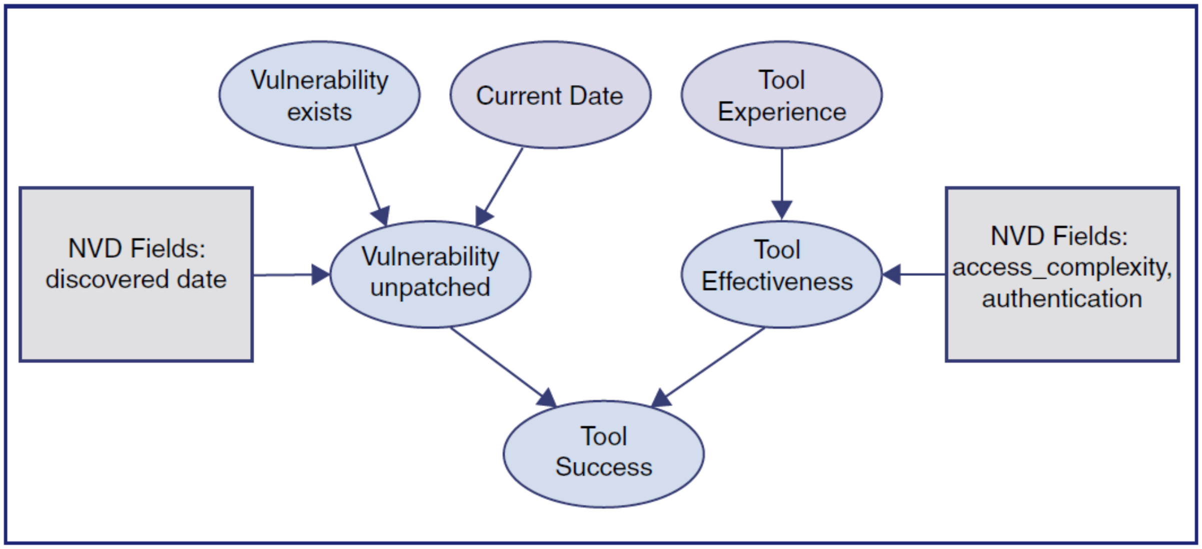


Figure 15: automated planning for remote penetration testing framework proposed by LGS Innovations-Bell Labs.

Other approaches relied on YARA rules which are pieces of programming language working on defining several variables that contain patterns found in a sample data and if some or all the conditions are met, depending on the rule, then it can be used to successfully attack vector identification. This approach is very limited to strict small number of attacks with a pre-defined structure [46].

3.11.7 Machine Learning approaches

The first attempt to optimize PT automation using machine learning methods was introduced by MIT researcher in form of ML module named AI2 that learn and replay attack planning from set of data input and processed accordingly [49]. In parallel, a similar approach was put forward by Core Security researchers [50] and implemented in their commercial PT system Core Impact with an addition of RL modelling from a previous research work, the proposed approach is basic and limited to small assets and attempt to capture security variables and solves scenarios based on the probability of success of actions and generated traffic. No mention of the variables and timing allowed for such small scenarios and the proposed software was tested in practice on a 50-machine network and fails fundamentally to address the required points namely test coverage and producing results within the allocated testing time frame. In related work, [53] attempted to automate PT by mitigating vulnerabilities and countermeasures by conducting comprehensive what-if analyses. A conceptual framework is provided to reason about mitigation actions applied to a network model. The approach determines optimal combinations that minimize attacker success following a holistic mitigation strategy [54].

In a recent work, [51] attempted to apply RL to solve capture the flag (CTF) scenarios. Fundamentally, CTF competitions are very specific scenarios which do not account for many variables in typical PT, but the study was significant for investigating the relevance of different RL techniques. An additional noteworthy proposal aimed to automate exploitation and post-exploitation by combining deep RL and the PowerShell empire post-exploitation framework [52]. RL agents pick a PowerShell module and use its internal features as action states and then compare the learning progress of three RL models: A2C, Q-Learning, and SARSA. The results showed that A2C is the most efficient and trained agent can eventually obtain the admin privileges of the domain controller system [54]. Another research investigated the application of model-free RL to pentesting and tested the standard Q-learning algorithm using both tabular and neural network solving implementations. The output was that both tabular and neural network implementations were able to find optimal attack paths for small size networks but fails to achieve acceptable results in medium and large networks due to the scalability problem for model-free representation which results in huge environments [55]. Finally, research focused into the automation of PT using deep learning which proposed a two-step approach: first using the Shodan search engine to collect relevant server data in order to build a realistic network topology, and second employing multi-host multi-stage vulnerability analysis to generate an attack tree for that network. The researchers employed deep Q-learning network (DQN) methods to discover the easiest to exploit attack path from the possible candidates out of thousands of input scenarios, and DQN and enabled optimal path discovery with an accuracy of 86%. Again, this approach is relevant for simple network and the second DQN approach fails above a certain network size [57-58].

Overall, we elaborated this graph summarises the research undergone and reading and summarizing the output of related works. We categorised the previous work on automated PT under 3 main approaches: blind or regular automation, targeted or script-based automation and optimized automation which the AI approaches fall under. Figure bellow categorises reviewed research works under each of the categories.

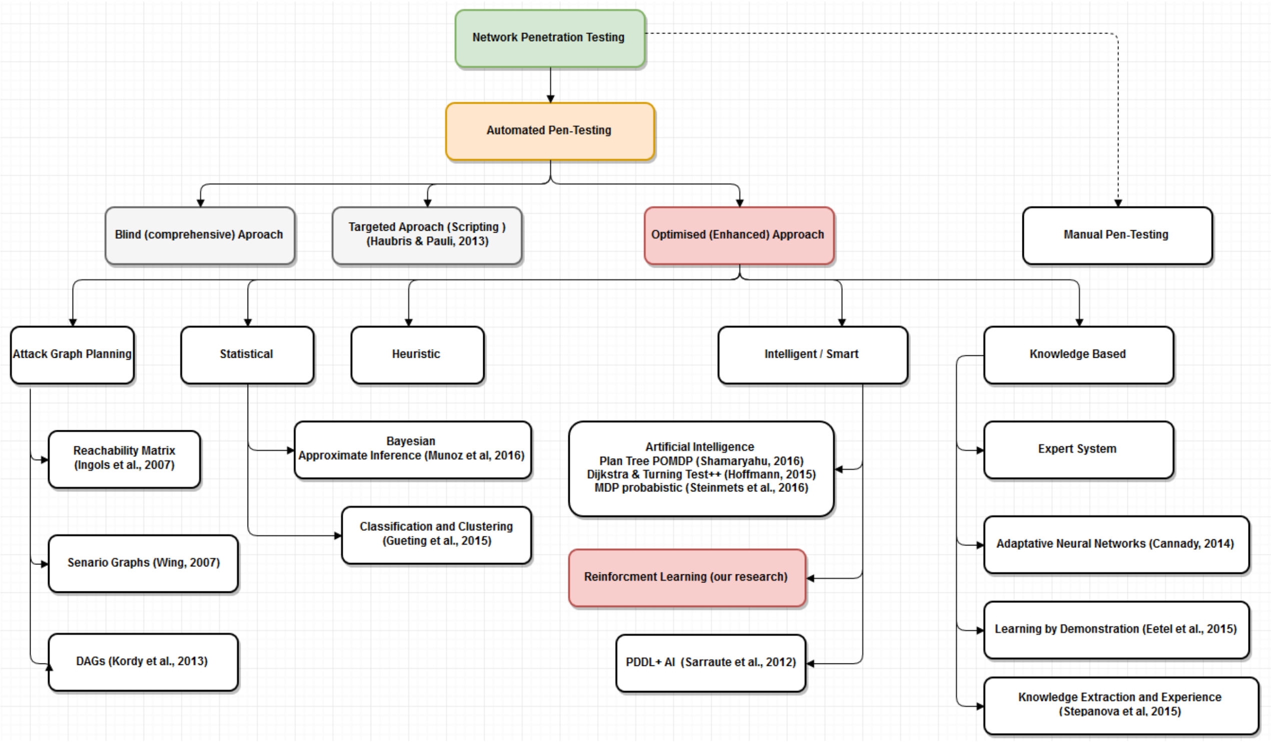

Network Penetration Testing
Automated Pen-Testing
Blind (comprehensive) Aproach
Targeted Aproach (Scripting)
(Haubris & Pauli, 2013)
Optimised (Enhanced) Approach
Manual Pen-Testing
Attack Graph Planning
Statistical
Heuristic
Intelligent / Smart
Knowledge Based
Reachability Matrix
(Ingols et al., 2007)
Senario Graphs (Wing, 2007)
DAGs (Kordy et al., 2013)
Bayesian
Approximate Inference (Munoz et al, 2016)
Classification and Clustering
(Gueting et al., 2015)
Artificial Intelligence
Plan Tree POMDP (Shamaryahu, 2016)
Dijkstra & Turning Test++ (Hoffmann, 2015)
MDP probabistic (Steinmets et al., 2016)
Reinforcment Learning (our research)
PDDL+ AI (Sarraute et al., 2012)
Expert System
Adaptative Neural Networks (Cannady, 2014)
Learning by Demonstration (Eetel et al., 2015)
Knowledge Extraction and Experience
(Stepanova et al, 2015)

Figure 16: summary of previous research work on PT automation in form of tree of approaches [23].

To conclude this chapter, a good amount of literature had been produced and been surveyed in this research after being organised in a scientific structured way that straightforward categorized literature review from traditional to the offensive cyber security notably VA and PT [59]. We elaborate a summarized list of the identified issues and shortfalls of previous related works as well as the limits of the proposed solutions summarized into the following points:

- ➢ Computer-generated plan and attack graph based on static data would isolate the tester from the complexity and dynamicity of the real-world network security and therefore make the accuracy and pertinence of the results very limited to the context in which the tests were made in and negatively impact the security posture of the organisations.

- Suitability of the proposed models and representation into dealing with versatile networks' topologies, configurations, and securities architectures along with the dynamic context, usually the proposed solutions were limited into efficiency by many assumptions.
- The continuous and inevitable need for a human expert supervising or controlling the system and making crucial decisions results in the proposed system being less accessible to non-experts as well as the lack of optimisation in the use of the knowledge gained during previous testing practices.
- The eventual compatibility of the proposed solution to be incorporated or embedded within the industry's PT systems and framework, are mostly planning solutions and eventual implementation will leave the problem of fully automation unsolved.
- Blindly automated PT indeed solves the problem of human labor but creates an even bigger problem of efficiency as the required time will increase sharply and often goes beyond the limits. The generated traffic will also create additional problems related to network congestion, security detection, and downtime which are all undesirable problems for any prudent network security professional. Attacks allow the user to perform reproducible tests, which opens the path to computing security metrics whose evolution would be a key indicator of the security posture of organisations.

Chapter 4: Reinforcement Learning

Reinforcement learning (RL) is a branch of machine learning that allows a system software agent to automatically determine the ideal behavior within a specific context by interacting with its environment and receive rewards for actions performed with the aim of maximizing its performance. RL provides a conceptual framework to address a fundamental problem in AI: the development of situated agents learning how to behave while interacting with the environment. The problem is formulated as an agent-centric optimisation in which the objective is to choose actions that result in the highest possible reward, in the long run, resulting and therefore learning the optimal actions to take from each state [60]. Over the last decade, RL has exerted a seismic influence on planning and sequential decision-making problem solving and cognitive science in general. Whereas it was initially treated as a repository of specific computational techniques, RL has become a general framework for thinking about motivating behavior and learning in humans' expertise in many fields. With the increasing popularity among AI researchers, the RL role has broadened and gradually moved beyond a classical gaming and robotics view of RL, tackling many fundamental real-world problems [61-62].

RL ultimately aims to determine optimal actions' selection policies in sequential decision-making processes as it deals properly with delayed rewarding. The general framework is based on sequential interactions between an agent and its environment, where the environment is characterized by a set of states and an agent that executes actions from the set of predefined actions. The agent interacts with the environment and transitions from one state to another following the execution of actions. The sequential decision-making process, therefore, consists of a sequence of states and a sequence of actions during a period starting from an initial state and finishing at a terminal state. For each transition or observation related to action taken and resulting in progress from one state to another, a reward (also called feedback) is allocated to the agent which the value is either positive (rewarding) when pursuing actions that lead to beneficial outcomes, or they may be negative (punishing) when pursuing actions that lead to worse outcomes. RL enable an autonomous computer system to learn from his own experience using the received rewards and punishments resulting from the performed actions. The RL agent is trained to take actions, giving the current environment parameters, that maximise a cumulative reward and therefore using trial and error to explore his environment [63]. A Markov decision process is one of the mathematical formalisms widely used to implement RL algorithms. The relevant components of this formalism are the state space, action space, transition probabilities, and rewards [64].

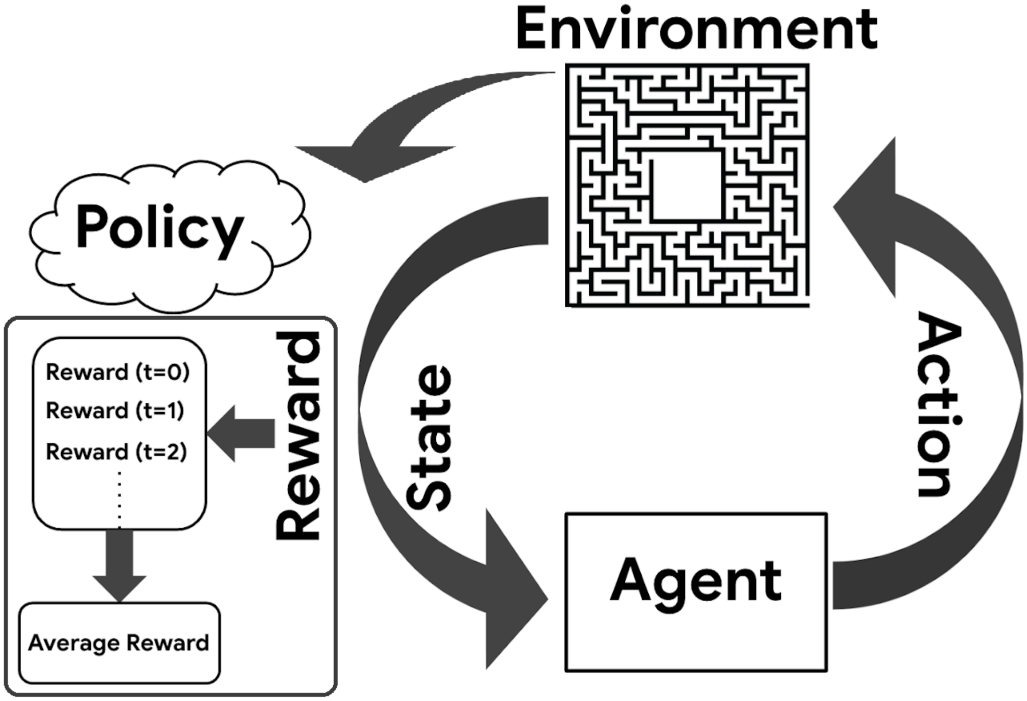


Figure 17: RL paradigm, the agent learns the optimal policy which represents a map of actions that lead to the greatest cumulative reward [64].

Figure 17 illustrates 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 a reward, from the environment that indicates the utility of the action [62]. In practice, there are many different formulations of RL are also known as approaches with each having different algorithms and implementation. As a matter of fact, RL is defined by a specific type of problem, and all its solutions are classed as RL algorithms although all require that the agent makes the best decision (action) possible based on his current state. A different approach has been developed to either accommodate various types of environments or to utilize slightly different learning mechanisms [65].

4.1 Reinforcement Learning

RL comes as a natural choice for solving real-world problems requiring sequential decision making when a human expert performs several tasks that depend one on the other in a very strict order. It is more relevant to the cases when this expert works with incomplete knowledge. Therefore, partially observable Markov decision processes (POMDPs) provide a natural representation for sequential decision tasks under uncertainty and particularly network PT. This representation roots into a well-established model of Markov decision processes (MDPs) [66] and is distinct by the feature of allowing the RL agent to act even when it fails to identify the exact environment state. The POMDP formalism is reputed to be efficient and powerful which helped to extend MDPs application to several real problems [67]. PT is a sequential decision process in which PT experts interact and perform actions and wait for the outcome. As with any sequential decision process automation using RL, it is expected to involve an agent that interacts synchronously with the external environment or system; the sequence of system states can be modelled as a stochastic process [68-69]. The agent's goal is to maximize reward by choosing appropriate actions. These actions and the history of the environment states determine the probability distribution over possible next states [66].

4.2 Markov Decision Process

The easiest way for modeling a sequential decision problem is the famous Markov decision process (MDP) model. MDP is an extension of Markov chains with a set of decisions (actions) and costs (reward) structure. For each state of the process, a decision must be made regarding which action should be taken. The chosen action affects both the transition probabilities and the costs (or rewards) incurred. The goal is to choose an optimal action in every state to increase some predefined measure of performance. The decision process for doing this is referred to as the Markov decision process [64]. In MDP, a state is a description of the agent's environment position at a particular point in time. Although we will deal with continuous state and action spaces when describing large problems, we generally assume that the environment can be in a finite number of states, and the agent can choose from a finite set of actions. Let S = $\{S_0, \ldots.S_N\}$ be a finite set of states. Since the MDP is stochastic, a particular state at any time $t \in$ T, can be viewed as a random variable S_t whose domain is the state space [70].

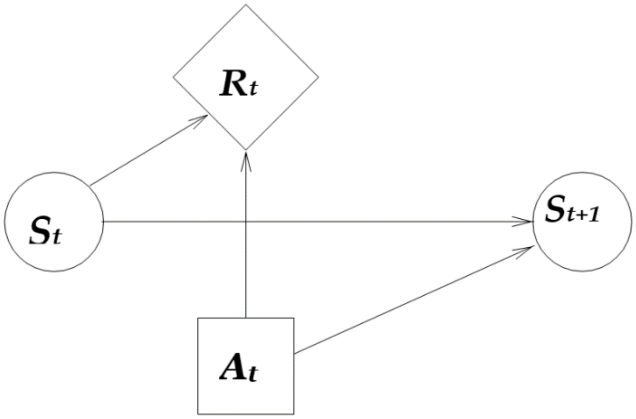


Figure 18: MDP relationships between states, actions At, rewards Rt received at stage R(St, At) and progressing to state St+1.

For a process to be Markovian, the state has to contain enough information to predict the next state. This means that the past history of system states (earlier than the current state) is irrelevant to predicting the future [71].

$$Pr(S^{t+1}|S^0, S^1, \ldots, S^t) = Pr(S^{t+1}|S^t)$$

At each stage, the agent can affect the state transition probabilities by executing one of the available actions. The set of all actions will be denoted by A and for each action a ∈ A is described by S × S state transition matrix, whose entry in an ith row and jth column is the probability that agent will move from state s_i to state s_j if action a is executed [72].

$$p_{ij}^a = Pr(S^{t+1} = s_j|S^t = s_i, A^t = a)$$

When we assume that the MDPs are stationary and thus the transition probabilities do not depend on the current time or system's states. The transition function *T:* S x Ai → Δ (S) summarises the effects of Δ (S) which is a function that for each state and action associates probability distributions over the states space S. Thus, for each s and a, the transition function T will determine the probability of transiting from a state to another when executing a specific action as follow:

$$T(s, a, s') = Pr(S^{t+1} = s'|S^t = s,\, A^t = a)$$

Finally, Giving *R*(*s, a*) is an immediate reward that MDP agent would receive for executing action a while being in state, the reward function that for each state and action is as follow R : S x A → R [66].

4.2.1 Partially Observable Markov Decision Process

In this section, we formally introduce the POMDP model and related decision-making concepts. The Markov decision process (MDP) is a modelling variant for solving an RL problem which entails sequential decisions. As with MDP, the goal of the agent is to act in such a way as to maximise some form of expected long-term reward. There are many cases where the Markovian assumption may not be valid [73]. Such cases include those where the agent either cannot perfectly observe the state information, in which case the problem is referred to as a partially observable Markov decision process (POMDPs) [74], or if there is a long temporal dependence between states and the feedback provided. Approaches to solving these types of problems often include retaining some form of the state history [63-67], such as by using recurrent neural networks or a more complex variant that relies on long short-term memory (LSTM) [71].

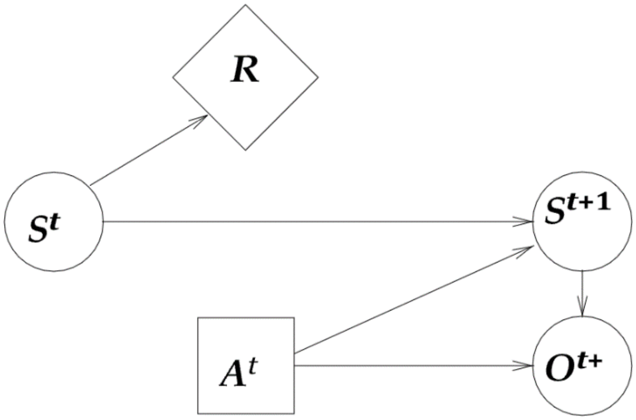


Figure 19: POMDP relationships between states, actions, rewards, and observations.

POMDP shares many elements with the fully observable MDP and in practice is a tuple <S *, A ,T, R, O , Z>*, consisting of the exhaustive set of possible state space S , the exhaustive set of possible actions that can be taken space A , transitions or observations probabilities of transitioning between the various states given actions *T* and *O*, reward function *R* and observation function *Z*. POMDP's ultimate objective is to develop a policy which is a graph mapping actions to state aiming to produce the greatest possible cumulative rewards. Nonetheless, what distinguishes a POMDP from MDP is that the agent now perceives an observation o ∈ Ω, instead of observing states directly. The set of observations Ω = {o_1, ..., o_n} represent all possible observations the agent can receive. The observation is therefore conditioned by state s, action a, and follow the observation function O : S ×A× Ω → [0,1]. The probability of observing o in state s_0 after executing a is O (s_0, a, o). Note that, for O to be a valid probability distribution over possible observations it is required that s∈S, a∈A, o∈Ω, O (s_0 , a , o)≥ 0 and $\sum_{o\in\Omega}$ O(s_0 , a , o) = 1 [68].

4.2.1.1 Observation function

A POMDP is comprised of an underlying MDP, extended with an observation space O and observation function *Z*. Let O be a set of observations an agent can receive. In MDPs, the agent has full knowledge of the system state. In partially observable environments, observations are only probabilistically dependent on the underlying environment state [23]. Determining which state the agent is in becomes problematic, because the same observation can be observed in different states. Z: S x A →Δ(O) is an observation function that specifies the relationship between system states and observations [71-74]. Z (s,a,o) is the probability that observation o will be recorded after an agent performs action a and lands in state as:

$$Z(s', a, o') = Pr(O^{t+1} = o' \mid S^{t+1} = s',\ A^t = a)$$

4.2.2 Process histories

History in POMDP stand for the log of everything that happened during the execution. Thus, POMDP complete history from the beginning until the time t is a sequence of triples:

$$\langle S^0, O^0, A^0\rangle, \langle S^1, O^1, A^1\rangle, \ldots, \langle S^t, O^t, A^t\rangle$$

The set of all complete histories is denoted as H. Rewards depend only on visited states and executed actions, and system history is used to evaluate RL agent’s performance and represented as a sequence of states and actions. The system history h from the set of all system histories Hs provides an external, objective view about the process; value functions will be defined on the set s in the next subsection. In POMDP, an agent cannot fully observe the underlying world state and thus it can only base its decisions on the observable history as the agent has prior beliefs about the world that are summarized by the probability distribution called initial belief b_0 and cover all states. The agent starts by executing some action a_0 based solely on initial belief b_0 and the observable history until time step t is then a sequence of action and observation pairs (A_0, O_1), (A_1, O_2),…, (A_{t-1}, O_t). The set of all possible observable histories will be denoted as H_o. Finally, it is important to highlight that representing H_o impact directly POMDP solution algorithms and the policy output as the observable history will used by RL agent internal memory [64].

4.2.3 Performance measures

At each step in POMDP, the agent has to decide what action to perform based on its internal observable history, the policy π: H→A is a set of rules that map observation into actions and is defined as a probabilities distribution over all possible sequences of states and actions starting by the initial belief distribution b_0. The RL agent goal is to pick a policy that maximises the objective function that is defined on the set of system histories H_s called value function V. This function is essential for the learning as it assign a real number to each h_s, a system history h_i is prioritized over h_j only when $V(h_i) > V(h_j)$ [37]. Overall, the value function is a mapping from the set of RL agent histories into real numbers:

$$V : \mathcal{H}_s \mapsto \mathbb{R}$$

In POMDP formulations, the value function V have a structure that makes it much easier to represent and evaluate and generally V is additive thus the value of a given system history is the sum of rewards accrued at every step. In the case where the decision process stops after a finite number of steps H, the problem is called finite horizon problem and usually aims to maximize the total expected reward. The value function for a RL system h of length H is simply the sum of rewards attained at each stage [68]:

$$V(h) = \sum_{t=0}^{t=H} R(s^t, a^t)$$

The sum of rewards over an infinite horizon is unbounded and therefore we introduce a discount factor γ to mathematically address the problem so the rewards received later get discounted and will impact lesser than current rewards. The value function for a total discounted reward problem is [74]

$$V(h) = \sum_{t=0}^{\infty} \gamma^t \, R(s^t, a^t), \; 0 \leq \gamma < 1$$

4.2.4 Policy Graph representations

POMDP agent's task is to determine the best course of actions in an uncertain environment following a given criterion of optimality. This can be in context of infinite horizon the discounted sum of rewards. The POMDP agent's behavior is determined by the policy π which represent a general mapping from the observation histories to actions π : H_o i→ A**.** while the history is represented as:

$$h^t = \langle a^0, o^1 \rangle, \langle a^1, o^2 \rangle, \ldots, \langle a^{t-1}, o^t \rangle$$

Therefore, the action is represented the policy π at time t as: $a_t = \pi(h^t)$. The expected policy value when considering the initial belief distribution b_0 is represented as Prob (h|π, b_0) for histories Hs. The expected policy value for the policy π:

$$EV(\pi) \equiv V^\pi = \sum_{h \in \mathcal{H}_s} V(h) Pr(h|\pi, b_0)$$

Thus, the value of the policy π at a given starting state s_0 will be denoted:

$$EV(\pi) = \sum_{s \in \mathcal{S}} b_0(s)\, V^\pi(s)$$

The agent's goal is to find a policy π that maximal expected value V^π. Finally, it is important to highlight that the general policy format as mapping arbitrary observation histories to actions is not practical and POMDP solving algorithms exploit value and observation functions to calculate tractable policies where observable histories can be represented as probability distributions over system states. Generally, any POMDP where the agent can fully observe all state is reduces to MDP, the sequence of states forms a Markov chain impose that next state depends only on the current state making the history of the previous states irrelevant [75-79].

4.2.5 Finite versus Infinite horizon

In finite horizon MDP, knowing the current state and stage is sufficient for the agent to represent thewhole observability and thus maximizing total reward whether using a discount rate or not. A policy π is therefore reduced to a map of states to actions π : S × T ⟶ A. π(*s, t*) is the policy at state *s* when *t* stages are left to end the process, the expected value of a policy is calculated following Bellman recurrence as:

$$V_0^\pi(s) = R(s, \pi(s, 0)),$$
$$V_t^\pi(s) = R(s, \pi(s, t)) + \gamma \sum_{s' \in \mathcal{S}} T(s, \pi(s, t), s')\ V_{t-1}^\pi(s')$$

The value functions in the set Vt^π (0≤t≤H) is t-step where H is the number of stages the process goes through. A policy π* is optimal if $V_H^{\pi*}$ (s) ≥ $V_H^{\pi'}$(s) for all H policies π and all states where optimal value function is in fact a value function with optimal policy. Therefore, the Bellman's principle of optimality [77] allows to calculate the optimal t-step value function as follow:

$$V_t^*(s) = \max_{a \in \mathcal{A}} \left[R(s, a) + \gamma \sum_{s' \in \mathcal{S}} T(s, a, s')\ V_{t-1}^*(s') \right]$$

For infinite horizon problems, optimal decisions can be calculated based only on the current system state, since at any stage, there is still an infinite number of time steps remaining [76-78]. Therefore, the value of a stationary policy π can be determined by a recurrence analogous to the finite horizon as bellow:

$$V^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} T(s, \pi(s), s')\ V^\pi(s')$$

As the POMDP agent goal is to find the optimal policy π* that maximise the value function V for all system states, The value function is:

$$V^*(s) = \max_{a \in \mathcal{A}} \left[R(s, a) + \gamma \sum_{s' \in \mathcal{S}} T(s, a, s')\ V^*(s') \right]$$

Value-iteration methods calculate optimal value functions directly and derive the optimal policies implicitly from value functions [80]. We introduce here the notion of a Q-function, *Q*(*s, a*) is the value of executing action *a* at state *s*, and then following the optimal policy:

$$Q(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s,a,s')\, V^*(s')$$

Thus, infinite horizon optimal policy is a greedy of value function:

$$\pi^*(s) = \arg\max_a Q(s,a)$$

4.2.6 Stochastic policies

In MDP, a stochastic infinite horizon policy is the generalisation of a deterministic policy which assigns a distribution for all actions to a state instead of mapping every action to a state ψ : S → Δ(A). Mapping states to actions ψ(s, a) is the probability that action a will be executed at state s and when we add expectation over actions, we can rewrite the value function as:

$$V^{\psi}(s) = \sum_{a \in \mathcal{A}} \psi(s,a)\, R(s,a) + \gamma \sum_{s',\, a} \psi(s,a)\, T(s,a,s')\, V^{\psi}(s')$$

Generally, stochastic policies are not advantageous in context of infinite horizon MDPs but very useful in context of POMDPs as it enables us to convert the discrete action space to a continuous space of distributions actions, then the value function is optimized [81-82].

4.2.7 Policy trees

In POMDP environments, RL agent can only base its decisions on the history of its actions and observations and not simply mapping states to actions which results in complicated form of policy graph as:

$$\sum_{t=0}^{t=H-1} |\mathcal{O}|^t = \frac{|\mathcal{O}|^H - 1}{|\mathcal{O}| - 1}$$

Like MDPs, when only one stage left, the agent can only to execute an action while it can execute an action with two stages left, it will receive an observation, and after it will execute the last action [83].

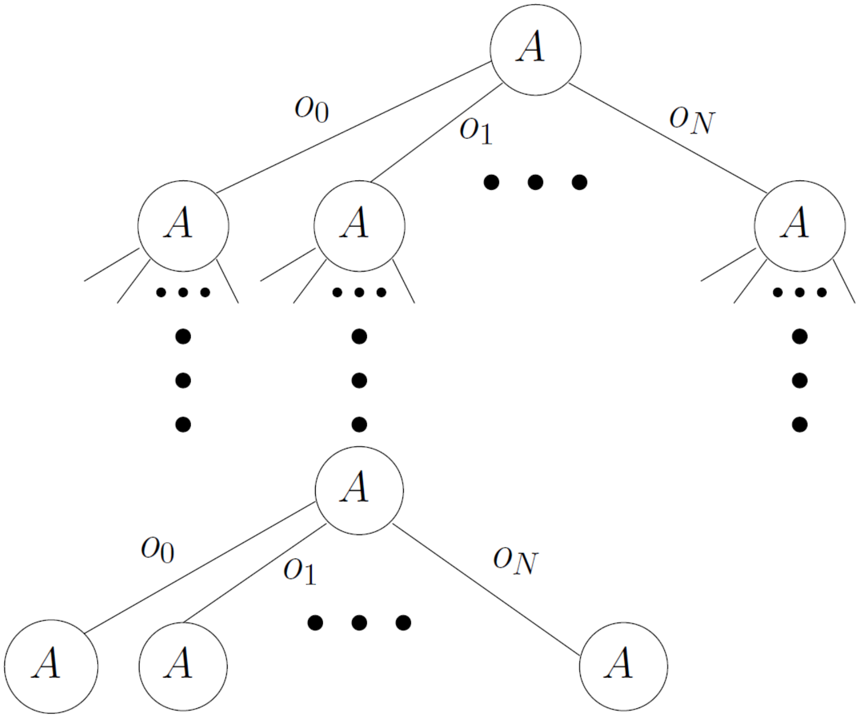


Figure 20: A policy tree for horizon t. For each observation, there is a branch to nodes at a lower level. Each node can be labeled with any action from the set A [76].

In a finite horizon of length H, a policy is a tree of height H where all policies for H are represented in finite policy trees where each node prescribes an action to be taken at a particular stage along with the observation received that determines the next branch to follow [84-85]. A policy tree size in horizon H will have a size of possible H- horizon policy trees of:

$$|\mathcal{A}|^{\frac{|\mathcal{O}|^{H}-1}{|\mathcal{O}|-1}}$$

In context of recursive policy trees, we rely on the notion of conditional plans σ ∈ Γ is a pair <a, ν> where a is an action, and ν : O →Γ is the observation strategy. The set of all observation strategies will be denoted as ΓO; obviously, its size is Γ |O|. Γ_t be the set of all conditional plans available to an agent with t stages left $\Gamma_t = \{(a, v_t) \mid a \in \mathrm{A},\ v_t \in \Gamma^{O}_{t-1}\}$. In this case, representing policy trees as conditional plans allows us to write down a recursive expression for their value function [77]. The value function of a non-stationary policy π_t is formulated as bellow where $\sigma_0(s)$ is the action to be executed at the last stage

$$V_0^{\pi}(s) = R(s, \sigma_0(s)),$$
$$V_t^{\pi}(s) = V_t^{\sigma_t}(s) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s,a,s') \sum_{o \in \mathcal{O}} Z(s',a,o)\, V_{t-1}^{\nu_t(o)}(s'),$$

In POMDP, the actual system state is not fully observed, the value of a particular policy tree with respect to the initial belief state b is:

$$V_t^{\pi}(b) = V_t^{\sigma_t}(b) = \sum_{s \in \mathcal{S}} b(s)\, V_t^{\sigma_t}(s)$$

Therefore, an optimal t-step value function for the belief state b is the simple result of the enumeration of all policy trees for Γt:

$$V_t^{*}(b) = \max_{\sigma \in \Gamma_t} \sum_{s \in \mathcal{S}} b(s)\, V_t^{\sigma}(s)$$

Thus, the t-step value function for the continuous belief simplex b is represented by the max of the finite (often doubly exponential of t) conditional plans [81-84].

4.3 From POMDP to α-vectors and belief state MDPs

POMDP value function is linear and convex [66]. The value of any policy tree V^{σ} is linear in b, V* is therefore the upper surface of the collection of all value functions of policies for Γ_t. ασ be a vector of size |S| with the entries are the of policies tree corresponding to σ in each state:

$$\alpha^{\sigma} = [V^{\sigma}(s_0), V^{\sigma}(s_1), \ldots, V^{\sigma}(s_N)]$$

When applying the *α*-vectors representation we have a V^*_t which contains all *t*-step *α*-vectors corresponding to *t*-step policy trees and sufficient to produce an optimal *t*-horizon value function:

$$V_t^{*}(b) = \max_{\sigma \in \Gamma_t} \sum_{s \in \mathcal{S}} b(s)\, \alpha^{\sigma}(s) = \max_{\alpha \in \mathcal{V}_t} \sum_{s \in \mathcal{S}} b(s)\, \alpha(s)$$

The optimal value function $\mathbf{V_t}$ is represented by the upper surface of the α-vectors V* (figure bellow) where the worst case any policy in Γ^t might (rarely in reality) be in some beliefs superior. Many vectors in within $\mathbf{V_t}$ are often dominated and thus omitted from representing in the optimal value function. In Figure 21, the vector α_3 is completely dominated by α1, whereas vector α_1 is jointly dominated by both vectors α_0 and α_2 together [85].

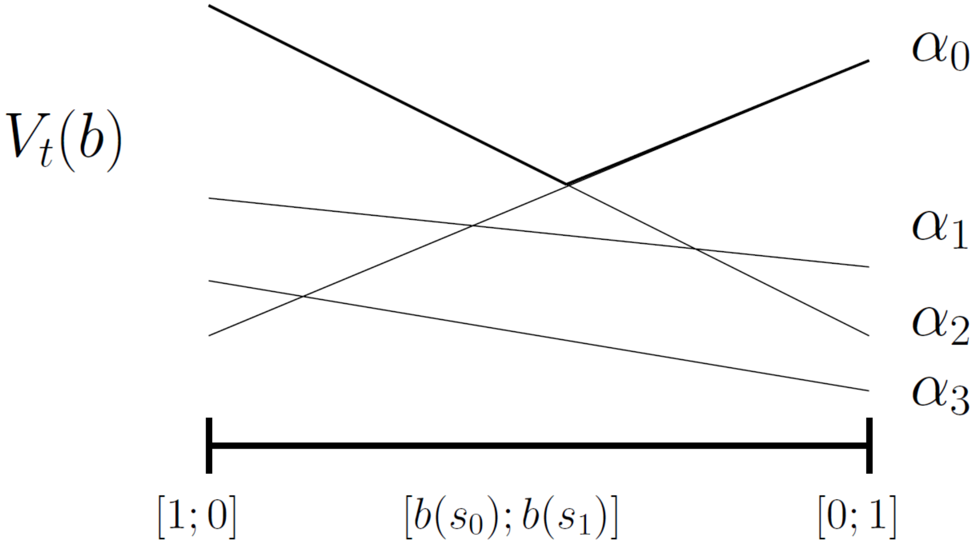


Figure 21: example of two-state POMDP representing the whole belief space B for value function Vt(b) [85].

Therefore, it is possible clean (often called prune) which will result in lowering the number of α-vectors representing the optimal value function $V_t{}^*$ as follow:

$$V_t^*(b) = \max_{\alpha \in \mathcal{V}_t} \sum_{s \in \mathcal{S}} b(s)\ \alpha(s) = \max_{\alpha \in \mathcal{V}_t^-} \sum_{s \in \mathcal{S}} b(s)\ \alpha(s)$$

In the resulting parsimonious set, all α-vectors which are representing policy trees) are relevant [86-89]. A vector α is useful if R (α, V) is non-empty and is not dominated by other vectors as follow:

$$\mathcal{R}(\alpha, \mathcal{V}) = \{b \mid b \cdot \alpha > b \cdot \alpha',\ \alpha' \in \mathcal{V} - \{\alpha\},\ b \in \mathcal{B}\}$$

In practice, the existence of such these regions is done using linear programming (LP) in many value-based POMDP solving algorithms regardless the adopted methods for pruning the α-vectors in V_t.

4.3.1 Implicit POMDP policies

An explicit t-step POMDP policy can be either represented using policy tree or using recursive conditional plan. In a given b_0, the optimal t-step policy is determined by locating value function Max in the set of useful policy trees, then, the RL agent will perform actions at the nodes, and follow the observation links to the determined policy sub-trees [76]. the optimal policy at b with t-stages remaining is:

$$\pi^*(b, t) = \arg\max_{a \in \mathcal{A}} Q_t(b, a)$$

Instead of keeping the entire policy trees, we can simply preserve the useful vectors in each t-stage. Therefore, POMDP implicit t-step policy is defined by performing greedy one-step lookahead. We introduce the Q-value function $Q_t(b, a)$ as a value of taking action a at belief state b while continuing throughout the optimal policy for t-1 remaining stages while b_o^a is the belief state that results from b after taking action a and receiving observation o:

$$Q_t(b, a) = \sum_{s \in \mathcal{S}} b(s) R(s, a) + \gamma \sum_{o \in \mathcal{O}} Pr(o|a, b) V^*_{t-1}(b^a_o))$$

4.3.2 Belief state MDPs

In a finite horizon POMDP, policy is a mapping from belief states and stages to actions π: B × T⟶ A. previous research demonstrated that a proper update of the probability distribution for the entire state space S is enough to summarize all the observable history for the POMDP agent without any loss of optimality [75]. Therefore, a POMDP can be converted into a continuous MDP where belief states is fully observable and annotated as quadruple <B, A, T^b, R^b>. In the following representation:

➢B = Δ(S) is the new continuous state space.

➢A is the action space exactly the same as in POMDP.

➢T^b : B × A i→ B is the belief transition function as follow:

$$\begin{aligned} T^b(b,a,b') &= Pr(b'|b,a) \\ &= \sum_{o\in\mathcal{O}} Pr(b'|a,b,o)\, Pr(o|a,b) \\ &= \sum_{o\in\mathcal{O}} Pr(b'|a,b,o) \sum_{s'\in\mathcal{S}} Z(s',a,o) \sum_{s\in\mathcal{S}} T(s,a,s')\, b(s), \end{aligned}$$

$$Pr(b'|a,b,o) = \begin{cases} 1 & \text{if } b_o^a = b', \\ 0 & \text{otherwise.} \end{cases}$$

After the execution of action, and observation, the updated belief b_0^a can be calculated from the previous belief b as follow:

$$b_o^a(s') = \frac{Z(s',a,o)\, \sum_{s\in\mathcal{S}} T(s,a,s')\, b(s)}{Pr(o|a,b)}$$

- R^b : B × A → R is the reward function:

$$R^b(b,a) = \sum_{s\in\mathcal{S}} b(s)\, R(s,a)$$

Then, the RL agent will have just to execute the action prescribed by the policy, and then update its probability distribution in order to follow the policy. Nonetheless, in context of infinite horizon the value function remains convex but not anymore linear. The optimal policy for infinite horizon problems a stationary mapping of action from beliefs π: B→ A [77-79]. Thus, it is determined using greedy one-step lookahead from the optimal value V* as follow:

$$\begin{aligned} Q(b,a) &= \sum_{s\in\mathcal{S}} b(s) R(s,a) + \gamma \sum_{o\in\mathcal{O}} Pr(o|a,b) V^*(b_o^a), \\ \pi^*(b) &= \arg\max_{a\in\mathcal{A}} Q(b,a). \end{aligned}$$

4.4 Policy Graph for Finite-state POMDP

The optimal infinite horizon value function V∗ is often arbitrarily approximated to a set of finite-horizon value functions V0, V1,... , Vt, with t tend to infinity where all optimal policies are often linear and convex despite some situation where they are convex but contain infinitely many facets [70]. Thus, optimal value are linear so whoever two successive V_t and V_{t+1} are equal will results in optimal value:

$$V^* = V_t = V_{t+1}$$

Because each vector α in V* has an associated belief space region annotated R(α, V*) where it dominates all other remaining vectors represented as follow:

$$\mathcal{R}(\alpha, \mathcal{V}^*) = \{b \mid b \cdot \alpha > b \cdot \alpha',\ \alpha' \in \mathcal{V}^* - \{\alpha\},\ b \in \mathcal{B}\}$$

When an optimal value function V* is represented as a set (finite) of vectors, all belief states within one region are automatically considered new belief states for each given action and the associate resulting observation. In this case, the belief transitions constitute a policy graph (PG) where each node correspond to belief space when optimal actions and transitions are matched to observations [66-71]. Therefore, RL agent is not required to formally maintain its belief state b and perform the heavy computing operations of updating it while determining the optimal α-vector but simply using the starting node which is by default optimised for use as initial belief b_0. When the POMDP does not allow the representation of the infinite horizon policies in form of finite policy graph the above operation is slightly modified to extract from a suboptimal value function the near-optimal policy graph. This will raise of course the tracabilty issue as only with a limited size of PG that the tracabilty of solving is achieved and remains approximate otherwise as it depends on the whole history of observations and actions which is often sacrificed to ease the solving of the POMDP problem by assuming that RL agent is constrained in term of memory and only required (the agent) to execute a policy that are present in the mapping internal states-actions [66-68].

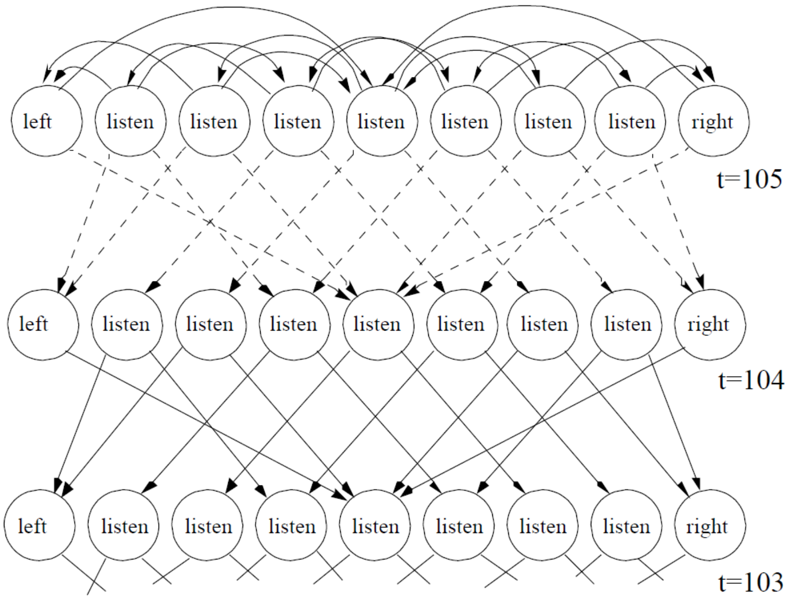


Figure 22: illustration of policy tree branches rearrangement to produce stationary policy [78].

4.4.1 Finite State Controller model

A deterministic PG π is a triple <N , *ψ*, *η*>, where:

- N the set of nodes *n* constituting the internal memory states.
- *ψ* : N→ A the action selection function that for each node *ψ*(*n*).
- *n* : N x O → N the node transition function that for each which assign to every observation the successor node n which in form of observation strategy already defined as trees or conditional plans. In a stochastic FSC, the function ψ and the internal transition function η are therefore stochastic.

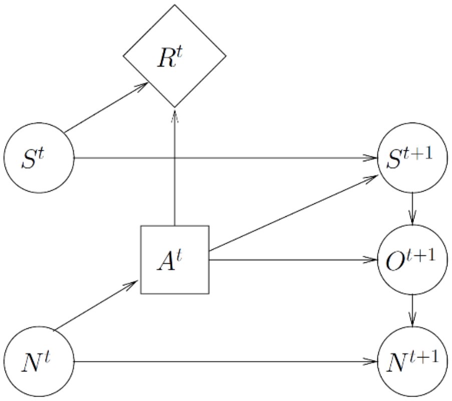


Figure 23: POMDP policy graph joint influence diagram.

Thus, ψ :N →Δ(A) which is a stochastic action selection function that for each node distribution for all system actions:

$$\psi(n,a) = Pr(A^t = a | N^t = n)$$

While *n*: N x O → Δ(N) is the node transition function when assign a probability distribution for every node and observation as:

$$\eta(n,o',n') = Pr(N^{t+1} = n' | N^t = n,\, O^{t+1} = o')$$

4.5 Policy graph value

In the POMDP environment, the agent stored PG π =<N,ψ, η> is used to determine policy graph can be calculated using Bellman's equation when T^π is the transition matrix is as follows:

$$V^\pi(s) = R^\pi(s) + \gamma \sum_{s'} T^\pi(s,s')\, V^\pi(s')$$

Given the stochastic nature of the functions ψ and η, the transition matrix T^π will account for expectation over actions a and observations o is formulated as follows:

$$T^\pi(\langle n,s\rangle, \langle n',s'\rangle) = \sum_{a,o} \psi(n,a)\, \eta(n,o,n')\, T(s,a,s')\, Z(s',a,o)$$

When R^π is consequently the reward vector:

$$R^\pi(\langle n,s\rangle) = \sum_a \psi(n,a) R(s,a)$$

4.6 Value iteration

In MDPs. the standard method is to find the optimal infinite horizon policy π* using a sequence of optimal finite horizon value functions V_0^*, $V_1^*\ldots$, V_t^* [90]. The difference between the optimal value function and the optimal t-horizon value function described earlier is that t goes to infinity:

$$\lim_{t\to\infty} \max_{s\in\mathcal{S}} |V^*(s) - V_t^*(s)| = 0$$

given the Bellman error E, the optimal value function can be calculated in a finite number of steps as follow:

$$V_{t+1}(s) = \max_{a\in\mathcal{A}} \left[R(s,a) + \gamma \sum_{s'\in\mathcal{S}} T(s,a,s')\ V_t(s') \right]$$

In POMDP context, it is first reduced into a continuous belief-state MDP to enable then the calculate optimal infinite horizon POMDP policies where the value function is calculated as the following:

$$V_{t+1}(b) = \max_{a\in\mathcal{A}} \left[\sum_{s\in\mathcal{S}} b(s)R(s,a) + \gamma \sum_{o\in\mathcal{O}} Pr(o|a,b)V_t(b_o^a) \right]$$

Where:

$$Pr(o|a,b) = \sum_{s'\in\mathcal{S}} Z(s',a,o) \sum_{s\in\mathcal{S}} T(s,a,s')\, b(s)$$

The ultimate aim of any value-iteration algorithm is to find the set t+1 representing value function V_{t+1}, given the previous set of α-vectors t. Algorithms largely differ in how they compute value function representations with most naive way is to construct the set of conditional plans V_{t+1} which is done by enumerating all both sets of actions and observations and mappings to the set V_t with the size of V_{t+1} equals $|A||V_t||O|$ accounting for many vectors in the Vt that are to be dominated [92-95].Some algorithms calculate V'_{t+1} by generating V_{t+1} of size $|A|$ x $|V_t|^{|O|}$ and later pruning by eliminating the dominated α-vectors using linear programming. Incremental pruning algorithm such as IP proposed by [78] and others build the set V'_{t+1} directly from V'_t without considering any useless conditional plans making these method on of the most efficient exact value-iteration solving that can solve medium size POMDPs when adequate computational power is supplied.

4.7 Policy Search

Policy iteration algorithms works differently from value iteration as they proceed by iteratively improving the policies themselves generating a sequence $\pi^0, \ldots, \pi^t$ which converges to the optimal infinite horizon policy π^*, as t converges to ∞. Policy Iteration works on two stages basis; policy computing then policy improvement. For MDP policy iteration, it starts with initializing $\pi^0(s)$, then repeat the policy iteration and improvement steps until $\pi^{t+1}(s) = \pi^t(s)$ meaning that the policy does not change [102]. At a first stage, the value of policy is calculated as:

$$V^{\pi_t}(s) = R(s,\pi_t(s)) + \gamma \sum_{s'\in\mathcal{S}} T(s,\pi_t(s),s')\ V^{\pi_t}(s')$$

Later the policy evaluation is calculated based on Q-function as:

$$Q_{t+1}(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s,a,s')\, V^{\pi_t}(s')$$

Finally, the policy improvement is computed for all states as:

$$\pi_{t+1}(s) = \arg\max_{a \in \mathcal{A}} Q_{t+1}(s,a)$$

Therefore, MDP policy iteration converge much faster with the only disadvantage of the higher computational requirements as it calculate for each step, a policy evaluation requiring a |S| × |S| linear system. In POMDP policy iteration, as the importance is to represent a policy so that its value function can be calculated easily. This there is a use of FSC in the explicitly representation which is independent from the V^π. Hansen proposed an approach to directly represent a policy using FSC and the algorithm execute policy iteration in set of MDPs. After, it performs regular policy iteration steps with evaluation and improvement done using dynamic programming update to results in a sequence of finite-state controllers π_0, $\pi_1 \ldots, \pi_t$ converges to the optimal policy π^* as t converges to ∞. The policy evaluation works at each controller node which corresponds to an α-vector. Because of the deterministic nature of the PG, ψ(n) outputs the action with the *α*-vector representation of a V is formulated as:

$$V^\pi(\langle n,s\rangle) = R(s,\psi(n)) + \gamma \sum_{s',o} T(s,\psi(n),s')\, Z(s',\psi(n),o)\, V^\pi(\eta(n,o),s')$$

When:

$$V^\pi(\langle n_i,s\rangle) \equiv \alpha_i(s)$$

With the policy evaluation having a maximum running time of $|N \times S|^2$.

4.8 Policy improvement

The policy improvement consists of performing dynamic programming backup for each step t in which the POMDP value function V^π represented in form of finite set of α-vectors is replaced by an improved value function V'. This later is then represented by another most of exact solving algorithms notably the Witness and Incremental pruning perform very well in medium size POMDPs [99-102].

For every step, a set of α-vectors V^{π} is calculated from the finite-state controller π as described above and later is processed using dynamic programming backup to produce a new V^{π} as follow:

- For each vector α ∈ V where action and linked α are similar to the action and CP existing in the initial π, the node will be maintained in the π',
- When α is dominating more than one node in π it is then inserted into the π by changing the action and successor links to the vector α,
- Else, if any new action and observation strategy associated with α is found it is added π.

Finally, the pruning is done by removing any useless node in π which have no correspondence with α-vector in V. Note that the policy improvement does not affect in any case the FSC which remains optimal. In practice, POMDP policy only requires a few steps to converge as the policy evaluation complexity is negligible when it is compared to the worst-case exponential complexity of the associated dynamic programming improvement which gives this solving approach clear advantage over value iteration in solving large POMDPs especially when the initial belief state is known [103].

4.9 Gradient-based optimization

Exact solving approach of POMDPs is famous for being time and computational power consuming in addition to the high intractability. This is mainly due to fact that optimal policies can be either very large or sometime infinite. The example illustrating this last issue is the number of numbers of controller nodes which grow twice exponentially for a given H which is also the case for α-vectors number required to represent the value function which multiplied exponentially in value iteration. Therefore, the obvious solution is the approximation through the restriction in the set of policies with the aim of determining the best policy in the set [94].

All POMDPs policies can be represented in form of policy graphs PG which is an efficient restriction to limit the set of infinite policies to be representable by finite PG or a limited size FSC which achieve a compromise or balance between the requirement of observable history and the ability to reduce the size and complexity of the policy space taking into consideration that exact policy iteration does not place any constraints on the policy graph structure but some algorithms exploit computational advantage of searching in restricted FSCs to improve performances. Research works [95-97] and [99] proposed the search for optimal memoryless policies while [88] and [100] worked on finding sequences of reactive policies with a direct search in an imported policies set using called finite policy graphs. In this research, the restricted policy space that we will be consider is representable by a limited size stochastic finite-state controller and use of a gradient-based policy search method. The major contribution of this gradient based POMDP policy search methods is the reformulation of finding optimal POMDP policies problem as a nonlinear numerical optimization problem for which the gradient of V [101].

4.10 Policy graph values:

The value of a stochastic PG π, with V and R vectors with a of length |N|x|S| and γ <= 1, is now summarizing in form of matrix of a size |N|x|S| by |N|x|S| is as follow:

$$V^{\pi} = R^{\pi} + \gamma T^{\pi} V^{\pi}$$

$$V^{\pi} = (I - \gamma T^{\pi})^{-1} \; R^{\pi}$$

the vector V^{π} is therefore optimised by picking the functions ψ and η which enable converting the problem into nonlinear optimisation [102].

4.11 Initial beliefs

V^{π} contains the total discounted cumulative reward which depends on the state s and node n in which POMDP agent starts from called initial beliefs and annotated b_0. In reality this is the best representation that agent have about the environment and is represented as follow:

$$\sum_{n,s} b(\langle n,s\rangle) = 1,$$

$$b(\langle n,s\rangle) \geq 0 \text{ for all } n \in \mathcal{N}, s \in \mathcal{S}$$

With a total cumulative discounted reward:

$$E^{\pi} = b_0 \cdot V^{\pi}$$

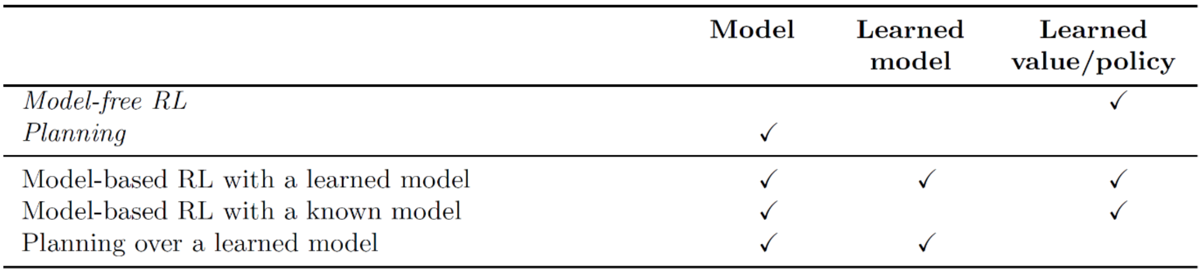
In POMDP, we often assume that agent starts from the first node and its prior knowledge about the world is formulated as:

$$b(\langle n,s\rangle) = \begin{cases} b(s), & \text{if } n = n_0, \\ 0, & \text{otherwise.} \end{cases}$$

4.12 Reinforcement Learning Approaches:

There are two RL approaches for agent functioning: model-based and model-free. In the model-based approach, the RL agent will use a predictive model of the environment to learn by attempting different actions in each state and choosing the best one. The model-based approach uses past experiences in the form of a sequence of instances <a_t, o_t, r_t> to learn a POMDP model that is likely to generate the sequence. Modern algorithms and computing capabilities allow for a tractable solution for POMDPs with reasonably sized state spaces. In model-based, the agent uses a predefined internal model, one that both predicts action outcomes and estimates the immediate reward associated with specific situations. Decisions are made not based on stored action values, but instead following a planning approach. This approach makes use of the internal model and therefore the suitability and efficiency of candidates will depend on their behavior in the specific context [88].

Table 1: a comparative study of RL modelling, learning, and solving approaches [88]



	Model	**Learned model**	**Learned value/policy**
Model-free RL			✓
Planning	✓		
Model-based RL with a learned model	✓	✓	✓
Model-based RL with a known model	✓		✓
Planning over a learned model	✓	✓	

Over the past decade, many research works investigated possible roles for model-free RL in human experts' decision-making. In the model-free approach, the modelling step is intentionally omitted to allow direct learning of the policies. Model-free RL assumes that learning occurs without access to any internal representation of the environment structure. Instead of building such an internal model, the RL agent will simply store estimates for the expected values of the actions available in each state and this will be shaped to become a history of direct interaction with the environment. In our research and giving the PT context, we opted for the model-free approach and later for a more direct method of solving RL which is Policy Search which will be fully justified in the next chapter [104-105].

4.13 Model-based vs Model-free modelling approach

In this section we describe different POMDP modelling approaches namely model-free and model-based which both will be initially considered four our proposed framework. We initially discuss here the two RL modelling approach and highlighting the pros and cons of each. Finally, we will sum up in selecting the model-free approach after excluding the model-based approach [106].

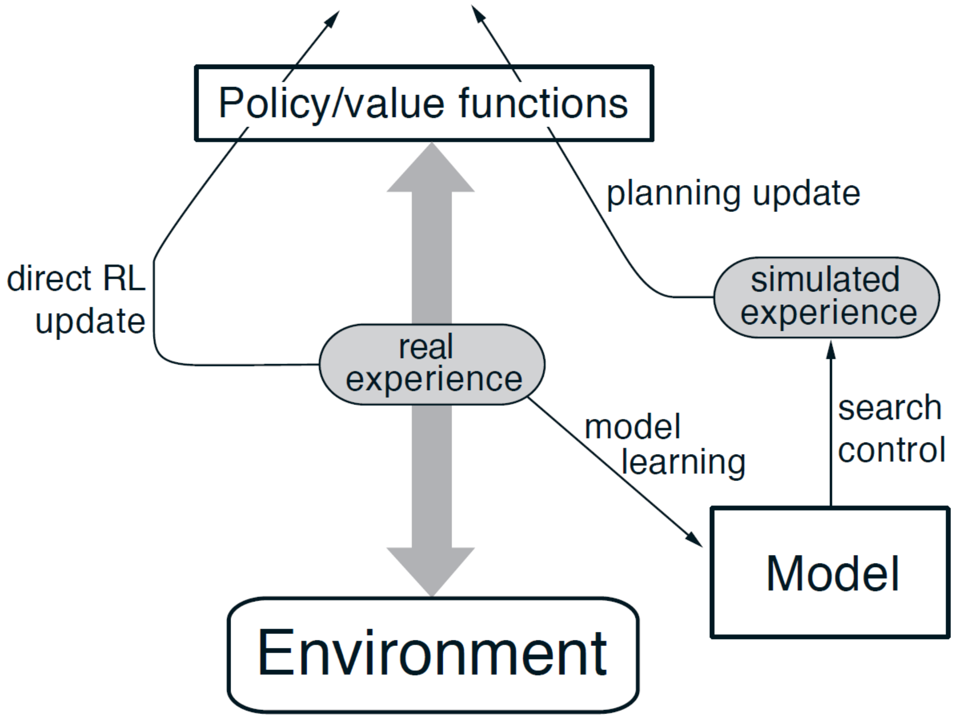


Figure 24: different RL learning and modelling approaches with solving methods [108].

4.13.1 Model-free POMDP

Model-free approach RL agent will use algorithm that will compute optimal policy without utilizing the POMDP environment transition, and reward functions. In practice, a model-free RL agent estimates a value function or the policy directly by interacting with the environment without requiring neither the transition function T nor the reward function R. Thus, a value function can be thought of as a function which evaluates a state (or an action taken in a state), for all states and from it a policy can then be derived [109].

In fully observable MDP the main assumption is that the RL agent might be unaware of the state transition and reward probabilities and try to navigate the solve the problem, the agent simply relies on some trial-and-error experience for action selection [110]. This latter is problematic in POMDP where agent is not aware of any state space but only some aspects of the problem that structural mainly such as the actions it can execute and the expected rewards without real accuracy. Model-free POMDP [111] attempt to learn how acting without learning the major parameters of the model which make it a trivial approach to cope with an unknown state space observed through the use the observation space rather than relying on the completely unknown state space, thus the main assumption here is that observation could correspond to a state entry. This use of observations often generates two major issues in POMDP [112]. The first is when the agent is misled by varied observation output which are very different or irrelevant to the environment and the RL agent is required to make the difference between the relevant and irrelevant observations. The second problem with model-free POMDP is the size of the O space which might inferior to A which results in the RL agent suffering the well-known perceptual aliasing problem. This is worst when having numerous states correspond to the same precept can and thus the same optimal action causing the downsize of the S space. This results in the agent will be forced to perform different actions in each state despite the fact that s and s' generate the same observation which will harm deeply the calculation of the optimal policy graph [113].

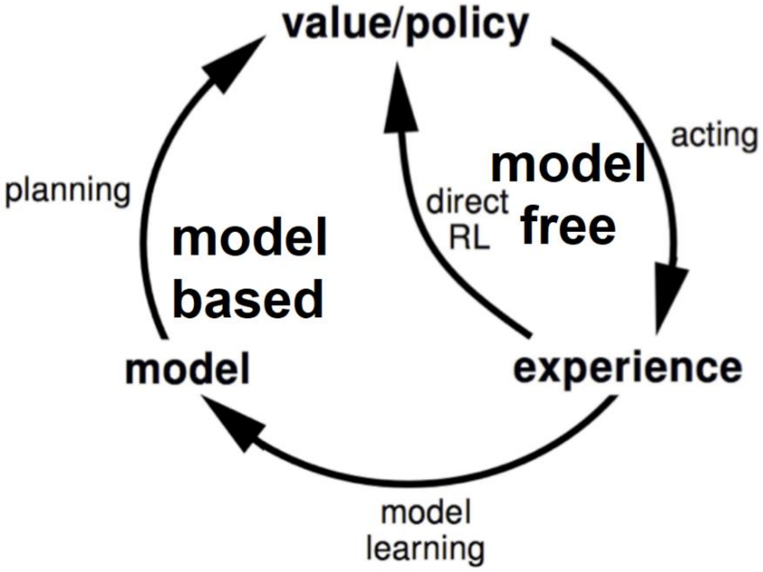


Figure 25: model-based versus model-free RL [66].

4.13.2 Model based POMDP

In model-based learning, the RL agent exploits a previously learned model to accomplish the task and therefore it has access to a model of the environment. In this context, a model-based algorithm is an algorithm that uses the transition function and the reward function in order to estimate the optimal policy. The agent might have access only to an approximation of the transition function and reward functions, which can be learned by the agent while it interacts with the environment, or it can be given to the agent. In general, in a model-based algorithm, the agent can potentially predict the dynamics of the environment (during or after the learning phase), because it has an estimate of the transition function (and reward function) [114]. However, note that the transition and reward functions that the agent uses in order to improve its estimate of the optimal policy might just be approximations of the "true" functions. Hence, the optimal policy might never be found (because of these approximations) [115-118]. The main advantage in model-based RL is that it allows the agent to plan by thinking ahead as it distills the results from planning ahead and translate it into learned policy. The main downside is that in medium and highly complex scenarios a true modelling of the environment is not usually available [119-121].

4.13.3 POMDP solving approach: value function vs policy search

In this section we describe different POMDP solving approaches, we will then highlight those main characteristics of both policy search and value function [89]. The aim of the POMDP agent is to find (take) actions which fulfill its task in the best possible way and thus learn an optimal policy. In fact, an optimal policy does not map the state to action but instead it maps observation and beliefs to actions $\pi(b)$. Contrary to MDPs, the policy $\pi(b)$ is a function or set of probability distributions over S. As already discussed in this chapter, a policy π can be characterized by a value function V^{π}: $\Delta(S) \rightarrow R$ which is defined as the expected future discounted reward $V^{\pi}(b)$ the agent can gather by following π starting from belief b. Because the POMDP representation complexity in term of elaborating the environment especially in large problems where the real-world effect impacts the complexity. We describe two major approaches of reinforcement learning: the first which learn a value function over states of the world, and the second which search in the space of policies directly [121].

There are many different approaches to solve a RL problem with two popular solving approach namely, value function and policy search [121]. The value function approach allows an RL agent evolving within the environment to select the sequences of actions that lead to maximizing the overall value which is often done on the long term and not only in the immediate future. On the other hand, policy-search approach looks directly in the space of policies for the best course of action. Constraining the policy space facilitates the search and may lead tractable (although approximate) POMDP solution algorithms [123-124]. Finite-state controllers (FSCs) are the policy representation of choice in such situation by providing a compromise between the requirement that action choices depend on certain aspects of observable history and the ability to easily control the complexity of policy space being searched [125].

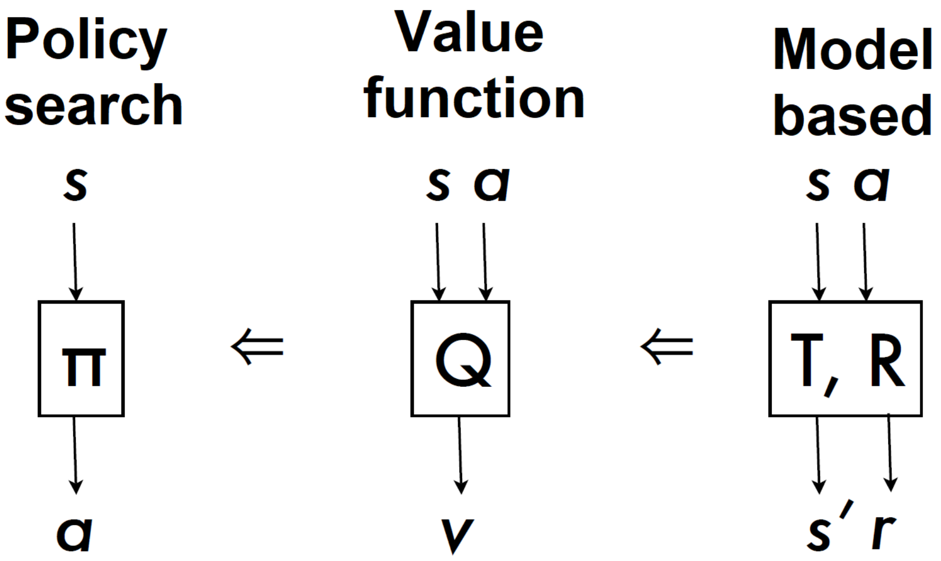


Figure 26: RL different approaches inputs and outputs [126].

4.13.4 Policy-search approach

The policy is the decision-making function which the agent adopts (or try to learn) to later follow. It specifies what action the agent should take in any of the situations it might encounter. In the RL policy-search approach, the agent "ultimate" target is to dress the best policy (decision sequences) that maximize the received award and achieve the initial pre-set objectives such as minimising the solving consumed time and/or number of episodes, thus the other RL components will be used and manipulated to improve the policy [110].

4.13.5 Value-function approach:

The value function is used in the learning process to control it over a long time, it specifies what is good in the long run. As a simple example, the value of a state is the total amount of reward the agent can expect to accumulate over the future when starting from the current state. Unlike the reward scheme which determines immediate desirability, the function value deals with the long-term desirability. In analogy to the human way of acting and thinking, rewards are an immediate pleasure (if high reward) or pain (if low) whereas values correspond to the more refined far-sighted judgment of how pleased or displeased we are that our environment is in a particular state. Most of the research work on improving RL technique focus on improving approximate value functions. The value of an action is its overall utility; for example, an action may bring a high reward, but lead to low-value states, making it low-value [127-129].

4.13.6 Reward function approach

In this RL approach, defining (finding or determining) the adequate reward function that maximizes the benefits received or produce the sought decision policies is the goal of the RL agent. It maps the state of the environment to a single number, a reward, indicating the intrinsic desirability of the state. The agent's objective is to maximise the total reward it receives in the long run. When adopting the policy search approach, the reward function is not learned directly as fixed rewarding mechanism will be used as input for the RL system. The transition probabilities also do not have to be learned [130]. The agent can directly learn the action values, or even directly the policy, with policy search approach method for instance. There are, however, techniques for which the reward and the transition probabilities must be either provided or learned [131].

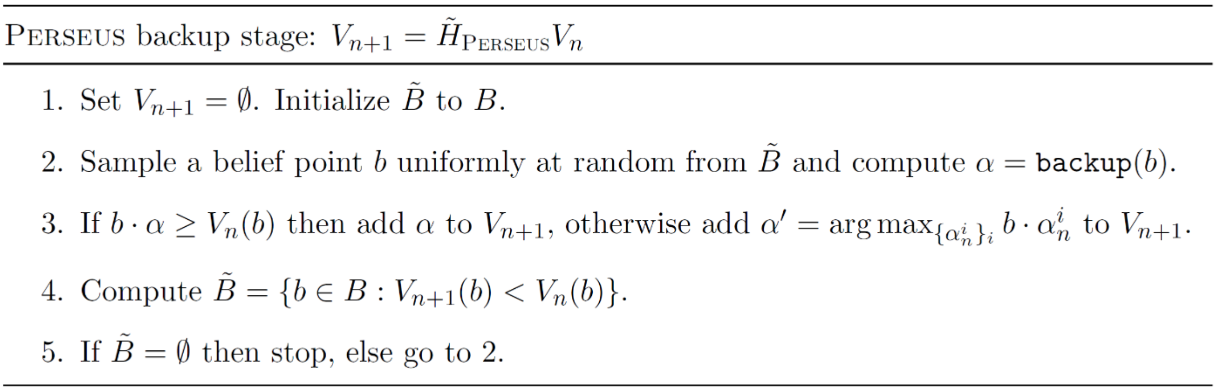
4.14 Solving POMDP algorithms:

4.14.1 Approximate solving:

We present here the most popular approximate solving algorithms. PERSEUS is a point-based value iteration algorithm for POMDPs designed by [132] in which the value function update scheme is implemented with a randomized approximate backup operator that increases the value of all belief points in B and thus exploiting the value function characteristics. In every stage (value backup), PERSEUS improves the value of all points in the belief set by updating the value with a random gradient selected from the available subset of the points [133].

At stage n and giving the value function Vn, PERSEUS calculate the next value function Vn+1 that improves the value of all values in B resulting in value function V_{n+1} that upper bounds V_n over B which removes the necessity of performing linear programming [134]. The RL agent will first randomly explore its environment and built B the set of reachable belief points which are then fixed during the entire algorithm execution phases. PERSEUS will then use the built, from the initial belief, the value function V_0 in form of vector. Then, starting with V_0, PERSEUS will keep performing backup stages until it reaches the convergence criterion as described in algorithm 1 below.

Table 2: PERSEUS Randomized Point-based Value Iteration approximate solving Algorithm [123].



```
PERSEUS backup stage: $V_{n+1} = \tilde{H}_{\text{PERSEUS}} V_n$
  1. Set $V_{n+1} = \emptyset$. Initialize $\tilde{B}$ to $B$.
  2. Sample a belief point $b$ uniformly at random from $\tilde{B}$ and compute $\alpha = \texttt{backup}(b)$.
  3. If $b \cdot \alpha \geq V_n(b)$ then add $\alpha$ to $V_{n+1}$, otherwise add $\alpha' = \arg\max_{\{\alpha_n^i\}_i} b \cdot \alpha_n^i$ to $V_{n+1}$.
  4. Compute $\tilde{B} = \{b \in B : V_{n+1}(b) < V_n(b)\}$.
  5. If $\tilde{B} = \emptyset$ then stop, else go to 2.
```

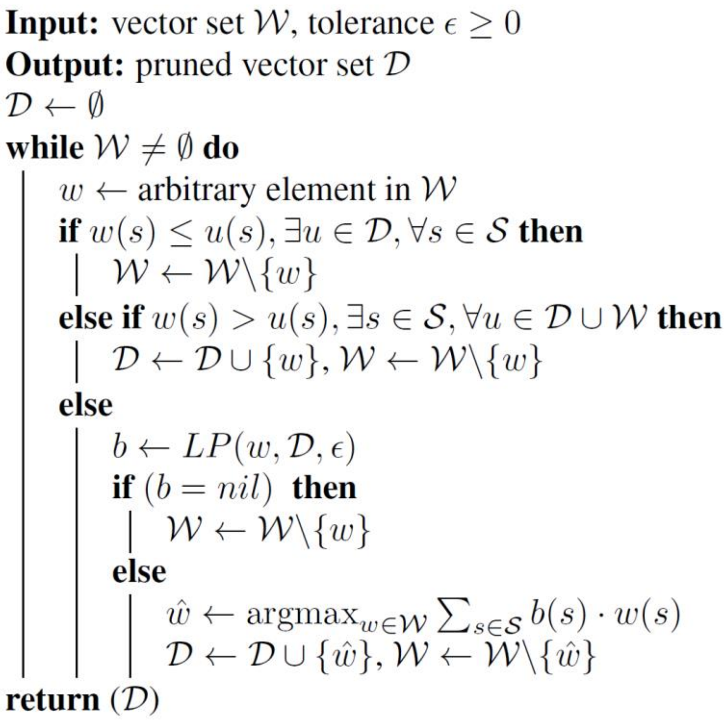
The most attractive feature in PERSEUS is its extensibility to solve large size POMDP enticements as it relies on the principle of 'improve–only' during its backup stage which involves a maximization over all actions in A. in small and medium size action space, PERSEUS can cache in advance the transition, observation, and reward models for all actions and thus achieve an optimised implementation of backups. In large and continuous action spaces where a full maximization over actions is infeasible, PERSEUS enable the use of max operator that performs the maximisation over a random subset of A [130]. This method enables PERSEUS to compute some sampled action which can generate temporary models that are cached for later use notably when the same action is reconsidered in next iterations. Finally, the key factor for picking PERSEUS algorithm is the efficiency in term of time and memory as picking a belief b which later backed up will the result into vector improvement for many belief points in B and not only the related value function of the picked belief and thus computing value functions with a smaller number of vectors [135].

4.14.2 Exact solving

4.14.2.1 Incremental Pruning

Incremental Pruning algorithm (detailed in algorithm 2) starts by generating $|A||V_n||O|$ vectors of the entire horizon H over V_n and then proceeds to the pruning of dominated vectors.

Table 3Basic Incremental Vector pruning algorithm [135].



```
Input: vector set $\mathcal{W}$, tolerance $\epsilon \geq 0$
Output: pruned vector set $\mathcal{D}$
$\mathcal{D} \leftarrow \emptyset$
while $\mathcal{W} \neq \emptyset$ do
    $w \leftarrow$ arbitrary element in $\mathcal{W}$
    if $w(s) \leq u(s), \exists u \in \mathcal{D}, \forall s \in \mathcal{S}$ then
        $\mathcal{W} \leftarrow \mathcal{W} \backslash \{w\}$
    else if $w(s) > u(s), \exists s \in \mathcal{S}, \forall u \in \mathcal{D} \cup \mathcal{W}$ then
        $\mathcal{D} \leftarrow \mathcal{D} \cup \{w\}, \mathcal{W} \leftarrow \mathcal{W} \backslash \{w\}$
    else
        $b \leftarrow LP(w, \mathcal{D}, \epsilon)$
        if $(b = nil)$ then
            $\mathcal{W} \leftarrow \mathcal{W} \backslash \{w\}$
        else
            $\hat{w} \leftarrow \mathrm{argmax}_{w \in \mathcal{W}} \sum_{s \in \mathcal{S}} b(s) \cdot w(s)$
            $\mathcal{D} \leftarrow \mathcal{D} \cup \{\hat{w}\}, \mathcal{W} \leftarrow \mathcal{W} \backslash \{\hat{w}\}$
return $(\mathcal{D})$
```

Incremental Pruning methods [122] save hugely in terms of computation time as it exploits the following feature:

$$\mathtt{prune}(G \oplus G' \oplus G'') = \mathtt{prune}(\mathtt{prune}(G \oplus G') \oplus G'')$$

Therefore, the number of constraints in the LPs used for pruning purposes will grow and leading to better performance. The basic IP algorithm exploits the specific context of the above equation when it computes V_{n+1} as bellow:

$$V_{n+1} = \mathtt{prune}(\bigcup_a G_a)$$

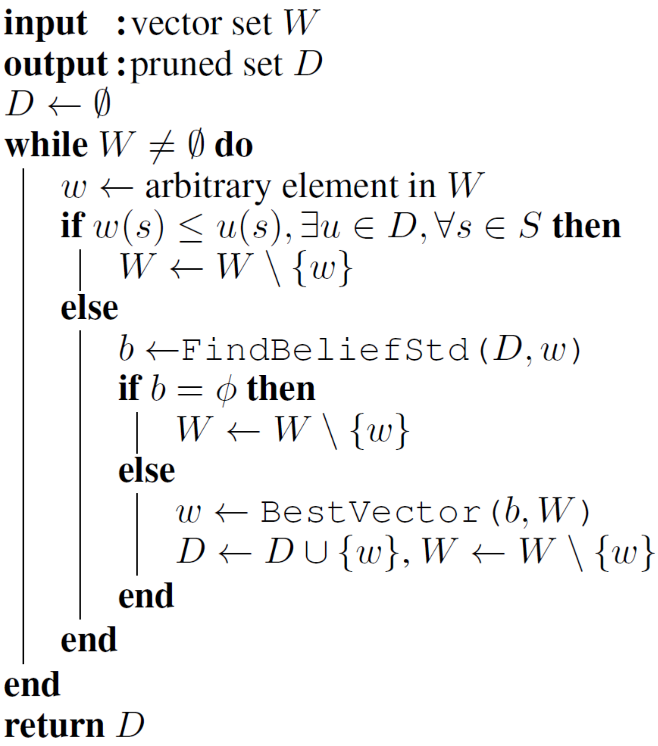
$$\begin{aligned} G_a &= \mathtt{prune}(G_a^1 \oplus G_a^2 \oplus G_a^3 \oplus \cdots \oplus G_a^{|O|}) \\ &= \mathtt{prune}(\cdots \mathtt{prune}(\mathtt{prune}(G_a^1 \oplus G_a^2) \oplus G_a^3) \cdots \oplus G_a^{|O|}) \end{aligned}$$

In general, computing exact solutions for POMDPs is an intractable problem [123] which increased the use of approximate solving algorithms [140]. Therefore, IP algorithm will determine the best action for the available state set in a very efficient manner by relying on comparison of the value function with the witness which led to superior performance and asymptotic complexity.

4.14.2.2 Generalized Incremental Pruning

We present here the most popular and efficient exact solving algorithms. Generalised Incremental Pruning (GIP) is an exact POMDP solving algorithm which computes optimal solutions for POMDPs following linear programming (LP) solving and pruning techniques. In fact, exact solving of POMDP is challenging because of the high computational requirements for LP based algorithms. GIP is different from major exact solving algorithms which utilise subroutines functions to prune dominated vectors in value functions and therefore require a huge number of linear programs (LPs) to be solved and it represents a large part of the total running time [141-143].

Table 4: implementation of incremental pruning combined with decomposed LP [137].



```
input  : vector set $W$
output : pruned set $D$
$D \leftarrow \emptyset$
while $W \neq \emptyset$ do
    $w \leftarrow$ arbitrary element in $W$
    if $w(s) \leq u(s), \exists u \in D, \forall s \in S$ then
        $W \leftarrow W \setminus \{w\}$
    else
        $b \leftarrow$ FindBeliefStd($D, w$)
        if $b = \phi$ then
            $W \leftarrow W \setminus \{w\}$
        else
            $w \leftarrow$ BestVector($b, W$)
            $D \leftarrow D \cup \{w\}, W \leftarrow W \setminus \{w\}$
        end
    end
end
return $D$
```

In GIP, pruning subroutines are decomposed using a Benders decomposition [111-114]. The resulting algorithm incrementally adds LP constraints and uses only a small fraction of the constraints. Our algorithm significantly improves the performance of existing pruning methods and the commonly used incremental pruning algorithm. The GIP variant of is the fastest optimal pruning based POMDP algorithm.

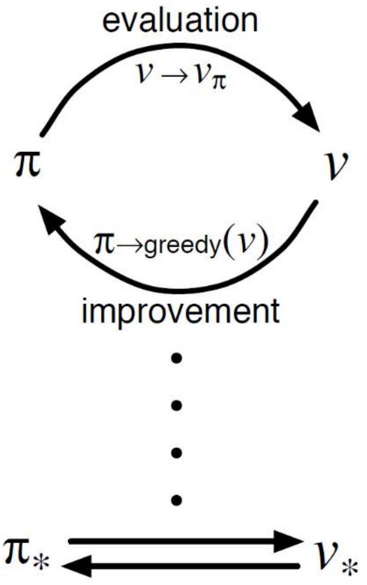
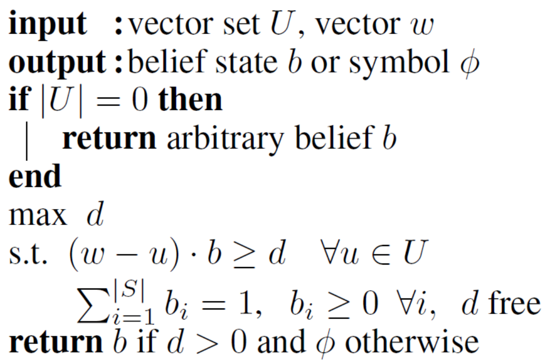


Figure 27: Generalized policy iteration scheme with interaction between value and policy functions [96].

GIP employ enhanced filter function as the intermediate value vectors are passed through to remove the irregularity by performing dynamic programming updates in POMDPs [109]. In practice. Nonetheless this decomposition is not automatically applied but only when in context of complex pruning for POMDPs resulting in GIP algorithm that only require a limited number of constraints in the original LP to find an optimal LP solving of the POMDP problem. GIP is memory efficient compared with other IP variants and proved as the fastest exact algorithm for solving small and medium size POMDP problems, the decomposition approach is behind the performance enhancement [87].

Table 5: Find Belief Std – computes the belief in which w improves U the most [138].



```
input  : vector set $U$, vector $w$
output : belief state $b$ or symbol $\phi$
if $|U| = 0$ then
    return arbitrary belief $b$
end
max  $d$
s.t.  $(w - u) \cdot b \geq d \quad \forall u \in U$
      $\sum_{i=1}^{|S|} b_i = 1, \quad b_i \geq 0 \;\; \forall i, \quad d$ free
return $b$ if $d > 0$ and $\phi$ otherwise
```

4.14.3 Challenges with POMDP

In practice, the realistic extension of MDPs dramatically increases the complexity of POMDPs, making exact solutions virtually intractable. Dealing with uncertainty and partial observability in sequential decision problems is a very challenging for the RL intelligent agents. POMDPs are widely approved as successful representation for sequential decision making and planning problem under uncertainty and have proved successful in most of real-world applications such as robotics and people assistance [134].

A significant number of research works focused on POMDPs and made many breakthroughs notably solving to optimality medium and size problem when enough processing power is provided. Although several efficient approximate methods for POMDPs exist such as PERSEUS [43-44], optimal solutions are more respected notably with recent advances in LP and the increasing computational power [135]. Furthermore, the advantage of exact (optimal) solutions is that they are independent of external parameters such as incremental pruning [137] which is the most popular method to determine optimal solving for POMDP basing on a subroutine that prune (eliminate) dominated vectors from value functions to enable subroutine solving of large number LPs. Nonetheless, checking whether a vector is dominated is often a costly operation which result into LP representing a major part of the total running time. Most research attempted to address the scalability of the LP subroutine by solving less LPs taking advantage of POMDP structure which allow the creation of LPs with fewer constraints and thus deriving much faster and efficient exact solving algorithms [99-100].

Unfortunately, the generality of POMDPs entails high computational cost. The problem of finding optimal policies for finite-horizon POMDPs has been proven to be PSPACE-complete [117]. The intractability of exact solving algorithms and notably GIP LP-Solve algorithm and current solution algorithms, especially those that use dynamic programming to construct (approximately) optimal value functions [139], the application of POMDPs remains limited to very small problems. In addition, to act optimally the RL agent might need to consider all the previous history of observations and actions, rather than just the current state it is in [123]. Finite-state controllers (FSCs) are the policy representation of choice in such work, providing a compromise between the requirement that action choices depend on certain aspects of observable history and the ability to easily control the complexity of policy space being searched.

While optimal FSCs can be constructed if no restrictions are placed on their structure [47], it is more usual to impose some structure that one hopes admits a good parameterization, and search through that restricted space. One way is to consider the problem of finding the best FSC of a given size for a completely specified POMDP. Even with the FSC size restriction constraint, the problem remains NP-hard [135-139]; therefore, gradient ascent (GA) has proven to be especially attractive for solving this type of problems because of its computational properties [46-49]. Unfortunately, gradient-based approaches can converge to arbitrarily bad local optimal.

Chapter 5: Proposed Model of Network Penetration Testing as RL problem

In this chapter we will detail the process of defining the RL model for network PT which enable IAPTF to move from IAPTF-Prep data and previous tests data directly inputted towards a POMDPs problem. This process is done several times for each testing as the output POMDP will change in line with testing data updates as result of PT progression. As we already discussed in Chapter 2 and Chapter 4, we opted for a model-based approach because of its relevance to our problem context notably the fact that model-based methods use the knowledge of the probabilistic environment as a guide and the RL agent plans and navigates the environment aiming to learn the best actions for each state which reflect the PT practice. We present here the different steps undertaken to progress from PT domain data and information to formulate it as a RL problem represented in the form of POMDP environments. We present an extensive explanation on actions, states, and observations' definition using the motivational network we presented in the previous section as illustrative example. In addition, we will allocate a section to explain the rewarding calculation processes developed for this modelling purpose. Finally, we will present a novel hierarchical RL model which was introduced late in this research as result of the poor performances of solving large POMDP and difficulties in extracting expertise to enable future reuse [23-24].

5.1 Explanatory network example

We introduce here one of the 53 created different size networks which includes 50 machines that we will use as an explanatory example to introduce our proposed RL model of network PT practice. This network (figure 28) is a typical small and medium enterprise (SME) or regional corporate network with DMZ including WEB, DNS and GFI servers, internal storage including mail and data servers and other subnets notably sensitive internal storage, production, Bring Your Own Device (BYOD), Microsoft HypeV, and VMWare ESX server along with employees' workspaces. We opted for such rich and complex network to provide detailed explanation of the proposed RL model and POMDP representation of PT practice within IAPTF. In total, this network includes 46 physical machines and 4 virtual machines consisting of computers, servers, networking devices such as routers and switches in addition to security devices that run an Operating System such as firewalls and IDPSs.

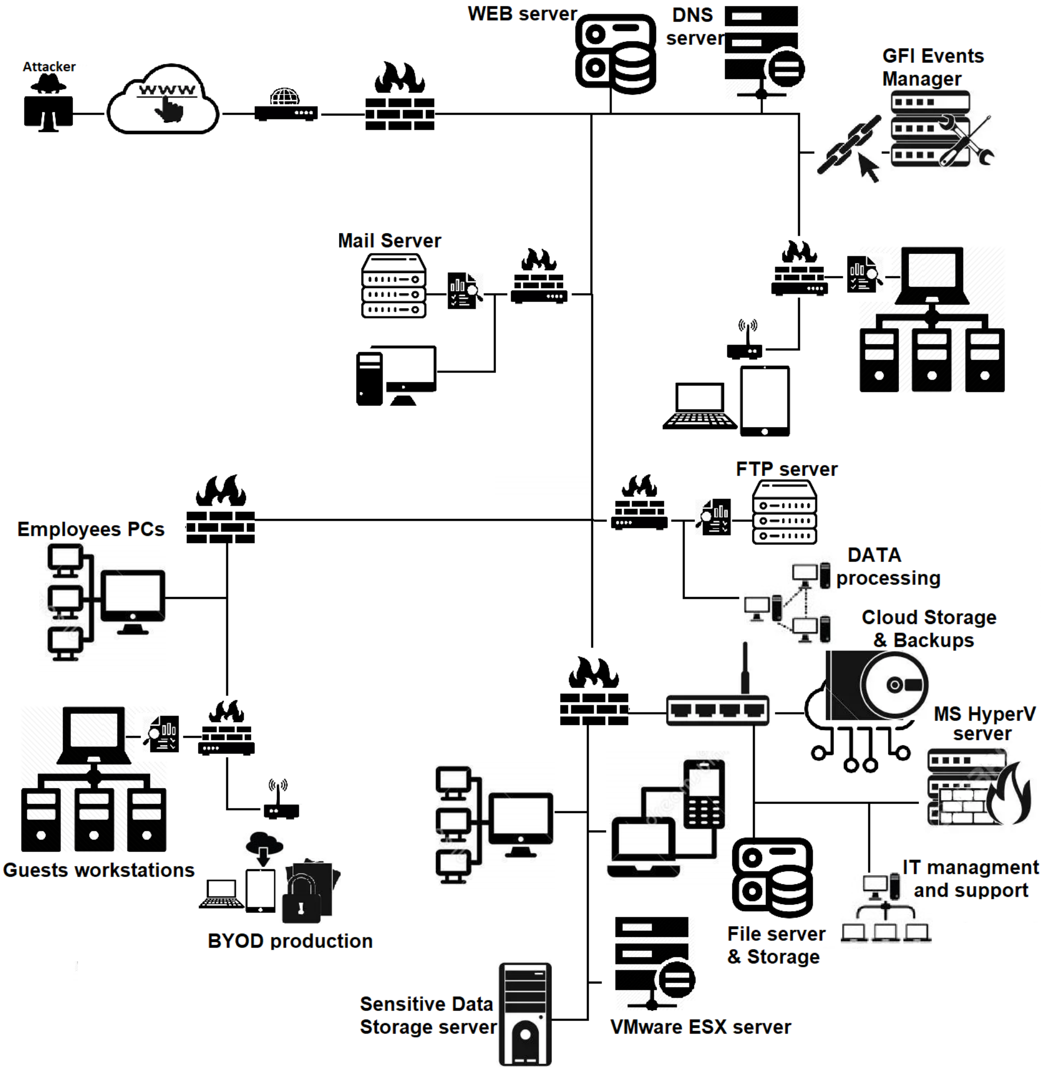


Figure 28: Medium size corporate LAN composed of 50 machines.

5.2 From PT data to POMDP environments

We describe here the full process of elaborating an RL environment for a sample PT starting from a given explanatory medium LAN example. The overall extraction and elaboration process is explained is the results of our domain understanding and PT activities, tasks and sub-tasks details grasping and the proposed model and consequently its POMDP formulation mirror the entire PT practice as explained in the functional diagram illustrated in Figure 29 bellow.

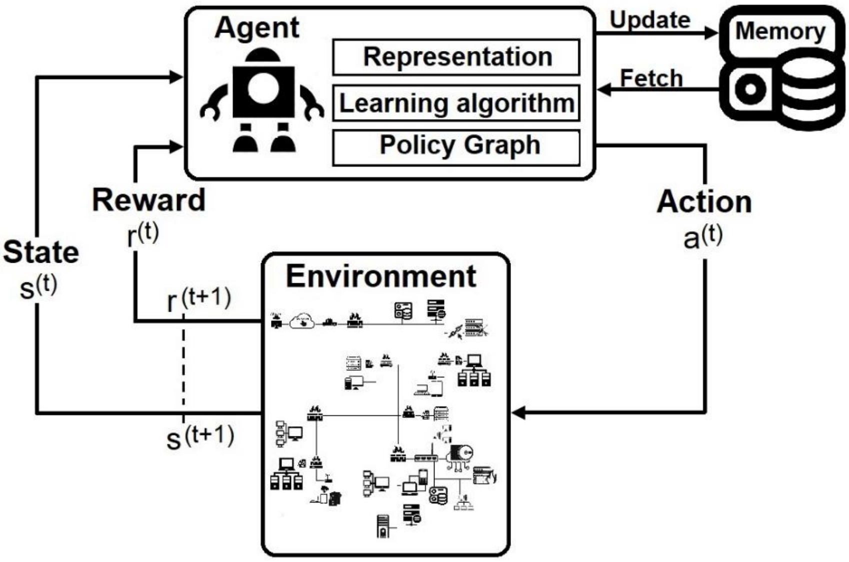


Figure 29: IAPTF reinforcement learning and memory management diagram.

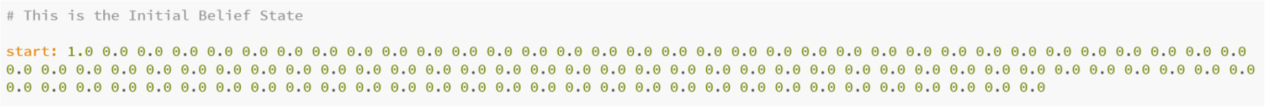
We also highlight the dynamic nature of the formulated POMDP and the associated frequent changes as the PT progress and information used in the elaboration are updated or upgraded. The RL POMDP environment is made from:

- States Space
- Actions Space
- Initial Belief
- Observation Space
- Transitions Probabilities
- Observations Probabilities
- Rewards

In addition, each POMDP file will include information about the value of the discount rate (factor) which is a real value between 0 and 1. At early stage of the model elaboration we attempted to determine the best discount rate value which guarantees the balance between efficiency and effectiveness or the RL solving algorithm. We thus tested five possible discount rates “0.99”, “0.95” “0.9” and “0.7” to finally settle on the discount rate of “0.95”. Finally, the POMDP files header also includes detail about the adopted solving approach, and we opted for reward as values.

5.2.1 Initial Belief

Previous test data is directly used to artificially boost the POMDP problem solving when the pre-processing output indicate a certain degree of similarity with at least one previous test. This operation is performed by a script part of the IAPTF-Prep module which compares current IAPTF-PrepOut.txt file with stored text files stored in IAPTF-Memory. This comparison aims to identify similarity in machine configuration such as OS versions, service pack/ version, open ports and running services. During IAPTF early lifecycle, this operation is optional and can be adapted by the human CEH who will rely on their expertise to only include the adequate data and discard the rest of the data. It is important to highlight that the output of the past testing either successful, failure or incomplete is directly fed to POMOP Initial Belief after being adapted to the new environment along with current information gathering and discovery data. In case of retesting process, IAPTF-Memory will directly import the data from the last testing output [24]. As the aim is to replace human expert in PT, the framework memory was built around the idea of favoring automation over the human expert which is left with the task of dealing with failure into performing some PT tasks or successfully carrying out tests. Similar to CEH operative mode, IAPTF includes an internal evaluation logging procedure to recognize that what has been done could be useful in another context or with minor amendments when required by CEH. Also, CEH will initially, 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, IAPTF will be mainly dealing with the policies stored into the PG file which constitutes the outcome of the POMDP problem solver [123]. Below is an example of POMDP initial belief for an environment of 100 states and where the only known information refereed by 1.0 value is the first state which is Internet. This reflects a Blackbox PT practice when the tester has no prior knowledge about the assessed network.



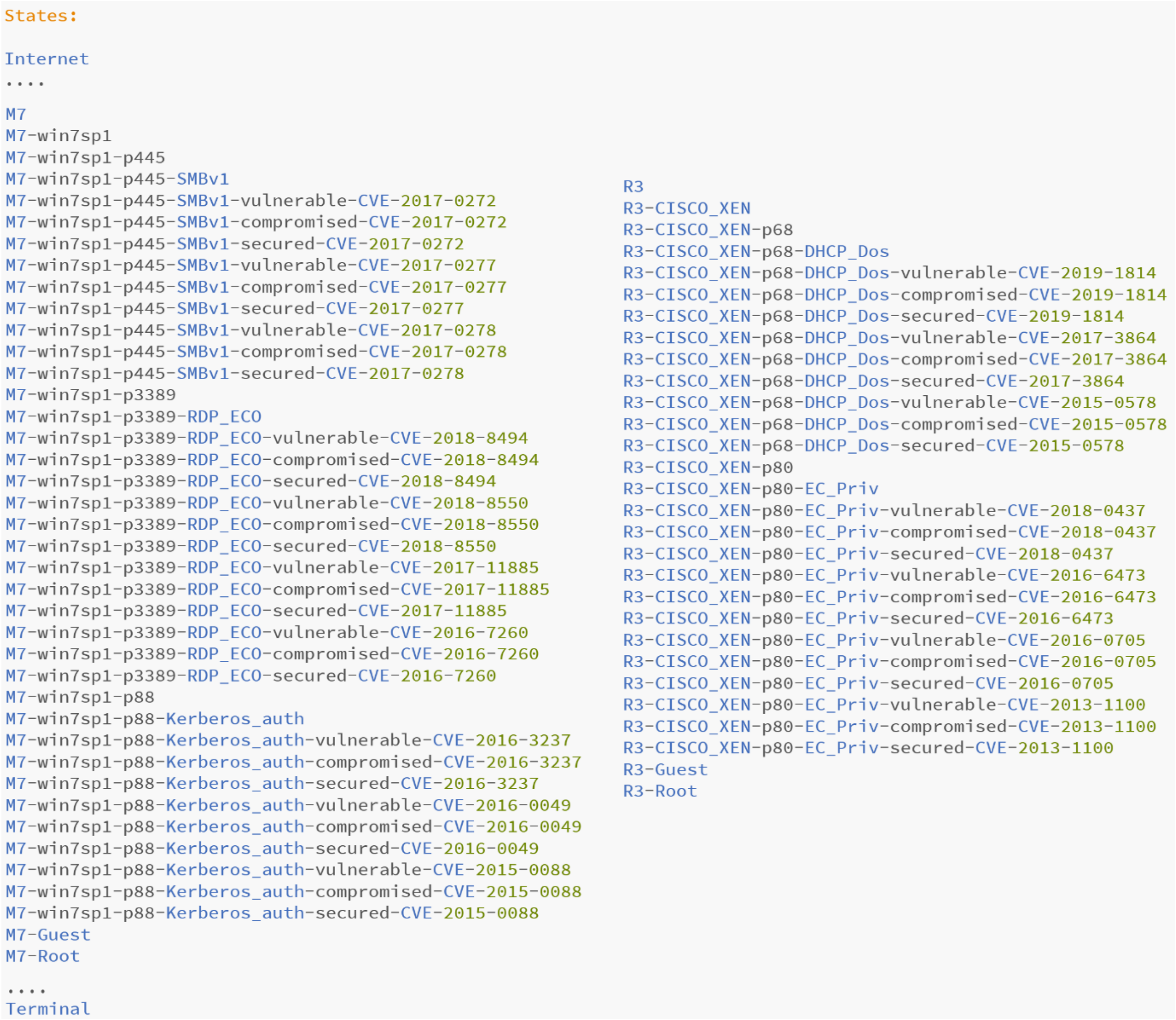
```
# This is the Initial Belief State

start: 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
```

Figure 30: Initial Belief state when nothing is known for the RL agent, and the starting point is Internet with a value of 1.0

5.2.2 State space

The state space S is defined as the collection of all known states for each machine, networking, or security components constituting the network including virtual appliances and cloud backup servers. In other words, a machine is any appliance running an OS and using services and applications. S 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, OS patches and updates. In addition, S contains 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 succinct. In practice, most of the State space is elaborated at an early stage mimicking modern PT techniques relying on strong discovery and starting knowledge to feed CEHs and automated system before launching the initial information gathered 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 [23].

Any machine or connecting device within the network will be assigned a unique number "i" and will be represented as either 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. The information initially represented are often updated as the discovering and scanning tasks progress to confirm previous probabilistic information or to add a new one. Furthermore, network routers are considered to be more than just transmission equipment, in fact, they can run operating systems and embed one or more security isolation and protection mechanisms notably FWs (firewalls), AVs (Anti-Viruses), IDPSs (intrusion detection/prevention systems), VLANs (virtual LANs) and others. Following this logic, network and firewalls are either considered as machines (running OS and thus having vulnerabilities) or simply as security isolation delimiter for clustering purposes. The figure 31summarizes the state space representation for two M7 computer and R3 router. We included the generic start state "internet" from which the hacker and ethical hacker start the PT along with the last state "Terminal" which, when reached, the PT is completed.

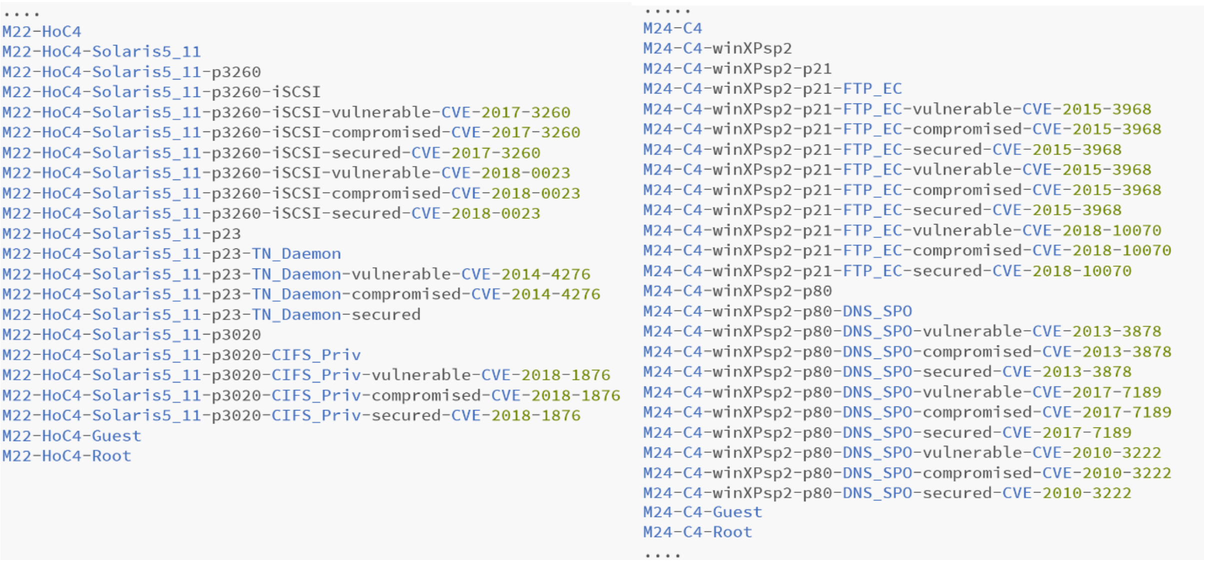


```
States:

Internet
....
M7
M7-win7sp1
M7-win7sp1-p445
M7-win7sp1-p445-SMBv1
M7-win7sp1-p445-SMBv1-vulnerable-CVE-2017-0272
M7-win7sp1-p445-SMBv1-compromised-CVE-2017-0272
M7-win7sp1-p445-SMBv1-secured-CVE-2017-0272
M7-win7sp1-p445-SMBv1-vulnerable-CVE-2017-0277
M7-win7sp1-p445-SMBv1-compromised-CVE-2017-0277
M7-win7sp1-p445-SMBv1-secured-CVE-2017-0277
M7-win7sp1-p445-SMBv1-vulnerable-CVE-2017-0278
M7-win7sp1-p445-SMBv1-compromised-CVE-2017-0278
M7-win7sp1-p445-SMBv1-secured-CVE-2017-0278
M7-win7sp1-p3389
M7-win7sp1-p3389-RDP_ECO
M7-win7sp1-p3389-RDP_ECO-vulnerable-CVE-2018-8494
M7-win7sp1-p3389-RDP_ECO-compromised-CVE-2018-8494
M7-win7sp1-p3389-RDP_ECO-secured-CVE-2018-8494
M7-win7sp1-p3389-RDP_ECO-vulnerable-CVE-2018-8550
M7-win7sp1-p3389-RDP_ECO-compromised-CVE-2018-8550
M7-win7sp1-p3389-RDP_ECO-secured-CVE-2018-8550
M7-win7sp1-p3389-RDP_ECO-vulnerable-CVE-2017-11885
M7-win7sp1-p3389-RDP_ECO-compromised-CVE-2017-11885
M7-win7sp1-p3389-RDP_ECO-secured-CVE-2017-11885
M7-win7sp1-p3389-RDP_ECO-vulnerable-CVE-2016-7260
M7-win7sp1-p3389-RDP_ECO-compromised-CVE-2016-7260
M7-win7sp1-p3389-RDP_ECO-secured-CVE-2016-7260
M7-win7sp1-p88
M7-win7sp1-p88-Kerberos_auth
M7-win7sp1-p88-Kerberos_auth-vulnerable-CVE-2016-3237
M7-win7sp1-p88-Kerberos_auth-compromised-CVE-2016-3237
M7-win7sp1-p88-Kerberos_auth-secured-CVE-2016-3237
M7-win7sp1-p88-Kerberos_auth-vulnerable-CVE-2016-0049
M7-win7sp1-p88-Kerberos_auth-compromised-CVE-2016-0049
M7-win7sp1-p88-Kerberos_auth-secured-CVE-2016-0049
M7-win7sp1-p88-Kerberos_auth-vulnerable-CVE-2015-0088
M7-win7sp1-p88-Kerberos_auth-compromised-CVE-2015-0088
M7-win7sp1-p88-Kerberos_auth-secured-CVE-2015-0088
M7-Guest
M7-Root

....
Terminal

R3
R3-CISCO_XEN
R3-CISCO_XEN-p68
R3-CISCO_XEN-p68-DHCP_Dos
R3-CISCO_XEN-p68-DHCP_Dos-vulnerable-CVE-2019-1814
R3-CISCO_XEN-p68-DHCP_Dos-compromised-CVE-2019-1814
R3-CISCO_XEN-p68-DHCP_Dos-secured-CVE-2019-1814
R3-CISCO_XEN-p68-DHCP_Dos-vulnerable-CVE-2017-3864
R3-CISCO_XEN-p68-DHCP_Dos-compromised-CVE-2017-3864
R3-CISCO_XEN-p68-DHCP_Dos-secured-CVE-2017-3864
R3-CISCO_XEN-p68-DHCP_Dos-vulnerable-CVE-2015-0578
R3-CISCO_XEN-p68-DHCP_Dos-compromised-CVE-2015-0578
R3-CISCO_XEN-p68-DHCP_Dos-secured-CVE-2015-0578
R3-CISCO_XEN-p80
R3-CISCO_XEN-p80-EC_Priv
R3-CISCO_XEN-p80-EC_Priv-vulnerable-CVE-2018-0437
R3-CISCO_XEN-p80-EC_Priv-compromised-CVE-2018-0437
R3-CISCO_XEN-p80-EC_Priv-secured-CVE-2018-0437
R3-CISCO_XEN-p80-EC_Priv-vulnerable-CVE-2016-6473
R3-CISCO_XEN-p80-EC_Priv-compromised-CVE-2016-6473
R3-CISCO_XEN-p80-EC_Priv-secured-CVE-2016-6473
R3-CISCO_XEN-p80-EC_Priv-vulnerable-CVE-2016-0705
R3-CISCO_XEN-p80-EC_Priv-compromised-CVE-2016-0705
R3-CISCO_XEN-p80-EC_Priv-secured-CVE-2016-0705
R3-CISCO_XEN-p80-EC_Priv-vulnerable-CVE-2013-1100
R3-CISCO_XEN-p80-EC_Priv-compromised-CVE-2013-1100
R3-CISCO_XEN-p80-EC_Priv-secured-CVE-2013-1100
R3-Guest
R3-Root
```

Figure 31: POMDP State space representation of two machines; computer M7 and router R3.

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 enable future hierarchical POMDP representation which will be discussed later in this chapter. The following example summarises the information captured about two machines Mi and Mj as Mi-Mj-TCP-SSH-0". In a later stage, as we introduced the security clustering method to divide the network into a set of security cluster, we added to the current POMDP representation the security cluster number j (Cj) to which the machine or networking device belong. Furthermore, the representation accounts for the Head-of-Cluster by adding the

HoCj annotation and the relevant security and networking configurations after the Machine ID as illustrated in figure 32 below:



```
....
M22-HoC4
M22-HoC4-Solaris5_11
M22-HoC4-Solaris5_11-p3260
M22-HoC4-Solaris5_11-p3260-iSCSI
M22-HoC4-Solaris5_11-p3260-iSCSI-vulnerable-CVE-2017-3260
M22-HoC4-Solaris5_11-p3260-iSCSI-compromised-CVE-2017-3260
M22-HoC4-Solaris5_11-p3260-iSCSI-secured-CVE-2017-3260
M22-HoC4-Solaris5_11-p3260-iSCSI-vulnerable-CVE-2018-0023
M22-HoC4-Solaris5_11-p3260-iSCSI-compromised-CVE-2018-0023
M22-HoC4-Solaris5_11-p3260-iSCSI-secured-CVE-2018-0023
M22-HoC4-Solaris5_11-p23
M22-HoC4-Solaris5_11-p23-TN_Daemon
M22-HoC4-Solaris5_11-p23-TN_Daemon-vulnerable-CVE-2014-4276
M22-HoC4-Solaris5_11-p23-TN_Daemon-compromised-CVE-2014-4276
M22-HoC4-Solaris5_11-p23-TN_Daemon-secured
M22-HoC4-Solaris5_11-p3020
M22-HoC4-Solaris5_11-p3020-CIFS_Priv
M22-HoC4-Solaris5_11-p3020-CIFS_Priv-vulnerable-CVE-2018-1876
M22-HoC4-Solaris5_11-p3020-CIFS_Priv-compromised-CVE-2018-1876
M22-HoC4-Solaris5_11-p3020-CIFS_Priv-secured-CVE-2018-1876
M22-HoC4-Guest
M22-HoC4-Root
```

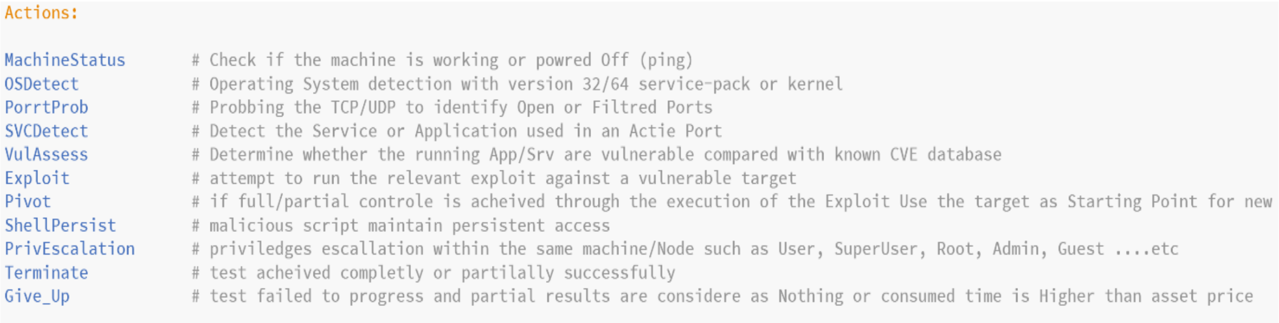
```
.....
M24-C4
M24-C4-winXPsp2
M24-C4-winXPsp2-p21
M24-C4-winXPsp2-p21-FTP_EC
M24-C4-winXPsp2-p21-FTP_EC-vulnerable-CVE-2015-3968
M24-C4-winXPsp2-p21-FTP_EC-compromised-CVE-2015-3968
M24-C4-winXPsp2-p21-FTP_EC-secured-CVE-2015-3968
M24-C4-winXPsp2-p21-FTP_EC-vulnerable-CVE-2015-3968
M24-C4-winXPsp2-p21-FTP_EC-compromised-CVE-2015-3968
M24-C4-winXPsp2-p21-FTP_EC-secured-CVE-2015-3968
M24-C4-winXPsp2-p21-FTP_EC-vulnerable-CVE-2018-10070
M24-C4-winXPsp2-p21-FTP_EC-compromised-CVE-2018-10070
M24-C4-winXPsp2-p21-FTP_EC-secured-CVE-2018-10070
M24-C4-winXPsp2-p80
M24-C4-winXPsp2-p80-DNS_SPO
M24-C4-winXPsp2-p80-DNS_SPO-vulnerable-CVE-2013-3878
M24-C4-winXPsp2-p80-DNS_SPO-compromised-CVE-2013-3878
M24-C4-winXPsp2-p80-DNS_SPO-secured-CVE-2013-3878
M24-C4-winXPsp2-p80-DNS_SPO-vulnerable-CVE-2017-7189
M24-C4-winXPsp2-p80-DNS_SPO-compromised-CVE-2017-7189
M24-C4-winXPsp2-p80-DNS_SPO-secured-CVE-2017-7189
M24-C4-winXPsp2-p80-DNS_SPO-vulnerable-CVE-2010-3222
M24-C4-winXPsp2-p80-DNS_SPO-compromised-CVE-2010-3222
M24-C4-winXPsp2-p80-DNS_SPO-secured-CVE-2010-3222
M24-C4-Guest
M24-C4-Root
....
```

Figure 32: POMDP State space representation including security clustering and Head-of-Cluster information.

Finally, the networking information are not static and thus we opted to represent it within the POMDP observations to reflect the real-world situation [23-24].

5.2.3 Action space

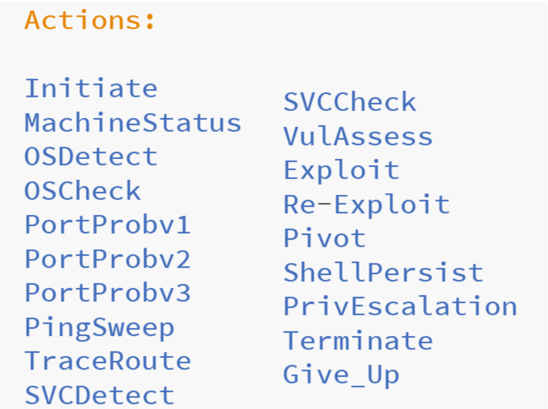
The action space A is the set of available actions that were deducted from our study of activities, tasks and sub-tasks performed by the pentesters and hackers. We elaborated the network POMDP model action space in a reflection of the sub-tasks performed by pentesters but in a structured way. We accounted for all PT tasks and sub-tasks following a concise annotation. As with any RL problem, the number of actions is known, static and limited and PT does not fall out of this logic. Initially, we opted for more generic actions that fits all type of activity without differentiate between doing the task for the first time or repeating it. We therefore proposed an Action space made from 11 actions namely MachineStatus, OSDetect PortProb, SVCDetect, VulnAssess, Exploit, Pivot ShellPersist, PrivEscalation, in addition to some generic action that will be used for control purpose by RL agent namely Terminate and Give_Up. Figure 33 illustrates the 11 action and brief description of each action.



```
Actions:

MachineStatus       # Check if the machine is working or powred Off (ping)
OSDetect            # Operating System detection with version 32/64 service-pack or kernel
PorrtProb           # Probbing the TCP/UDP to identify Open or Filtred Ports
SVCDetect           # Detect the Service or Application used in an Actie Port
VulAssess           # Determine whether the running App/Srv are vulnerable compared with known CVE database
Exploit             # attempt to run the relevant exploit against a vulnerable target
Pivot               # if full/partial controle is acheived through the execution of the Exploit Use the target as Starting Point for new
ShellPersist        # malicious script maintain persistent access
PrivEscalation      # priviledges escallation within the same machine/Node such as User, SuperUser, Root, Admin, Guest ....etc
Terminate           # test acheived completly or partilally successfully
Give_Up             # test failed to progress and partial results are considere as Nothing or consumed time is Higher than asset price
```

Figure 33: POMDP Actions space made from 11 specific actions and 2 generic actions

We then extended the action space to include 19 in total to reflect the real-world when different Probing levels which are different in term of intensity and reflection the Nmap software predefined probing profile respectively ProtProbv1 for NULL probing, ProtProbv2 for regular probing and ProtProbv3 for comprehensive and slow probing. We also added the retesting actions such as OSCheck for rechecking operating system and versions, PingSweep and TraceRoute for advanced discovery, SVCCheck for rechecking service detection and Re-Exploit for re-attempting the exploit with or without changing parameters as summarized in figure 34 .



```
Actions:

Initiate
MachineStatus   SVCCheck
OSDetect        VulAssess
OSCheck         Exploit
PortProbv1      Re-Exploit
PortProbv2      Pivot
PortProbv3      ShellPersist
PingSweep       PrivEscalation
TraceRoute      Terminate
SVCDetect       Give_Up
```

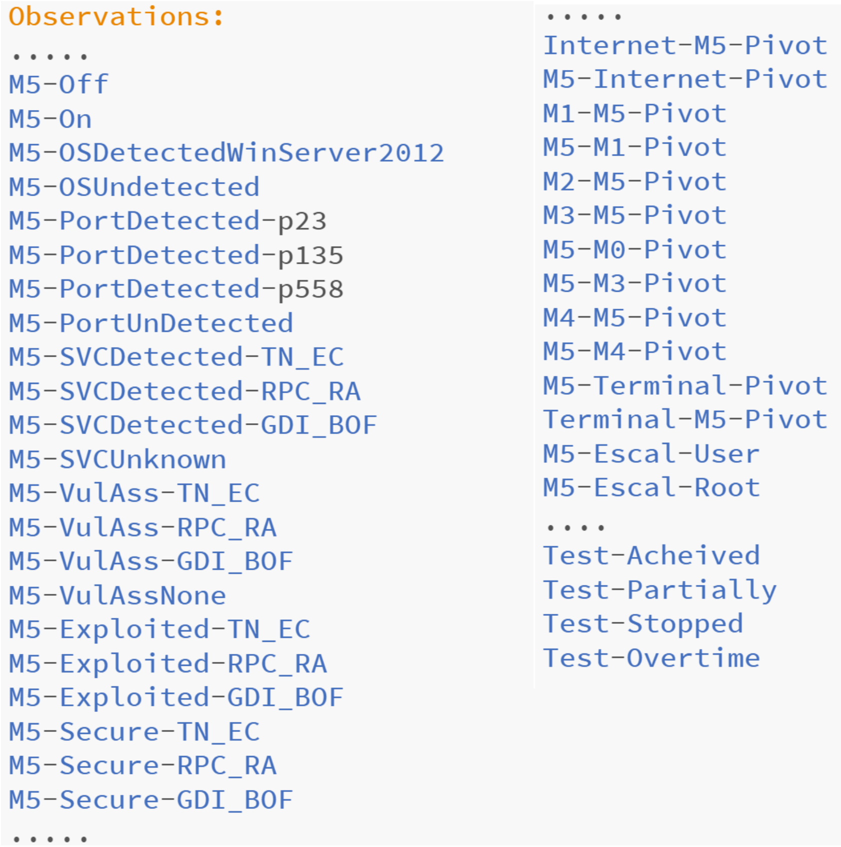
Figure 34: POMDP extended Actions space made from 17 specific actions and 2 generic actions

The number of actions that the expert can perform is huge and cannot be totally represented within the RL action space which led us to introduce these limited contextual actions. Furthermore, as in PT domain successful or failed action might require further or repeating actions, we defined some additional actions to differentiate between the original action and the other action. In practice, the purpose of such representation is to deal with the special and complex scenarios notably:

- 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, sub-tasks that 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.

5.2.4 Observation space

The observations space O is the set of available observations we deducted from the non-confirmed states. In a decision process, observations provide information to the decision maker for deciding the future course of action [88]. We propose to represent a comprehensive observation space which include all possible observations, this will enable our RL agent to establish and devise rich policies with a different course of action for each possible observation which ultimately will tend to select the same course of action for many different observations that share similar features. In our model we represent different observations for each machine such as the status Mi-Off when turned off and Mi-On when machine ais running. In addition, we also add four generic observations which reflect observation used by the RL agent to tackle some situations and ends the test respectively: Test-Achieved, Test-Partially, Test-Stopped and Test-Overtime. Finally, the networking and reachability data is modelled in form of pivoting observation. Detailed observation space representation is provided in figure 35.



```
Observations:
.....
M5-Off
M5-On
M5-OSDetectedWinServer2012
M5-OSUndetected
M5-PortDetected-p23
M5-PortDetected-p135
M5-PortDetected-p558
M5-PortUnDetected
M5-SVCDetected-TN_EC
M5-SVCDetected-RPC_RA
M5-SVCDetected-GDI_BOF
M5-SVCUnknown
M5-VulAss-TN_EC
M5-VulAss-RPC_RA
M5-VulAss-GDI_BOF
M5-VulAssNone
M5-Exploited-TN_EC
M5-Exploited-RPC_RA
M5-Exploited-GDI_BOF
M5-Secure-TN_EC
M5-Secure-RPC_RA
M5-Secure-GDI_BOF
.....
```

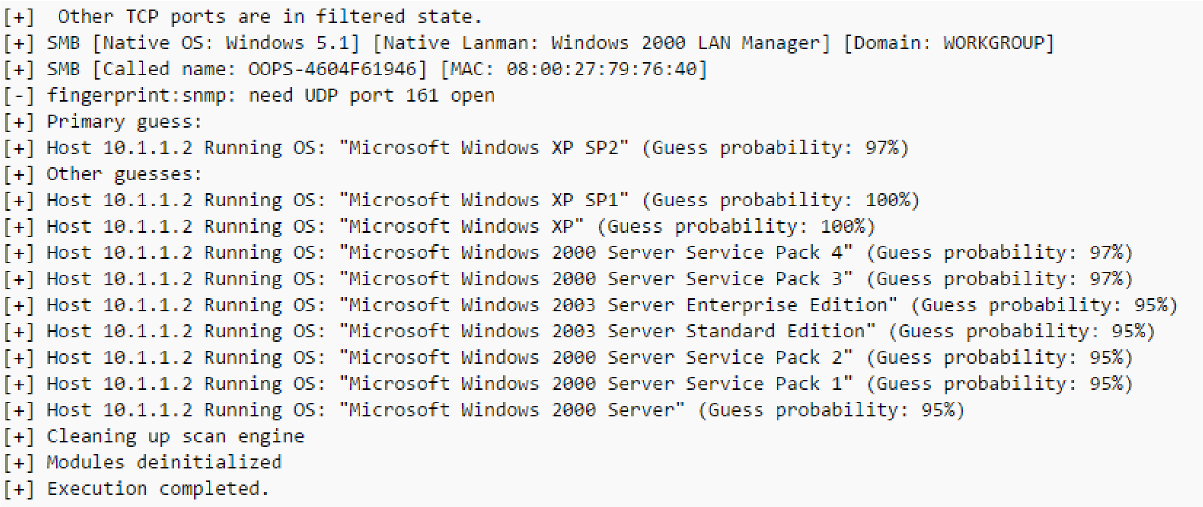
```
.....
Internet-M5-Pivot
M5-Internet-Pivot
M1-M5-Pivot
M5-M1-Pivot
M2-M5-Pivot
M3-M5-Pivot
M5-M0-Pivot
M5-M3-Pivot
M4-M5-Pivot
M5-M4-Pivot
M5-Terminal-Pivot
Terminal-M5-Pivot
M5-Escal-User
M5-Escal-Root
....
Test-Acheived
Test-Partially
Test-Stopped
Test-Overtime
```

Figure 35: POMDP extended Actions space made from 17 specific actions and 2 generic actions

The observations space reflects the probabilistic nature of PT practice where states are not always deterministic notably as results of action (scanning, fingerprinting, exploiting) which made us adopt the model-based approach and allocating the adequate probabilities for Transitions and Observations in order to mirror the real-world PT practice [130-133].

5.2.5 Transitions and Observations Probabilities

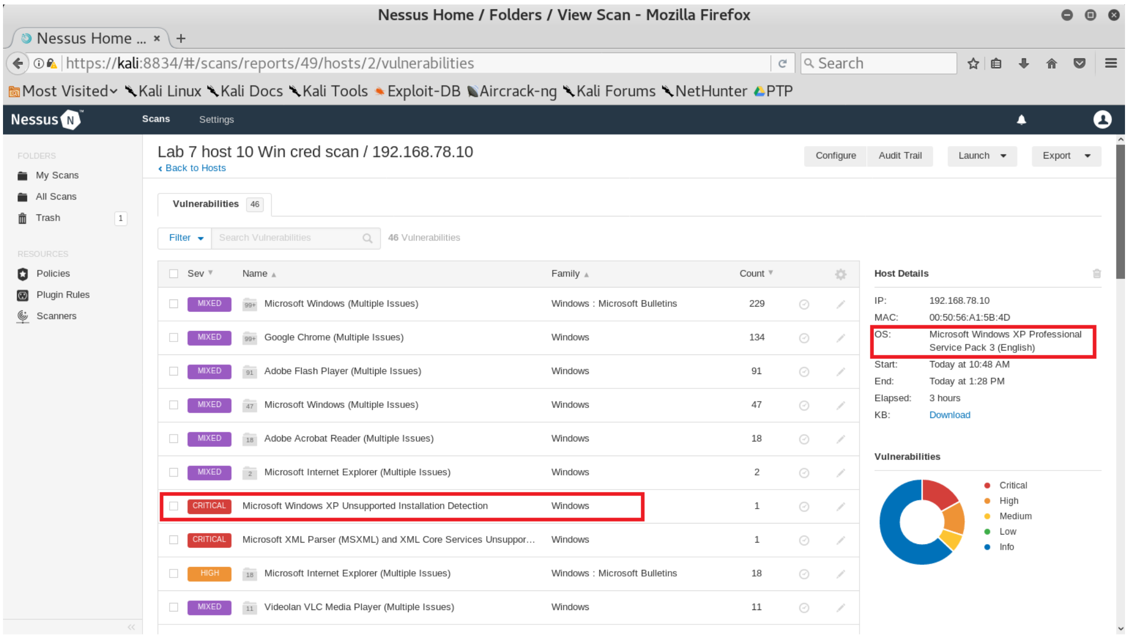
In this section, we describe and illustrate the transition and observation probabilities calculation based on real CVE and NVD data and mathematical formulation. Initially, all transitions and observations probabilities were uniformly sampled in the form **T: * : * : * X** and **O: * : * : * Y** with probabilities X= 1/total number of states and Y= 1/ total number of observations. For a realistic Transitions and Observations probabilities calculation we considered multiple approaches. We settled on the most appropriate calculation method based on Nmap NSE and Nessus output to determine Transition and Observation probabilities for each action. We will be relying on vulnerability assessment output (NMAP and NESSUS) and other sources to define vulnerabilities discovery and information gathering and Cyber Threat Intelligence. We rely om two major online databases VulDB and NESSUS Auditing DB. The starting point in transition and observation probabilities calculation is the IAPTF-Prep output covering information gathering (scanning and fingerprinting) and discovery which are mainly resulting from Nmap scanning results as illustrated in figure 36.



```
[+]  Other TCP ports are in filtered state.
[+] SMB [Native OS: Windows 5.1] [Native Lanman: Windows 2000 LAN Manager] [Domain: WORKGROUP]
[+] SMB [Called name: OOPS-4604F61946] [MAC: 08:00:27:79:76:40]
[-] fingerprint:snmp: need UDP port 161 open
[+] Primary guess:
[+] Host 10.1.1.2 Running OS: "Microsoft Windows XP SP2" (Guess probability: 97%)
[+] Other guesses:
[+] Host 10.1.1.2 Running OS: "Microsoft Windows XP SP1" (Guess probability: 100%)
[+] Host 10.1.1.2 Running OS: "Microsoft Windows XP" (Guess probability: 100%)
[+] Host 10.1.1.2 Running OS: "Microsoft Windows 2000 Server Service Pack 4" (Guess probability: 97%)
[+] Host 10.1.1.2 Running OS: "Microsoft Windows 2000 Server Service Pack 3" (Guess probability: 97%)
[+] Host 10.1.1.2 Running OS: "Microsoft Windows 2003 Server Enterprise Edition" (Guess probability: 95%)
[+] Host 10.1.1.2 Running OS: "Microsoft Windows 2003 Server Standard Edition" (Guess probability: 95%)
[+] Host 10.1.1.2 Running OS: "Microsoft Windows 2000 Server Service Pack 2" (Guess probability: 95%)
[+] Host 10.1.1.2 Running OS: "Microsoft Windows 2000 Server Service Pack 1" (Guess probability: 95%)
[+] Host 10.1.1.2 Running OS: "Microsoft Windows 2000 Server" (Guess probability: 95%)
[+] Cleaning up scan engine
[+] Modules deinitialized
[+] Execution completed.
```

Figure 36: Nmap NSE OS detection and fingerprinting sample output

In addition, we correlate the Nmap data with NESSUS vulnerability assessment output as illustrated in figure37 below. As highlighted, the VA output confirms the OS detection and advance that the assessed machine is running Windows XP Service Pack 3.



Lab 7 host 10 Win cred scan / 192.168.78.10

‹ Back to Hosts

Sev	Name	Family	Count
MIXED	229 Microsoft Windows (Multiple Issues)	Windows : Microsoft Bulletins	229
MIXED	99+ Google Chrome (Multiple Issues)	Windows	134
MIXED	91 Adobe Flash Player (Multiple Issues)	Windows	91
MIXED	47 Microsoft Windows (Multiple Issues)	Windows	47
MIXED	18 Adobe Acrobat Reader (Multiple Issues)	Windows	18
MIXED	2 Microsoft Internet Explorer (Multiple Issues)	Windows	2
CRITICAL	Microsoft Windows XP Unsupported Installation Detection	Windows	1
CRITICAL	Microsoft XML Parser (MSXML) and XML Core Services Unsuppor…	Windows	1
HIGH	18 Microsoft Internet Explorer (Multiple Issues)	Windows : Microsoft Bulletins	18
MIXED	11 Videolan VLC Media Player (Multiple Issues)	Windows	11

Host Details

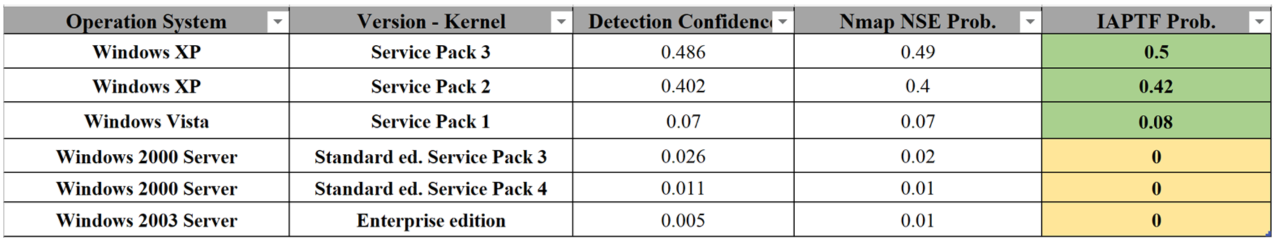
IP:	192.168.78.10
MAC:	00:50:56:A1:5B:4D
OS:	Microsoft Windows XP Professional Service Pack 3 (English)
Start:	Today at 10:48 AM
End:	Today at 1:28 PM
Elapsed:	3 hours
KB:	Download

Vulnerabilities

Figure 37: NESSUS vulnerability assessment output and OS detection validation

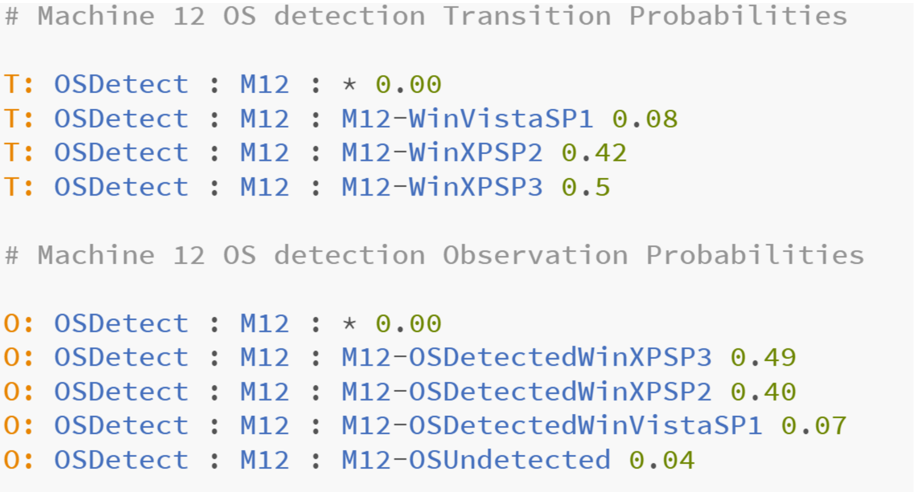
From all data available we create the following OS fingerprinting table which combines and collates all data. For efficiency reason, we artificially edit the probabilities to only leave a maximum of three possible OS out of six which reflect highest three probabilities and we discard the remaining ones as illustrated in the table 2. We provide two examples summarising IAPTF OS detection and fingerprinting in two major OS namely Microsoft Windows and Apple Mac OS. For the windows machine, only Windows XP SP3, Windows Vista SP1 and SP2 options are considered with probability respectively 0.75, 0.15 and 0.1.in Mac OS machine, only three possible OS out of eleven are maintained as summarised in table 2. note that the choice of maximal three options is purely functional and aims to reduce the size of the POMDP on one hand and enable a better solving convergence on the other hand. In practice, it is highly unlikely that a machine OS detection with probability of 0.1 is an accurate detection.

Table 6: OS detection output with transition and observation probabilities calculation example in Windows.



Operation System	Version - Kernel	Detection Confidenc	Nmap NSE Prob.	IAPTF Prob.
Windows XP	Service Pack 3	0.486	0.49	0.5
Windows XP	Service Pack 2	0.402	0.4	0.42
Windows Vista	Service Pack 1	0.07	0.07	0.08
Windows 2000 Server	Standard ed. Service Pack 3	0.026	0.02	0
Windows 2000 Server	Standard ed. Service Pack 4	0.011	0.01	0
Windows 2003 Server	Enterprise edition	0.005	0.01	0

The representation of machine OS and version detection is illustrated in figure 38:

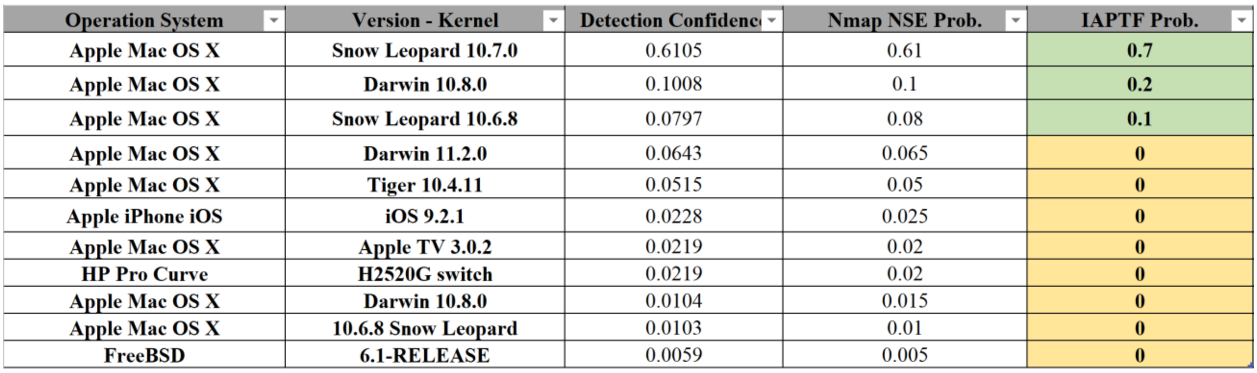


```
# Machine 12 OS detection Transition Probabilities

T: OSDetect : M12 : * 0.00
T: OSDetect : M12 : M12-WinVistaSP1 0.08
T: OSDetect : M12 : M12-WinXPSP2 0.42
T: OSDetect : M12 : M12-WinXPSP3 0.5

# Machine 12 OS detection Observation Probabilities

O: OSDetect : M12 : * 0.00
O: OSDetect : M12 : M12-OSDetectedWinXPSP3 0.49
O: OSDetect : M12 : M12-OSDetectedWinXPSP2 0.40
O: OSDetect : M12 : M12-OSDetectedWinVistaSP1 0.07
O: OSDetect : M12 : M12-OSUndetected 0.04
```

Figure 38: Windows Machine OS detection Transitions and Observations probabilities POMDP representation

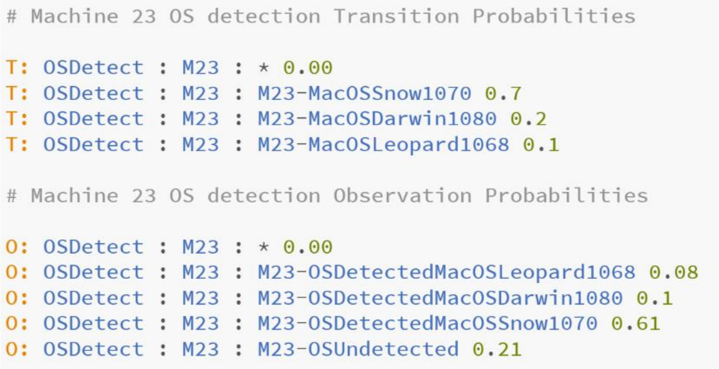
Another example of transition OS detection probabilities in Mac OS is illustrated in table 3.

Table 7: OS detection output with transition and observation probabilities calculation example in Mac OS machine.



Operation System	Version - Kernel	Detection Confidenc	Nmap NSE Prob.	IAPTF Prob.
Apple Mac OS X	Snow Leopard 10.7.0	0.6105	0.61	0.7
Apple Mac OS X	Darwin 10.8.0	0.1008	0.1	0.2
Apple Mac OS X	Snow Leopard 10.6.8	0.0797	0.08	0.1
Apple Mac OS X	Darwin 11.2.0	0.0643	0.065	0
Apple Mac OS X	Tiger 10.4.11	0.0515	0.05	0
Apple iPhone iOS	iOS 9.2.1	0.0228	0.025	0
Apple Mac OS X	Apple TV 3.0.2	0.0219	0.02	0
HP Pro Curve	H2520G switch	0.0219	0.02	0
Apple Mac OS X	Darwin 10.8.0	0.0104	0.015	0
Apple Mac OS X	10.6.8 Snow Leopard	0.0103	0.01	0
FreeBSD	6.1-RELEASE	0.0059	0.005	0

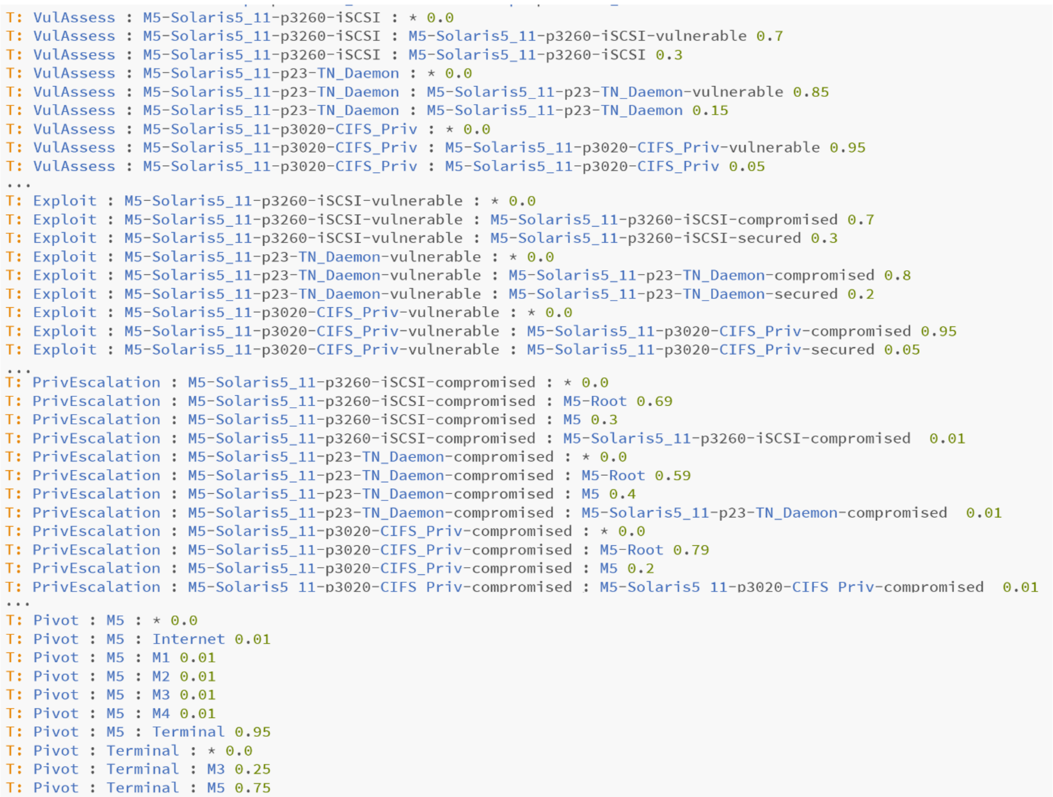
In this case the number of probable OS version is eleven and we narrow it down to three by discarding the least probable OS configuration. Nonetheless, the observation probability for undetected OS version will increase considerably and will represent the sum of the omitted 8 probabilities to reach 0.21 and so 21%. This is the perfect illustration of uncertainty in PT practice and which we reflect perfectly in our POMDP environments. Figure 39 illustrates the POMDP representation for machine number 23 in the explanatory network.



```
# Machine 23 OS detection Transition Probabilities

T: OSDetect : M23 : * 0.00
T: OSDetect : M23 : M23-MacOSSnow1070 0.7
T: OSDetect : M23 : M23-MacOSDarwin1080 0.2
T: OSDetect : M23 : M23-MacOSLeopard1068 0.1

# Machine 23 OS detection Observation Probabilities

O: OSDetect : M23 : * 0.00
O: OSDetect : M23 : M23-OSDetectedMacOSLeopard1068 0.08
O: OSDetect : M23 : M23-OSDetectedMacOSDarwin1080 0.1
O: OSDetect : M23 : M23-OSDetectedMacOSSnow1070 0.61
O: OSDetect : M23 : M23-OSUndetected 0.21
```

Figure 39: MacOS Machine OS detection Transitions and Observations probabilities POMDP representation

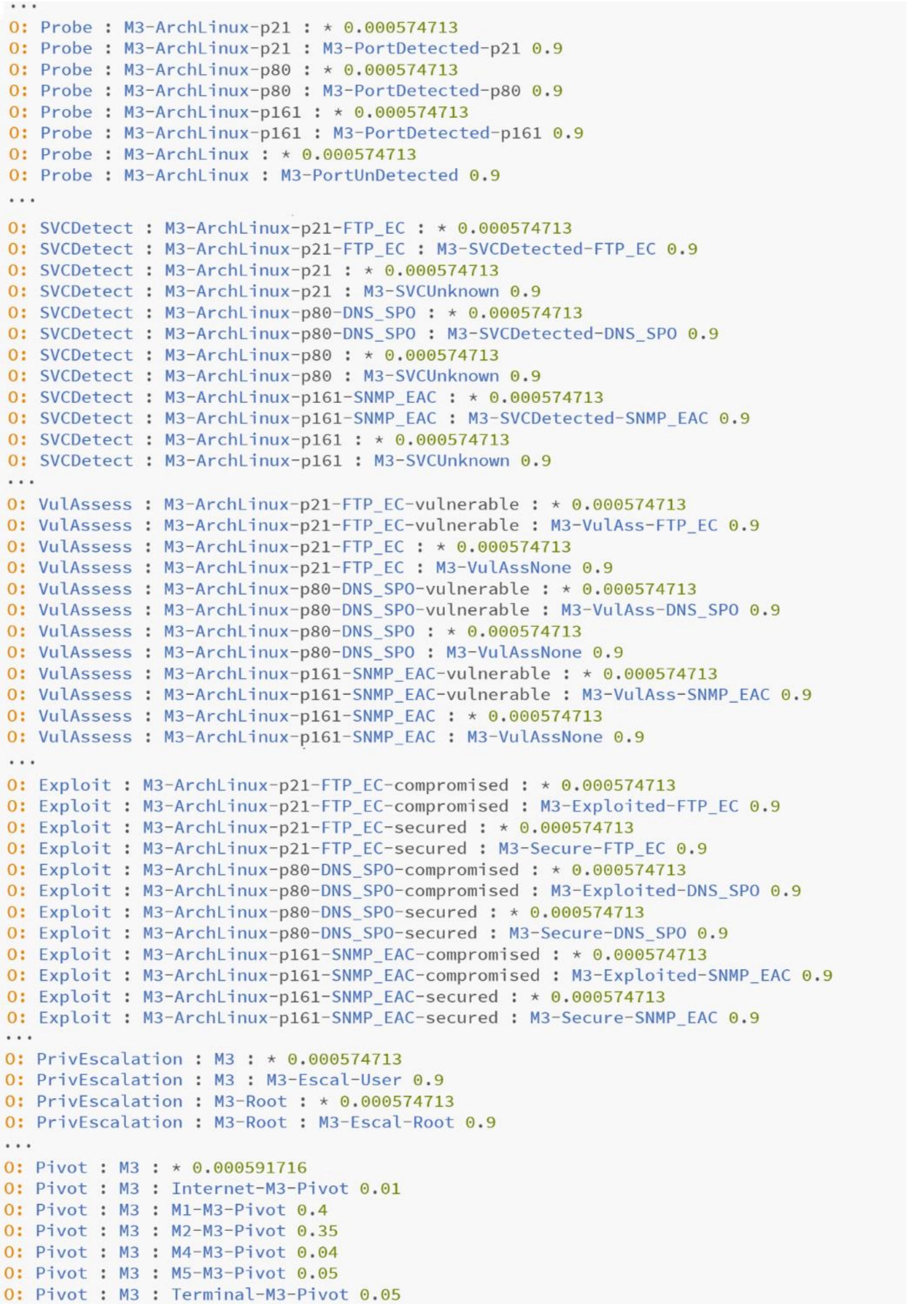
The next step in this part is probabilities allocation for port probing and service detection. This step is less problematic as port have only three statuses: Open, Closed and Filtered. Often, the simplest scan can determine whether the port is closed or not. On few occasions results are unclear as some ports are simply filtered and cannot tell if they are open as illustrated in Figure 40 Transitions.



```
T: VulAssess : M5-Solaris5_11-p3260-iSCSI : * 0.0
T: VulAssess : M5-Solaris5_11-p3260-iSCSI : M5-Solaris5_11-p3260-iSCSI-vulnerable 0.7
T: VulAssess : M5-Solaris5_11-p3260-iSCSI : M5-Solaris5_11-p3260-iSCSI 0.3
T: VulAssess : M5-Solaris5_11-p23-TN_Daemon : * 0.0
T: VulAssess : M5-Solaris5_11-p23-TN_Daemon : M5-Solaris5_11-p23-TN_Daemon-vulnerable 0.85
T: VulAssess : M5-Solaris5_11-p23-TN_Daemon : M5-Solaris5_11-p23-TN_Daemon 0.15
T: VulAssess : M5-Solaris5_11-p3020-CIFS_Priv : * 0.0
T: VulAssess : M5-Solaris5_11-p3020-CIFS_Priv : M5-Solaris5_11-p3020-CIFS_Priv-vulnerable 0.95
T: VulAssess : M5-Solaris5_11-p3020-CIFS_Priv : M5-Solaris5_11-p3020-CIFS_Priv 0.05
...
T: Exploit : M5-Solaris5_11-p3260-iSCSI-vulnerable : * 0.0
T: Exploit : M5-Solaris5_11-p3260-iSCSI-vulnerable : M5-Solaris5_11-p3260-iSCSI-compromised 0.7
T: Exploit : M5-Solaris5_11-p3260-iSCSI-vulnerable : M5-Solaris5_11-p3260-iSCSI-secured 0.3
T: Exploit : M5-Solaris5_11-p23-TN_Daemon-vulnerable : * 0.0
T: Exploit : M5-Solaris5_11-p23-TN_Daemon-vulnerable : M5-Solaris5_11-p23-TN_Daemon-compromised 0.8
T: Exploit : M5-Solaris5_11-p23-TN_Daemon-vulnerable : M5-Solaris5_11-p23-TN_Daemon-secured 0.2
T: Exploit : M5-Solaris5_11-p3020-CIFS_Priv-vulnerable : * 0.0
T: Exploit : M5-Solaris5_11-p3020-CIFS_Priv-vulnerable : M5-Solaris5_11-p3020-CIFS_Priv-compromised 0.95
T: Exploit : M5-Solaris5_11-p3020-CIFS_Priv-vulnerable : M5-Solaris5_11-p3020-CIFS_Priv-secured 0.05
...
T: PrivEscalation : M5-Solaris5_11-p3260-iSCSI-compromised : * 0.0
T: PrivEscalation : M5-Solaris5_11-p3260-iSCSI-compromised : M5-Root 0.69
T: PrivEscalation : M5-Solaris5_11-p3260-iSCSI-compromised : M5 0.3
T: PrivEscalation : M5-Solaris5_11-p3260-iSCSI-compromised : M5-Solaris5_11-p3260-iSCSI-compromised  0.01
T: PrivEscalation : M5-Solaris5_11-p23-TN_Daemon-compromised : * 0.0
T: PrivEscalation : M5-Solaris5_11-p23-TN_Daemon-compromised : M5-Root 0.59
T: PrivEscalation : M5-Solaris5_11-p23-TN_Daemon-compromised : M5 0.4
T: PrivEscalation : M5-Solaris5_11-p23-TN_Daemon-compromised : M5-Solaris5_11-p23-TN_Daemon-compromised  0.01
T: PrivEscalation : M5-Solaris5_11-p3020-CIFS_Priv-compromised : * 0.0
T: PrivEscalation : M5-Solaris5_11-p3020-CIFS_Priv-compromised : M5-Root 0.79
T: PrivEscalation : M5-Solaris5_11-p3020-CIFS_Priv-compromised : M5 0.2
T: PrivEscalation : M5-Solaris5_11-p3020-CIFS_Priv-compromised : M5-Solaris5_11-p3020-CIFS_Priv-compromised  0.01
...
T: Pivot : M5 : * 0.0
T: Pivot : M5 : Internet 0.01
T: Pivot : M5 : M1 0.01
T: Pivot : M5 : M2 0.01
T: Pivot : M5 : M3 0.01
T: Pivot : M5 : M4 0.01
T: Pivot : M5 : Terminal 0.95
T: Pivot : Terminal : * 0.0
T: Pivot : Terminal : M3 0.25
T: Pivot : Terminal : M5 0.75
```

Figure 40: A portion of POMDP Transitions probabilities representation .

Finally, we provide here the Observation probabilities allocation for the rest of actions on which we highlight the fact that we started by allocated a uniformed probabilities for all state-observation which the total sum represents 0.1 in order to allow the RL agent further exploration. In the figure 41, this uniform probability is set to 0.000574713 which reflects the portion of each of the 174 observations in this example.



```
...
O: Probe : M3-ArchLinux-p21 : * 0.000574713
O: Probe : M3-ArchLinux-p21 : M3-PortDetected-p21 0.9
O: Probe : M3-ArchLinux-p80 : * 0.000574713
O: Probe : M3-ArchLinux-p80 : M3-PortDetected-p80 0.9
O: Probe : M3-ArchLinux-p161 : * 0.000574713
O: Probe : M3-ArchLinux-p161 : M3-PortDetected-p161 0.9
O: Probe : M3-ArchLinux : * 0.000574713
O: Probe : M3-ArchLinux : M3-PortUnDetected 0.9
...
O: SVCDetect : M3-ArchLinux-p21-FTP_EC : * 0.000574713
O: SVCDetect : M3-ArchLinux-p21-FTP_EC : M3-SVCDetected-FTP_EC 0.9
O: SVCDetect : M3-ArchLinux-p21 : * 0.000574713
O: SVCDetect : M3-ArchLinux-p21 : M3-SVCUnknown 0.9
O: SVCDetect : M3-ArchLinux-p80-DNS_SPO : * 0.000574713
O: SVCDetect : M3-ArchLinux-p80-DNS_SPO : M3-SVCDetected-DNS_SPO 0.9
O: SVCDetect : M3-ArchLinux-p80 : * 0.000574713
O: SVCDetect : M3-ArchLinux-p80 : M3-SVCUnknown 0.9
O: SVCDetect : M3-ArchLinux-p161-SNMP_EAC : * 0.000574713
O: SVCDetect : M3-ArchLinux-p161-SNMP_EAC : M3-SVCDetected-SNMP_EAC 0.9
O: SVCDetect : M3-ArchLinux-p161 : * 0.000574713
O: SVCDetect : M3-ArchLinux-p161 : M3-SVCUnknown 0.9
...
O: VulAssess : M3-ArchLinux-p21-FTP_EC-vulnerable : * 0.000574713
O: VulAssess : M3-ArchLinux-p21-FTP_EC-vulnerable : M3-VulAss-FTP_EC 0.9
O: VulAssess : M3-ArchLinux-p21-FTP_EC : * 0.000574713
O: VulAssess : M3-ArchLinux-p21-FTP_EC : M3-VulAssNone 0.9
O: VulAssess : M3-ArchLinux-p80-DNS_SPO-vulnerable : * 0.000574713
O: VulAssess : M3-ArchLinux-p80-DNS_SPO-vulnerable : M3-VulAss-DNS_SPO 0.9
O: VulAssess : M3-ArchLinux-p80-DNS_SPO : * 0.000574713
O: VulAssess : M3-ArchLinux-p80-DNS_SPO : M3-VulAssNone 0.9
O: VulAssess : M3-ArchLinux-p161-SNMP_EAC-vulnerable : * 0.000574713
O: VulAssess : M3-ArchLinux-p161-SNMP_EAC-vulnerable : M3-VulAss-SNMP_EAC 0.9
O: VulAssess : M3-ArchLinux-p161-SNMP_EAC : * 0.000574713
O: VulAssess : M3-ArchLinux-p161-SNMP_EAC : M3-VulAssNone 0.9
...
O: Exploit : M3-ArchLinux-p21-FTP_EC-compromised : * 0.000574713
O: Exploit : M3-ArchLinux-p21-FTP_EC-compromised : M3-Exploited-FTP_EC 0.9
O: Exploit : M3-ArchLinux-p21-FTP_EC-secured : * 0.000574713
O: Exploit : M3-ArchLinux-p21-FTP_EC-secured : M3-Secure-FTP_EC 0.9
O: Exploit : M3-ArchLinux-p80-DNS_SPO-compromised : * 0.000574713
O: Exploit : M3-ArchLinux-p80-DNS_SPO-compromised : M3-Exploited-DNS_SPO 0.9
O: Exploit : M3-ArchLinux-p80-DNS_SPO-secured : * 0.000574713
O: Exploit : M3-ArchLinux-p80-DNS_SPO-secured : M3-Secure-DNS_SPO 0.9
O: Exploit : M3-ArchLinux-p161-SNMP_EAC-compromised : * 0.000574713
O: Exploit : M3-ArchLinux-p161-SNMP_EAC-compromised : M3-Exploited-SNMP_EAC 0.9
O: Exploit : M3-ArchLinux-p161-SNMP_EAC-secured : * 0.000574713
O: Exploit : M3-ArchLinux-p161-SNMP_EAC-secured : M3-Secure-SNMP_EAC 0.9
...
O: PrivEscalation : M3 : * 0.000574713
O: PrivEscalation : M3 : M3-Escal-User 0.9
O: PrivEscalation : M3-Root : * 0.000574713
O: PrivEscalation : M3-Root : M3-Escal-Root 0.9
...
O: Pivot : M3 : * 0.000591716
O: Pivot : M3 : Internet-M3-Pivot 0.01
O: Pivot : M3 : M1-M3-Pivot 0.4
O: Pivot : M3 : M2-M3-Pivot 0.35
O: Pivot : M3 : M4-M3-Pivot 0.04
O: Pivot : M3 : M5-M3-Pivot 0.05
O: Pivot : M3 : Terminal-M3-Pivot 0.05
```

Figure 41: A portion of POMDP Observations probabilities representation.

5.2.6 IAPTF Rewarding Scheme:

This section details the reward calculation and allocation approach adopted in the IAPTF generated POMDP environments. We present here two approaches: reward calculation and reward allocation. The reward calculation applies for exploitation and post-exploitation (privileges escalation and pivoting) activities and will rely on a well-defined mathematical formulation using IAPTF-Prep and National Vulnerability Database (NVD) CVSS data for each exploit (CVE). After considering multiple approaches, we settled on the most appropriate mathematical calculation inspired from the Common Vulnerability Scoring System (CVSS) established by the National Vulnerability Database which we esteem adequate in the context of PT. Additionally, we considered to use real-world data built from IAPTF past tests and enhanced by the human expert initially meant to passively supervise the IAPTF.

5.3 Reward calculation

5.3.1 Using CVSS probabilities to calculate the Rewards:

The proposed method recognizes that information gathering using network remote scanning and discovery tools and the system often produce an uncertain result. Thus, in this research, as we focus on black box penetration testing, we will assume that information gathering phase undertaken by the expert using the PT system facilities can only provide probabilistic information about the operating system and services running on the target machine. The reward is annotated as R : action a : state s : state s' : observation o reward-value. During this research, the probabilistic nature of the exploits and attacks (RL actions) is a crucial factor to be considered. As we aim to mirror to the best the real-world environment which will allow us to simulate real scenarios. Thus, we are going to use the well-established and standard sources for as an input for generating the action probabilities (actions, observations, transitions). Indeed, Common Vulnerability and Exploits (CVE, NVD …etc.) constitute a reliable online vulnerability/ exploit catalog and database of known proven vulnerabilities associated with different type of systems/ software which were, being or can be exploited to compromise these targets. The use of such sources is motivated by the rich content, easy accessibility, regular update and the available scoring function and mechanism such as CVSS.

The National Vulnerability Database (NVD) provides CVSS scores for almost all known vulnerabilities. As a starting point for this research, only individual vulnerabilities will be considered in the method of computing the Cost using the DVSS and a dynamic cost-centric framework. The Common Vulnerability Scoring System (CVSS) provides an open framework for communicating the characteristics and impacts of IT vulnerabilities. Thus, CVSS is the most suitable and reliable measurement approach for this research's vulnerability impact scores attribution (or success probabilities which will be derived from the

CVSS overall scores). Furthermore, we opted for Varying Rewards as we noted that the results of a generic rewarding scheme are very negative and that they are diverging we undertook few steps so we can improve through the introduction of special rewards for some actions. This is part of the process in RL that gives us control as to what the algorithm optimizes for. We want to discourage the algorithm from terminating before reaching the pivoting and privileges escalation and thus we introduced a small positive reward for each move action. The calculated scores of intrinsic, time-based, and ecological metrics by combining related sub-scores and modeling the problem's parameters into a mathematical framework to develop a unique severity cost. The third version of CVSS equations is done following the CVSS v3.0 equations [139].

5.3.2 Exploitation reward calculation:

To calculate the reward for each of the action/state combination we relied on a realistic approach in cyber risk assessment and used data from NVD namely CVSS version 3 calculator. The aim was to determine the reward value that the IAPTF RL agent should receive for each action in terms of exploitation, pivoting post-exploitation as being used NVD and CVE qualitative metrics translated into exploitability. In fact, CVSS scores are computed in sequence such that the Base Score is used to calculate the Temporal Score and the Temporal Score is used to calculate the Environmental Score. Impact is measured from the concatenation of Confidentiality Impact (C), Integrity Impact (I) and Availability Impact (A). The Exploitability measured from the concatenation Attack Vector (AV), Attack Complexity (AC), Privileges Required (PR), User Interaction (UI), Scope (S), and the overall impact value previously calculated. Finally, the Temporal Score Metrics is the concatenation of Exploit Code Maturity (E), Remediation Level (RL) and Report Confidence (RC) and the overall exploitability value previously calculated. The Base Score is a function of the Impact and Exploitability sub score equations defined as follow:

If (Impact sub score <= 0) *0 else,*

Scope Unchanged $Roundup(Minimum[(Impact + Exploitability), 10])$

Scope Changed $Roundup(Minimum[1.08 \times (Impact + Exploitability), 10])$

Scope Unchanged $6.42 \times ISC_{\text{Base}}$

Scope Changed $7.52 \times [ISC_{Base} - 0.029] - 3.25 \times [ISC_{Base} - 0.02]$

And the Exploitability sub score is defined as:

$$8.22 \times AttackVector \times AttackComplexity \times PrivilegeRequired \times UserInteraction$$

$$Roundup(BaseScore \times ExploitCodeMaturity \times RemediationLevel \times ReportConfidence)$$

The environmental score is defined as:

If (Modified Impact Sub score <= 0)   0 else,

If Modified Scope is Unchanged   Round up(Round up (Minimum [(M.Impact + M.Exploitability) ,10]) × Exploit Code Maturity × Remediation Level × Report Confidence)

If Modified Scope is Changed   Round up(Round up (Minimum [1.08 × (M.Impact + M.Exploitability) ,10]) × Exploit Code Maturity × Remediation Level × Report Confidence)

If Modified Scope is Unchanged $6.42 \times [ISC_{Modified}]$

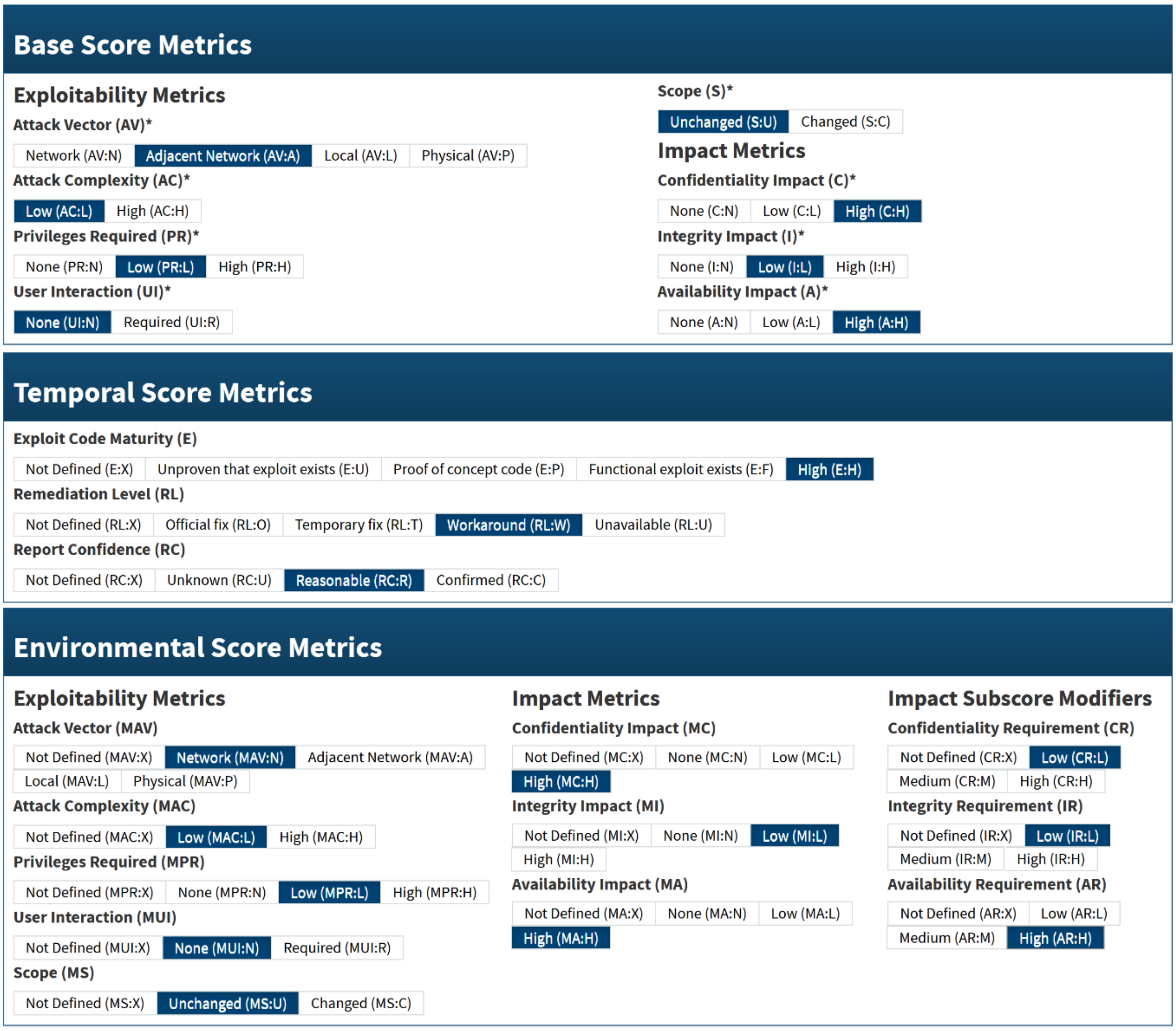
If Modified Scope is Changed $7.52 \times [ISC_{Modified} - 0.029]$-$3.25\times [ISC_{Modified} - 0.02]$

Finally, the Modified Exploitability sub score is defined as:

$$ISC_{Modified} = Minimum\ [[1 - (1 - M.\,IConf \times CR) \times (1 - M.\,IInteg \times IR) \times (1 - M.\,IAvail \times AR)], 0.915]$$

$$8.22 \times M.\,AttackVector \times M.\,AttackComplexity \times M.\,PrivilegeRequired \times M.\,UserInteraction$$

Figure 42 summarises different qualitative entries used to calculate each metrics.



Base Score Metrics

Exploitability Metrics

Attack Vector (AV)*

Network (AV:N) [✓ Adjacent Network (AV:A)] Local (AV:L) Physical (AV:P)

Attack Complexity (AC)*

[✓ Low (AC:L)] High (AC:H)

Privileges Required (PR)*

None (PR:N) [✓ Low (PR:L)] High (PR:H)

User Interaction (UI)*

[✓ None (UI:N)] Required (UI:R)

Scope (S)*

[✓ Unchanged (S:U)] Changed (S:C)

Impact Metrics

Confidentiality Impact (C)*

None (C:N) Low (C:L) [✓ High (C:H)]

Integrity Impact (I)*

None (I:N) [✓ Low (I:L)] High (I:H)

Availability Impact (A)*

None (A:N) Low (A:L) [✓ High (A:H)]

Temporal Score Metrics

Exploit Code Maturity (E)

Not Defined (E:X) Unproven that exploit exists (E:U) Proof of concept code (E:P) Functional exploit exists (E:F) [✓ High (E:H)]

Remediation Level (RL)

Not Defined (RL:X) Official fix (RL:O) Temporary fix (RL:T) [✓ Workaround (RL:W)] Unavailable (RL:U)

Report Confidence (RC)

Not Defined (RC:X) Unknown (RC:U) [✓ Reasonable (RC:R)] Confirmed (RC:C)

Environmental Score Metrics

Exploitability Metrics

Attack Vector (MAV)

Not Defined (MAV:X) [✓ Network (MAV:N)] Adjacent Network (MAV:A) Local (MAV:L) Physical (MAV:P)

Attack Complexity (MAC)

Not Defined (MAC:X) [✓ Low (MAC:L)] High (MAC:H)

Privileges Required (MPR)

Not Defined (MPR:X) None (MPR:N) [✓ Low (MPR:L)] High (MPR:H)

User Interaction (MUI)

Not Defined (MUI:X) [✓ None (MUI:N)] Required (MUI:R)

Scope (MS)

Not Defined (MS:X) [✓ Unchanged (MS:U)] Changed (MS:C)

Impact Metrics

Confidentiality Impact (MC)

Not Defined (MC:X) None (MC:N) Low (MC:L) [✓ High (MC:H)]

Integrity Impact (MI)

Not Defined (MI:X) None (MI:N) [✓ Low (MI:L)] High (MI:H)

Availability Impact (MA)

Not Defined (MA:X) None (MA:N) Low (MA:L) [✓ High (MA:H)]

Impact Subscore Modifiers

Confidentiality Requirement (CR)

Not Defined (CR:X) [✓ Low (CR:L)] Medium (CR:M) High (CR:H)

Integrity Requirement (IR)

Not Defined (IR:X) [✓ Low (IR:L)] Medium (IR:M) High (IR:H)

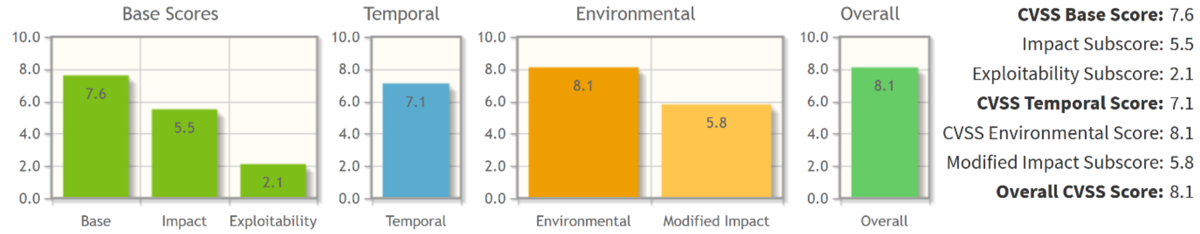
Availability Requirement (AR)

Not Defined (AR:X) Low (AR:L) Medium (AR:M) [✓ High (AR:H)]

Figure 42: qualitative metric entry to calculate Base, Temporal and Environmental scores.

The overall reward is the concatenation of Impact Metrics, Exploitability Metrics and Temporal Score Metrics calculated from the qualitative value and then used during the overall reward calculation. Below are two explanatory real-world examples to illustrate reward calculation for two CVEs IAPTF:

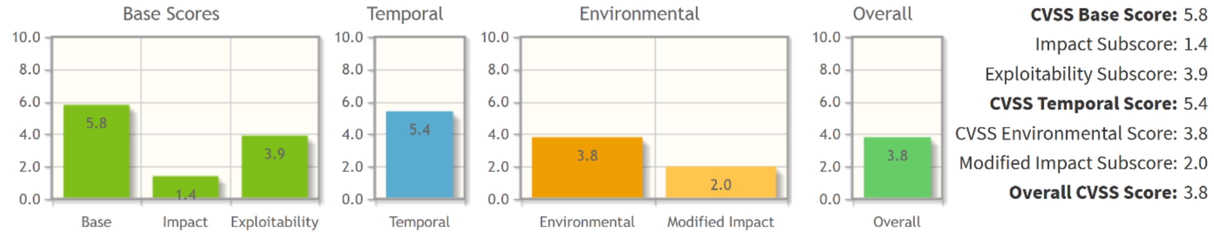
Explanatory Example 1: CVE-2011-0660 is windows Server Block Message (SMB) related vulnerability where a client could allow remote code execution on Microsoft Windows Server 2008 SP2 and R2 SP1, Windows 7 SP1, Windows 8.1, Windows Server 2012 Gold and R2, Windows RT 8.1, Windows 10 Gold, 1511, 1607, and 1703, and Windows Server 2016, allows an information disclosure vulnerability in the way that it handles certain requests. The security update addresses the vulnerabilities by correcting the manner in which the CIFS Browser handles specially crafted Browser messages and correcting the manner in which the SMB client validates specially crafted SMB response. To exploit the vulnerability, an attacker would have to be able to authenticate and send SMB messages to an impacted Windows SMB Server. Figure 43 summarises the calculated CVSS score for this CVE.



CVSS Base Score: 7.6
Impact Subscore: 5.5
Exploitability Subscore: 2.1
CVSS Temporal Score: 7.1
CVSS Environmental Score: 8.1
Modified Impact Subscore: 5.8
Overall CVSS Score: 8.1

Figure 43: Impact, Exploitability and Overall CVSSv3 score for CVE-2011-0660.

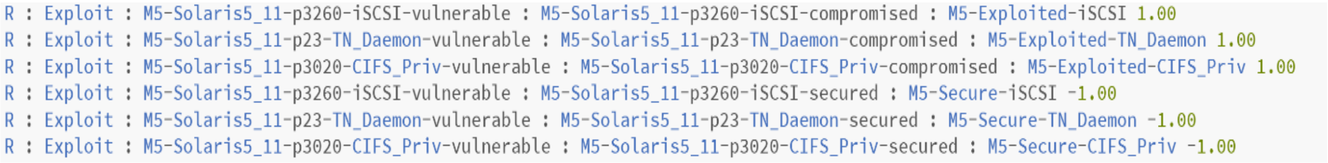
Explanatory Example 2: CVE-2016-9209 is a vulnerability in TCP processing we encounter often in CISCO FirePOWER system software that could allow an unauthenticated, remote attacker to download files that would normally be blocked. There are many Cisco products which are vulnerable such as Adaptive Security Appliance (ASA) 5500-X Series with FirePOWER Services, Advanced Malware Protection (AMP) for Networks. Overall CVSS score for CVE-2016-9209 is illustrated in figure 44.



CVSS Base Score: 5.8
Impact Subscore: 1.4
Exploitability Subscore: 3.9
CVSS Temporal Score: 5.4
CVSS Environmental Score: 3.8
Modified Impact Subscore: 2.0
Overall CVSS Score: 3.8

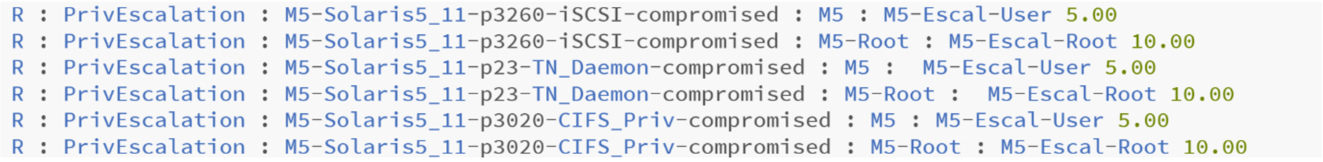
Figure 44: Impact, Exploitability and Overall CVSSv3 score for CVE-2016-9209.

Finally, after obtaining the overall CVSS score, the reward for the Exploit, PrivEscalation and Pivot is respectively 5, 10 and 15 added to the overall CVSS score. For instance, the execution of Exploit CVE-2011-0660 against the machine M22 running Windows 7 SP1 and having port 445 open will receive a positive reward of 13 for exploitation and 18 for admin privilege escalation and 23 for installing a rootkit and perform pivoting as illustrated in figure 45,46 and 47.



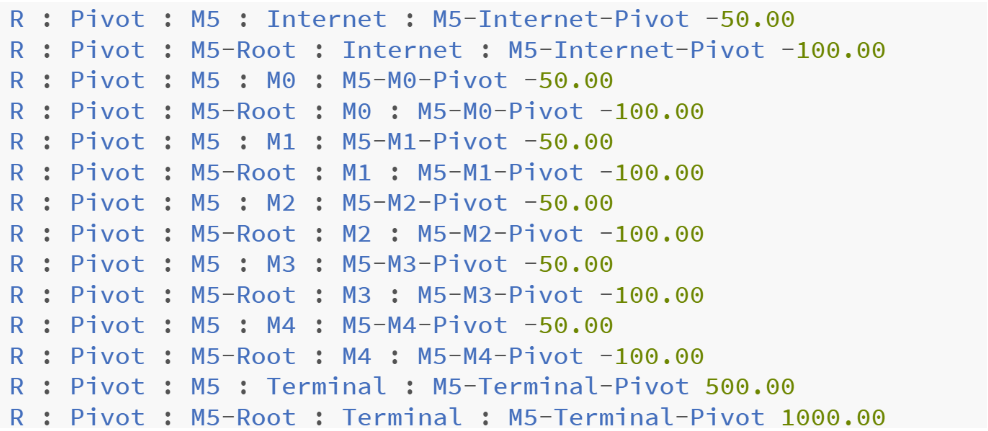
```
R : Exploit : M5-Solaris5_11-p3260-iSCSI-vulnerable : M5-Solaris5_11-p3260-iSCSI-compromised : M5-Exploited-iSCSI 1.00
R : Exploit : M5-Solaris5_11-p23-TN_Daemon-vulnerable : M5-Solaris5_11-p23-TN_Daemon-compromised : M5-Exploited-TN_Daemon 1.00
R : Exploit : M5-Solaris5_11-p3020-CIFS_Priv-vulnerable : M5-Solaris5_11-p3020-CIFS_Priv-compromised : M5-Exploited-CIFS_Priv 1.00
R : Exploit : M5-Solaris5_11-p3260-iSCSI-vulnerable : M5-Solaris5_11-p3260-iSCSI-secured : M5-Secure-iSCSI -1.00
R : Exploit : M5-Solaris5_11-p23-TN_Daemon-vulnerable : M5-Solaris5_11-p23-TN_Daemon-secured : M5-Secure-TN_Daemon -1.00
R : Exploit : M5-Solaris5_11-p3020-CIFS_Priv-vulnerable : M5-Solaris5_11-p3020-CIFS_Priv-secured : M5-Secure-CIFS_Priv -1.00
```

Figure 45: POMDP reward representation in POMDP for Exploitation actions.



```
R : PrivEscalation : M5-Solaris5_11-p3260-iSCSI-compromised : M5 : M5-Escal-User 5.00
R : PrivEscalation : M5-Solaris5_11-p3260-iSCSI-compromised : M5-Root : M5-Escal-Root 10.00
R : PrivEscalation : M5-Solaris5_11-p23-TN_Daemon-compromised : M5 :  M5-Escal-User 5.00
R : PrivEscalation : M5-Solaris5_11-p23-TN_Daemon-compromised : M5-Root :  M5-Escal-Root 10.00
R : PrivEscalation : M5-Solaris5_11-p3020-CIFS_Priv-compromised : M5 : M5-Escal-User 5.00
R : PrivEscalation : M5-Solaris5_11-p3020-CIFS_Priv-compromised : M5-Root : M5-Escal-Root 10.00
```

Figure 46: POMDP reward representation in POMDP for Privilege Escalation actions.



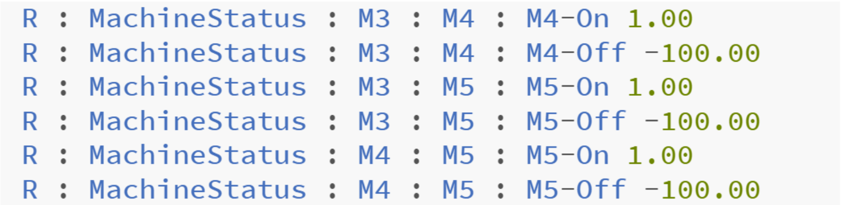
```
R : Pivot : M5 : Internet : M5-Internet-Pivot -50.00
R : Pivot : M5-Root : Internet : M5-Internet-Pivot -100.00
R : Pivot : M5 : M0 : M5-M0-Pivot -50.00
R : Pivot : M5-Root : M0 : M5-M0-Pivot -100.00
R : Pivot : M5 : M1 : M5-M1-Pivot -50.00
R : Pivot : M5-Root : M1 : M5-M1-Pivot -100.00
R : Pivot : M5 : M2 : M5-M2-Pivot -50.00
R : Pivot : M5-Root : M2 : M5-M2-Pivot -100.00
R : Pivot : M5 : M3 : M5-M3-Pivot -50.00
R : Pivot : M5-Root : M3 : M5-M3-Pivot -100.00
R : Pivot : M5 : M4 : M5-M4-Pivot -50.00
R : Pivot : M5-Root : M4 : M5-M4-Pivot -100.00
R : Pivot : M5 : Terminal : M5-Terminal-Pivot 500.00
R : Pivot : M5-Root : Terminal : M5-Terminal-Pivot 1000.00
```

Figure 47: POMDP reward representation in POMDP for Pivoting actions.

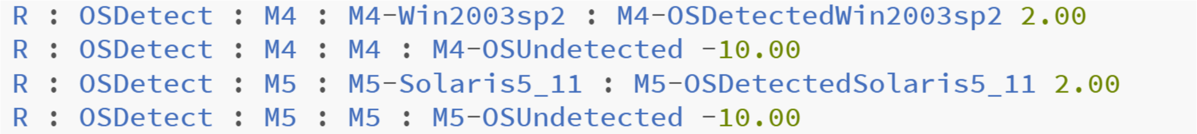
5.3.3 Reward allocation

In the second part of IAPTF rewarding scheme rely on the allocation of reward values following a human CEH expert rewarding grid and the use of some default rewarding values are used for generic situations. Rewarding the performed actions will be predefined by a human CEH who will have to decide the adequate reward for each action performed depending on his/her overall insight he has on the practice, experience and testing achievements. Afterward, IAPTF 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 utilised, 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 degree of generated traffic and thus associated risk of detection. We detail below the manual reward allocation rules:

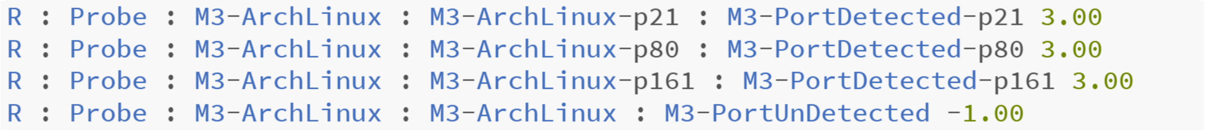
- **Estimated machine value:** The objective of a PT is to successfully fingerprint or discovery vulnerabilities in specific machine in a network. Here we thus propose to assign a fixed reward for each successful exploit on an uncontrolled or partially compromised machine which will vary from 1 to 10 depending on the position (DMZ, Workspace, Sensitive), and future prospect beyond this achievement.
- **Time Consumed:** Each scanning, discovery or probing action requires an execution time and may be achieved within or beyond the allocated time. Thus, the more time is consumed the less is the reward allocated. We define a discount function which multiplies the maximum forward into the percentage of time allocated to complete the action (MachineStatus, Prob, OSDetect, SVCDetect, and VulAssess), so that the expected duration of the PT tasks may be minimised by assigning each transition, an action that goes overtime will results in negative reward.
- **Risk of Detection:** this is explicitly represented by assigning a discounted reward for aggressive actions causing large traffic and thus having high risk of detection and prevention by the network IDSs.



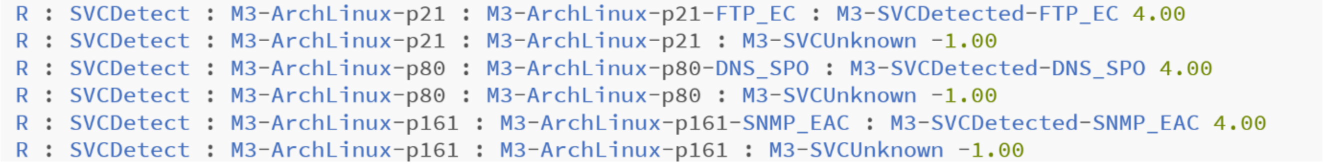
```
R : MachineStatus : M3 : M4 : M4-On 1.00
R : MachineStatus : M3 : M4 : M4-Off -100.00
R : MachineStatus : M3 : M5 : M5-On 1.00
R : MachineStatus : M3 : M5 : M5-Off -100.00
R : MachineStatus : M4 : M5 : M5-On 1.00
R : MachineStatus : M4 : M5 : M5-Off -100.00
```



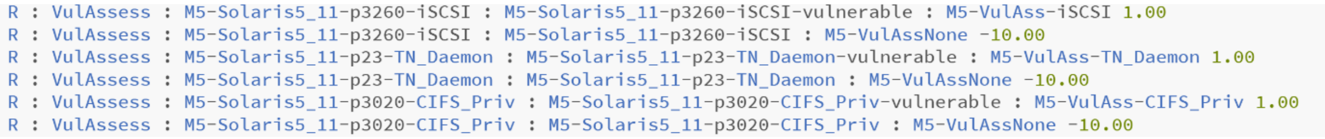
```
R : OSDetect : M4 : M4-Win2003sp2 : M4-OSDetectedWin2003sp2 2.00
R : OSDetect : M4 : M4 : M4-OSUndetected -10.00
R : OSDetect : M5 : M5-Solaris5_11 : M5-OSDetectedSolaris5_11 2.00
R : OSDetect : M5 : M5 : M5-OSUndetected -10.00
```



```
R : Probe : M3-ArchLinux : M3-ArchLinux-p21 : M3-PortDetected-p21 3.00
R : Probe : M3-ArchLinux : M3-ArchLinux-p80 : M3-PortDetected-p80 3.00
R : Probe : M3-ArchLinux : M3-ArchLinux-p161 : M3-PortDetected-p161 3.00
R : Probe : M3-ArchLinux : M3-ArchLinux : M3-PortUnDetected -1.00
```



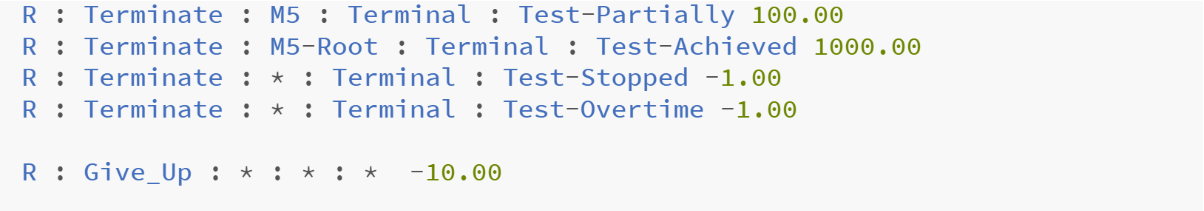
```
R : SVCDetect : M3-ArchLinux-p21 : M3-ArchLinux-p21-FTP_EC : M3-SVCDetected-FTP_EC 4.00
R : SVCDetect : M3-ArchLinux-p21 : M3-ArchLinux-p21 : M3-SVCUnknown -1.00
R : SVCDetect : M3-ArchLinux-p80 : M3-ArchLinux-p80-DNS_SPO : M3-SVCDetected-DNS_SPO 4.00
R : SVCDetect : M3-ArchLinux-p80 : M3-ArchLinux-p80 : M3-SVCUnknown -1.00
R : SVCDetect : M3-ArchLinux-p161 : M3-ArchLinux-p161-SNMP_EAC : M3-SVCDetected-SNMP_EAC 4.00
R : SVCDetect : M3-ArchLinux-p161 : M3-ArchLinux-p161 : M3-SVCUnknown -1.00
```



```
R : VulAssess : M5-Solaris5_11-p3260-iSCSI : M5-Solaris5_11-p3260-iSCSI-vulnerable : M5-VulAss-iSCSI 1.00
R : VulAssess : M5-Solaris5_11-p3260-iSCSI : M5-Solaris5_11-p3260-iSCSI : M5-VulAssNone -10.00
R : VulAssess : M5-Solaris5_11-p23-TN_Daemon : M5-Solaris5_11-p23-TN_Daemon-vulnerable : M5-VulAss-TN_Daemon 1.00
R : VulAssess : M5-Solaris5_11-p23-TN_Daemon : M5-Solaris5_11-p23-TN_Daemon : M5-VulAssNone -10.00
R : VulAssess : M5-Solaris5_11-p3020-CIFS_Priv : M5-Solaris5_11-p3020-CIFS_Priv-vulnerable : M5-VulAss-CIFS_Priv 1.00
R : VulAssess : M5-Solaris5_11-p3020-CIFS_Priv : M5-Solaris5_11-p3020-CIFS_Priv : M5-VulAssNone -10.00
```

Figure 48: POMDP reward allocation for vulnerability assessment and discovery action.

Furthermore, we allocate the following reward for these special cases reward allocation. For Give Up action will receive a positive reward (10) if the RL already achieved at least one of the predefined targets (either as admin/root or lower privileges). Nonetheless, if the RL did not yet achieve any target Give Up action receive a very bad reward (-100). Action leading to exceed the maximal allowed time (Run Overtime) will receive simply a null reward. Finally, the Terminate action is a twofold problem from rewards allocation point of view. The RL agent that reaches this state can be either considered as a good or bad outcome depending on the context and thus rewarding (positive value) or punishing (negative reward) is allocated basing on previous states and history H of the agent but not ignoring the previous achievements prior reaching such state. Thus, a double weight of measure will be adopted and the allocated Negative or Positive values. When the previous achieved results were so far interesting (potentially valid attack path) and only the last action to the terminal state the reward for this action will be highly positive (+100). In some situation a Null (zero) reward is allocated despite being relatively bad situation which will allow the RL agent to expand further for a couple of state before facing a negative reward forcing the termination. Nonetheless, the action leading to a terminal state from a poor or null previous state should imperatively be severely punished by allocating a negative value (such –100) which will force the termination of the sub-test. It is worth to highlight that we adopted a progressive rewarding approach thus the closer to the target/aim the higher is reward value as shown in figure 49.



```
R : Terminate : M5 : Terminal : Test-Partially 100.00
R : Terminate : M5-Root : Terminal : Test-Achieved 1000.00
R : Terminate : * : Terminal : Test-Stopped -1.00
R : Terminate : * : Terminal : Test-Overtime -1.00

R : Give_Up : * : * : *  -10.00
```

Figure 49: POMDP reward allocation for termination actions.

5.4 Hierarchical POMDP for medium and large networks

In this section, we detail the methodology adopted to address our research problem which is addressing the scaling-up issue in solving PT RL problem (large and complex POMDP environment) in context of large computer networks. Initially, we will re-introduce the proposed POMDP model which will be serving as a starting point to the introduction of the new hierarchical RL model for representing large network PT. To achieve this goal we had to consider two options: the PT phases separation and security cluster separation and the later was undoubtedly the most adequate in term of efficiency and relevance.

In this research, our objective turned into finding the perfect way in dealing with medium and large PT associate POMDP environments in terms of consumed solving time and memory. This scalability issue became more impactful with larger environment as discussed in the RL chapter and thus the urgent need for a hierarchical representation of large problem through dividing POMDP environment which are often the results of large LANs and MANs networks, we investigated different options of enhancing the efficiency of IAPTF especially in solving medium and large size POMDP which are the logic result of representing PT in medium and large LANs and MANs. The obvious approach to address scalability issue in POMDP solving is by solving smaller environments which is consequently dividing large environments into many smaller ones reflected by either splitting large network into a number of sub-networks or by considering each phase activities and tasks separately. We detail here the two approaches considered and implemented initially within IAPTF, then we will justify the choice of the clustering approach to achieve an efficient hierarchical RL representation of PT.

5.4.1 Option 1: task-based approach

PT is a task-oriented practice with a well-established sequential order and some repetition of some tasks depending on the context. A natural way of tackling the scaling problem in RL solving is by dividing large POMDP environments following task or group of tasks approaches. In fact, the manual and automated PT multi-step procedure with the general principle behind is to evaluate and test the security of any computing resource. The approach considered into breaking down the complex PT practice into phases each grouping several costumed tasks. The idea came from the vulnerability assessment and security auditing industry where each activity is performed separately but in a sequential order and where the output of the first phase is used as input for the second one. By considering each of the PT phases separately, such as Vulnerability Assessment, Exploitation, Post-exploitation, and Validation, we can elaborate a POMDP problem with the data relevant for each phase and will be resulting into smaller POMDP environments which are easier and faster to process and solve and therefore enhancing IPATF performances. As described in Chapter 2, the PT procedures are better tackled when they are grouped into phases which are a set of tasks and sub-tasks (figure 50). Below are the main phases, activities, tasks, and sub-tasks are being considered in our aim of representing the PT into a hierarchical RL and thus dividing it into many sub POMDP environments.

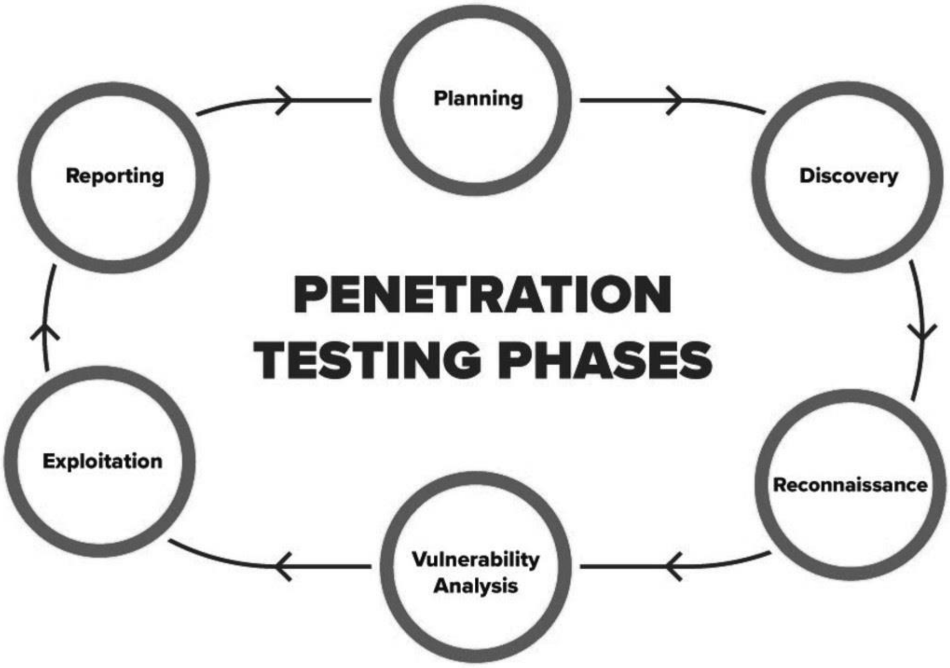


Figure 50: Standard PT practice cyclic activities diagram [27].

The first phase in PT is to perform vulnerability assessment (VA) of the target network which is a set of activities such as network discovery, port scanning & probing, vulnerability analysis and finally vulnerability identification and validation. Each of these activities is multi-task such as scanning and probing which is a set of tasks varying from hosts and servers' discovery, port scanning and finally

OS fingerprinting. For each task, many sub-tasks are implicitly performed taking the example of host discovery which is done through launching first a ping sweep then following up by doing traceroute discovery. Note that the PT versatility impose that tester/system to identify additional computing resources that have the potential to affect one or more mission critical systems which is often based upon the assessed asset functioning. The associated network POMDP environments will contain exclusively the vulnerability assessment and discovery information and aim to solve it as separate RL problem with the aim of serving as input for the next phase POMDP.

The second phase is Exploitation with all the required prioritization of the launched exploits, payloads with the available resources in term of exploitability, running time and importance level. This is done through multiple activity such as validating the relevance of the exploit based on several factors, ordering and sorting the planed exploits following a logical diagram such as data sensitivity, frequency of usage, type of application and cost and finally launching the exploits and assessing the outcome. Each of these activities is itself a set of tasks and sub-tasks such as exploitation, which is a set of shortlisting the relevant exploits, establishing an order, customize the exploit variables to match the target IP, architecture, kernel, and variant. Again, all associated network POMDP environments will contain exclusively exploitation and attacking information and aim to solve it as separate RL problem with the aim of serving as input for the next phase POMDP.

The third phase in our view of PT in IAPTF is the post-exploitation which group all activities done by the tester to follow-up the exploitation phase. In practice this phase is mainly constituted of privilege escalation and pivoting activities which are multi-tasks and highly versatile. Tasks are typically related to establish guest control or executing the relevant buffer-over-flow to force privilege escalation. Then, this later outcome is used ti achieve full control over the asset and to launch new attach against adjacent computers by repeating the phase 1 and 2 activities which should produce a different output. In this phase, all associated network POMDP environments will contain post-exploitation and attacking information in addition to a small fraction of vulnerability assessment and exploitation phases as there will be cyclic activities requiring the inclusion of such information. Again, this POMDP will be solved separately with a major difference related to the number of POMDO problems to reflect the version for each of the pivoting point representation and this will be processed later to serve ss input for the next phase POMDP. The fourth phase is the validation where all activities related to checking the status of the targeted machines and results in term of success and failure are grouped together and examined. In addition, the different adopted attack paths are extracted and reconstructed to produce a clear attacking scenario checklist. This is highly automated phase and human interaction will be simply validation or changing the stored data.

Finally, the fifth phase is the reporting and recommendation which is fully automated and does not require any further RL involvement. In this phase, activities such as summarising the output pf tests and attacks launched and effects along with the recommendations are achieved using Expert System database where such information are stored and in form of pre-defined recommendations. In fact, the CLIPS expert system embedded with IAPTF will base on the determined threats and impact levels notably crucial, critical, serious, and minor to address the security weaknesses and how they should be dealt with, depending on the threat, it is eliminated by applying various security measures, such software patching, reconfigurations, managing access control permissions, network monitoring and encryption. This phase also will output a generic testing report (document) which is a simple format that offer guide on the minimization of consequences if an attack occurs in future.

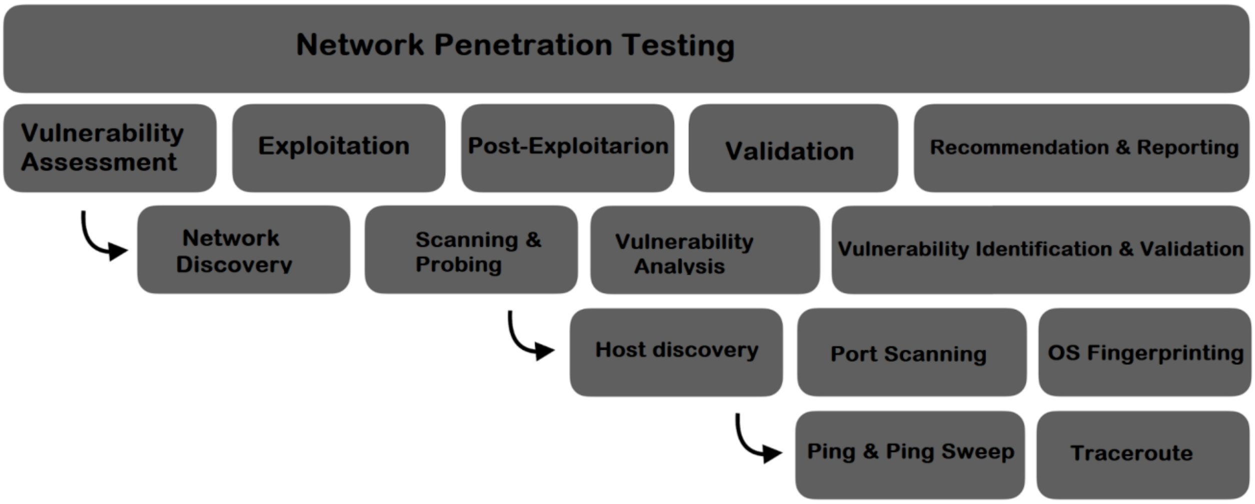


Figure 51: Task based hierarchical modelling of network PT practice.

5.4.2 Option 2: Security Cluster approach

The second and most plausible option is the decomposition of the network itself and maintain the regular PT multi-phases approach in proceeding. In fact, this approach lies on well-established practice in network security maintenance and development where the large networks are initially divided into smaller sub-networks following a given rule. The rules are often related to the functioning, configuration, or location. While such approaches is commonly adopted, it lacks depth from a security point of view as it reflects more the administrator and technician view of the network. We propose a novel approach of dividing network in security cluster which we adopt the security separation and isolation of the cluster (grouping several computers) having the same security protection level and online exposure. This approach mimics the hacker behavior as they see the network form security point of view and class section based on their security level.

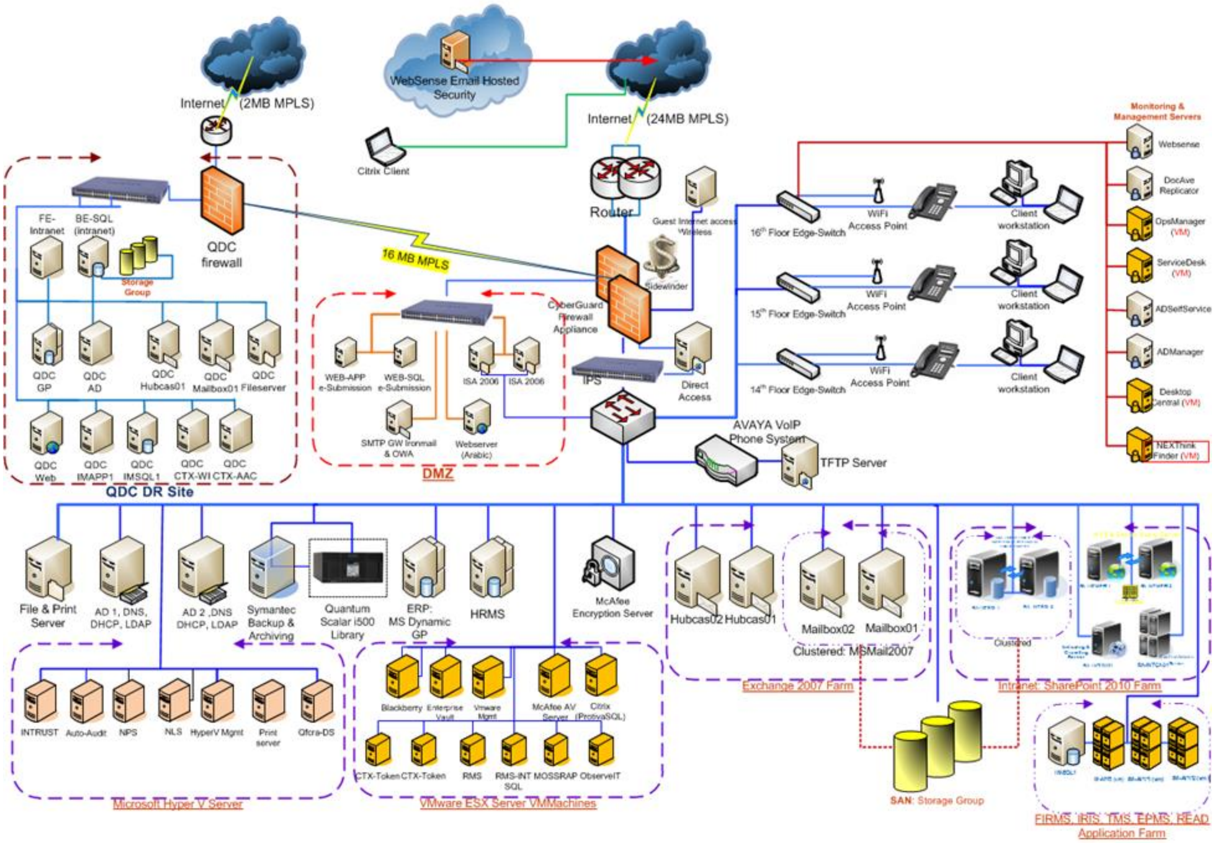


Figure 52: large corporate LAN architecture with security isolation illustration [45].

Performing PT different tasks requires from the PT expert to meet some basic requirements. One of the most important requirements in PT is to clearly identify and understand the assessed network security mechanism and most importantly the firewalls functioning and the resulting isolation. This knowledge is totally extracted from the vulnerability assessment and reconnaissance tasks which is a combination of topology and network defense. This will result in identifying the relevant attack paths based on the knowledge of existing connections between machines and/or sub-networks and firewall filtering. This is a crucial phase as any cyber attacker will, after performing the initial reconnaissance, try to sketch a logical security map of the target network and assess the ability to reach the targeted machine. Often, tester can initially reach only few machines from his external position (high-value targeted machines cannot initially be reached) and the overwhelming majority of assessed network machines are unreachable directly form the Internet. Thus, a progressive approach is unavoidable, the tester will attempt to gain partial or full control of the reachable machines and then use these controlled machines as a launching point for the future attacks as it is likely that they share a sub-network with one of them: and those which are unreachable from any controlled computer.

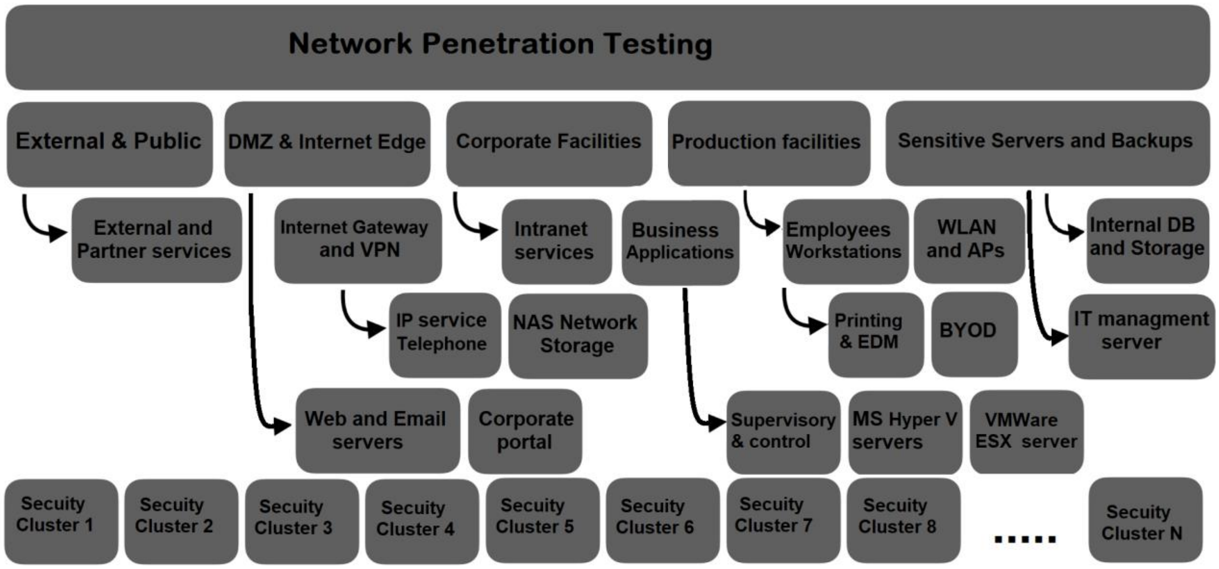


Figure 53: Security cluster based hierarchical modelling of network PT practice.

The network clustering is a crucial phase for our research as this will allow an easy and simple RL modelling of large network PT environment which mirror the real-world hacking operative mode. In fact, it is hard to deal with each network connectivity separately and this will generate larger environment problem and increase solving complexity notably when representing the full LAN topology. The network clustering is therefore our proposed method to incorporate the information about the network connectivity within the RL model along with maintaining the solvability of the problem in an adequate amount of time. The full technical details about how the security clustering is done will be detailed in the next chapter.

5.4.3 IPATF adequate Hierarchical RL approach

In our quest for better performances in IAPTF, we identified two main approaches for automatically decompose large POMDP environments into smaller POMDPs. Secondly, we are concerned with the computation of optimal policies, using hierarchically decomposed POMDPs. Most of the work to date has targeted this second problem. In practice, this first hierarchical approach which relies on a division of PT phases will generate higher-level POMDPs which have a direct control over more specialized POMDPs. At the high level, the POMDP policy consists of selecting appropriate sub-modules (i.e. hierarchical actions), while at the lower-level, specialized POMDPs have policies involving direct actions onto the problem domain (i.e. non-hierarchical actions). This division of tasks is not strict, and POMDPs of all levels can have both hierarchical and non-hierarchical actions, except for lowest-level POMDPs which have only non-hierarchical actions. Belief states are always maintained over all POMDPs of the hierarchy and are updated after each action/observation pair. When a specific specialized POMDP is selected, it is identified as an ``active module” and is responsible for the learner's action selection at that given time step. The belief state is updated according to a default action, thereby ensuring that they capture the entire history and will produce the appropriate behaviour when later selected as an ``active module".

We have shown that in a domain such as a conversational speech interface, structure can indeed be exploited to obtain a policy much faster than with a conventional POMDP, and furthermore allowed us to build a larger POMDP-based dialogue manager than was possible with non-hierarchical POMDP representations. Such structure is not present in all domains, however is found in a great variety of applications, ranging from the dialogue task, to robot navigation tasks where the domain could be divided into a set of separate problems according to building topology (e.g. each room gets a separate POMDP). The speedup obtained by applying our hierarchical approach to POMDPs can be remarkable in domains with appropriate structure. The hierarchical POMDPs generator will first develops a high-level PT plan to sequence the overall (complex) PT models to be converted into a detailed continuous state plan. This hierarchical planning approach results in a decomposition of the POMDP planning problem into smaller sub-parts that can be solved with significantly lower computational costs. The ability to sequence the visitation of local dynamics models also provides a powerful way to leverage the hybrid dynamics to reduce state uncertainty. We evaluate the proposed planner on a navigation task in the simulated domain and on an assembly task with a robotic manipulator, showing that our approach can solve tasks having high observation noise and nonlinear dynamics effectively with significantly lower computational costs compared to direct planning approaches.

Chapter 6: Intelligent Automated Penetration Testing Framework

In this chapter, we will gradually introduce the detailed methodology which enabled us to produce IAPTF through the different phases of conception, design, modelling, and implementation. The proposed framework has been built in several steps. Initially we focused on the core reinforcement learning module which include the software RL agent, the memory where POMDP stored and POMDP solving mechanism. The RL agent communicate via scripts with MSF Console. Nonetheless, the development of IAPTF-Core module sits in the heart of IAPTF and illustrate a real-world implementation of our proposed RL modelling of the network PT practice detailed in previous chapter. The proposed model implementation had been through two major milestones related to the adopted model for representing the PT domain as RL problem in form of POMDP, the two milestones are namely, the regular RL which was initially developed, implemented, and tested, and the hierarchical RL (HRL) which was later introduced to address scalability encountered in solving large POMDP as described in chapter 5. The second phase of the development was the design, development and implementation of all complementary modules and systems including the data extraction, preprocessing, security clustering, POMDP elaborator, expert system, attack vectors generator and post-testing system. In other words, this phase covers all modules, scripts and system constituting IAPTF except the RL referred to in this work as IAPTF-Core. Figure 54 illustrates the full functioning of IAPTF and summarises the main modules and system utilized. The next sections of this chapter will gradually introduce keys components of the IAPTF framework including the IAPTF-Core module where the proposed RL model is implemented alongside with the IAPTF-Memory which are the focus of this research work.

6.1 IAPTF anatomy and functional diagram

We present in this section the proposed representation of the PT problem in the form of POMDP environments. The proposed representation which is a core component of our research is introduced here through a series of illustrative examples for a smooth introduction of the main concepts and method used in our proposed IAPTF. In the context of PT, there is no need to fully represent the state of the network to describe the current situation. The modelling will only focus on the features and aspects that are relevant for the task at hand. An example of the network topology and security architecture is not required to be included within the RL state space as in networking the main network topology and defense is assumed to be static and known. But it will have to account for the configuration and status of each computer on the network. The detailed environment description is detailed as follows.

Our research output is twofold: RL model of PT practice and a framework named Intelligent Automated Penetration Testing Framework (IAPTF). Figure 54 illustrates the final design in form of a functional diagram of the proposed framework. Overall, IAPTF starts by using automated scripts to process existing and acquired data from early-stage PT tasks to perform the network clustering and networking and defense data processing. These tasks are done through automated python scripts with many XML and log files as inputs and produce core data for elaborating the initial PT POMDP environment including identifying the asset composition and different security clusters which will be detailed later in this chapter. The data generated from this operation is then stored in IAPTF memory in two distinct formats: first raw format which will be handled later by IAPTF and second, generalised data which will serve future use of the framework. The generalisation is done through the removal of all specific information associated with the data such as IP and MAC addresses, machines names or other irrelevant information from a future use point of view. In other words, the output of each phase will be made general and then stored with IAPTF memory and used also within the Expert System implemented and embedded in IAPTF.

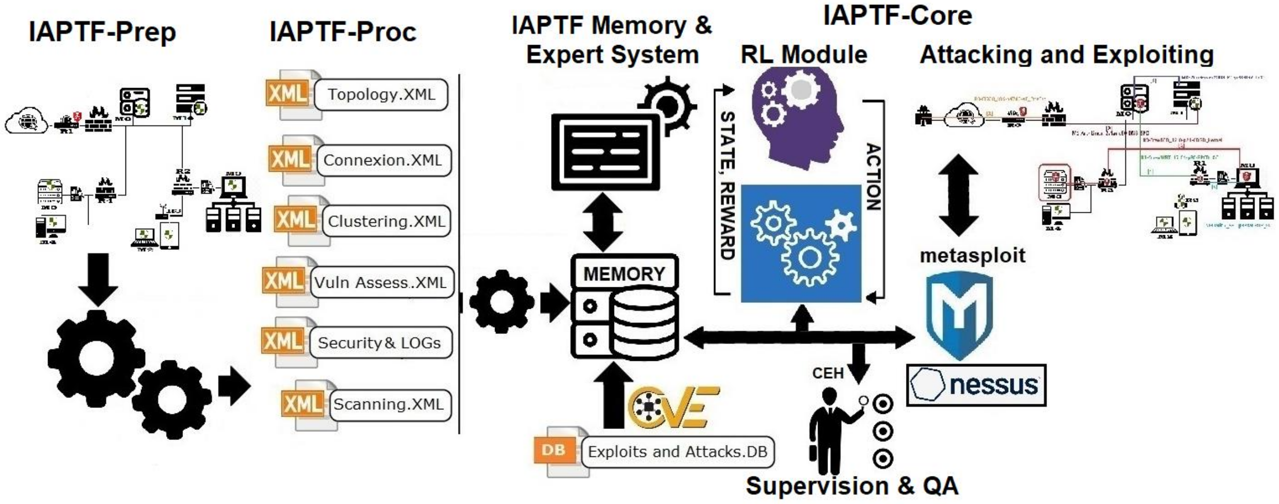
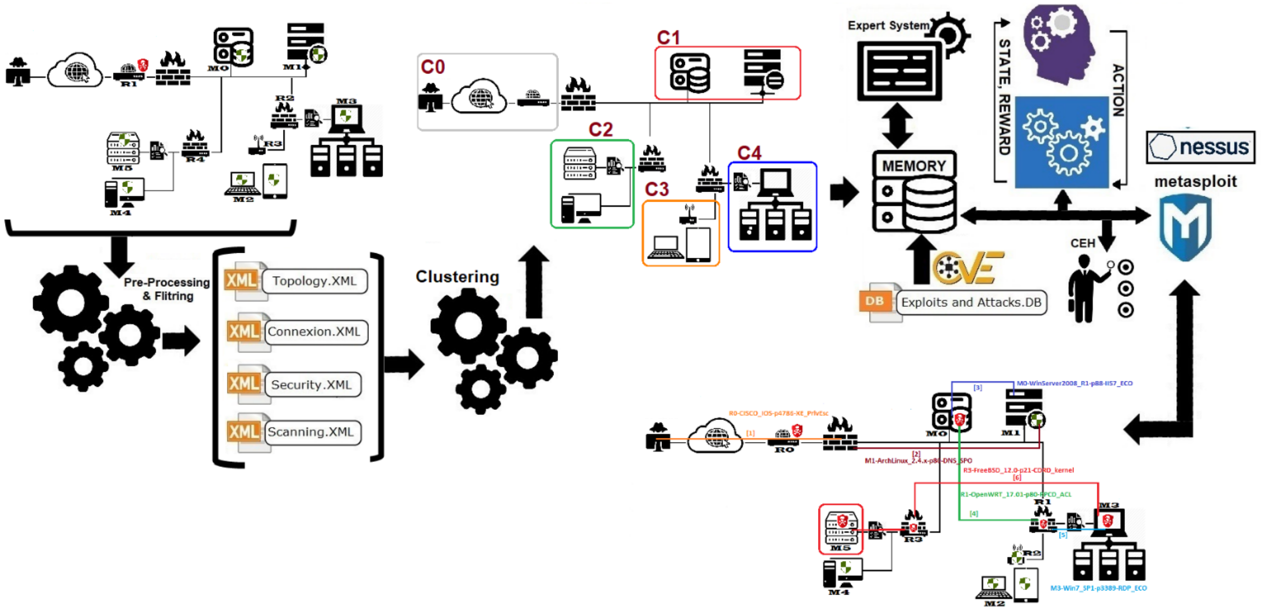


Figure 54: IAPTF anatomy and key modules and components.

On the other hand, the functional diagram is crucial to highlight the contribution of each module as well as the interaction and complementarity between them. IAPTF operates in a sequential order but highly interactive. The output of each module serves as the input for the next one. Notably the output of IAPTF-Prep in the form of data acquisition and filtering output is fed directly into IAPTF-Proc for processing and outputting structured and cleaned data that will be fed to the IAPTF-Memory to elaborates the different POMDPs environments files which will be then complemented and solved by IAPTF-Core. The figure 55 illustrates the full functional diagram for IAPTF.



R1
M0
M1
R2
R3
M3
M5
R4
M4
M2
Pre-Processing & Filtring
XML Topology.XML
XML Connexion.XML
XML Security.XML
XML Scanning.XML
Clustering
C0
C1
C2
C3
C4
Expert System
STATE, REWARD
ACTION
MEMORY
CEH
nessus
metasploit
DB Exploits and Attacks.DB
R0-CISCO_IOS-p4786-XE_PrivEsc
[1]
R0
[2]
M1-ArchLinux_2.4.x-p80-DNS_SPO
[3]
M0-WinServer2008_R1-p88-IIS7_ECO
R3-FreeBSD_12.0-p21-CDRD_kernel
[6]
R1-OpenWRT_17.01-p80-RPCD_ACL
[4]
[5]
M3-Win7_SP1-p3389-RDP_ECO

Figure 55: IAPTF overall functioning diagram and inter-modules interaction.

In the next session we will introduce the different modules following the logical sequence of the functional diagram.

6.2 IAPTF-Preparation module

In this section we will describe the details of different modules and system constituting IPATF-Prep. As described in Chapter 5, POMDPs files and notably the state space should include, for each phase or cluster of the system, all relevant information from a networking and security point of view. It will include, but not limited to, the OS, services, topology, security architecture, patches and knowledge base, version, subnetting and other networking information. This information will be represented under a specially introduced annotation designed to be brief, clear, precise, and easy to handle from point of view programming and data processing. In IAPTF-Prep, the different modules will attempt to extract, reconstruct, structure and represent all the information which the POMDP-Generator module requires to build the POMDP and especially state spaces which also come with some uncertainty related to the nature of the PT practice. Data extraction and Filtering programs and scripts are networking data, topology data, reachability data, along with vulnerability assessment data.

6.2.1 The networking data

The network data that defines corporate networks are often referred by connections and configurations of machines and limited in this research to the following sources: subnetworks, topology, machines network interfaces and networking services. Every network machine could have several services at different levels of machine. The extracted data at this stage is an abstract which includes relevant details of a real-world networks. In addition, at this stage of data capturing, there is no requirement nor benefit for developing an intelligent (AI-led) module to perform such activities as stand-alone software and is enough to capture and extract networking data including topology, sub-netting, and connections (reachability) between machines along with location of switches, routers, firewalls and IDSs in the network. The reason for this abstraction is to try and keep the simulator as simple as possible and at the level of representation that will be used in elaborating the POMDP environments. The proposed scripts are expected to work on determining which networking data can be used on top of vulnerability scans and assessments to store and to be used against which machine and in what order. Moreover, some specific details related to VA and PT such as ports status, service configuration along with details on application specific implementations are also copied to another raw XML file to be included in the Security data later. This choice comes from PT nature which is highly interactive even at automated levels with systems, toolkits and frameworks such as Metasploit which customize and utilises data on different exploits depending on the scenario and launch it.

6.2.2 Sub-networks and Topology

Computer networks are made of multiple subnets, single machines and devices. A subnet is a smaller network within the larger network that is composed of several machines that are reachable to each other. Each subnet has its own address, which is the first number in any machine's 32-bist IP address and comes with subnet mask to define the network, subnet, and machine address. In IAPTF, we opted for a security-oriented approach in regard to network splitting which we will detail in the security-clustering section. Nonetheless, it is important to capture, extract and process subnets data as we require the maximum information about network functioning and machines' reachability in one hand, and also enable a full abstraction of the network addressing. In fact, we often deal with single machines as target and not the full subnet despite the fact that all machines within a subnet can fully communicate and have relatively the same security protection and restriction in contrast to the Inter subnets communication is overseen by network topology, filtering, defense settings. The subnets data is therefore extracted and stored in the Connxion.XML file for future use.

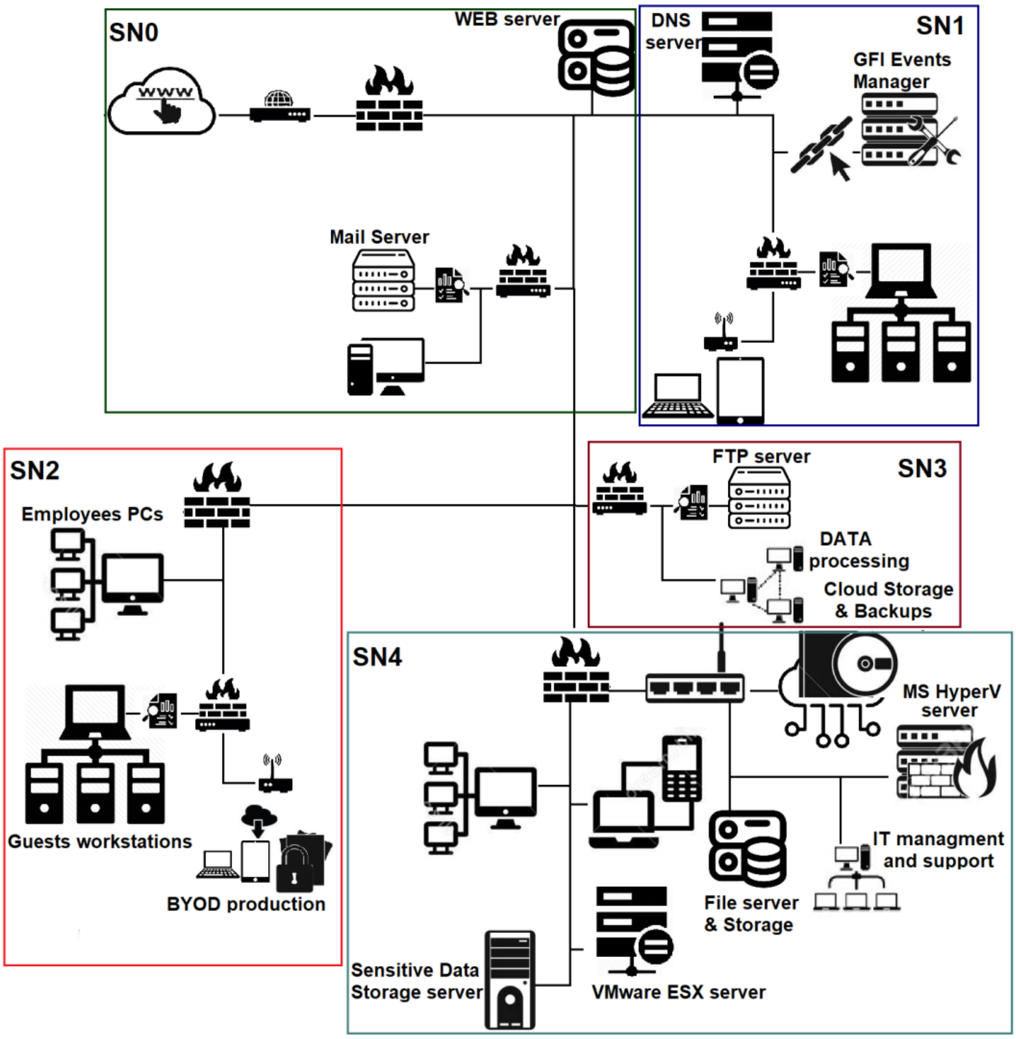


Figure 56: Sub-networking division of test bed output example.

The network topology (also called reachability) is defined as subnets inter-connection based on existing controls and restriction mechanisms which regulate which subnets can communicate directly with each other and with the external network. As an illustrative example,in the network in figure 56is composed of five subnets SN0, SN1, SN2, SN3 and SN4. SN0 and SN1 are the subnets in the part of network that is connected to the Internet often referred to as a demilitarized zone (DMZ) while subnets SN2 and SN3 are located in the part connected to internal workspace and have thus a different addressing. SN4 is here for internal storage and production. Note that machines can only communicate with the ones within the subnet or adjacent ones if they are adjacent and reachable.

6.2.3 Connectivity and Reachability

In IAPTF network topology is represented by an adjacency matrix, with rows and columns representing the different subnets. An example matrix is as shown in figure 57. In practice, IAPTF-Prep will use a simple python script Topology.py which will, from the addressing data, extract the different existing subnets. Then another script named Reachability.py will combine data from the topology and Security.xml to elaborate the full security-oriented network topology in form of adjacency matrix. The ones in the matrix stand for a direct connection between different subnets where one or more machine in the subnet can reach at least one machine in the second subnet. The reachability values are crucial from an offensive security point of view and therefore IAPTF reachability script named test-reachability.py are deployed within IAPTF-Prep and facilitate using Internet Control Message Protocol (ICMP) shortly known as ping. The scripts will run on the entire network using different machines in different subnets as starting points and initiate an ICMP request between source and destination host and only validate the reachability if the ICMP request is successfully. In the case where the destination host is behind a firewall and thus non-responsive, ping echo reply from the firewall reply should not be considered and the matrix value is set to zero “0”. Finally, it is worth to highlight the reciprocity nature of the reachability matrix which completely different from the proposed security clustering where the firewall one-way filtering characteristic is captured and utilised in determining security clusters’ composition.

Table 8: Example network topology for the network represented using an adjacency matrix.

.

subnet	SN0	SN1	SN2	SN3	SN4
SN0	**1**	**1**	**0**	**1**	**0**
SN1	**1**	**1**	**1**	**0**	**0**
SN2	**0**	**1**	**1**	**1**	**0**
SN3	**1**	**0**	**1**	**1**	**1**
SN4	**0**	**0**	**0**	**1**	**1**

6.2.4 Network Vulnerability Assessment

The vulnerability assessment data includes static (initial output) discovery, scanning, enumeration and vulnerabilities assessment. Enumeration data are embedded in IAPTF and used to represent fully or partially information about ports, services and version of software running on different machine in the target network. These data are, to a certain extent, analogous to vulnerability assessment software such as Nessus with special focus on the security side of services, the data are validated to extract service and security information that hackers would exploit and simply ignore all other non-vulnerable running services and software. Each service is defined by its name or a short abbreviation such as FTP, SMB (for Samba), or HTTP which are unique and specific in term of version, so it would be possible to track vulnerabilities and know what services require patching SMB-v3.6.x. The ultimate goal of is to achieve root or admin control throughout executing the adequate Meterpreter session against the set target. Thus, the service version and port scanning and enumeration will help to narrow down the required activities so this goal is achieved and thus IAPTF agent to earn full control over the target machine. In IAPTF, these activities are launched at first phase independently from the Metasploit MSF using small pieces of software and scripts (Nmap NSE and Nessus) with the aim of building enough memory for IAPTF functioning. The following steps are performed when the initial enumeration and scans are launched:

- scan all reachable (from the internet hacker computer) for open ports and identify running services

- complete the enumeration of all identified services
- validate the results and launch the initial vulnerability assessment
- confirm existing vulnerability and export all outputs into the relevant XML files to be uploaded into IAPTFR-Prep

As with manual PT process, IAPTF-Prep enumeration module will initiate by extracting the important data using externally crafted Nmap Python scripts together with two built-in NSE (Nmap Scripting Engine) scripts Vulscan and Vulners. The basic concept behind NSE is the offered automation and flexibility to weaponize Nmap features. Nonetheless, it remains heavy to run and thus often avoided by PT experts who adopt more hand-crafted simple scripts to automate a wide variety of tasks related to a given scenarios and which can be launched in parallel with the enhanced efficiency. IAPTF-Prep imports and uses directly Nmap's Vulners and Vulscan to identify, fingerprint and enumerates ports services along with finding relevant information about the CVE of a service such as SSH, RDP or SMB. Moreover, external scripts are embedded in the IAPTF-Prep module to improve data capturing and validate information related to vulnerable services. Vulners script offers the possibility of working online and querying the CVE and NVD databases every time the NSE script is used. Vulscan queries to either a online database on internal pre-downloaded database synchronized with ExploitDB. Vulners scripts are compatible with VulDB and NVD which are used as references for defining vulnerabilities severity ranging from: Critical, High, Medium, or Low which match the CVSSv3 criticality score between 0 and 10. After this step, IAPTF-Prep second module will oversee searching the extracted data aiming to identify initial vulnerabilities out of NSE scans. Then comes the role of PreProcessing.py main program which will process the data into IAPTF memory following CLIPS expert system format which was designed to fit our research RL needs and structured in the form of decision trees that remain compatible with other AI applications. Final data processed will contains seven fields, namely:

- Machine OS, version/architecture, and patch/service pack
- Security clustering and subnets defense
- Open and Filtered Port as found by Nmap
- Running Service with version and variant if applicable

- Imported CVE database set with relevant vulnerabilities
- Available exploits and payloads.

6.2.5 Security and logging data

The final component of the network model are the firewalls that exist along the connections between any subnets and between the network and the external environment. Firewalls act to control which services can be communicated with on machines in a given subnet from any other connection point outside of the subnet. They function to allow certain services to be used and accessed from machines within a subnet with the correct permissions, while blocking access to that service from unwanted entry points. Each firewall is defined by a set of rules which dictate which service traffic is permitted for each direction along a connection between any two subnets or from the external network. Figure 58 shows an example firewall that sits between subnets and which allow access to the SSH and FTP services on machines on SN3 from machines on SN1 and SN2 and access to FTP and HTTP services on machines on subnet SN1 from machineson SN3. In a real-world setting firewall rules are typically set by defining which port can be accessed, however for simplicity and since for most cases the same services are run on the same port numbers, we have decided to instead define rules by service rather than port.

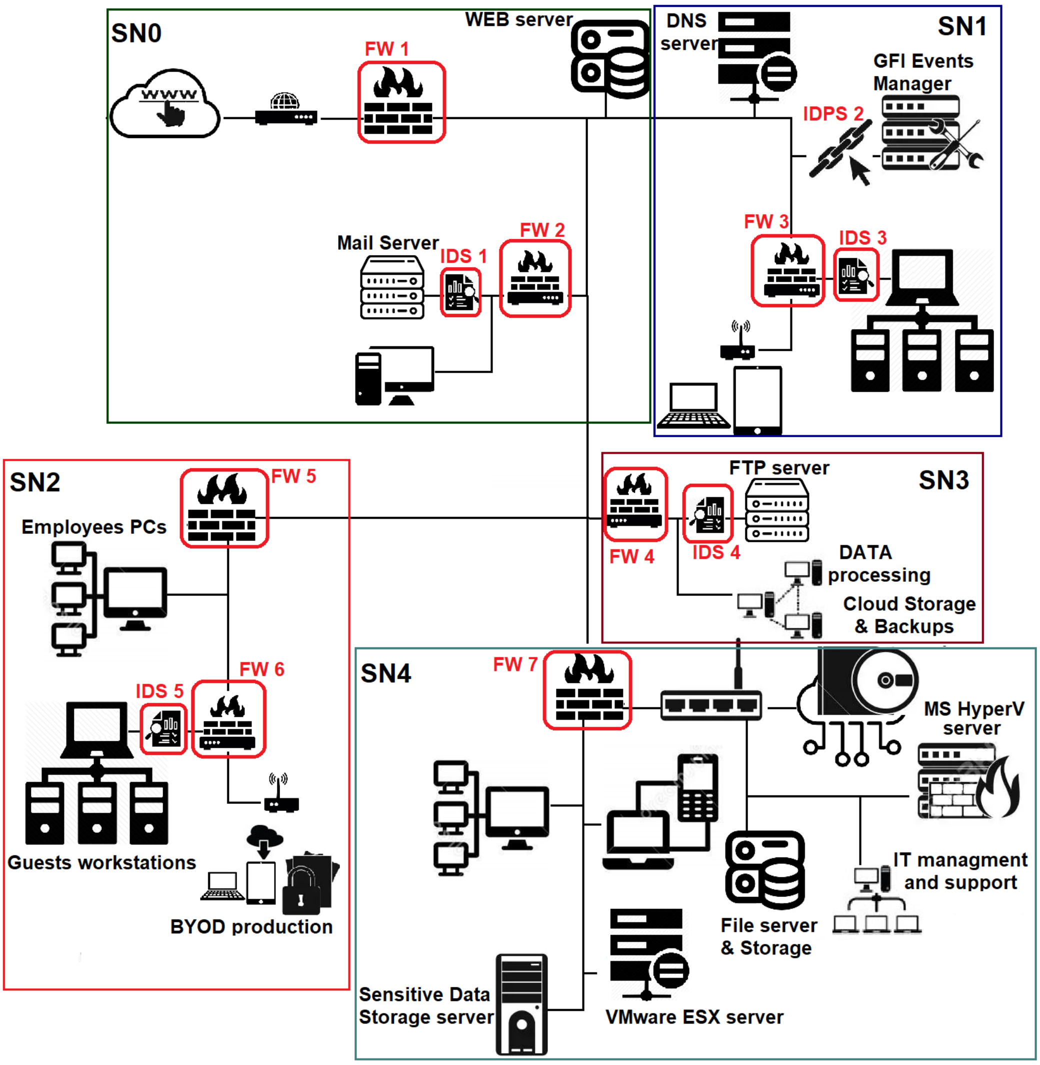


Figure 57: Example firewall and Intrusion detection systems location and impact on the connection between subnets SN0, SN1, SN2, SN3 and SN4 and defines which services are permitted in each direction.

Finally, the IAPTF-Prep include a python scripty named VulnAssessment.py which imports directly Vulnerability Assessment to concatenated it to the data extracted regarding security, topology, and networking. IAPTF-Prep imports direct data from the Nessus-NM and Tenable-IO in form of vulnerability scanning and assessment output XML files.

6.3 IAPTF-Processing module

This section will detail the first modules of IAPTF namely IAPTF-Prep and IAPTF-Proc which are in fact a set of program and scripts enabling the acquisition, extraction, and processing of the different data from different sources. This module oversees moving from Networking, Security and Vulnerability Assessment data to build the different POMDP environments which fits our model representation of the real-world PT practice environment. As we discussed in Chapter 5, the POMDP representation of PT practice requires projecting to the best all features and characteristics of the real environment. The system will thus require a memory that increases alongside its development and capturing of features and data increasing. The RL agent memory and environment model often consume most of the storage capacity. If there are S states and A actions, then a complete model will take up space proportional to |S| x |S| x |A| because it maps state-action pairs to probability distributions over states. By contrast, the reward and value functions might just map states to real numbers and thus be of size S. As an exception, RL learning policy search approach where a second value function determines is in the agent agenda (not the reward function) the system memory can increase further.

As the proposed RL model focuses on the relevant aspect of the PT practice which are also the cyber attackers used key information, this module aims to produce POMDPs files from the raw data received from IAPTF-Prep and using Python scripts to perform the pre-processing, processing and structuring and ultimately produce optimized representation of the PT domain in the form of POMDPs problem. In IAPTF-Proc some fundamental networking features and aspects were neglected and not considered in the environment modelling. This is due to the irrelevance of this data for the PT practice and include such useless details will result in enlarging the domain size and thus compromise the efficiency. The sought after IAPTF-Proc is a multi-scripts module which facilitates the processing of different format extracted data notably networking, vulnerability assessment and security LOG and XML file to produce cleaned IAPTF-memory format data which then be used in the representation of the early-stage PT POMDP environments. As we had already discussed in the previous chapter, the RL can be of great help to the PT community only when properly embedded notably by striking a balance between contribution and cost in terms of time and computational resources. With computer networks expanding in size, the PT task represented in the form of POMDP environment size will increase too, resulting in high scalability issue as the solving becomes time and resources consuming. The reason behind human expert success in PT is that testers often take advantage of the context and utilise this to reduce the workload by performing multiple pre-elimination of the useless or non-productive testing scenarios where the tester is sure that the outcome would be negative, or the cost would exceed the expected reward. The IAPTF-Processing system is a crucial part towards a fully automated and intelligent PT framework by preparing the field for the IAPTF-Core to solve relevant and potentially rewarding problems which enhance considerably the testing time and reduce the use of resources along with generating meaningful and consistent results. The IAPTF-Proc is made of many scripts and a large module. The main scripts are configuration.py and security.py and the security-clustering.

6.3.1 Machines configurations and defense

The most primitive security entity of network security is the machine. A machine in IAPTF refers to any IT device that is connected to the network and hence can be reached and exploited. In this module, each machine i is defined by its address which includes the following:

- machine name in the form of M_i (machine), R_i (networking router or switch), V_i (virtual machine) or T_i (IoT device),
- subnet to which this machine belongs,
- security protection (DiD score) level.

An example machine definition can be for a given web application server in DMZ called M2 belonging to subnet SN1 and having a medium host security protection of 3 out of 5, this machine is represented as M2_SN1_3. It is important to highlight that the security protection level is completely independent from the machine value from the point of view of sensitivity, as such machines that the hackers and thus pentesters would want to gain access to or that the owner wants to protect. The other important data is the running services that are used for communication purposes only. This can be either within the same subnet or any reachable subnets. The services available on each machine are defined on its configuration profile and thus for each machine on the network as this data is largely heterogeneous and machines does not necessarily have the same configuration such as workstations, Web servers, file storage. Note that these services present on a machine will be added later to the other services when it comes to build the vulnerability profile which aims to list services vulnerable or potentially vulnerable to attackers aiming to exploit.

6.4 Medium and Large networks Security Clustering

Performing PT different tasks requires the pentester expert to meet some basic requirements. One of the most important requirements to know and understand the assessed network “general” security topology which will result in identifying the relevant attack paths based on the knowledge of existing connections between Machines (systems) and/or Sub-Nets. This is a crucial phase as any cyber attacker will, after performing the initial reconnaissance, try to sketch a logical security map of the target network and assess the ability to reach the targeted machine. Often, the PT expert can initially reach only a few machines from his external (Internet) position (high-value targeted machines cannot initially be reached) and the overwhelming majority of assessed network machines are unreachable directly from the Internet. Thus, a progressive approach is unavoidable, the tester will attempt to gain partial or full control of the reachable machines and then use these controlled machines as a launching point for future attacks as it is likely that they share a sub-network with one of them: and those which are unreachable from any controlled computer.

The network clustering is a crucial phase in the IAPTF as it will allow an easy and simple RL modelling of a large network PT environment which mirrors perfectly the reality. In real-world PT, it is hard to deal with each network connectivity separately and this case will create an additional problem to the existing ones: the large environment problem and complexity of representing the actual topology. The network clustering is therefore our proposed method to incorporate the information about the network connectivity within the RL model along with maintaining the solvability of the problem in an adequate amount of time.

The following network clustering method aims to divide the medium and large size network (or even a small network) into a certain number of clusters. The definition of the word cluster here is a set of machines and systems that:

- belong to the same security level (protected by the same set of security devices and mechanisms) and,
- reachable to each other (direct connection without passing through any network security component excluding the host-based systems and software) and,
- serving a similar or complementary purpose within the network such as Data Server, Workstation, or Printing.

As we already mentioned in the previous chapter, the proposed clustering of the network is quite simple and does not obey always conventional networking criteria and rules (sub-net, LAN, MAN, WAN, WLAN ….). Instead, we will utilises a security-based method by:

- Building a complete and comprehensive map of the assessed network including the existing divisions following the networking (functional) rules (WLAN, LAN, Internal, Public …) map,
- Determine network reachability data which reflect the ability for a machine to reach other machines without passing through the existing security measures. This is done by exploiting data from the network equipment ensuring the isolation such as firewalls, IDSs, Routers, DMZ.

As illustrative example, the assessed small LAN in figure 59 composed of 18 machines including 11 computer machines named M0, M1 …, M10 and 7 networking devices (firewalls, routers and IDSs) in addition to the ethical hacker machine which is connected through the Internet (external) but will be considered later as an independent cluster for the modelling purpose. The 18 Machines are divided from networking point of view into two sub-networks namely SN0 and SN1. Nonetheless, the proposed clustering approach will divide the network into security clusters following a rigorous and realistic approach inspired from the real-world situation of security isolation and how hackers perceive the network (clustering approach detailed in next section) resulting into five different security clusters (C0, C1, C2, C3 and C4) with each cluster grouping several machines such as C_1 which is the DMZ (M0 Application Server and M1 Web Server).

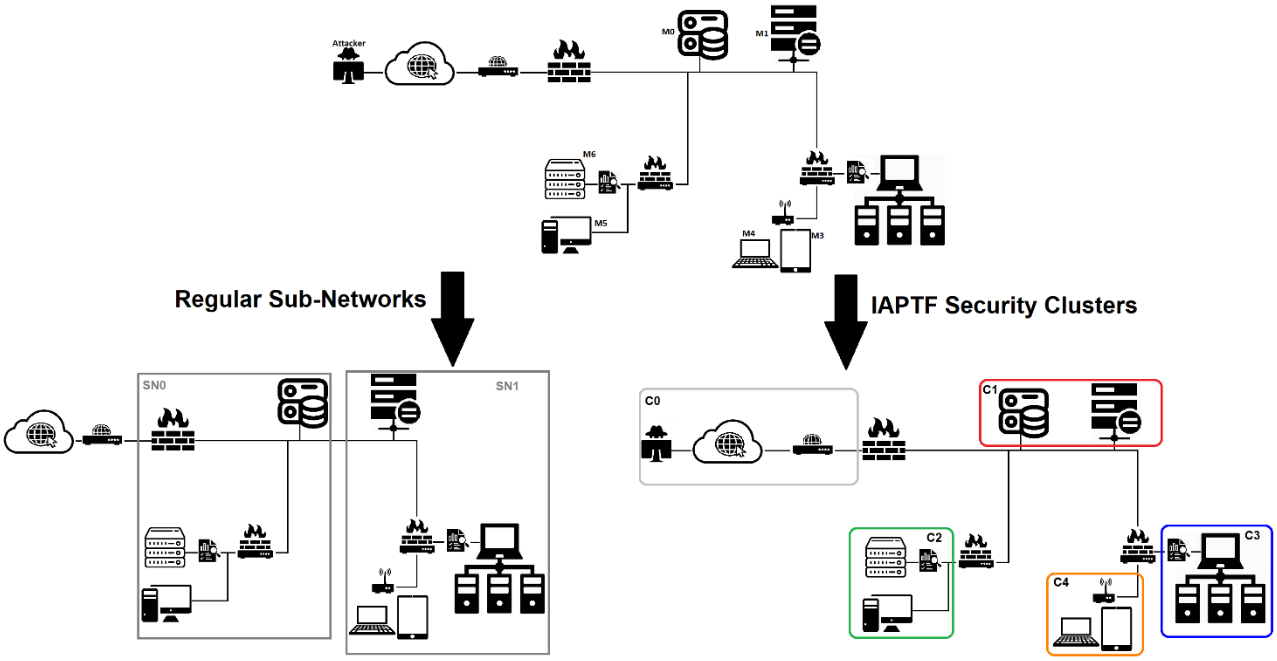


Figure 58: An illustrative example of small LAN made of two sub-networks (SN0 and SN1) and parallelly five clusters (C0,C1,C2, C3 and C4).

In our proposed POMDP model only details about the machine-to-machine connections information are included initially. This is in reality the reflection of cluster-to-cluster (inter-cluster connectivity) which provides information regarding the type of connections (active and available) along with the type of the used protocol, security and other relevant information such as the number of hopes.

The number of hopes is purely related to the existing security mechanism separating two different machines (clusters) such as firewalls, IDSs, IPSs, routers. In the connectivity representation only one route is mentioned for the same two entities giving the same connection protocol, as example is M0 and M2 can be connected via an TCP-FTP connection via more than one route, giving the fact that such scenario (FTP connection) has the same security protection for all the connectivity only the shortest (with less hopes) route is represented as follow:

- C0-C2-TCP-FTP-1: Direct intra-cluster connection with a firewall (FW0) separating the cluster C0 and cluster C2.
- C0-C2-TCP-TelNet-2: connection via a third cluster (C1) which will result in passing through an additional security mechanism (FW2)

- C0-C2-TCP-SSH-2: composite route including connection passing through two or more other clusters' security isolation and filtering to reach the other end.

The most important component of IAPTF-Prep is the security clustering scripts. This module output is twofold. Firstly, the clusters constitutions in term of name and IP address of each machine and the belonging cluster. Secondly, the head (or heads) of each cluster which is in theory the most vulnerable machine with root/admin control. The proposed approach focuses on a key characteristic of network PT practice namely security isolation overview. This is a common approach for hackers and cyber attackers as they oversee the target network from a security point of view and not simply networking functioning in the aim of extracting key information about the security isolation and reachability. It will be noticed in the proposed approach we call clustering that some fundamental networking features and aspects were neglected and not considered in the task of dividing large LANs and WANs and thus modelling them into smaller POMDP environments. Finally, it is important to highlight that one-way filtering approach adopted in firewalls, routers and IDPSs is a key element in our proposed clustering approach making the clustering approach fundamentally different form the regular subnetting approach which is by default symmetric (reciprocity of reachability between subnets) as illustrated in the figure 60 where SN0 and SN1 are considered the only subnets in the network following the symmetric reachability approach.

On contrast to the subnetting, the proposed security clustering produce a completely different result. This proposed approach backs a hierarchical structure of POMDPs inspired from the corporate network architecture which is, from a hacking point of view, characterized by multi-layered security protection (also known as Defense-in-Depth or shortly DiD). In this module, data regarding DiD and other security and isolation device locations and configuration are used directly and sometimes assisted by hand-crafted scripts, to find a security-based clustering of the LAN and MAN. We then investigated a more efficient technique to automate the decomposition process and designed a security-oriented clustering scripts which use conventional networking data such as reachability, routing table, access lists (ACLs) along with the defensive cyber security information such as firewalls, IDPSs position and functioning to provide an automatic decomposed of medium and large size networks in an appropriate manner. In few cases, where the size of some cluster exceeds 50 machines, this cluster is divided into 2 to 3 sub-clusters which will be accounted as distinct security clusters. Such decision was made after observing IAPTF performance decline in large networks, and it allows the computation of best policies, while most reducing the complexity required to compute these policies. In term of adapting the proposed POMDP representation, we opted for a two-levels approach. First is to consider each security cluster as a separate network and only represent the data about machines in that cluster in the low-level POMDP environments. Then we represent the network of the head of each cluster and including all possible connections including machine-to-machine and cluster-to-cluster connections information to fully reflect the real-world inter-cluster connectivity. In the Clustering module we implemented three scripts. The first script defines the cluster composition and named Cluster_composition.py which the output is the full security clustering of the network. In other words, the result is many clusters (number will depend on the size of the network, configurations and security setting DiD but will be at least 3) which each contain many machines belonging to the same cluster will be treated similarly. The idea behind this assumption is the nature of the networking environment. The machine belonging to the same sub-net often have a similar defense and protection. In fact, an attacker who gains a control of a machine will easily progress to the rest of the machines on the same sub-net or cluster as a trust relation (in some applications) exist between the infected machine and the remaining allowing the attacker to take advantage to run (execute) the exploit against those machines. The second script named Clusters_Connectivity.py oversees capturing and processing information regarding the type of connections (active and available) along with the type of the used protocol, security, and other relevant information such as the number of hopes. The number of hopes is purely related to the existing security mechanism separating two different machines (cluster) such as firewalls, IDSs, IPSs, routers etc. The full detail about the representation is detailed in the previous chapter. The following Network Clustering method aims to divide the medium and large size network (or even a small network) into a certain number of clusters. The definition of the security cluster concept in IAPTF is simply a set of machines and system that have to same security protection and have the same attack surface (exposure) from a hacker point of view, and this is elaborating following the concatenation of networking, security, and vulnerability assessment data. As results, applying the Clustering python script, to the output is illustrated in figure 60 where each cluster group several machines such as C1 which is in this case the DMZ (M0 Application Server and M1 Web Server).

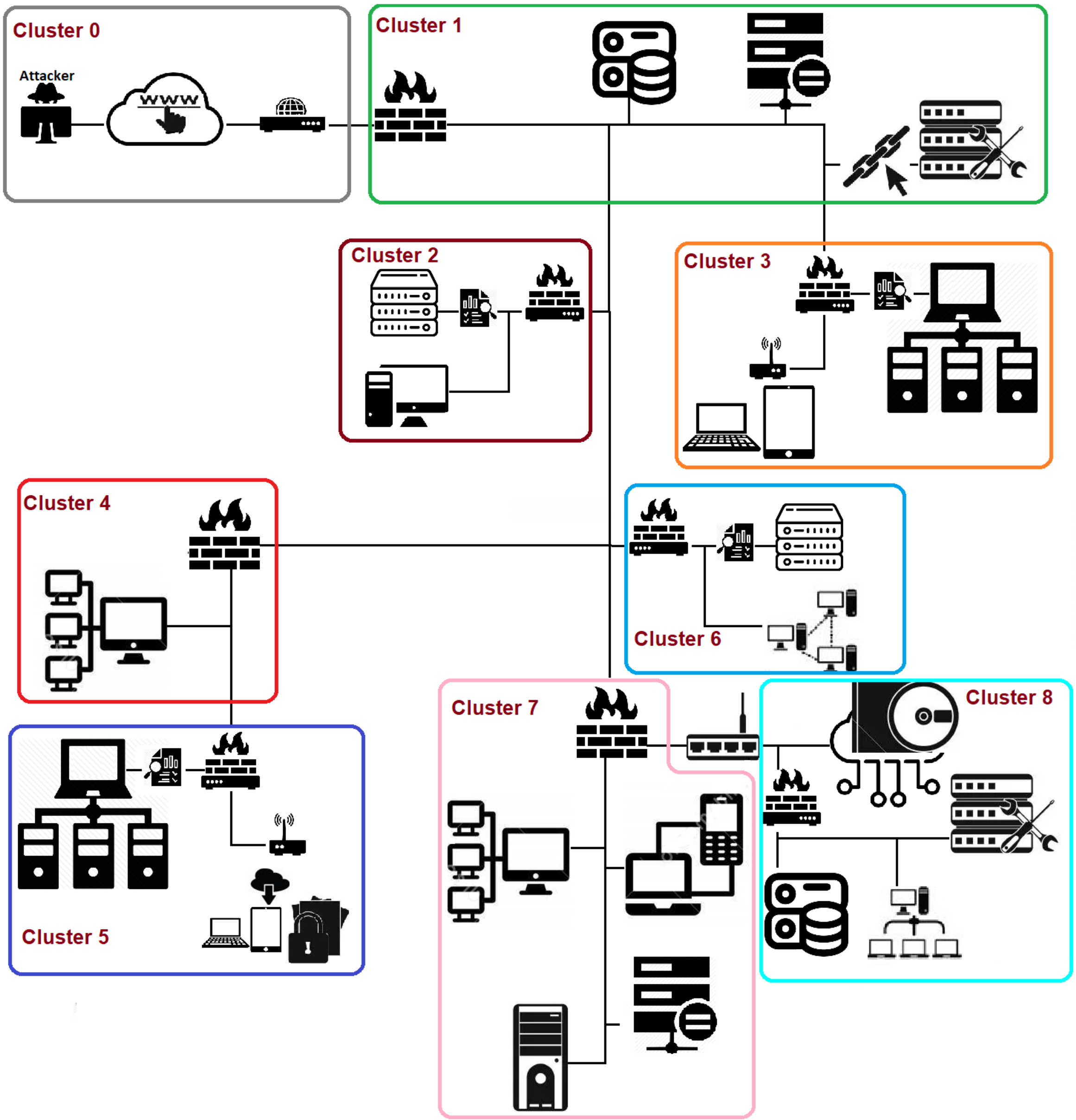


Figure 59: An example network made of nine security cluster (initially five sub-networks) C0, C1…C8.

The third scripts named HoC.py is in charge of identifying the head of clusters HoC (on some occasion more than one machine is designated as HoC). The idea starts by reducing the low-risk machine first and only focus on machine with a large attack surface. Attack surface stands for the number of open ports, running services and associated vulnerabilities in the machine making it more likely to be targeted and exploited by hackers. We exclude any honeypot machine (or even a honeynet) from being HoC. An illustrative example with the output of HoC.py on the test-bed network basing on IAPTF-

Prep vulnerability assessment output data is in the figure 61. In this research will initially deal with Clusters as Machine and thus all the machines belonging to the same cluster will be treated similarly. The idea behind this assumption is the nature of modern corporate networking environment. The machine belonging to the same sub-net often has a similar defense and protection. In fact, an attacker who gains control of a machine will easily progress to the rest of the machine on the same subnet or cluster as a trust relation (in some applications) exists between the infected machine and the remaining allowing the attacker to take advantage to run (execute) the exploit against those machines. It is obvious that reducing the size of the inter-clusters network is not a straightforward activity. As shown above many security clusters will have more than one machine maintained such as C1, C7 and C8. And some clusters will have a set of machines with identical configuration maintained such as C4 and C7 again. Therefore, we opted for a rigorous approach to reducing the size of the final Inter-Clusters network and only include the strict minimum so the associated POMDP environment size can be maintained to small size and therefore enabling a fast exact solving. The figure 61 illustrates how Head of Clusters are selected. Therefore, we introduce the head of cluster notion. Initially, this notion was defined to serve simplicity purpose as IAPTF will need to complete the PT tasks including the inter-cluster testing and to do so we will deal with the entire cluster as one single machine from security point of view and thus all the machines.

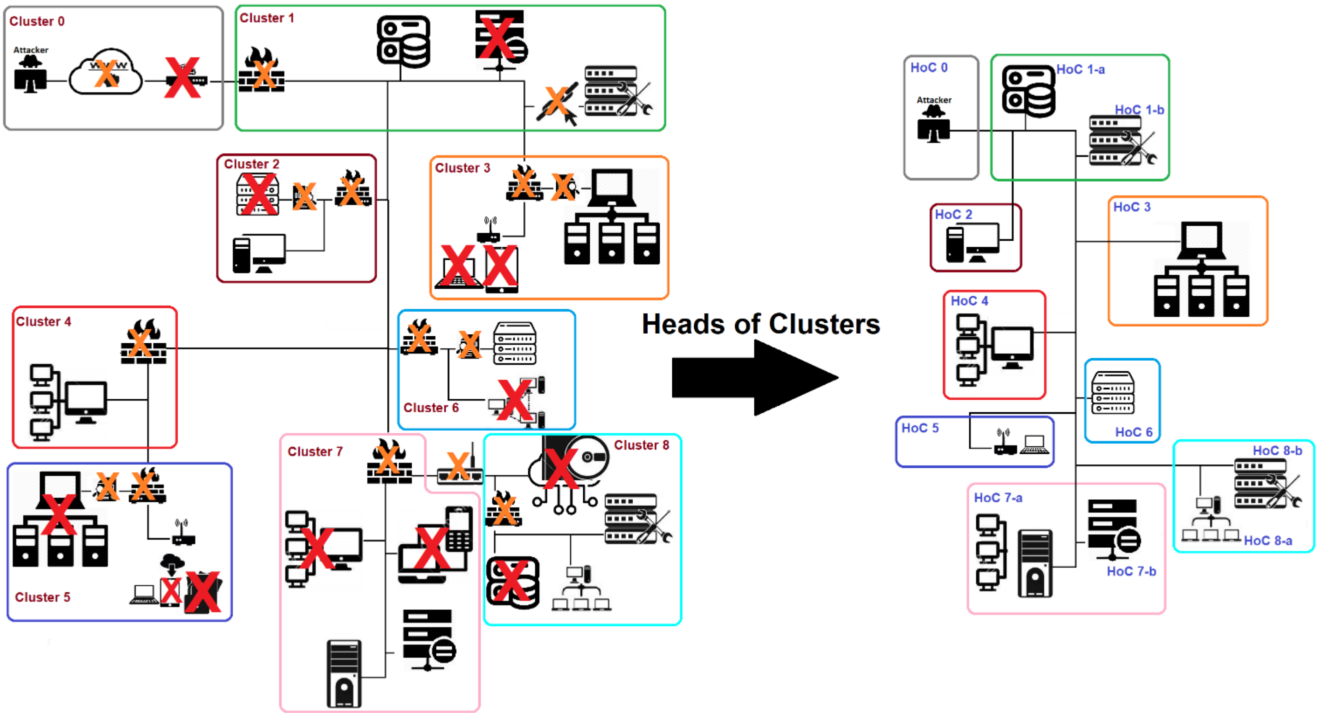


Figure 60: illustrative example of clusters' size reduction by excluding low-risk, and redundant machines following by the selection of heads of clusters (HoCs).

Thus, Internet (public) cluster will have hacker's machine as the head of Cluster and thus the starting point. The second challenging case is often the DMZ which is C1 in this case as machines in this zone are often more exposed and have a larger attack surface. In this case we strike a balance between reliability and flexibility and implemented a multi-HoC function which compares the vulnerabilities scores of each of the finalist candidates HoC and only select two (worst case) and designate them as HoCi-a and HoCi-b with i is the cluster ID. The final output of the clustering phase is 09 small size LANs and a network constituted from Heads of Clusters. This enables us to represent two-layers POMDP representation of any medium or large LAN as illustrated in figure 62.

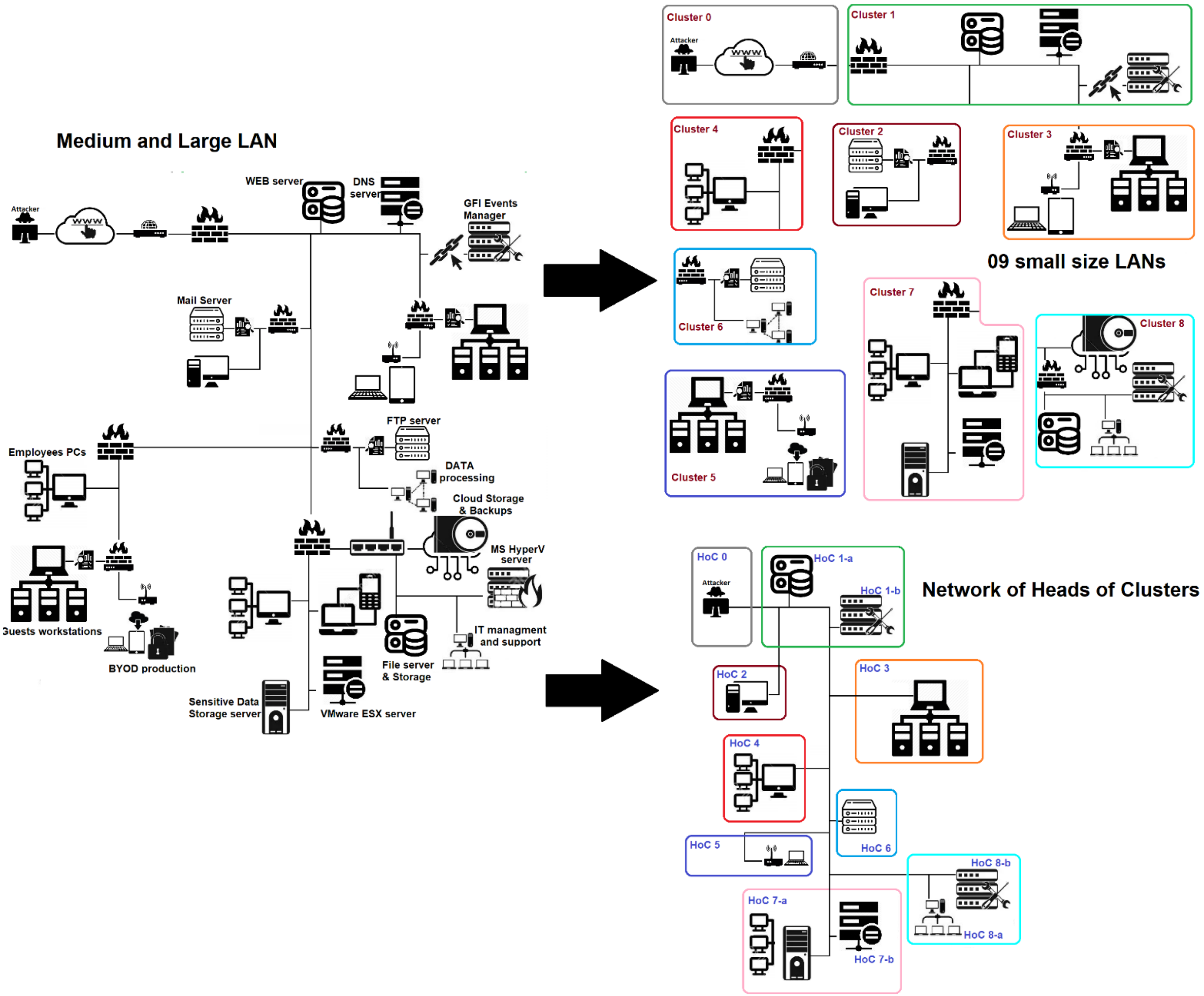


Figure 61: Overall security clustering output including cluster composition definition, election of head of cluster.

6.5 Heads of cluster Network

In the connectivity representation only one route is mentioned for the same two entities given the same connection protocol. As an example C0 and C2 can be connected via an TCP connection via more than one route, given the fact that such scenario (FTP connection) has the same security protection for all the connectivity only the shortest (with less hopes) route is represented as follow:

- C0-C2-TCP-FTP-1: Direct intra-cluster connection with a firewall (FW0) separating the cluster C0 and cluster C2.
- C0-C2-TCP-FTP-2: connection via a third cluster (C1) which will result in passing through an additional security mechanism (FW2)
- C0-C2-TCP-FTP-4: composite route including connection passing through two or more other clusters' security isolation and filtering to reach the other end.

Finally, the network of HoCs will be represented as POMDP and then solved in the same manner as with small network case with the only difference of using the resulting Policy Graph as an input for a global attacking vector script named Overall_Attacking.py. this later will produce a comprehensive attack vector policies covering the entire medium and large size network by combining PGs resulting from intra-cluster solving and the PG resulting from the inter-clusters. This script will be operating directly on data stored in IAPTF memory and after establishing the global PG the results will be processed again using another scripts named Generalisation_PG.py which will remove any specific information or machine ID and make the decision policy general for future use by the Expert System CLIPS.

Table 9: IAPTF-Prep modules, programs and scripts roles and description.

File	Description
Connexion.py	Script to capture (import) networking and connection data, process and structure i according to the IAPTF requirements
Topology.py	Script to capture and determine network topology and subnetting, process and structure it according to the IAPTF requirements
Scanning.py	Program to capture (import) all network discovery, fingerprinting and scanning tools output, process, and structure it according to the IAPTF requirements

LOGs.py	Script to import, clean and structure according to the IAPTF requirements all available networking and security devices and software log files
Security.py	Script to capture and process security defense (DiD) data and network security layout and structure it according to the IAPTF requirements.
VulnAssess.py	Program to collate and process all captured data output from Connexion.py, Scanning.py and Security.py to then structure it according to the IAPTF requirements
Clustering.py	Program to collate and process all captured data output from Connexion.py, Topology.py, LOGs.py and Security.py to then defines security clusters.
Generator.py	Script to collate and generate the POMDPs environments from the input

6.6 IAPTF memory and expertise handling

In our proposed framework, the main challenge we face is the expertise capturing and re-usability which is a crucial component in context of PT practice that is highly repetitive and the decision-making it a cornerstone in cyber security field. To take full advantage of this particularity rather than having it as counterweight to our framework performances, we proposed a module within IAPTF that is dedicated to extracting the knowledge output during PT tasks and make it general (perform generalization processing) then store it into an Expert System for future use. The expertise capturing, generalization and handling activities are performed mainly within two modules: IAPTF-Memory and IAPTF-Expert_System. We distinguish two functioning diagrams in IAPTF: the first only account for the expert system and expertise capturing and processing programs which require the supervision of the human expert (mainly in early life stage pf IAPTF) and a second comprehensive functioning diagram which embed the Expert System and Expertise acquisition within the IAPTF-Core where the RL Agent acts instead of the human expert. Figure 63 details the comprehensive functioning diagram.

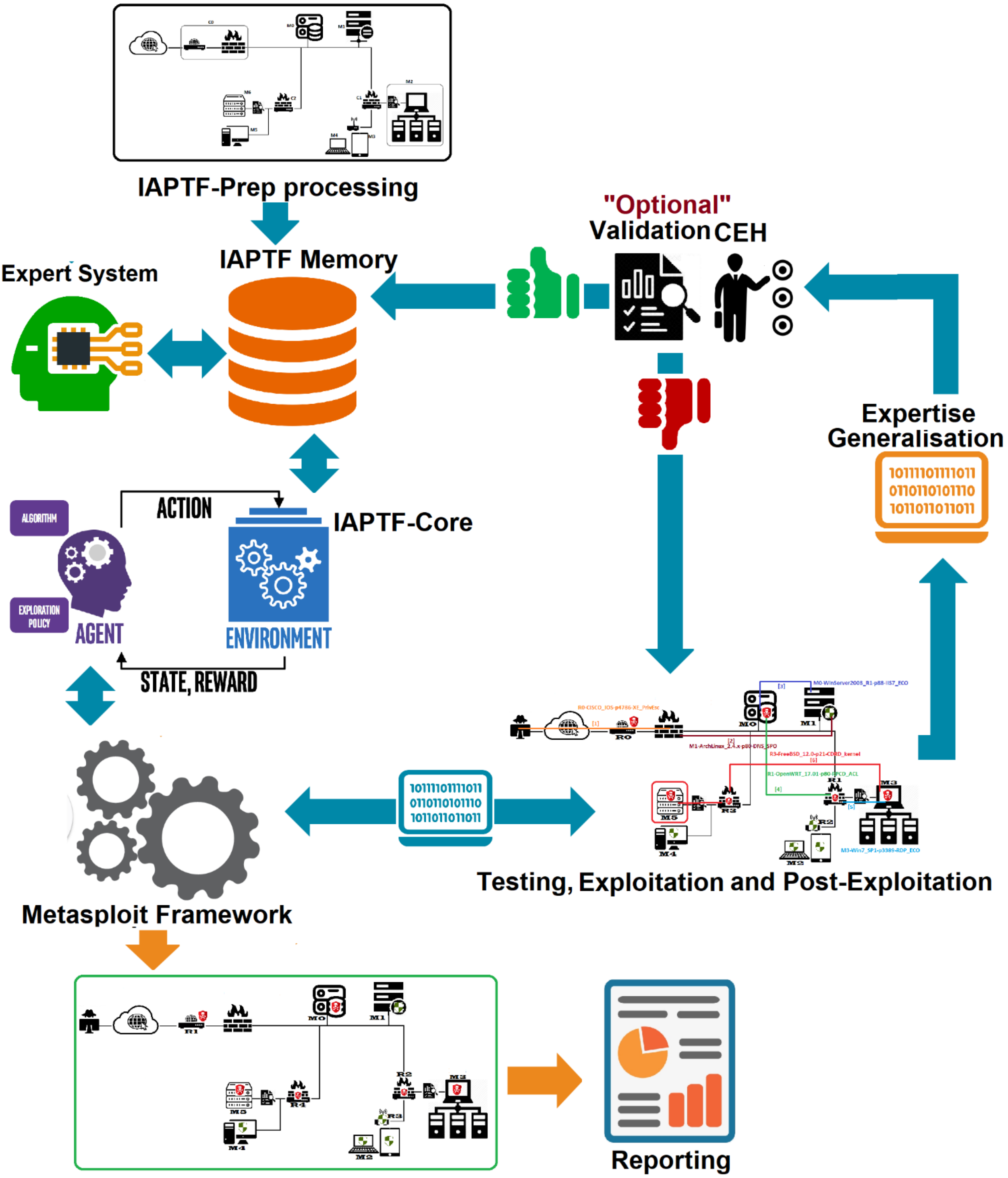


Figure 62: IAPTF-Core with Expert System extraction, validation, and replay diagram.

The first part of this activity is the generalisation tasks which are done through python scripts directly on the output XML files of the POMDP pentesting solving results. Once done, we progress into storing this precious knowledge in a basic Expert System (we annotated ES) using CLIPS. The diagram of IAPTF shown in figure 64 refers to the ES. There are several different modules in the proposes architecture, but we only focus in this section on functions which offload the POMDP solving XML files and extract from the policy graph (PG), based on standard formula and input regarding network configuration the decision (acting) policy made in each situation which is extracted in their original context to avoid irrelevant generalization. Then, a python script named "ES_Generalization.py" is applied to data to produce a general format from which as specific data is removed such as IP addresses, Machine name, non-generic data.

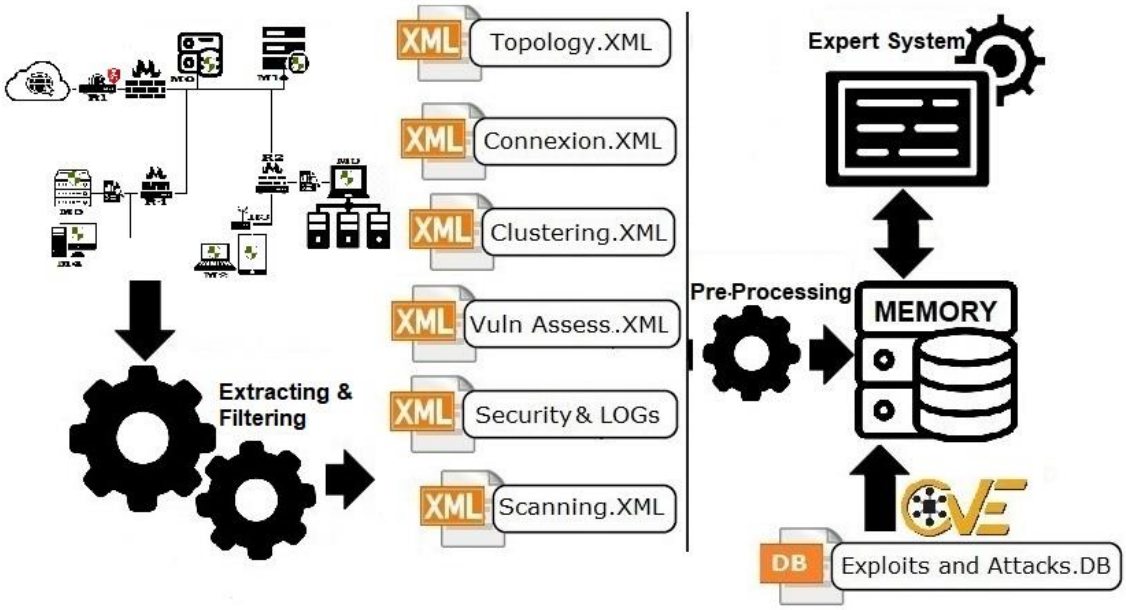


Figure 63: IAPTF-Preparation and IAPTF-Processing detailed functioning diagram.

6.7 CLIPS Expert System

The next step in building the IAPTF memory is the implementation of the Expert System which will oversee storing and reusing of decision policies. Since the aim of this research is mainly applying RL in PT practice, the ES comes in second priority, and we decided that we will not implement a heavy weight ES within IAPTF and only rely on CLIPS which is a public domain software tool for building expert systems. We will briefly describe how the general production system tool CLIPS is used to extract, process, store and reuse expertise for network PT purposes using previous testing captured experiences. The proposed system can also be utilised in case of decision-making assistance notably when retesting the same asset. In term of expertise capture and handling, we opted for a direct application of CLIPS Expert System to achieve our objective of capturing and replaying human CEH expertise and knowledge. CLIPS is a complete environment for developing IAPTF expert systems which includes an integrated editor and a debugging tool and enable inferences or reasoning. The CLIPS provides the three key elements of: memory for data, knowledgebase, and rule-base. The written program consists of rules, facts, and objects with the inference engine to select rules (action) to be executed for a given object. In IAPTF, we built a PT expert system by performing some modification into the default CLIPS code by introducing features such as single and multiple string pattern matching, certainty factors and timestamp with uses of MSF plugins adapted for pre-processing. The complexity of MSF in term of data handling and storing add nonetheless more complexity and challenges for our proposed Expert System. To overcome these shortcomings, we proposed an integration of our developed CLIPS Expert System with “MSF POSTGRESQL” database. Thus, IAPTF allows the simplification of the complex data workflow by considering complete testing and attacking scenarios instead of atomic actions. Finally, the modular structure of IAPTF enables us to propose a second variant of the expertise handling diagram which doesn’t account for the RL IAPTF-Core. This is particularly useful in case for retesting without changes (security compliance and auditing) and in context of small networks not requiring an advanced exploration but instead a check-list testing. Figure 65 illustrates the IAPTF framework expertise handling when excluding the IAPTF-Core RL module and testing results on different network size scenarios are detailed in chapter 7.

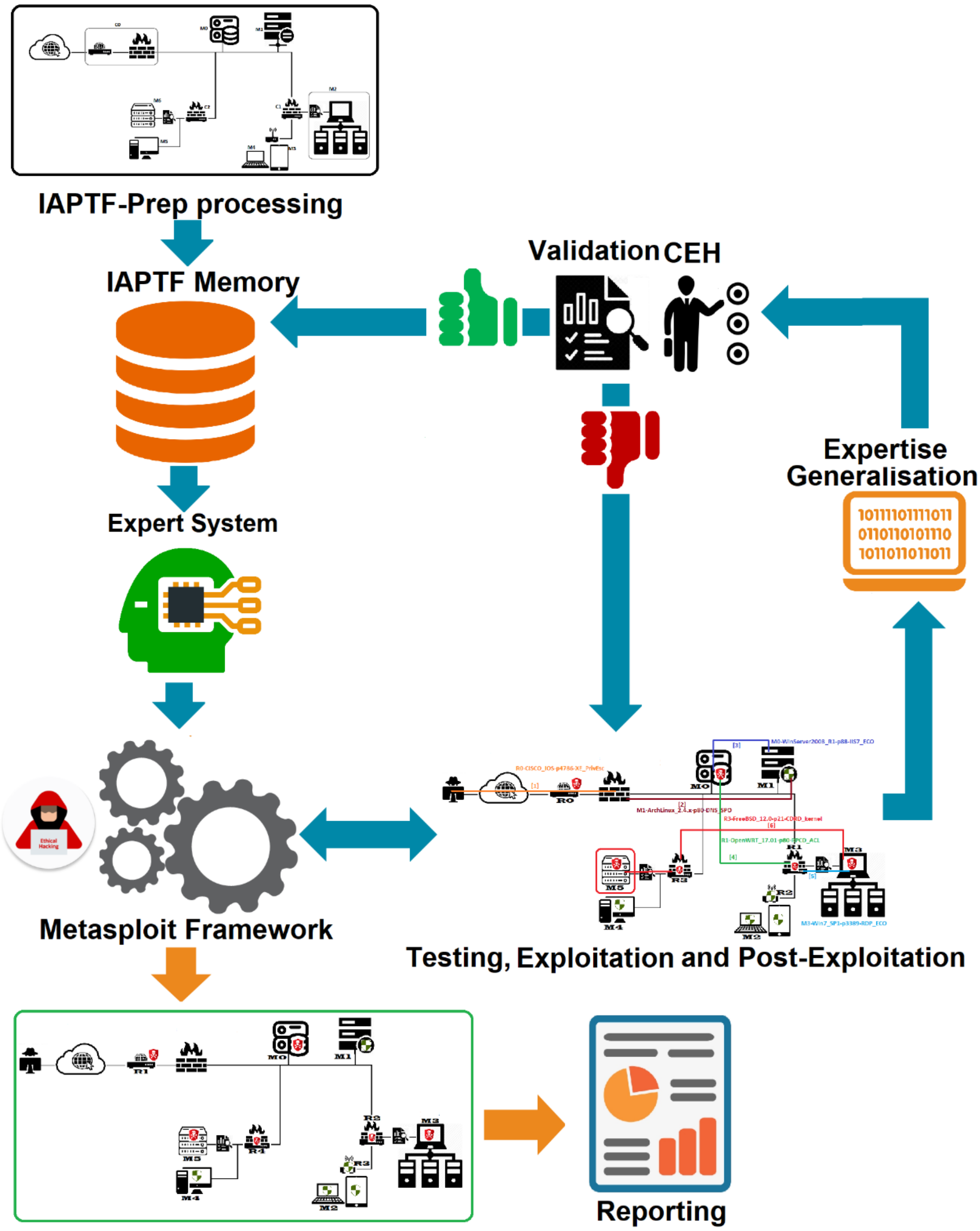


Figure 64: IAPTF-memory and Expert System expertise extraction, validation, and replay diagram.

6.7.1 IAPTF-Core:

The main component of IAPTF is the RL module which rely on the proposed POMDP modeling of PT practice as detailed in Chapter 5. In this section we will detail IAPTF-Core functional diagram and the different arrangement made to enable this module to access POMDP files stored into the IAPTF-Memory along with the extracted expertise sored in the CLIPS Expert System. The proposed IAPTF-Core is fully independent, modular, and optional and can be embedded with any other industrial PT framework. The current version is associated with Metasploit framework (MSF) and Nessus as an external module communicating via customized python scripts imputed through Metasploit MSFRPC API.

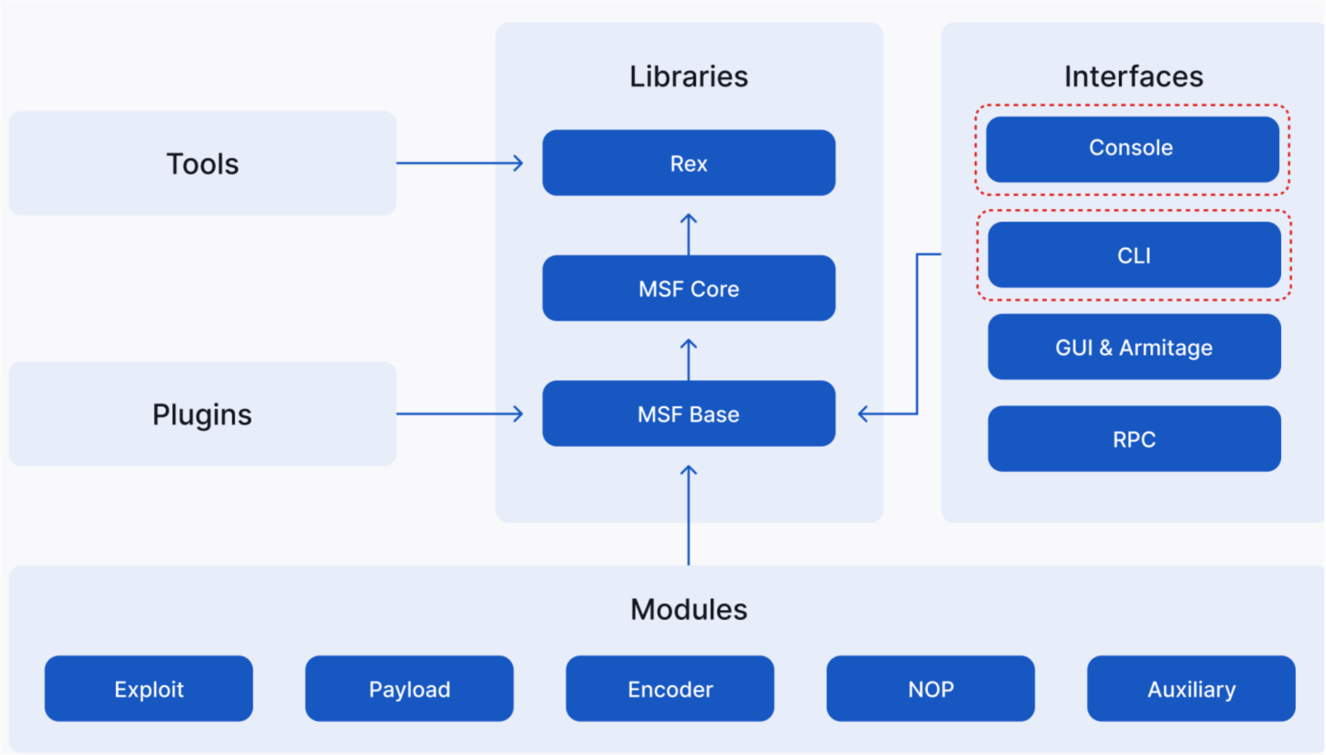


Figure 65: Metasploit Community and Professional framework architecture [102].

The purpose of such configuration is to avoid modifying the core component of the MSF and allowing us, for research purposes, to measure the IAPTF performances independently from the used PT system or framework. The Console and CLI circled in red are the entry points for IAPTF-Core to communicate and exchange data with MSF. The figure 67 illustrated the different components, functional diagram and interaction mechanisms adopted in IAPTF-Core.

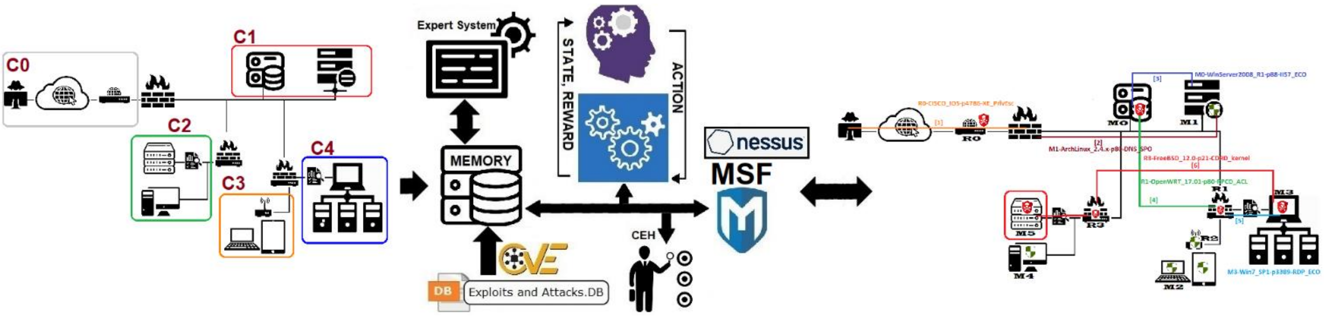

C0
C1
C2
C3
C4
Expert System
MEMORY
STATE, REWARD
ACTION
CEH
CVE
DB
Exploits and Attacks.DB
nessus
MSF
R0-CISCO_IOS-p4786-XE_PrivEsc
[1]
R0
M0
M1
M0-WinServer2008_R1-p88-IIS7_ECO
[3]
[2]
M1-ArchLinux_2.4.x-p80-DNS_SPO
R3-FreeBSD_12.0-p21-CDRD_kernel
[6]
R1-OpenWRT_17.01-p80-RPCD_ACL
R1
[4]
[5]
M3
M5
R3
R2
M4
M2
M3-Win7_SP1-p3389-RDP_ECO

Figure 66: IAPTF-Core components, functional diagram, and interaction mechanisms.

The Exploit Databases and attacking payloads and program import and fetching is twofold. The MSF Console imports the Exploits and the Payloads used for exploitation and post exploitation in addition to DB-Nmap NSE discovery scripts imported within the MSF POSTGRESQL. On the other hand, we developed the CVEs importing and fetching named Exploit_import.py which is a python program that oversees importing and restructuring CVE and NVD database along with creating a local mirror within the IAPTF-Memory. Key features of this module are:

- Storing and structuration format enabling CVE use and search by the IAPTF-Core directly basing on customized research criteria
- Enabling the usage of a lighter version of the large database both in term of number of CVEs and the description information stored within the original databases CVE and NVD. Only relevant information from PT point of view is kept in IAPTF-Memory.
- Direct interaction with Metasploit MSF console to enable real-time search narrowing and prioritisation based on NVD score as calculated per CVSS v3

6.8 IAPTF operative modes

PT involves complex and versatile activities which often come with automation challenges. To ensure that IAPTF is effective and able to achieve the sought aim of an optimised automated PT which is achieved by relying on AI techniques specifically RL and ES. IAPTF is designed and implemented to perform network PT under different contexts and operative modes notably as stand-alone, under-supervision or user-assistant and learning-only by capturing the knowledge from human expert. Therefore, IAPTF has four different operating modes with the selection of the adequate will depend on the context and the level of maturity (learning) as follow:

6.8.1 Stand Alone (Level 3):

this is the most advanced level of automation and optimisation that IAPTF can reach and constitute the aim of this research. In this mode IAPTF is fully intelligent and autonomous in performing the PT tasks after reaching the pre-set level of learning enabling it to act at least in the same way as a human expert Certified Ethical Hacker would do. In practice, the framework does not need to learn how to behave with all possible scenarios (which is technically impossible) but the assessed networks configuration will play a key role in the choice of the operating mode as previously completed testing with a degree of similarity will enable the IAPTF to use the acquired knowledge to achieve at worst an acceptable result. In this mode, the system will be responsible for auto-reward the RL agent actions by assigning automatically the relevant rewards following a pre-determined rewarding function, and updating and improving its decision policies when better attack paths are obtained.

6.8.2 Under supervision (Level 2):

this is a mode where IAPTF is partially autonomous which is in practice the standard mode for all first testing activities and early stage of commercial use. IAPTF in this mode is autonomous but remain under human CEH supervision for validation purposes. The system is then under constant and continuous supervision (locally or remotely) by a high-caliber (CEH at least) human pentester expert. The logic is that after a certain level of learning the framework will be enabled to take over the human expert to perform the penetration under supervision which is done for quality and consistency purposes by modifying the rewarding and evaluating mechanism. This will enable the human expert to be in charge of evaluating the accuracy, relevance and persistence of the testing by scrutinising the different attack vectors launched and evaluate the effectiveness and efficacy and allocating the relevant rewards. In this mode, human CEH will be only in charge of evaluating, which is done basing on his expertise, thus if IAPTF is performing above the threshold in each step (each action undertaken in a given state) the POMDP reward will be maintained. It is worth to mention that in many occasions, IAPTF will follow different attack paths which were never considered and/or explored by the expert which will be evaluated separately by the CEH to enable the framework to proceed further and carry on penetrating the network along with ensuring that the followed path does not lead to exceeding the maximum allocated resources. Upon completion, all the newly discovered path will be subject to a generalisation and stored in the IAPTF CLIPS expert system for future use.

6.8.3 Background assistant (Level 1):

in this mode IAPTF will be operating in the background of the manual human testing and only assist the expert in the decision making. The framework will capture the data and proceed to computing and processing, on a real-time basis, of the network variables and then suggest better alternatives if they exist and were missed by the expert. In the meanwhile, IAPTF will continue the learning by capturing the key information to be used in the POMDP representation optimisation and observing the CEH approach to extract the decision policy and assign the adequate rewards for any accepted suggestion it makes. The idea here is to accelerate the construction of the expert system knowledge which will at first stage help the ES to evaluate its decision policy before storing and introduce the relevant adaptation if required. Furthermore, the ES will store any decision alongside the associated context data in its memory after performing the Generalisation required for future use.

6.8.4 Early-stage (Level 0):

IAPTF is here in a learning only mode and will be using artificial data created from simulations. The idea here is to run IAPTF on as much cases as possible to allow the validation of the performances alongside the construction of internal knowledge base from repeating the test already completed by high caliber tester and extract the relevant data, policies and rewarding. In this situation, the framework can learn from the decision (actions and choices) undertaken by the expert during his work and extract the relevant knowledge from repeated test to build its memory.

Chapter 7: Testing, Results and Discussion

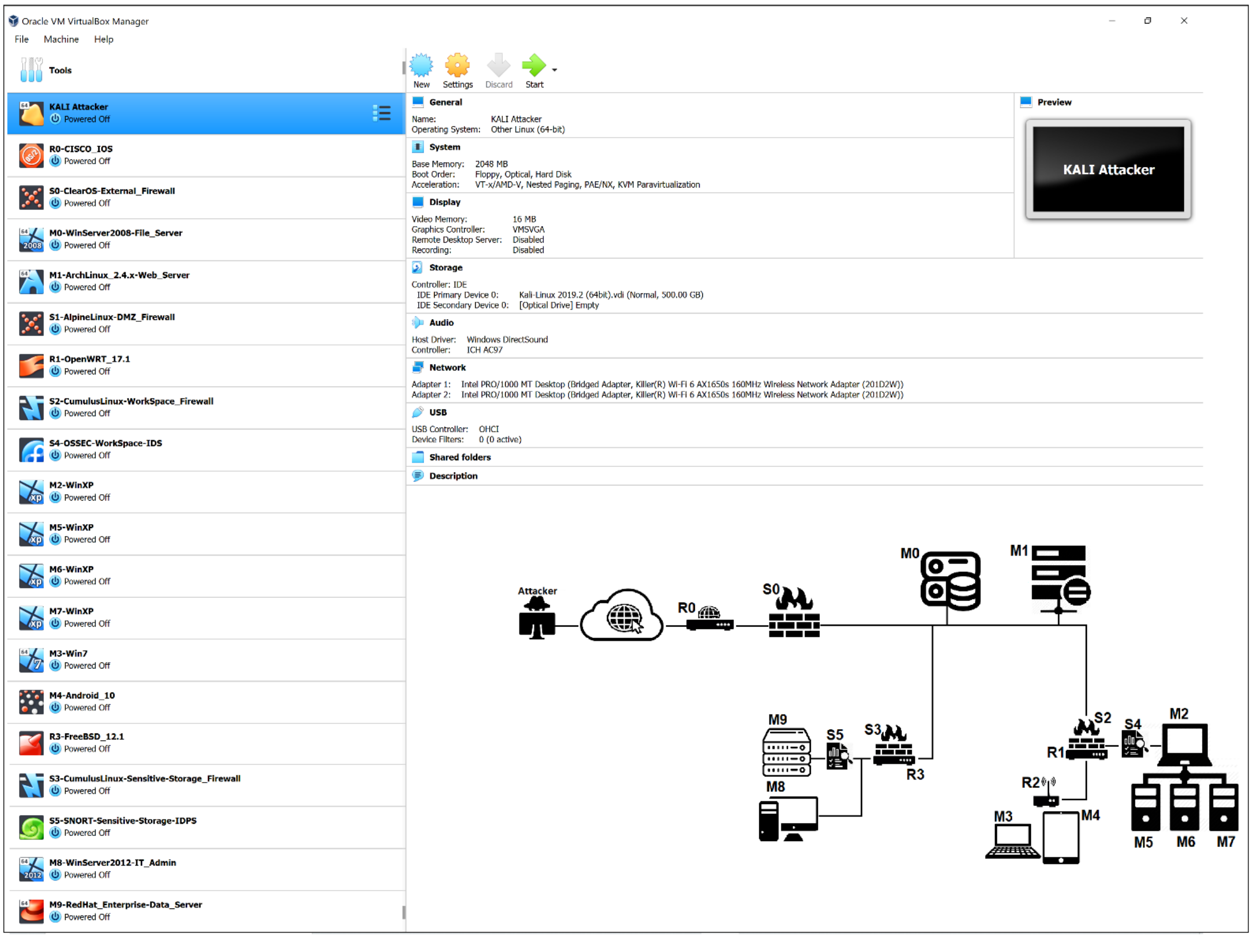
In this chapter, we will detail the different phases undertaken to develop test bed networks, test IAPTF and notably the RL module IAPTF-Core in different scenarios and conditions to validate RL modelling of PT practice. We also implement and test the new Hierarchical RL modelling for medium and large LANs following the security clustering approach and the functioning of the rule-based Expert System and Framework Memory in expertise capturing, generalization and replay

7.1 Setup of Experiments

The POMDP problem solving experiments were run on high-performance HP Z2 server with CPU Intel Xeon Processor E7-2176, 8 Core, 20MB Cache and 3.70GHz, and unbuffered RAM of 64GB DDR4 2666 DIMM ECC, Graphical NVIDIA Quadro P4000 8GB GFX. This machine runs Linux Calculate 20 kernel 5.4.6 64-bits which is a fast and resource-efficient Linux distribution based on Gentoo and maintains an optimal balance between state-of-the-art processing libraries and a renowned stability. The remaining IAPTF Modules, Script and Programs were hosted and run on a different machine DELL XPS 15 with CPU Intel Core i7- Processor 10750H, 6 Cores and 2.60GHz and unbuffered RAM of 16GB DDR4 dual-channel, Graphical NVIDIA GeForce GTX 1659Ti 4GB. This machine is running KALI Linux 2019.2 64-bits on Bare Metal mode which grant direct installation and access to the machine hardware to enhance performances.

7.2 Testbed Networks construction

In this section we provide a detailed explanation of the data capturing and reproduction of virtual LANs used for testing IAPTF. Building virtual LANs which can be used within the virtual context for simulating PT requires the use of realistic data, architecture and configurations which mimic the real networks that we cannot perform live testing due to existing restriction notably the GDPR regulation and Legal Framework in the UK and many other countries. Nonetheless, it is completely legal to capture the physical network data and recreate virtual clones with the same configuration, some functional sacrifices are compulsory as the virtual environments remain restricted thus some configurations will be imitated and not completely replicated. The starting point is the virtual machines (computers, servers, mobile, networking devices and security devices) which are either virtualized directly form the sources when possible or we use an existing pre-configured equivalent from online repository such as vulnhub.com [100] and darknet.org.uk [101] which provides VMware and VirtualBox materials allowing security practitioners to gain practical hands-on experience with digital security, computer applications and network administration on virtual machine environments. On few occasions, and this is relevant to servers and security devices cloning, we opted for the creation of the vulnerable virtual appliance from scratch using SecGen [102] which is a free utility to create lab environments for hacking challenges such as CTF. In total, for testing IAPTF we designed and implemented 53 VirtualBox LANs with different sizes varying from 2 machines to 250 machines. The used testbeds are, to the best of our knowledge, an illustration of the real-world networks widely adopted by corporates and organizations which include Internet-connected side, DMZ, intranet, and internal sensitive segments where crucial data is kept securely. The results of the virtualization phase on a 20-machines LAN constituted from 10 end-user machines (desktop, laptop, server or mobile), 4 networking devices (router or switch) and 6 security devices (firewall or IDS) is illustrated in figure 67.



KALI Attacker
Powered Off

R0-CISCO_IOS
Powered Off

S0-ClearOS-External_Firewall
Powered Off

M0-WinServer2008-File_Server
Powered Off

M1-ArchLinux_2.4.x-Web_Server
Powered Off

S1-AlpineLinux-DMZ_Firewall
Powered Off

R1-OpenWRT_17.1
Powered Off

S2-CumulusLinux-WorkSpace_Firewall
Powered Off

S4-OSSEC-WorkSpace-IDS
Powered Off

M2-WinXP
Powered Off

M5-WinXP
Powered Off

M6-WinXP
Powered Off

M7-WinXP
Powered Off

M3-Win7
Powered Off

M4-Android_10
Powered Off

R3-FreeBSD_12.1
Powered Off

S3-CumulusLinux-Sensitive-Storage_Firewall
Powered Off

S5-SNORT-Sensitive-Storage-IDPS
Powered Off

M8-WinServer2012-IT_Admin
Powered Off

M9-RedHat_Enterprise-Data_Server
Powered Off

General

Name: KALI Attacker
Operating System: Other Linux (64-bit)

System

Base Memory: 2048 MB
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, PAE/NX, KVM Paravirtualization

Display

Video Memory: 16 MB
Graphics Controller: VMSVGA
Remote Desktop Server: Disabled
Recording: Disabled

Storage

Controller: IDE
IDE Primary Device 0: Kali-Linux 2019.2 (64bit).vdi (Normal, 500.00 GB)
IDE Secondary Device 0: [Optical Drive] Empty

Audio

Host Driver: Windows DirectSound
Controller: ICH AC97

Network

Adapter 1: Intel PRO/1000 MT Desktop (Bridged Adapter, Killer(R) Wi-Fi 6 AX1650s 160MHz Wireless Network Adapter (201D2W))
Adapter 2: Intel PRO/1000 MT Desktop (Bridged Adapter, Killer(R) Wi-Fi 6 AX1650s 160MHz Wireless Network Adapter (201D2W))

USB

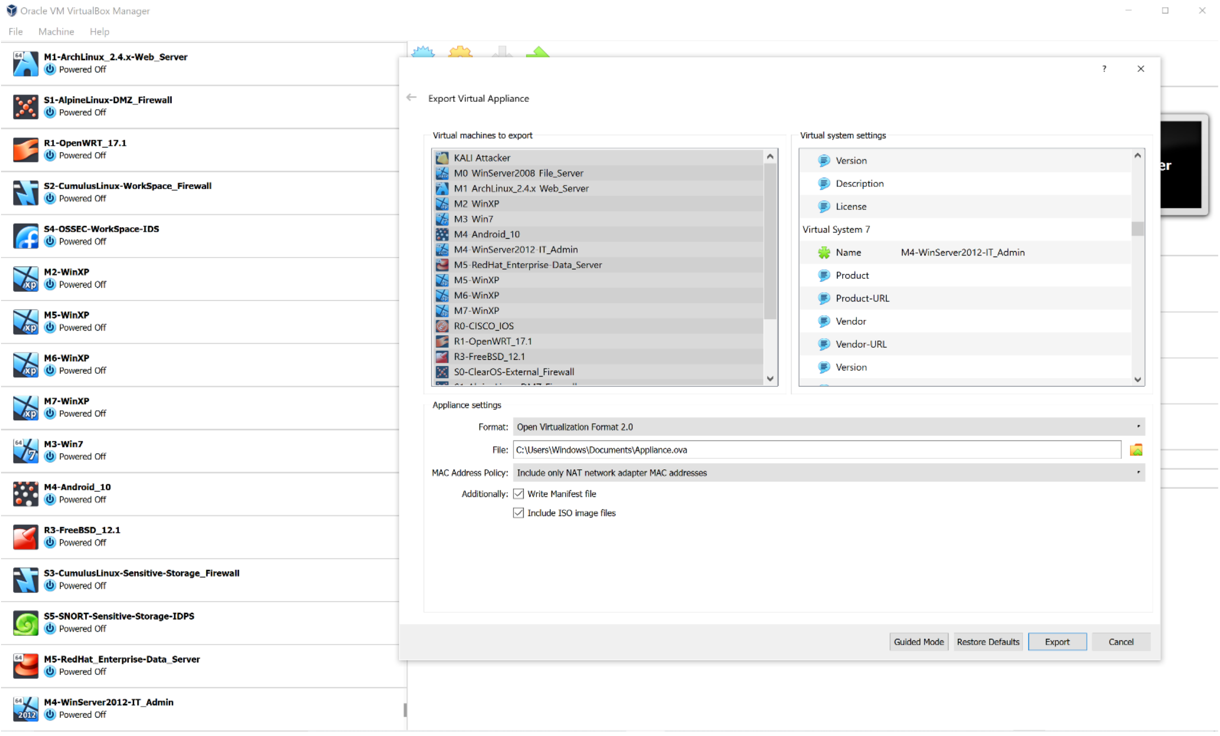
USB Controller: OHCI
Device Filters: 0 (0 active)

Shared folders

Description

Figure 67: Small corporate LAN of 20 machines replication on VirtualBox

The replicated LANs are then exported following open virtualization format 2.0 (*.ova file) and deployed into the testing server using the same virtual environment parameters. Note that not only machine data is exported in form of appliances, but we include networking and addressing data along the .ova files. Figure 68 illustrates the .ova file generation for the previous 20-machines LAN.



M1-ArchLinux_2.4.x-Web_Server
Powered Off
S1-AlpineLinux-DMZ_Firewall
Powered Off
R1-OpenWRT_17.1
Powered Off
S2-CumulusLinux-WorkSpace_Firewall
Powered Off
S4-OSSEC-WorkSpace-IDS
Powered Off
M2-WinXP
Powered Off
M5-WinXP
Powered Off
M6-WinXP
Powered Off
M7-WinXP
Powered Off
M3-Win7
Powered Off
M4-Android_10
Powered Off
R3-FreeBSD_12.1
Powered Off
S3-CumulusLinux-Sensitive-Storage_Firewall
Powered Off
S5-SNORT-Sensitive-Storage-IDPS
Powered Off
M5-RedHat_Enterprise-Data_Server
Powered Off
M4-WinServer2012-IT_Admin
Powered Off

Virtual machines to export

KALI Attacker
M0 WinServer2008 File_Server
M1 ArchLinux_2.4.x Web_Server
M2 WinXP
M3 Win7
M4-Android_10
M4-WinServer2012-IT_Admin
M5-RedHat_Enterprise-Data_Server
M5-WinXP
M6-WinXP
M7-WinXP
R0-CISCO_IOS
R1-OpenWRT_17.1
R3-FreeBSD_12.1
S0-ClearOS-External_Firewall

Virtual system settings

Version
Description
License
Virtual System 7
Name M4-WinServer2012-IT_Admin
Product
Product-URL
Vendor
Vendor-URL
Version

Appliance settings

Format: Open Virtualization Format 2.0
File: C:\Users\Windows\Documents\Appliance.ova
MAC Address Policy: Include only NAT network adapter MAC addresses
Additionally: ☑ Write Manifest file
☑ Include ISO image files

Figure 68: virtual LANs replicate exportation following open virtualization format.

7.3 Evaluating PT generated POMDP problems solving using SolvePOMDP

In the first phases of this research, we aimed to assess the effectiveness of the proposed POMDP modelling 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 Chapter 4. In this work, we tested IAPTF performance on different size experimental networks composed of variant number of machine and networking routers varying from 2 to 200 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 69 shows a sample large LAN with 100 machines. In this research we adopted a gradual simulating and testing approach to evaluate the test our proposed framework. We thus elaborated a multi-stage approach for validating the adequacy and relevance of our proposed RL model and POMDP representation. At early stage, we attempted to calibrate SolvePOMDP which is a Java program that solves POMDPs by executing value iteration and policy search algorithm to find both exact and approximate solutions for POMDP developed by Erwin Walraven in [22-24]. POMDPSolve is reputed to be flexible, reliable, and efficient and which we picked instead of reimplementing the solving algorithms. We run several tests to decide on the relevance if the proposed RL model and to calibrate SolvePOMDP using different solving approaches and three solving algorithms namely PERSEUS-LPSolve, GIP-LPSolve and the modified GIP-LPSolve_InitialBelief.

7.4 IAPTF optimal Discount Factor value

To determine which discount rate γ value (between 0 and 1) is best for our SolvePOMDP program which thus produces the best results and strike the balance time-efficiency vs results quality, we opted for a quantitative approach by attempting to solve POMDP environments resulting from PT representation as RL problems using different discount rate values between 0.8 and 0.99. As discussed in the RL chapter, the objective is to maximise the sum discounted reward using a discount factor $\gamma <=1$ as optimality criteria with different algorithm to find the optimal policy. The results (figures 70) clearly show that $\gamma=0.95$ (in amber) is the best value so solving algorithms PERSEUS and the two variants of GIP strike a balance between efficiency and accuracy of solving and producing the optimal policy graph. In the first tests, we run solving tests on different networks size varying from 2 to 200 machines and we attempted to manipulate the discount rate (factor). SolvePOMDP was run without time horizon limit until a target precision Epsilon equals 0.0001 is reached. We tested several options of discount rate varying from 0.8 to 0.99 and evaluate the suitability of each discount rate in accordance with the quality of the PG and the convergence time for the solving algorithm will briefly discuss below the effect of changing discount rate. figure 70 show the obtained results for PERSEUS, GIP-LPSolve and GIP-LPSolve_InitialBelief for 26 different LANs with size varying from 2 to 200 machines. The approximate solving with PERSEUS shows that 0.99 discount rate slows down considerably the solving algorithm especially on large LANs with number of machines more than 100. In fact, for the biggest LAN, PERSEUS solves the POMDP problem in approximately $4x10^5$ seconds compared with $2.75x10^5$, $2.4x10^5$ and $2x10^5$ with 0.95, 0.9 and 0.8 discount rates respectively as shown in figure 69.

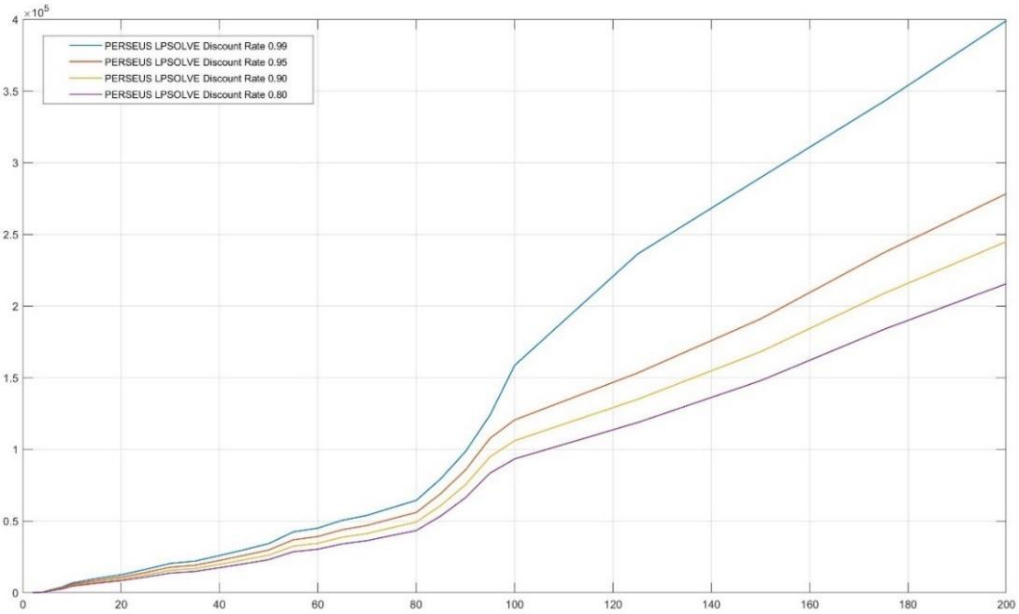


Figure 69: Solving different LANs POMDPs using PERSEUS-LPSolve with Initial Belief and variable discount rates.

The impact of smaller discount rate is more noticeable in exact solving methods such as GIP-LPSolve where in 200-machine networks the required solving time is 2.4x10^{6} for 0.99 compared with 1.7x10^{6}, 1.65x10^{6} and 1.55x10^{6} with 0.95, 0.9 and 0.8 discount rates respectively as shown in figure 70.

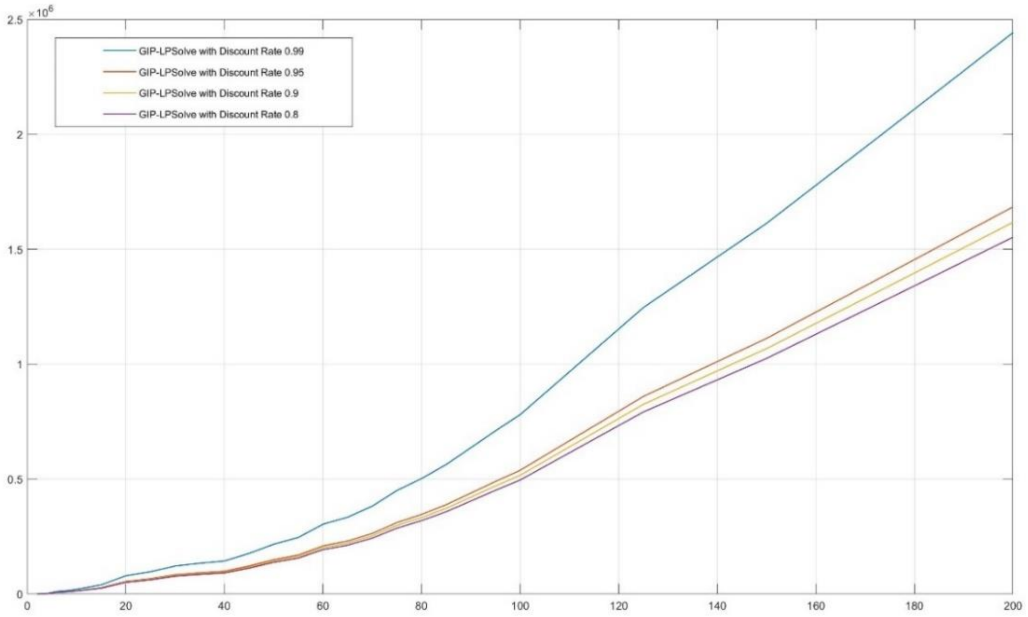


Figure 70: Solving different LANs size associated POMDPs using GIP-LPSolve and variable discount rates

Furthermore, with the projection of heavily using the modified GIP version on which the Initial Belief is manipulated and fed directly to the solving algorithm through a Java parser, we performed the same testing to confirm that the selection of the discount rate will not impact the performances. The obtained results show that solving with GIP-LPSolve_InitialBelief using the 0.99 discount rate slows down considerably the solving algorithm especially on large LANs as the 200-machine LAN POMDP problem is solved in approximately 17.25x10^5 seconds compared with 9.75x10^5, 9x10^5 and 8.25x10^5 with 0.95, 0.9 and 0.8 discount rates respectively as shown in figure 71. Full results are presented in Appendix 1 Discount Rate table.

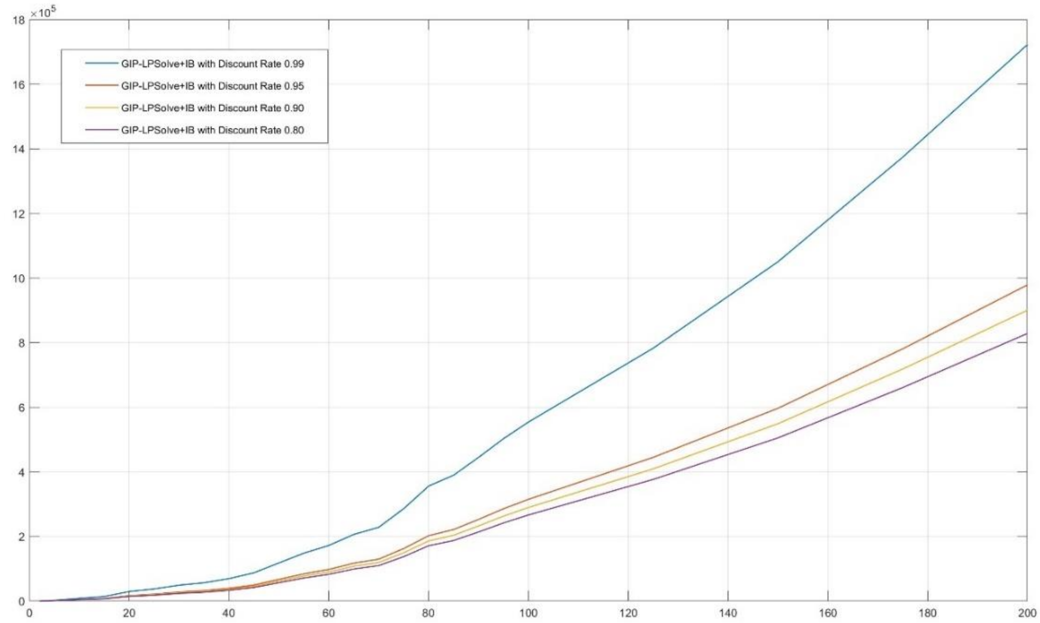


Figure 71: Solving different LANs size associated POMDPs using GIP-LPSolve with Initial Belief and variable discount rates

The obtained results illustrated in figures 72 show the deep impact, for the three algorithms of the variant discount factor value γ=0.99 on slowing down the solving performances and thus increasing consumed time. Where such discount rate remains strongly recommended in small network (typically smaller than 20 machines), it is therefore avoided in medium and large LANs. We noticed that γ=0.95 is the only candidate that meets both requirements for time efficiency and for policy graph quality in term of exploration and relevance. γ=0.95 and γ=0.99 discount rates produced nearly the same Policy Graph in testing scenarios where the network size is more than 100 machines. On the other hand, and as illustrated in figure 72 the time consumed for γ=0.9 and γ=0.8 is not significantly different from the γ=0.95 and this leads us to conclude that not enough solving time is saved for these two values compared with the decreasing in the policies quality which is noticed at the number of vectors in the raw PG and consequently the number of the valid attack vector that our framework will attempt if such values are adopted. To sum up, the discount rate testing enable us to conclude with confidence that γ=0.95 is the perfect balance between time efficiency and solving accuracy. To conclude this section, the obtained results with all three algorithms and for different testbed LANs back the election of a discount rate (factor) equal to 0.95 which is the balance between improving performance by in one hand requiring shorter solving (convergence) time by 1/3 compared with a discount rate of 0.99, and in another hand allowing enough exploration and policies improvement to reach a quality output PGs compared with 0.8 and 0.9 discount rates.

7.5 IAPTF Testing variable and evaluation metrics

In this section, we test and evaluate the performance of IAPTF-Core module on generated virtual LANs, the IAPTF-Processing output POMDP environments as result of the POMDP-generator Python program. This testing will be based in metrics we extract from IAPTF-Processing output as follow:

- Number of machines N in the tested network,
- Number of identified vulnerabilities **V**,
- Number of available exploits **E**,
- Number of security clusters **C** (in HRL context),
- Number of unchanged machines **I** (in re-testing context).

In re-testing, IAPTF-Processing compares the configuration of each machine in reference to the old configuration stored in IAPTF-Memory and determines the number of unchanged machine configuration **I** since the last testing. On the other hand, the expertise extraction and reusage are also considered during the retesting comparison. Furthermore, the evaluation metrics adopted in this research are:

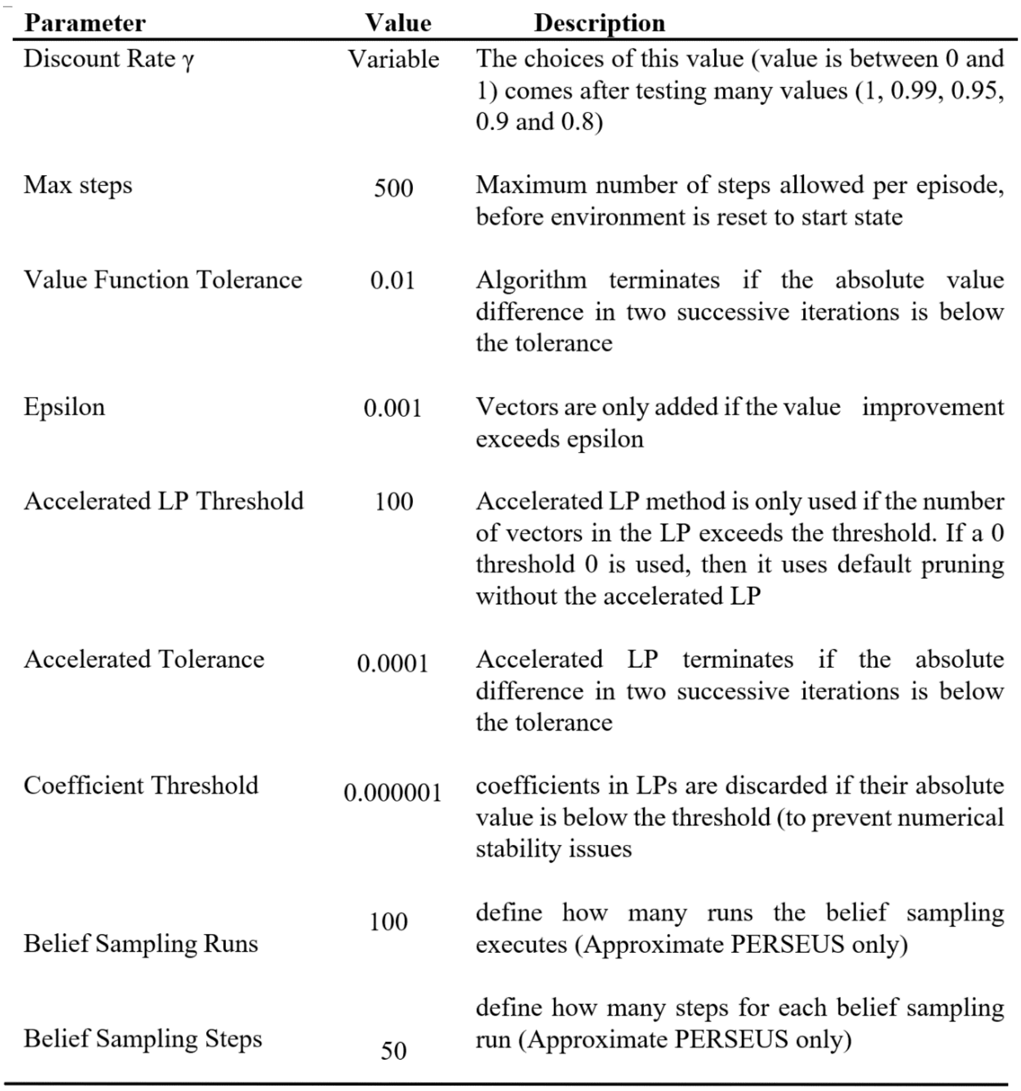
- Total testing time including RL algorithm solving, pre-processing and post-processing
- Number of attack vectors attempted which we use to estimate the network traffic generated
- Exploration and coverage measured by the number of valid attack vectors attempted.

For consistency purposes, we opted to introduce an additional quantitative value to be used in comparing different approach on the top of time consumed and testing coverage of the proposed IAPTF with both human expert (CEH) and industrial automated solution (MSF Pro). The proposed ratio is set to evaluate the exploitation of IAPTF in comparison with both human expert (CEH) and industrial automation (MSF Pro). We introduce the notion Exploitation Ratio which is calculated for CEH, IAPTF and blind automated MSF as follow:

$$Exploitation\ Ration = \frac{Network\ size\ (N) * Valid\ Attack\ vectors}{Congestion\ Factor\ (\varDelta) * \sqrt{number\ of\ Vuls\ (V) * number\ of\ Exploits\ (E)}}$$

With the number of Valid Attack vectors being measured in both IAPTF and MSF Pro but only estimated in CEH context basing on the author experience. We define Congestion factor as the impact of PT on the assessed network and will equal to 1 for human-led testing as CEH will only launch relevant attacks, while the Congestion Factor *Δ* defined on 2 to 5 scale for IAPTF and MSF Pro with 5 allocated to high number of attacks compared with network size. For each scenario we tried to keep as many settings as consistent as possible, to better elucidate the effect of the variable of interest. The table 6 provides details of the different parameter values used. The max steps value chosen as this is intended to produce good performance across a range of scenario sizes during preliminary testing. Similarly for the values of sensitive machines and action costs.

Table 10: SolvePOMDP Experiments parameters and their values.



Parameter	**Value**	**Description**
Discount Rate γ	Variable	The choices of this value (value is between 0 and 1) comes after testing many values (1, 0.99, 0.95, 0.9 and 0.8)
Max steps	500	Maximum number of steps allowed per episode, before environment is reset to start state
Value Function Tolerance	0.01	Algorithm terminates if the absolute value difference in two successive iterations is below the tolerance
Epsilon	0.001	Vectors are only added if the value improvement exceeds epsilon
Accelerated LP Threshold	100	Accelerated LP method is only used if the number of vectors in the LP exceeds the threshold. If a 0 threshold 0 is used, then it uses default pruning without the accelerated LP
Accelerated Tolerance	0.0001	Accelerated LP terminates if the absolute difference in two successive iterations is below the tolerance
Coefficient Threshold	0.000001	coefficients in LPs are discarded if their absolute value is below the threshold (to prevent numerical stability issues
Belief Sampling Runs	100	define how many runs the belief sampling executes (Approximate PERSEUS only)
Belief Sampling Steps	50	define how many steps for each belief sampling run (Approximate PERSEUS only)

The IAPTF-Core testing was rigorous and extensive, being aware of the volatility in solving performances of POMDP and notably when using approximate solving or when relying on linear programming (LP) and the associated complexity of finding optimal policies for POMDPs. This later are then solvable in polynomial time which is much efficient and enable huge solving-time reduction compared with PSPACE problem [125-128]. The testing was repeated several times for each network and calculated the mean value and standard deviation for each test scenario. Given the time constraints, we opted for a threefold testing approach. We repeated the testing 10 times for small LANs (2-25 machines) and 5 times for medium LANs (30-95 machines) and only 3 times for large LANs (100-200 machines). The reason of reducing the number of testes in large LANs is the time required to solve POMDPs which exceeds 468.05 hours (19.5 days) for exact solving of 200-machines POMDPs. The IAPTF-Core was gradually tested by initially using approximate solving to validate the RL modeling and POMDP representation, then the exact solving was applied in two phases. First the regular RL modelling where the entire network was considered at once. Secondly the proposed hierarchical RL model where networks are divided into security cluster and associated POMDPs are built then solved as described in Chapter 6. The obtained results illustrate the complexity of solving medium and large PT problems notably where the aim is reaching the optimal policy. The PERSEUS approximate solving algorithm is confirmed to be very efficient with all network size but remain unreliable notably in the quality of the output PG and the testing coverage (exploration).

As anticipated, the exact solving is time consuming and despite the use of the linear programming with GIP named GIP-LPSolve the solving time is significantly high and largely exceed the acceptable threshold for PT practice in all medium and large LANs despite the use of LPs. Upon the obtained results, we decided to introduce some changes within the solving algorithm GIP aiming a better performance from IAPTF on a short-term basis. We opted for prioritised transitions and observations through the manipulation of the associated probabilities along with introducing some customisation into the initial beliefs sampling. In detail, the initial belief is now extracted directly from the output of previous testing which reflect the better the real-world situation in PT where tester rely on reports of previous testing to elaborate a full awareness of the assessed network. Indeed, the networks changes over time as result of hardware/software upgrade and update, security architecture changes, configurations, and architecture changes notably by adding new machines and removing others.

Nevertheless, the vast majority of machines configuration and defence remain unchanged over short period (weeks to months). The IAPTF exploits this crucial feature in the quest of performance enhancement in solving medium and large size POMDP by opting-out the heavyweight GIP initial belief sampling and simply use python scripts to process last testing output and elaborate a complete initial belief data that is directly injected into the POMDP environment and thus processed by GIP. This new variant of GIP is named here GIP-LPSolve with Initial Belief. The modified GIP-LPSolve with Initial Belief produce a huge improvement (35 to 50%) in GIP performances which is due to the reduction of the observation space and the narrowing of RL exploration area, this comes of course with a major drawback which constitute the construction of the initial belief input either manually or with a customized script.

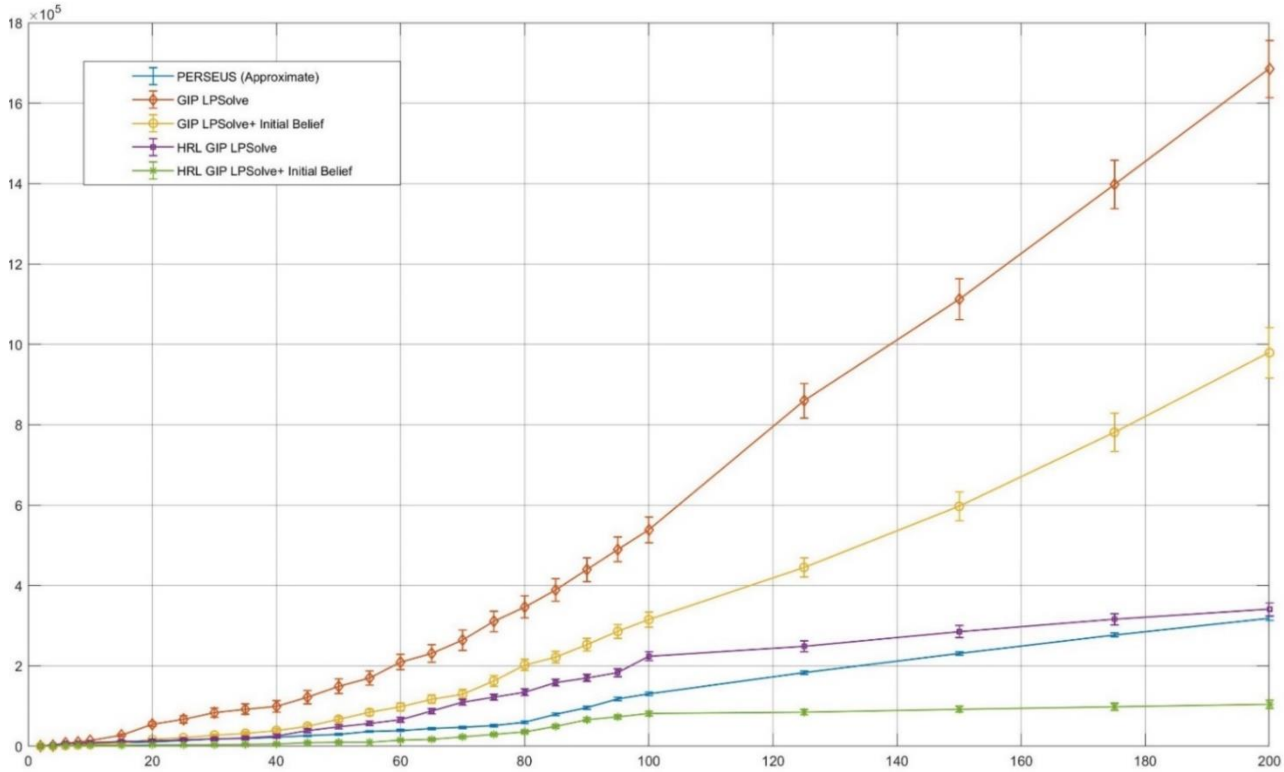


Figure 72: Solving variant size LANs associated POMDP time consumed and standard deviation when using different algorithms.

The performance enhancement in large LANs with size exceeding 100-machine (determined threshold) is clear when adopting the hierarchical modelling with the obvious security clustering function reaching the best performance in large networks by dividing networks into an optimal size of 7 to 10 machines per cluster and thus enabling fast solving of POMDPs of the dozens of clusters and the POMDP of HoCs. Finally, given the encountered difficulties when solving large network POMDPs and especially when the GIP is used, we introduced the HRL modelling which split the large POMDP problem into sub-problems following a security clustering approach and solve each of the cluster POMDP separately before solving the POMDP constructed from the Heads of Clusters only. As discussed in Chapter 6, this approach is very close to the human hacker operating mode by considering different LANs machines based on their cyber exposure, attack surface and security protection regardless of their sub-network. The HRL involve solving a number of small POMDPs equals to C+1 where C is the number of security cluster, rather than solving large POMDP which is time consuming. The obtained results were better than expected in term of efficiency and performance enhancement. HRL GIP-LPSolve performed closely to the fast approximate solving PERSEUS and especially in large LANs with a projected equality in 200+ machines networks. On the other hand, HRL GIP-LPSolve with Initial Belief outperformed the approximate solving and all other algorithms and perform particularly well in large networks (100+ machines) with solving time equal to 40% of the approximate PERSEUS and 10% of the time requires for GIP-LPSolve with Initial Belief. In small LANs, some loss of performance was noticed in HRL-GIP-LPSolve due to the extra-processing added for defining a hierarchical PODMP problems for each security cluster and HoC POMDP. Since we use value approximation at the higher nodes of the hierarchy to represent the value of selecting the various specialized lower-level POMDPs. Further testing showed that this loss of performance is minimal in very small LANs where the use of hierarchy is unjustified due to irrelevance on dividing a small POMDP into two or more smaller POMDP causing performance loss because of the extra computing required for different HRL functions and programs. It is worth to mention that as we calculated the complete testing time required for IAPTF to complete full PT exercise on different LANs which include the POMDPs HRL-GIP-LPSolve time, pre-processing and post-processing, we compare the overall result with both human expert CEH approximate consumed time along with a calculated time from Metasploit which is a python-based automation of MSF framework integrated with Armitage and Nessus .

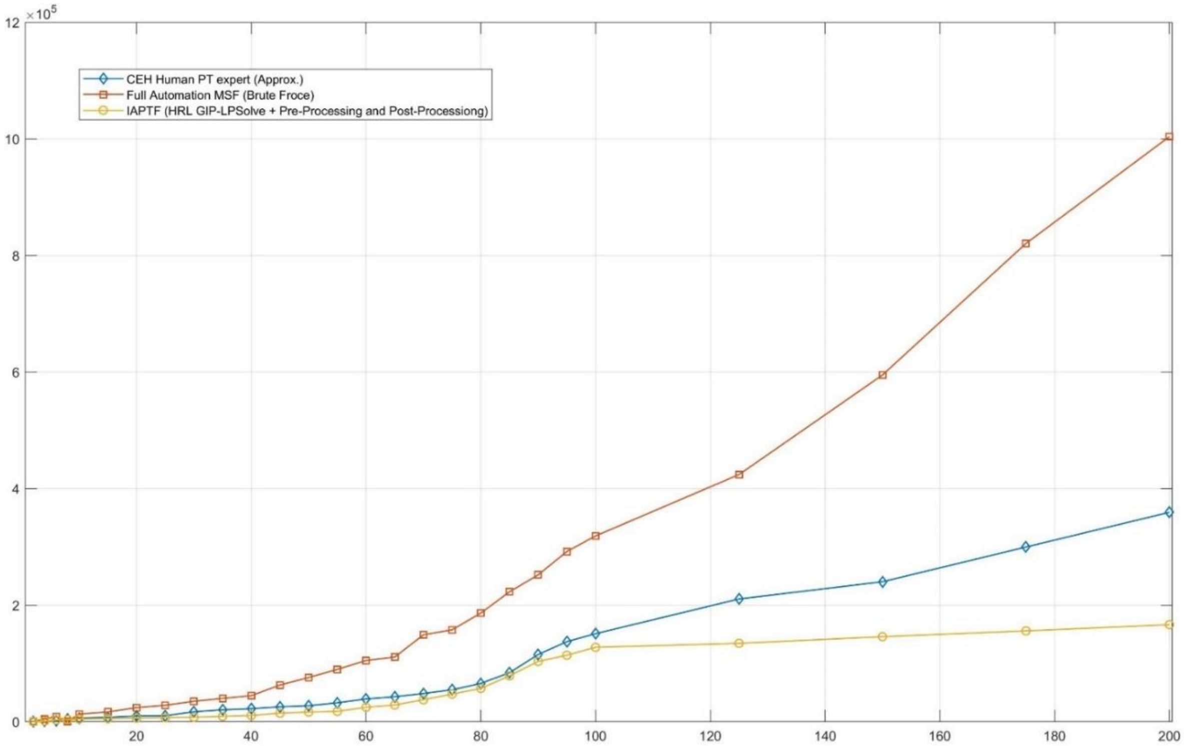


Figure 73: Time-efficiency comparison of IAPTF (HRL), CEH and fully automated MSF.

The figure 73 illustrates the comparison in term of consumed time and shows that IAPTF outperforms both CEH with the gap widen in large LANs. The efficiency of IAPTF compared with the blind automation is striking with IAPTF requiring only 17% of the MSF consumed time to complete 200-machine LANs testing.

7.6 Exploration- PT coverage

The second metric for testing the RL model of PT practice and the proposed IAPTF is by measuring and comparing the exploration and exploitation capabilities. In fact, the POMDP solving time is not enough to be considered as deterministic metric for evaluation of the proposed RL model and implementation and notably the quality of the solving which remain the key factor in PT practice. As with human expert, RL-led PT practice is performed following software RL agent decision-making which involves fundamental choices and results in exploring the POMDP environment, This test is designed to measure the exploration of IAPTF-Core in terms of the number of covered attack vectors.

The number of attack paths is defined as the number of full paths concatenated (re-ordered and gathered) Policy Graphs (PGs) resulting from the solving. In fact, the PGs are concatenated to form a complete attack path where the starting point is the hacker machine, or any machine controlled at admin/root level and thus used as pivot point for lateral movement and the end is a machine state either secure (fail exploit) or successful exploitation and privilege escalation. This reconstruction is done automatically using a python script and does not account for the feasibility and the relevance of the attack path in question. Figure 74 shows the total number of attempted attack vectors obtained with an exact solving of the POMDPs for different size LANs reconstructed from the output PGs of IAPTF-Core. Figure 74 presents a comparative illustration of number of attacks attempted by the four solving algorithms in different size LANs.

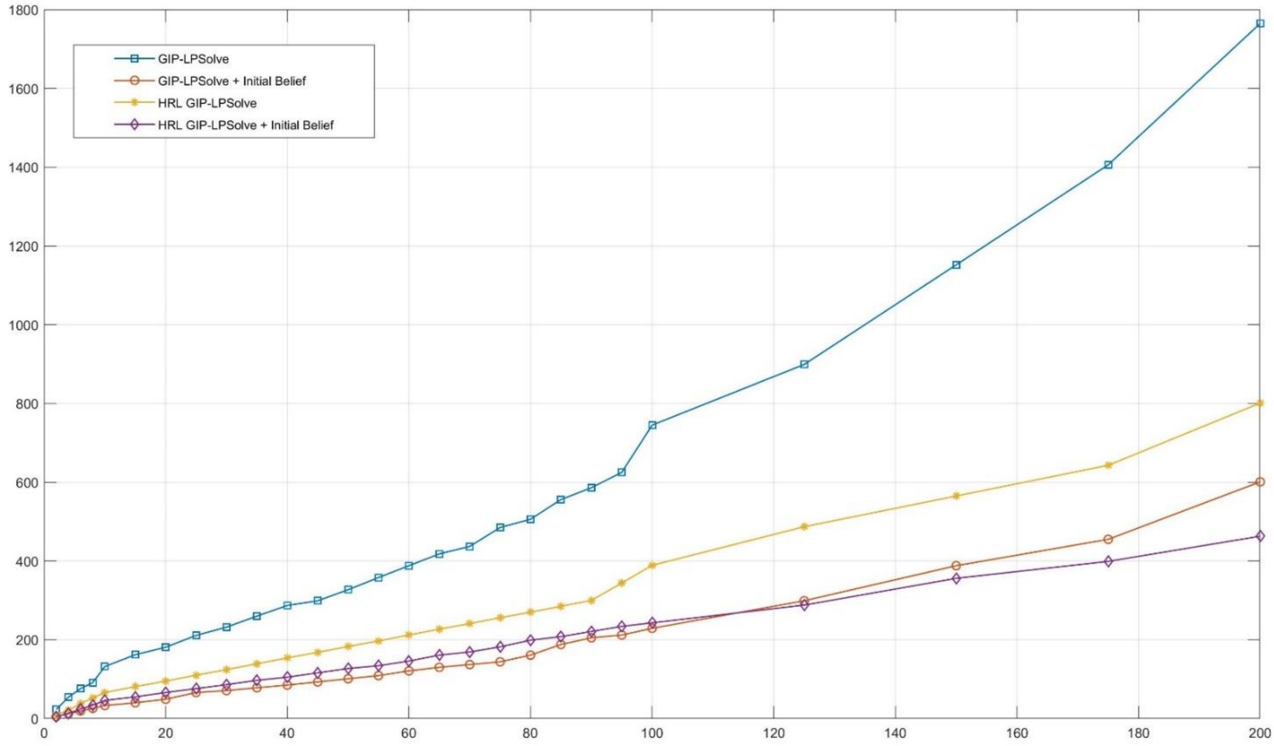


Figure 74: Number of attack vectors reconstituted from PGs covered by each if the solving algorithms in different size LANs.

From the obtained results, it is obvious that the classic implementation of GIP-LPSolve covers more attack vectors despite being more time-consuming. The output policy graph (PG) in GIP-LPSolve is larger as this solving algorithm perform more action and observe more states with the aim to compute the best possible policy starting without prior knowledge and pre-eliminated testing directions. HRL-GIP-LPSolve output approximately half the number of attack paths which is explained by the fact that the hierarchical RL algorithm will solve a number of small POMDPs equal to the number of security clusters plus the head of clusters (HoC) POMDP (C+1) resulting into RL performing a local exploration and thus covering less attack vectors. On the other hand, both customized initial-belief variant of GIP (RL and HRL) produced approximately the output a third of GIP-LPSolve. These two implementations rely on the use of last testing output as customized Initial Belief to feed the solving algorithm pre-eliminate many paths and push the RL agent to ignore certain paths which reduces the solving time but also reduces the number of attempted attacks by relying on the fully observable state and thus reducing the number of attacks possibilities which will operate as attack pre-elimination mechanisms.

The total number of attack vectors is important to measure the exploration but insufficient. We here highlight the fact that PGs size and number of attack vectors attempted, known as attack coverage rate, is subjective and often misleading as not all attack vectors are valid and complete. The program introduced in IAPTF is a set of python scripts that examine all attack vector and eliminate non-relevant, repetitive, cyclic and unworthy attack victors. The results for the four algorithms is unexpected as the gap between GIP-LPSolve and other three candidate because marginal which validate again the HRL assumption made and the robustness of the proposed model. We introduced, for comparison purposes, the average number of attack vector covered by CEH in figure 75.

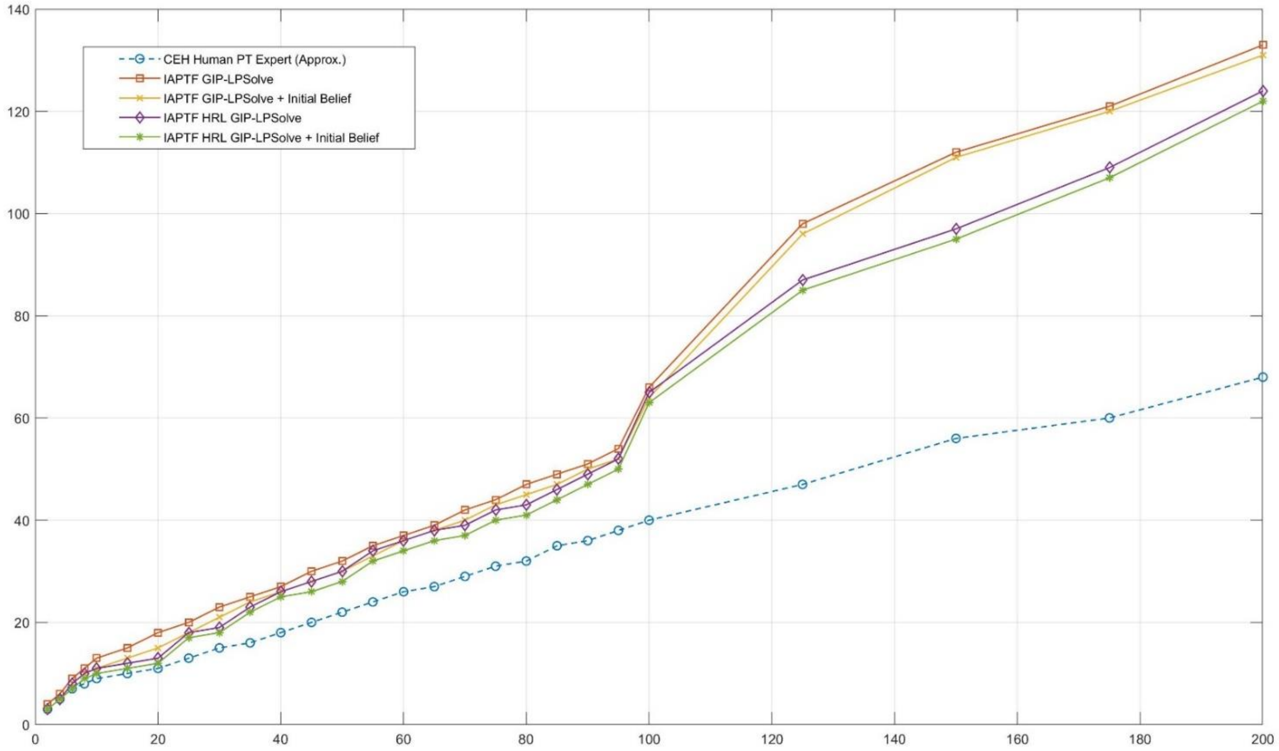


Figure 75: Comparison of total number of valid attack vectors for the four algorithms CEH.

It is obvious that on the top of the overall performance enhancement and notably, GIP LPSolve with initial belief and HRL- GIP-LPSolve algorithms produce a similar quality of the produced decision policies which remain was beyond human expertise especially in the case of the medium and large LANs when the size and complexity impact human CEH and therefore omit a significant number of composite (non-obvious) attacking vectors s. On the other hand, IAPTF explores more attack vectors and attempt higher number of composite attacks resulting in discovering additional attack vectors which most of CEH would ignore.

Finally, we elaborated an illustrative graph for comparing both time efficiency and valid attack vector coverage for IAPTF using HRL- GIP-LPSolve with Initial belief as it is the most efficient implementation in term of time and valid attack vector along with the estimated values for human expert CEH and a calculated values for MSF fully automated testing illustrated in figure 76.

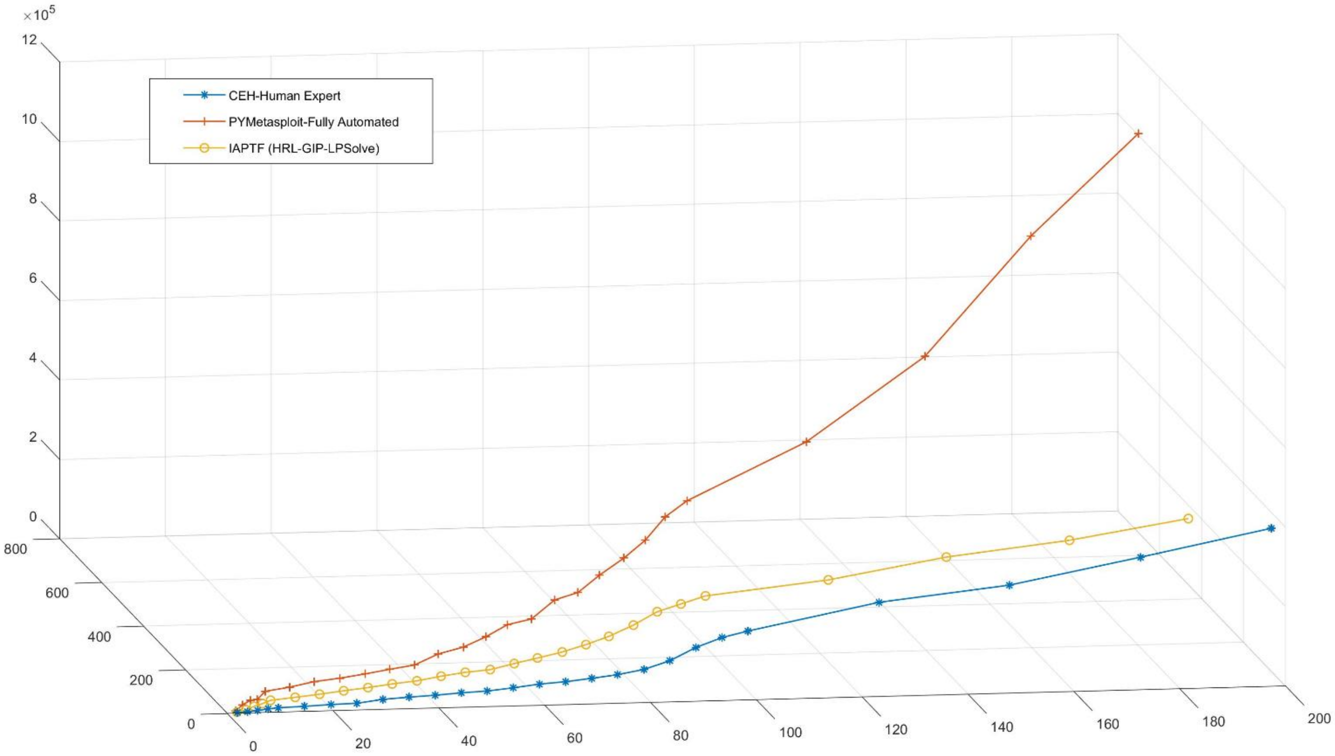


Figure 76: 3-D comparison of IAPTF, CEH and MSF performances in term of time and valid attack vectors covered.

It is obvious that IAPTF is the fastest PT approach and second best explorative behind the blind automation (brute-force) which remain far more time costly. IAPTF is faster and more explorative than the costly human CEH.

7.7 Exploitation-Testing Validation

In PT practice, success in not binary or measured by successful attacks and exploitation as a failed attack of attempted properly is a success in term of confirming that an asset is secure against the attack and will be reported accordingly for compliance purposes. In addition, notion such as false positive and false negative are not possible to measure and explain in PT practice. We thus introduced a new metric named exploitation ratio which is synonym of testing validation and represent the cumulative number of successful of failed valid attacks attempted and thus to be reported at the end of testing. We calculated this ratio for IAPTF, CEH and MSF and reported obtained results in figure 77.

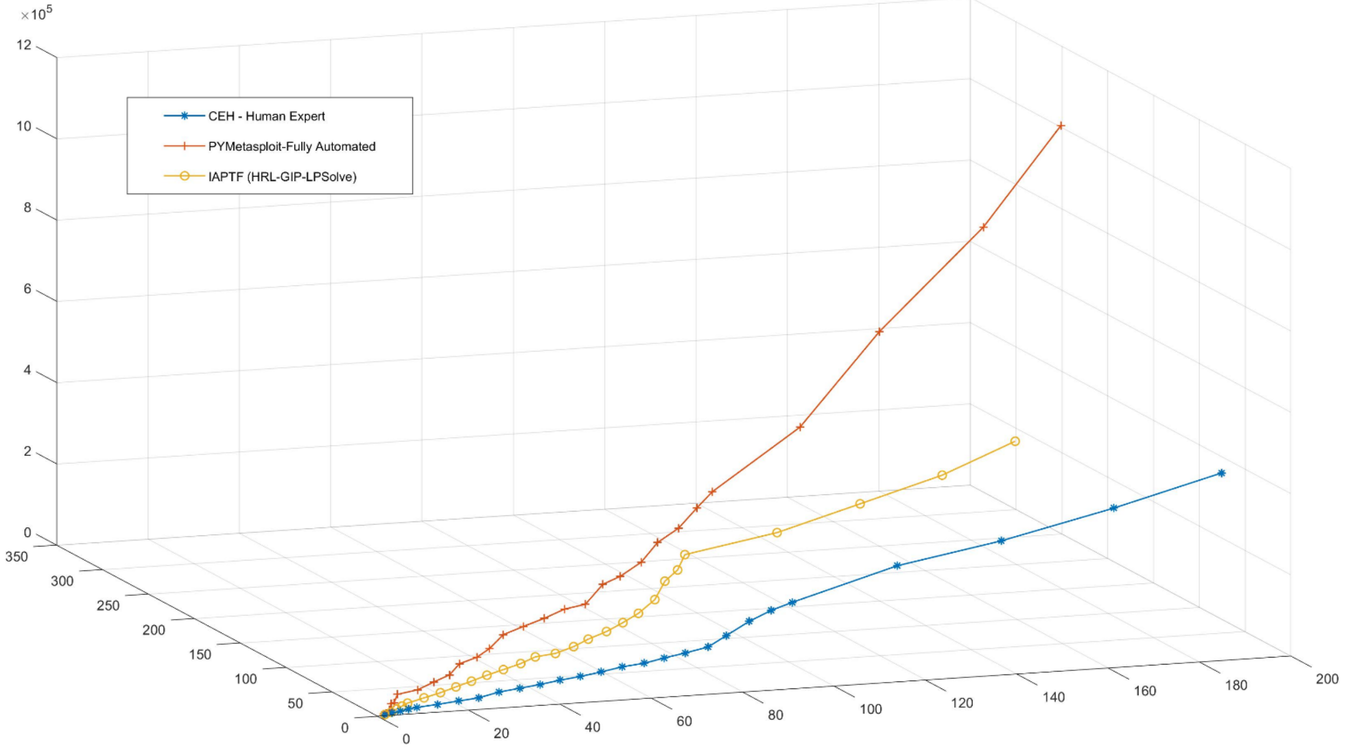


Figure 77: 3-D comparison of IAPTF, CEH and MSF performances in term of time and exploitation ratio.

The obtained results have once again backed IAPTF generally and the HRL modelling of PT specifically. In this occasion IAPTF outperforms both the lengthy blind automation (MSF) and the unreliable CEH where despite being time efficient remain costly and report less valid tests.

7.8 PT generated traffic

PT efficiency is often assessed by measuring the disturbance (network downtime) caused during its execution in term of network congestion or eventually downtime. This is measured by the amount of generated reconnaissance, discovery, exploitation, and post-exploitation traffic. Furthermore, the amount of generated traffic is usually considered proportionate to the risk of detection as most of network security mechanisms and especially Intrusion Detection Systems rely on the abnormal traffic monitoring for attack detection. In this research, we estimated the amount generated traffic by calculated the number of testing sub-tasks performed by IAPTF, CEH and MSF and we represented in figure 78 the obtained results in parallel with the time required for completing different LANs size testing.

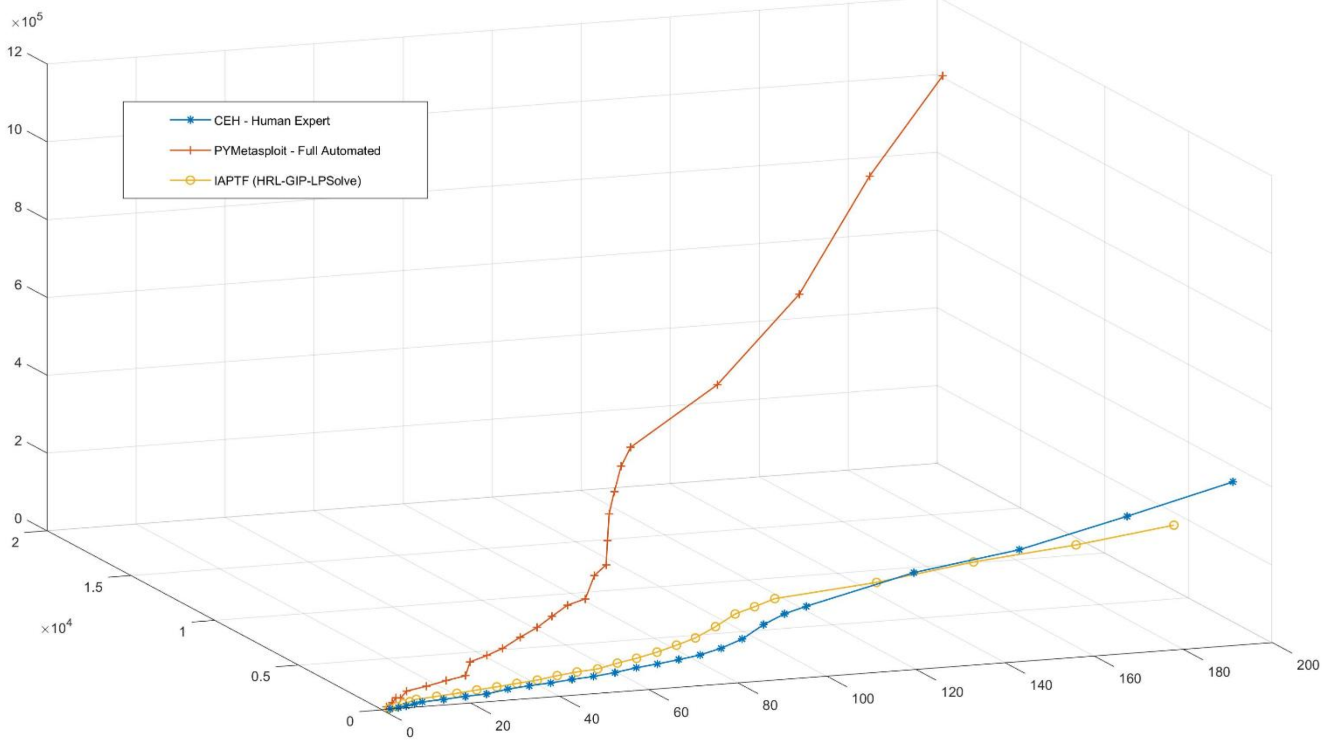


Figure 78: 3-D comparison of IAPTF, CEH and MSF performances in term of generated network traffic.

Obtained results illustrated in figure 78 strengthen the RL-led PT position as IAPTF remains far less noisy in term of generated network traffic than the blind automated MSF which is completely expected as it attempts fewer attacking vectors. Nonetheless, CEH remain stealthier as human expert is particularly keen on not trigger security mechanism in the network but this approach is less reliable (effective) because of the small amount of valid tests produced and also lower testing coverage.

7.9 IAPTF- Expertise extraction and Retesting

As we discussed in Chapter 2 and specifically the research contribution, the proposed RL model of IAPTF is introduced and implemented to tackle two major issues in the PT practice, having presented in previous sections tested and validated the RL and HRL modelling of PT practice and proved the performance enhancement of such AI-led PT in term of testing time, testing exploration, testing exploitation, and reducing network congestion. Furthermore, the use of RL enabled the implementation of IAPTF expertise extraction, generalisation, and re-use mechanisms which we tested to assess their impact on performance enhancement. We proceeded to re-test the same network by introducing gradual changes to certain number of machine configurations mimicking the real-world situation where networks change over time as result of hardware/software upgrade and update, security architecture changes, configurations, and architecture changes notably by adding new machines and removing others. The testing was carried out in two phases. First the regular RL implementation of GIP-LPSolve with customised Initial Belief and then in a later stage the introduced HRL-GIP-LPSolve with customised Initial Belief, The tests used the output of previous testing (performed before introducing the changes into the LANs) and measured the time required for solving. The IAPTF utilise this crucial feature in the quest of performance enhancement in solving medium and large size POMDP by opting out of the heavyweight GIP initial belief sampling and simply use python scripts to process last testing output and elaborate a complete initial belief data that is directly injected into the POMDP environment and thus processed by GIP. This new variant of GIP is named here GIP-LPSolve with Initial Belief.

The obtained results were surprisingly better than expected in term of performance enhancement. GIP-LPSolve with Initial Belief performed much better than the classic GIP in terms of consumed time and policy graph (PG) accuracy were calculated by dividing the number of attack vectors by the total number of policy graph. The performances are unanimous and showed in Figure 79. Furthermore, to assess the contribution of IAPTF expertise extraction, generalisation and reuse and its impact on 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 100- to 200-machine LANs were extremely encouraging and nearly halved the consumed time in solving as shown in Figure 79.

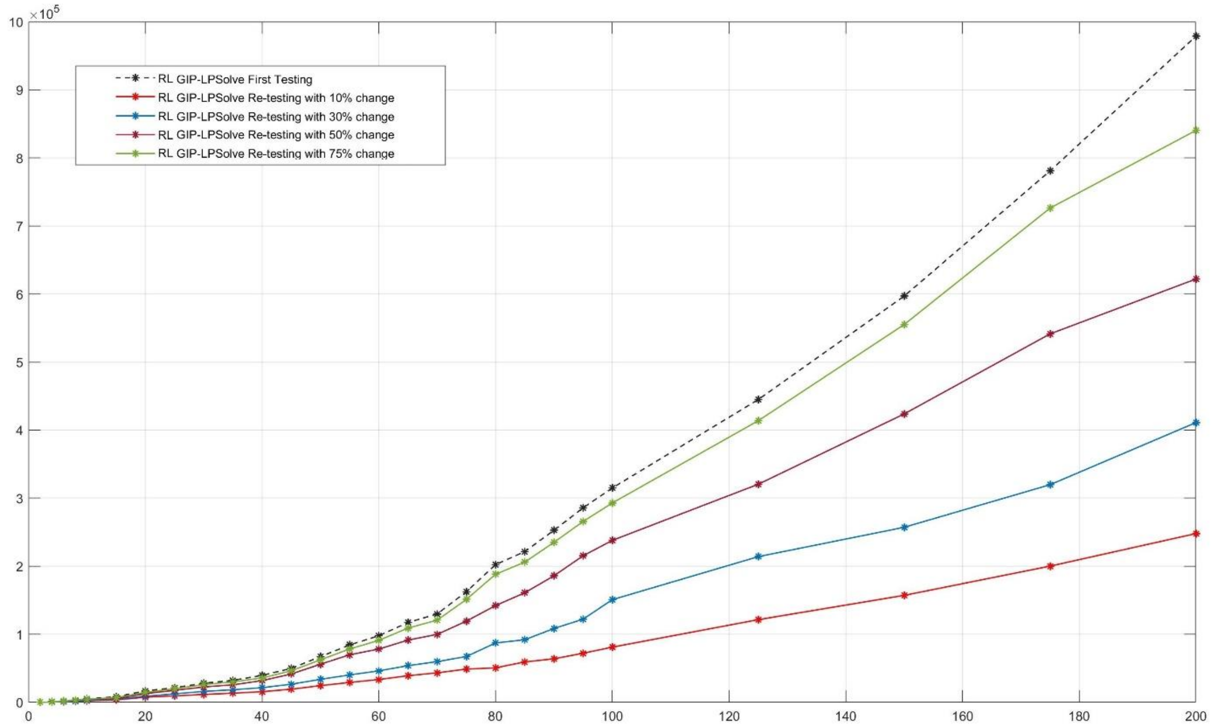


Figure 79: Re-testing the same network with introducing a percentage of change using RL-GIP-LPSolve.

The obtained results were unanimous. The performance enhancement in small LANs were marginal but started to increase in medium LANs to reach very efficient results in 100-machine LANs. The large LANs context is further better, and results were extremely encouraging as shown in Figure 79. Overall, retesting RL-GIP-LPSolve produced very good results in small and medium LANs but remained relatively high in large LANS. The results show some loss of performance for HRL (compared with regular RL for very small networks) with number of machines up to 10 machines (4 clusters, 33 vulnerabilities, 24 exploits). This issue is completely justified by the fact that clustering and cluster processing is useless and only slow down IAPTF. In small networks, security clustering produce often a big number of security clusters and thus many very small (2-3 machines) POMDPs on the top of the POMDP representing the Heads of clusters. This will result in forcing IAPTF in executing a big amount of data manipulation and POMDPs' solving which are in fact unnecessary. The fact that Regular RL solving of entire POMDP is faster. However, HRL-GIP effect is largely appreciated in larger networks and reach a very good rate in 100-machine network (25 clusters, 102 vulnerabilities, 80 exploits). HRL approach requires 224087.118 ± 12564.7 (2.6 days) compared with

538318.624± 31964.2 (6.2 days) in regular RL-GIP. Going beyond the 100-machine size, HRL is at least 4 times more efficient and reaching 200-machine size (52 clusters, 153 vulnerabilities, 115 exploits), HRL-GIP performed almost as well as approximate PERSEUS and required 340582.592 ±16297.8 (3.9 days) compared with 1685011.539± 71160.5 (19.5 days) for RL-GIP and 278369.056 ± 5236.9 (3.2 days). When we repeat the tests using the output of previous testing as initial belief (after processing), GIP-HRL surpasses PERSEUS performance and only required 1.2 days compared with 3.2 for approximate. We then tested the HRL-GIP-LPSolve with different LANs and changes percentages, as shown in figure 80.

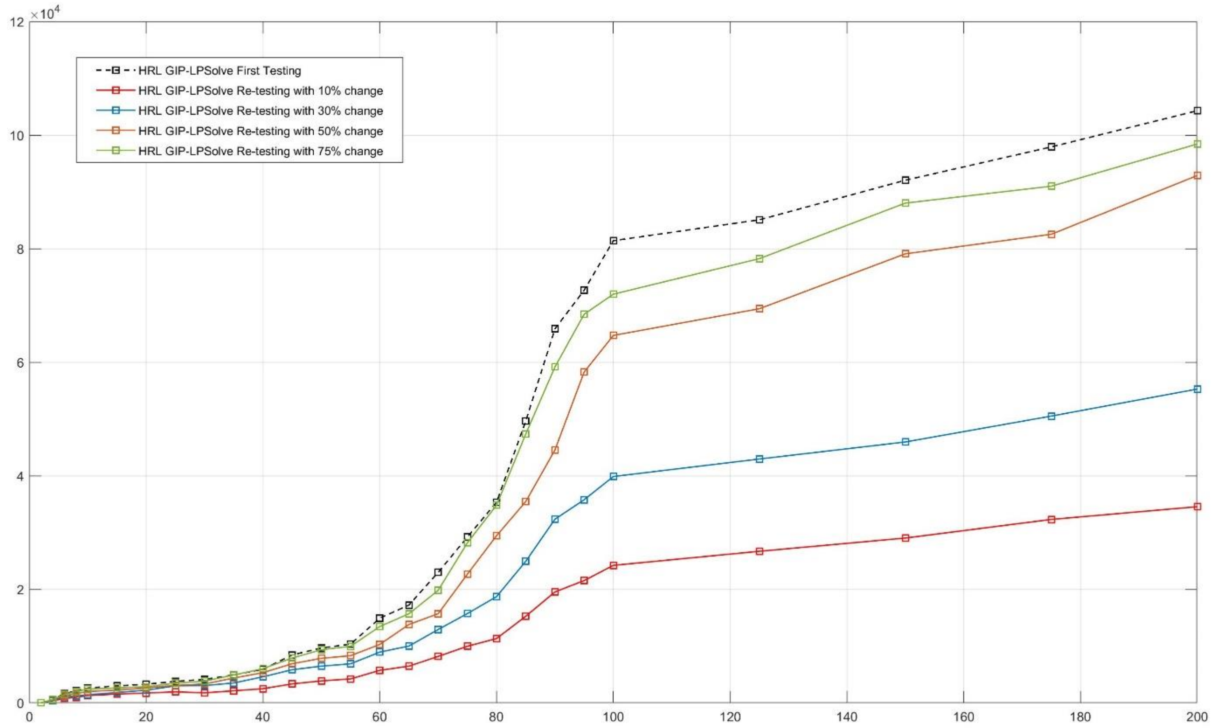


Figure 80: Re-testing the same network with introducing a percentage of change using HRL-GIP-LPSolve.

The obtained results outperformed the regular RL representation in all medium size LANs. In context of large LAN as illustrated in figure 81 the performance were beyond expectation with a clear performance enhancement in the retesting with 10% and 30% changes and with the 100-machines threshold for HRL-GIP-LPSolve retesting compared with first testing.

Finally, it worth to highlight that despite the re-testing covered different percentage of change, the 10% and 30% scenarios are the most probable and realistic as in networking the vast majority of machines configuration and defence remain unchanged or only experience few changes (not exceeding 10% over short period (weeks to months) and 30% over medium period (up to 2-years). The figure 81 illustrate IAPTF the amount of contribution of both hierarchical RL modelling and Initial Belief customization by using last test output data.

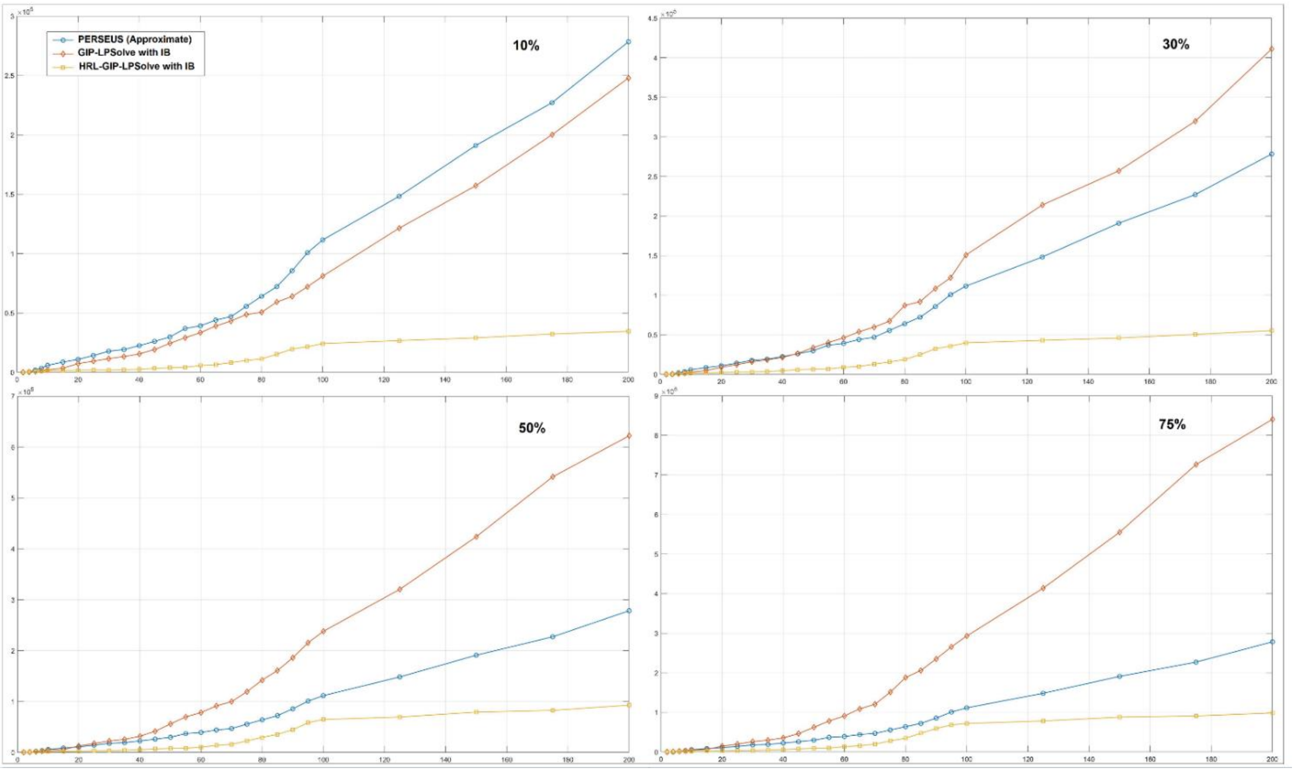


Figure 81: IAPTF re-testing performances' enhancement by algorithm on small, medium and large size LANs.

The performances enhancement in retesting is unanimous with obtained results surprisingly better in term of performance enhancement. HRL-GIP-LPSolve performed much better than the classic GIP in term if consumed time with results in the context of 100 to 200 machines LANs extremely positive.

7.10 Discussion and reflections

Discussing IAPTF results is a multi-step operation starting by discussing RL choice and validating the proposed model and learning approach. Later, we examine and discuss the obtained results in solving real-world PT problems modelled as POMDPs, then we proceed with discussing the obtained results obtained with different algorithms, RL representation and Initial Belief handling approaches as well different security auditing re-testing and compliance scenarios. This discussion focuses on the five evaluation metrics set initially which are: consumed time, coverage (exploration), reliability (exploitation), congestion and finally expertise extraction and reply in retesting with or without changes.

7.10.1 Reinforcement Learning and modelling choices

The choice of POMDP over MDP which are quite tractable to solve and easy to specify is because this later only consider the perfect knowledge of state which is mostly not the case in PT practice [36]. In fact, POMDP accounts for all sources of uncertainty uniformly and allows for continuous network discovery and information gathering actions. Nonetheless, POMDP choice comes with the price of enormous intractability in solving the problem optimally [23-24]. Consequently, we opted for modelling the POMDP as Belief-State MDP, this approach comes with the excellent news related to the fact that Value-Iteration is an exact method for determining the value function of POMDPs and the optimal action can be read from the value function for any belief state but the time complexity of solving POMDP in value-iteration is exponential in term of number of actions, number of observations and the dimensionality of the belief space which grows with number of states [140-141].

In term of solving approach, which is Policy Iteration for POMDPs, we opted for this option as it mimics the best possible way the PT expert decision making and self-learning and improvement. In general, Policy Iteration algorithms choose a policy and then determine the value function based on the current policy to later update the value function, based on Bellman’s equation, and finally update the policy and optionally iterate. Policy Iteration algorithms for POMDPs considered for this research are first GIP which is the improvement of original algorithm that was complex by introducing an efficient evaluation mechanism to assess local value function from policy and also the representation of policy using finite-state controller [49]. The second option considered was the PERSEUS Point-Based Value Iteration [65] which solve POMDP for finite set of belief points through initializing linear segment for each belief point and iterate while occasionally and keep adding new belief points as long as the improvements fall below a threshold. Finally, it is important to highlight that despite the use of a high performance server and optimised solving algorithms, large networks with size exceeding 200-machine result in very large POMDPs problems that are outside the scope of tractable exact solutions. The HRL modelling was then proposed to overcome this specific issue and making IAPTF universal and operational in all scenarios.

7.10.2 IAPTF performance

Overall, the obtained results consolidate prior assumptions on the role of ML and specifically RL in the automation of decision-making, performance enhancement and optimisation in the use of resources in offensive cyber security and notably PT practice. 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. HRL modelling is very efficient when used in the appropriate context (medium and large LANs), and IAPTF results are additional evidence of the drastic performance enhancement compared to an average human expert.

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 to define the overall consumed time that IAPTF will take to perform a full testing on the test-bed networks. The obtained results shown in the Figures 71-83 illustrate an initial comparison of different RL solving algorithms performances on different LANs which were also compared with manual PT consumed time based on the author's experience as a PT consultant and CEH as well as empirical estimation of the overall time required to perform an automated PT with no optimisation. It is clearly obvious that IAPTF 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 IAPTF.

In regular RL model, IAPTF performs better than CEH expert in small and medium size LANs when accounting for the gain associated to expertise extraction and cost cut. This was expected as solving large POMDP problems and time wasted in interactions will slow down the IAPTF. Nevertheless, the weakness in the performances is covered by IAPTF 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 case 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 IAPTF performance on large size LANs decreases 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 IAPTF.

Nonetheless, regular RL approach required to solve large POMDP environment generated from a medium size LAN required a huge amount of time of 149.5 hours (6.2 days) for a network of 100 machines and which is an unreasonable amount of time. The poor performances in medium networks of 100-200 machines was expected as the exact POMDP solving is a P-SPACE complete problem compared with NP-complete in approximate solving, thus the time required for solving became computationally intractable. We tested the new hierarchical representation of PT which meant solving several small size POMDP problem for each cluster then solving the inter-clusters POMDP problem. We accounted for the overall time required. The obtained results for five solving approaches, namely PERSEUS, RL-GIP-LPSolve, RL-GIP-LPSolve+Initial Belief HRL-GIP-LPSolve, and HRL-GIP-LPSolve+Initial Belief are plotted in figure 79 showing the mean values and standard deviations. In GIP as with other exact solving with linear programming (LP) algorithm is PSPACE-hard problem which in computational complexity theory can be solved using an amount of memory that is polynomial in the input length (polynomial space) and thus requiring polynomial time for finding optimal policies for POMDPs when that solution by linear programming is possible. When the decision horizon is large the use of discounted factor is crucial with the role of LP to ensure that the problem is solvable in polynomial time which is a huge reduction from the above PSPACE-hard making the solving equivalent to a MDPs which are fully observable POMDPs.

The performance enhancement in large LANs with size exceeding 100-machine is clear in hierarchical modelling where the security clustering is impactful as the security cluster size become more consistent (large corporate networks clustering produce typically 6 to 15 machines per cluster) and HRL GIP-LPSolve solving becomes more time-efficient in large LANs which is the ultimate validation of the HRL modelling designed and implemented to solve large LANs. In small LANs, we noticed that HRL GIP-LPSolve and HRL GIP-LPSolve with Initial Belief are relatively slower than GIP-LPSolve and GIP-LPSolve with Initial Belief This loss of performance is expected and justified by the extra processing done when defining a hierarchical structure for a problem, since we use value approximation at the higher nodes of the hierarchy to represent the value of selecting the various specialized lower-level POMDPs. The multiple testing shows that this loss of performance is minimal in small LANs with number less than 20 machines which is not the best scenario where the use of hierarchy is justified by the structure of the domain. We are looking at techniques to predict bounds on the performance loss as a function of the selected hierarchy. Finally, it worth to mention that despite the fact that the testing was limited to 200-machines HRL GIP-LPSolve with IB version is expected to solve extra-large LANs efficiently and effectively.

7.10.3 Expertise extraction and retesting

In terms of re-testing, the performed test on different LANs after introducing four different percentage of change in the assessed network with the changes introduced represent 10%, 30%, 50% and 75%. The obtained results confirmed one of this research hypotheses namely RL expertise extraction and reuse crucial impact when used as prior knowledge (initial belief) on algorithms' overall performances by notably accelerating the convergence toward optimal policy graph. The obtained results in the context of 100- to 200-machine LANs were outstanding and reduced considerably consumed time in 10% and 30% context. The retesting brilliant results are mainly due to the use of the enhanced GIP-LPSolve which utilises 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 labelled as good in results quality but bad in performance, but the introduced changes in initial belief sampling and managing along with prioritising 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 machines 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 has deep roots in 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 performance. This applies for the two versions of PERSEUS, GIP -LPSolve and GIP-LPSolve_IB with different levels depending on the solving approach and optimiaation. On the other side, the exact solving algorithm and notably the GIP reputed by their bad performances but good results and utilising this algorithm within IAPTF 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. Therefore we implemented a customised 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 customisation brought the sought impact on the IAPTF performance.

7.10.4 Exploration and Exploitation

RL is the only branch of machine learning that can enable, within a predefined environment, a software agent to explore and perform sequential decision-making under uncertainty and produce decision policies (also known as policy graphs) that have a high degree of accuracy and relevance along with acceptable performances in relation to complex tasks. The RL approach considered for PT practice is black-box (sometimes grey-box when initial belief is provided) as the RL agent explore and make decisions with respect to RL environment input data. These decisions are PGs which are translated into attack vectors using automated scripts. The output PGs and thus attack vector interpretation is a tricky task and require deep understanding of IALTF and especially RL model functioning along with mechanisms used such as scripting for cleaning and constructing attack vectors. Exploitation measurement is rooted in the deep problem of "explainable AI" and in this work we focused on the results output and its explanation. Form the obtained results, we confirm that IAPTF exploration and exploitation back the third hypothesis for this research as a RL-led PT practice produce more reliable result in term of testing different vulnerabilities and more accurate results in term of percentage of valid attack vector converted into confirmed testing output. All variant algorithms implemented in

IAPTF exceed by far any CEH performance with the number even doubled in large networks making IAPTF more reliable in terms of PT output confidence. In practical terms, using the adequate RL algorithm and adopting a new learning scheme enabled IAPTF to produce a very optimised attack policies when targeting the Machine M9 suspected to contain sensitive information and defined as the most secured machine within the test-bed network as illustrated in Figure 84. Indeed, the produced policy is from an attacker point of view obvious but getting an automated system to opt for such attack vectors despite being not minimal in terms of cost of the exploits and consumed time is the novelty in IAPTF which is able to sacrifice simplicity for a higher objective. IAPTF exploring and large coverage capabilities were able to find a very complex and non-obvious attack path in medium-size networks where, relying on authors experience, no human tester will directly consider adopting the below attack path despite being relevant and exploitable by hackers but instead will mainly focus on shorted and more direct attack paths.

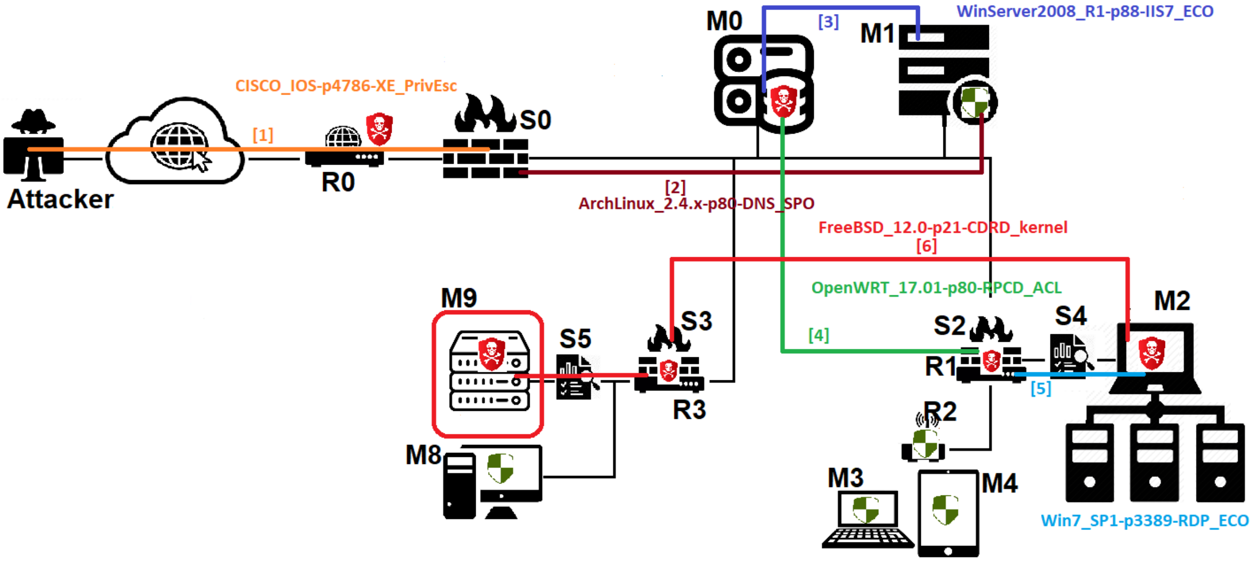

M0
[3]
M1
WinServer2008_R1-p88-IIS7_ECO
CISCO_IOS-p4786-XE_PrivEsc
[1]
S0
Attacker
R0
[2]
ArchLinux_2.4.x-p80-DNS_SPO
FreeBSD_12.0-p21-CDRD_kernel
[6]
OpenWRT_17.01-p80-RPCD_ACL
M9
S5
S3
S2
S4
M2
[4]
R1
[5]
R3
R2
M8
M3
M4
Win7_SP1-p3389-RDP_ECO

Figure 82: Partial illustration of IAPTF output attacking vectors on a 20-machines LAN.

The situation in large LANs is completely different as the attack vectors and notably composite and complex are less obvious to grasp by human CEH and often omitted or ignored. The figure 85 summarises few attack vectors (the most relevant). The high priority target encircled in red are located in different security clusters which requires the RL agents to procced to many pivoting (lateral movement) operations which is illustrated by the different attack vectors.

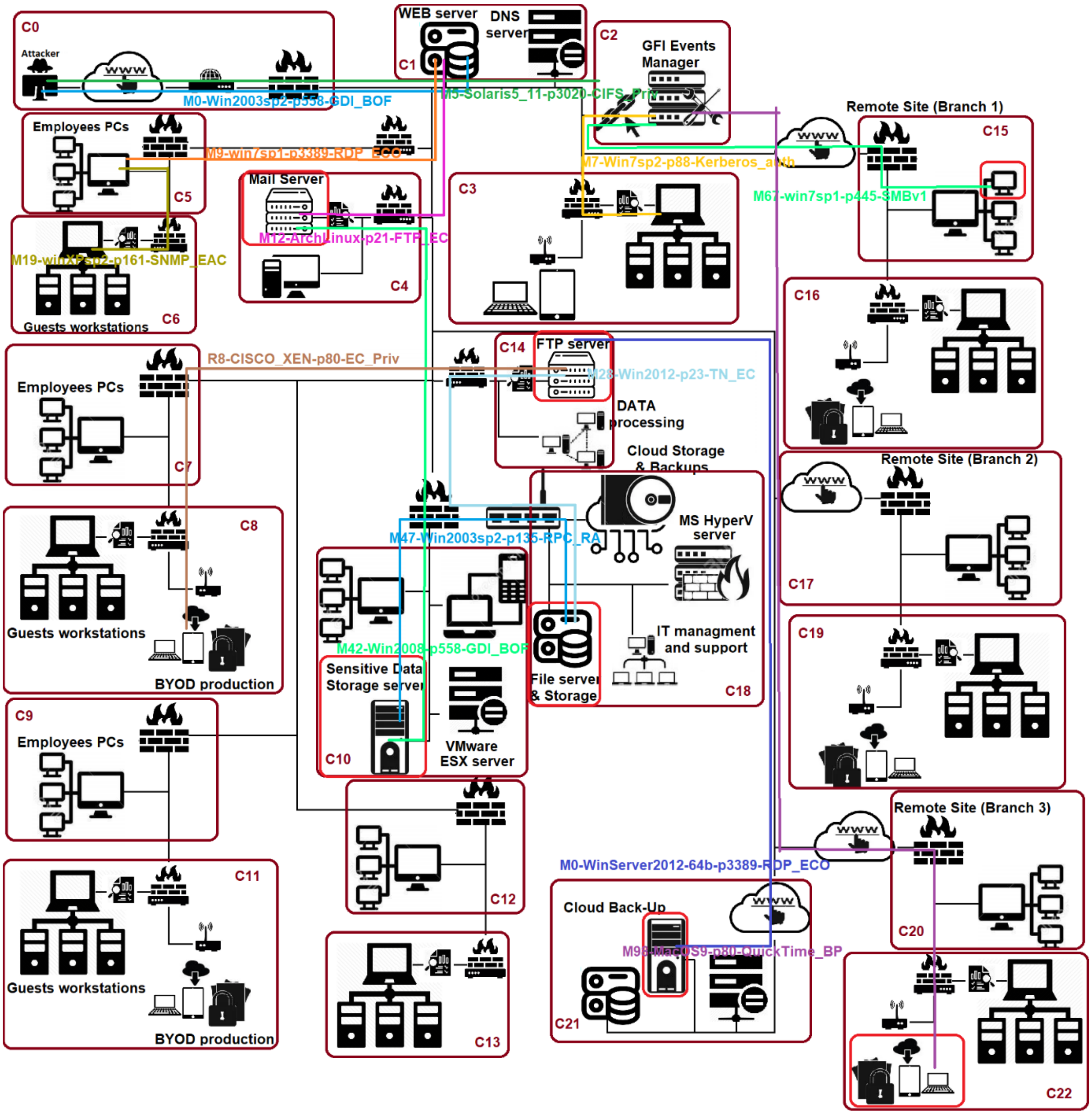


Figure 83 : Partial illustration of IAPTF output attack vectors on a 100-machines LAN.

To sum up, IAPTF does not only outperform CEH and blind automated system in term of time but also in term of exploration (covered attack) and exploitation (relevance and pertinence) of the executed and tested attacks. Several other positive outcomes were noticed, notably the pertinence of the produced result as solving POMDP resulted in PGs converted into valid attack vectors and assessment validated that IAPTF outperform CEH and blind automation in all five-assessment metrics. We estimate the evaluated the overall contribution of IAPTF to be twice better than human CEH in 200-

machine network and 5 to 6 times better than blind automation and this despite the framework heavyweight pre-processing and post-processing.

Chapter 8: Conclusion

8.1 Research Output Discussion

This section reviews the findings of the experiments conducted during this research to evaluate the proposed RL and HRL models and the performance of IAPTF in automating the testing of different size LANs recreated out of real-world data in our virtual environment.

PT is a complex and labor-intensive practice which despite the current use of tools and mostly automated systems, the complexity of PT tasks is also a major concern in current practice. This research has developed a new model and novel framework for PT using RL. Experiments conducted during this research have demonstrated the effectiveness of using RL to enable a software agent controlling an automated PT system to conduct vulnerability testing, exploration, and validation. There were a variety of POMDP solving algorithms used in the experiments: exact solving GIP and approximate solving PESEUS with both using linear programming and different approaches in handling and generating the initial belief. These RL solving approaches and algorithms were employed in different scenarios and inputs to real computer networks data represented in the form of POMDPs. All the solving approaches looked at achieved efficiency in term of consumed time, accuracy in term of vulnerability exploitation, and relevance in term of decision policies precision. The different solving approaches and algorithms were used into different networks varying in size from 2 to 200 machines created for this purpose and obtained results were compared as described in Chapter 7.

Overall, the proposed RL approach implemented within IAPTF outperforms fully automated PT systems by far in all testing scenarios including small LANs. During the first phase of IAPTF when only regular RL implementation were introduced, the performances of the framework were equivalent to the human expert (CEH) in small LANs but the gap started to widen in medium LANs (over 25 machines) to become huge in large LANs. This was due to our choice of opting into an exact solving of POMDP which is complex and costly, nonetheless the testing coverage and relevance of IAPTF is far superior than the CEH measured by the number of overall covered attack vectors and valid attack vectors respectively. Furthermore, this research proposed and implemented with the IAPTF the Hierarchical RL model of PT to tackle the scalability issue in large LANs. The obtained results showed that IAPTF performance outperforms any human expert (CEH) both in terms of consumed time (efficiency) and in terms of testing coverage and test relevance (effectiveness) [141].

Finally, we highlight that the IAPTF includes a further feature which is crucial towards the optimisation of the current PT automation, namely the expertise extraction, generalization and replay which enable fast retesting notably in security compliance scenarios. IAPTF permits the re-usability of the testing output by either learning and/or capturing the expertise during the test and storing it in the system memory for future use. It was proved to be very efficient in re-testing scenarios (very common in PT) and nearly similar cases when the testing time and accuracy of the produced results were exceptional. At a later stage of this research, we tackled the scalability issue raised in medium and large size networks by adopting and implementing a two-level hierarchical representation of the PT environment as POMDP problems. The first level tackled a set of small networks (security clusters) considered as independent small POMDP problem to solve, whereas the second level tackled the inter-clusters network composed from the head of clusters machines which are elected to be the most vulnerable and likely to be used by hackers for pivoting after achieving an accepted privilege escalation. This approach was then confirmed and validated in this research as the most realistic and optimal strategy to help a full RL-led automation of PT along with producing an efficient result in term of timing and expertise capturing. The HRL representation enabled a better explanation of the obtained results and more visibility about the testing coverage as well as an effective execution of different PT activities by notably excluding impossible scenarios which are often time consuming [141].

The methodology for conducting the research was done after a comprehensive review of related works and PT literature in general which enabled us to complete the domain understanding and identify the research gap and elaborate research questions. Then we proceeded with decomposing network PT practice in activities, tasks and subtasks to identify and understand the human expertise. Next we reviewed different AI techniques to elect the most relevant ones for our research which revealed to be RL because of its suitability to the domain of sequential decision making under uncertainty. At an early stage we proposed a POMDP representation of PT problem which does not only cover the planning but the entire practice. Then this research produced a novel application of RL techniques to the interactive part (and not the planning) of 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 IAPTF can act as a module and integrate with most of the industrial PT frameworks to significantly improve the efficiency and accuracy of medium and large networks context. The proposed modelling 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 IAPTF is integrated into the most 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.

8.2 Review of Contributions

The major contribution of this approach is to apply RL techniques to a real-world problem of automating and optimising PT practice. The research 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. Our work 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 scenarios (very common in PT) and nearly similar cases when the testing time and accuracy of the produced results were exceptional.

IAPTF performance, when adopting regular RL model, on relatively medium and large networks is far superior to the current industry baseline covering acceptable time usually allocated to PT expert. When we introduced the Hierarchical RL model, the performance improvement became striking notably when comparing blind automation and CEH with the novel hierarchical POMDP model of PT practice. The HRL approach relies on a complex processing during 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 has been tested and results obtained were carefully examined. The validated outcomes are excellent as HRL solves two major issues faced during the IAPTF testing: the performance enhancement as the system will be solving several small POMDP problems rather than dealing with one large and complex environment. In addition, the hierarchical model simplified and enhanced the process of expertise capturing and handling as this later is easily identified and reconstructed following the two levels (intra-cluster and inter-clusters) and then generalised and stored in the system memory employing expert system for future use which will depend on the changes introduced in the assessed network [142].

8.3 Identify and Address Limitations

Despite the fact of addressing the scalability and re-usability issues raised in the middle of this research notably by adopting the hierarchical RL model to tackle large LANs and also the use of expert system for experience generalisation and reply, this work encountered three limitations which are beyond the scope of this research. The first limitation of IAPTF is the need of high-caliber human expert supervision during early learning phases where this expert will perform or closely supervise PT activities undergone by IAPTF 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 second limitation of this research is that IAPTF and more specifically the RL and HRL model is designed and built to cover computer networks and infrastructures PT and thus does not account for application testing, web testing and IoT testing. This limitation is due to the adopted approach of applying RL in the form of POMDP which requires a set of modules to capture and process real-world PT data and then use a software agent to solve the problem and enable an optimised automation of PT practice along with improving the performances and also enhancing the output results' reliability and persistence which will lead optimistically to a PT system free from human error. Therefore, upgrading IAPTF capability to cover new testing variant and context should imperatively pass through the extension or eventually the development of new RL model and thus POMDP. The third limitation of IAPTF is the ability to capture expertise from unusual and complex testing vector choices when the human expert cannot validate the relevance and adequacy to enable the CLIPS expert system to process (make it general) and store it for future use without the intervention of IAPTF-Core module.

8.4 Directions for Future Work

Research and sciences always open the door for more questions rather than only answering existed questions. For that, in this research I assume further research on embedding AI techniques and optimising current automation of Vulnerability Assessment and PT is highly recommended and needed for the benefit of improving cyber resilience and security and offer a better protection to IT infrastructures form the fast-emerging cyber-threats and making cyber space a secured domain as possible. Future research direction in this domain will include:

- ➢ Investigating and implementing multi-agent solving as an alternative to improve performance on complex and large POMDP environments. This opting can be very beneficial in case of parallel testing where more than one session is launched on the system and in case of hierarchical modelling of large networks where the environment representation (states, actions, observations, transitions, and reward) associated with each phase of PT will be stored separately.

- Attempting a multi-layer hierarchical model which account for all phases, activities, tasks and sub-tasks of PT practice. This approach would address the expertise identification, capturing and generalisation and thus completely get rid of human expert intervention at early stage if learning.
- Extend the RL and HRL model to account for more testing scenarios such as application, web and IoT testing.
- Generalising the PT RL model itself to include web-penetration testing and eventually IoT and extend the IAPTF modules to accommodate for such upgrade.
- Attempt to extend the use of RL in more cyber security domain notably where human expert is heavily required such as incident handling and digital forensics.

8.5 Closing Remarks

In this research, I have worked towards a proof-of-concept framework which rely on RL branch of AI and other techniques to replace human expert in conducting PT and vulnerability assessment activities in an optimised and autonomous way. The sought-after framework is not intended to be stand-alone software but to integrate commercial and open-source existing systems such as Metasploit Pro, Core-Impact, Nessus and others. This research validated the theoretical model of network PT practice as a POMDP environment interaction problem and not just as planning problem, and which was then adapted to larger networks by adopting a hierarchical POMDP representation following the splitting of large networks into several security clusters and dealing with each one of them separately to then process the network composed from the head of clusters. Furthermore, during this research and before reaching the proposed formal RL and HRL models we contributed into formalising the network PT practice and advancing the domain understanding in relation to offensive cyber security evolution in parallel with real-world hackers and cyber-criminal organisations development.

References

[1] He, L., Bode, N. Network penetration testing. In: Blyth, A. (ed.) EC2ND. pp 3–12. Springer, London (2006).

[2] Bacudio, A., Yuan, X., Chu, B., Jones, M.: An Overview of Penetration Testing. International Journal of Network Security & Its Applications 3(1-2), 19–38 (2011). https://doi.org/10.5121/ijnsa.2011.3602

[3] Backes, M., Hoffmann, J., K¨unnemann, R., Speicher, P., Steinmetz, M.: Simulated penetration testing and mitigation analysis. (2017)ArXivabs/1705.05088.

[4] Phong, C., Yan, W.: An overview of penetration testing. International Journal of Digital Crime and Forensics (IJDCF) 6, 50–74 (2014). https://doi.org/10.4018/ijdcf.2014100104

[5] Abu-Dabaseh, F., Alshammari, E.: Automated penetration testing: An overview. Computer Science and Information Technology (2018)

[6] Yaqoob, I., Hussain, S., Mamoon, S., Naseer, N., Akram, J.: Penetration testing and vulnerability assessment. Journal of Network Communications and Emerging Technologies (JNCET) 7, 12–21 (2017)

[7] Singleton, C, Moore, S. and McMillen, D. Top 10 Cybersecurity Vulnerabilities. (2021). Security intelligence report, IBM, USA.

[8] Ghanem, M., Chen, T.: Reinforcement learning for efficient network penetration testing. Information 11, 6 (2019). https://doi.org/10.3390/info11010006.

[9] Sarraute, C., Buffet, O., Hoffmann, J.: Pomdps make better hackers: Accounting for uncertainty in penetration testing., proceedings of the twenty-sixth aaai conference on artificial intelligence, pp. 1816–1824 (2012).

[10] Boddy, M., Gohde, J., Haigh, T., Harp, S.: Course of action generation for cyber security using classical planning., proceedings of the 15th international conference on automated planning and scheduling. ICAPS'05, pp. 12–21 (2005).

[11] Sarraute, C., Richarte, G., Luc´angeli Obes, J.: An algorithm to find optimal attack paths in nondeterministic scenarios., proceedings of the acm conference on computer and communications security. (2013). https: //doi.org/10.1145/2046684.2046695

[12] Zennaro, F.M., Erdodi, L.: Modeling penetration testing with reinforcement learning using capture-the-flag challenges and tabular q-learning. ArXiv abs/2005.12632 (2020)

[13] Maeda, R., Mimura, M.: Automating post-exploitation with deep reinforcement learning. Computers & Security 100, 102108 (2021). https: //doi.org/10.1016/j.cose.2020.102108

[14] Bertoglio, D.D., Zorzo, A.F.: Overview and open issues on penetration test. Journal of the Brazilian Computer Society 23(1), 1–16 (2017)

[15] Mohan, V. (2014). A comparison of various reinforcement learning algorithms to solve race track problem. https://pdfs.semanticscholar.org.

[16] Schaul, T., Quan, J., Antonoglou, I., and Silver, D. 2016. Prioritized experience replay, Google DeepMind. Conference paper at ICLR 2016.

[17] Singhal, A. and Ou, X. (2011). Security risk analysis of enterprise Networks using probabilistic attack graphs. NIST Interagency Technical Report 7788.

[18] PETS Technical Guidelines. (2021). Available: http://www.penteststandard.org/index.php PTES Technical Guidelines.

[19] OSSIG. Penetration Test Framework (PTF). (2022). Available: http://cuchillac.net.

[20] Scarfone, K., Souppaya, M., Cody, A. and Orebaugh, A. Technical Guide to Information Security Test and Assessment. 2022.

[21] Kennedy, D., O'Gorman, J., Kearns, D. and Aharoni, M. Metasploit. The Penetration Tester's Guide. San Francisco. 2011.

[22] Layton, T. penetration studies: a technical overview.(2019). https://www.sans.org/reading-room/whitepapers.

[23] Schreuders, Z. C. and Ardern, L. Generating randomised virtualised scenarios for ethical hacking and computer security education: SecGen implementation and deployment. In Workshop on Cybersecurity Training & Education (VIBRANT15), 2015.

[24] Sawilla, R.E., Ou, X., 2008. Identifying critical attack assets in dependency attack graphs. In: Jajodia, S., Lopez, J. (Eds.), Computer Security - ESORICS 2008. In: Lecture Notes in Computer Science, volume 5283. Springer Berlin Heidelberg, pp. 18–34. doi: 10.1007/978- 3- 540- 88313- 5 _ 2 .

[25] Ou, X. , Singhal, A. , 2012. Quantitative Security Risk Assessment of Enterprise Networks. Springer.

[26] Mell, P. , Scarfone, K. , Romanosky, S. , 2006. Common vulnerability scoring system. Secur. Privacy 85–89.

[27] Bistarelli, S. , Dall'Aglio, M. , Peretti, P. , 2006. Strategic games on defense trees. In: International Workshop on Formal Aspects in Security and Trust. Springer, pp. 1–15 .
[28] Boddy, M.S. , Gohde, J. , Haigh, T. , Harp, S.A. , 2005. Course of action generation for cyber security using classical planning. In: ICAPS, pp. 12–21 .
[29] Borbor, D. , Wang, L. , Jajodia, S. , Singhal, A. , 2017. Securing networks against un-patchable and unknown vulnerabilities using heterogeneous hardening options. In: IFIP Annual Conference on Data and Applications Security and Privacy. Springer, pp. 509–528 .
[30] Polatidis, N., Pavlidis, M. and Mouratidis, H., 2017. Cyber-attack path discovery in a dynamic supply chain maritime risk management system. Computer Standards International conference.
[31] Creasey, J. and Glover, I. CREST: A guide for running an effective Penetration Testing program. http://www.crest-approved.org (2017).
[32] Almubairik, N. and Wills, G. Automated Penetration Testing Based on a Threat Model. The 11th International Conference for Internet Technology and Secured Transactions (ICITST), (2016).
[33] Denis, M., Zena, C. and Hayajneh, T. Penetration testing: Concepts, attack methods, and defense strategies. In 2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT), (2016).
[34] Morgan, J.P. Global cybersecurity spending exceeded 1 trillion dollars for 2017-2021. Cybercrime Magazine, (2021).
[35] Ferreira, A. and Kleppe, H. Effectiveness of automated application penetration testing tools, (2011).
[36] Conheady, S. Social engineering in IT security: Tools, tactics, and techniques. McGraw-Hill Education Group, (2014).
[37] Broad, J. and Bindner, A. Hacking with Kali: practical penetration testing techniques. Newnes, (2013).
[38] Haubris, K. and Pauli, J. Improving the efficiency and effectiveness of penetration test automation. 10th International Conference on Information Technology IEEE, pages 387-391. (2013).
[39] Jaswal, N. Mastering Metasploit. Fourth Edition Packt Publishing, (2020).
[40] Mirjalili, M., Nowroozi, A. and Alidoosti. M. A survey on web penetration test. Advances in Computer Science International Journal, 3(6):107-121, (2014).
[41] Nickerson, C., Kennedy, D., Smith, E., Rabie, A. and McCray, J. Penetration testing execution standard. http://www.pentest-standard.org. 2014.
[42] Shah, S., Babu M. and Mehtre, M. An overview of vulnerability assessment and penetration testing techniques. Journal of Computer Virology and Hacking Techniques, 11(1):27-49, (2015).
[43] Stefinko, Y., Piskozub, A. and Banakh, R. Manual and automated penetration testing, benefits and drawbacks, modern tendency. In 13th IEEE International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET), pages 488-491. (2016).
[44] Khan, S. and Parkinson, S., Review into state of the art of vulnerability assessment using artificial intelligence. Guide to Vulnerability Analysis for Computer Networks and Systems, pages 3-32. (2018).
[45] Samant, N. (2011). Automated Penetration Testing. Master thesis submitted to San Jose State University, Spring 2011. B. Schneier. Attack trees. Dr Dobbs Journal, 1999.
[46] Bacudio, A., Yuan, X., Chu, B. and Jones, M., An overview of penetration testing. International Journal of Network Security & Its Applications, 3(6), page 19. (2011).
[47] Obes, J.L., Richarte, G., and Sarraute, C. (2013). Attack Planning in the Real World. Journal Article, CoRR, abs/1306.4044.
[48] Sarraute, C. Buffet, O., and Hoffmann, J. (2012). POMDPs make better hackers: Accounting for uncertainty in penetration testing. Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI'12), pages 1816– 1824, Toronto, ON, Canada, July 2012. AAAI Press.
[49] Obes, J.L., Richarte, G., and Sarraute, C. (2013). Attack Planning in the Real World. Journal Article, CoRR, abs/1306.4044.
[50] Sarraute, C. 2012. Automated attack planning. Instituto Tecnologico de Buenos-aires, Ph.D. Thesis, Argentina, submitted on 02/07/2012.
[51] Sarraute, C., Richarte, G. and Lucangeli, J. An algorithm to find optimal attack paths in non-deterministic scenarios. In Workshop on Security and Artificial Intelligence, pages 71–80, 2011.
[52] Jimenez, S., De-la-rosa, T., Fernandez, S., Fernandez, F. and Borrajo, D. 2009. A Review of Machine Learning for Automated Planning. The Knowledge Engineering Review, Vol. 00:0, 1–24.c 2009, Cambridge University Press.
[53] Pasquale, L, Hanvey, S., Mcgloin, M. and Nuseibeh, B. (2016). Adaptive Evidence Collection in the Cloud Using Attack Scenarios. Preprint submitted to Computers and Security Journal February 8, 2016.
[54] Hoffmann, J. (2015). Simulated penetration testing: From "Dijkstra" to "Turing Test++". Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15). AAAI Press, 2015.
[55] Russell, S. and Norvig, P. (2009). Artificial Intelligence: A Modern Approach. 3rd Ed. Prentice Hall, 2009.

[56] Phillips, C. Swiler, L. P. (1998). A graph-based system for network-vulnerability analysis. In Proceedings of the New Security Paradigms Workshop 1998.
[57] Thaier, H. (2014). Attack Graph Approach to Dynamic Network Vulnerability Analysis and Countermeasures, Ph.D. thesis submitted to the University of Bedfordshire.
[58] NIST. 2017. Computer Security Resource Center - National Vulnerability Database. https://nvd.nist.gov. Last Accessed 23/05/2021.
[59] Qiu, X., Jia, Q., Wang, S., Xia, C. AND Shuang, L. (2014). Automatic generation algorithm of penetration graph in penetration testing, 2014 Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing.
[60] Dean R., McKinnel, T., Dargahi, A., and Kim-Kwang, R. A systematic literature review and meta-analysis on artificial intelligence in penetration testing and vulnerability assessment. Computers & Electrical Engineering, Volume 75, 175-188, (2019).
[61] Sheyner, O., Haines, J., Jha, S., Lippmann, R. and Wing, J. Automated generation and analysis of attack graphs. In Proceedings 2002 IEEE Symposium on Security and Privacy, pages 273-284. IEEE, (2002).
[62] Shmaryahu, D., Shani, G., Hoffmann, G. and Steinmetz, M. Constructing plan trees for simulated penetration testing. In The 26th international conference on automated planning and scheduling, volume 121, (2016).
[63] Thrun, S. and Littman, M.L., Reinforcement learning: an introduction. AI Magazine, 21(1), pages 103-143. (2000).
[64] Applebaum, A., Miller, D., Strom, B., Korban, C. and Wol, R. (2016). Intelligent, automated red team emulation. ACSAC '16 Proceedings of the 32nd Annual Conference on Computer Security Applications, Pages 363-373. ISBN 978-1-4503-4771-6/16/12.
[65] Backes, M. Hoffmann, J., Kunnemann, R., Speicher, P. and Steinmetz, M. (2017). Simulated Penetration Testing and Mitigation Analysis. http://arxiv.org/abs/1705.05088.
[66] Durkota, K., Lisý, V., Bošanský, B., Kiekintveld, C. and Pěchouček, M., 2019. Hardening networks against strategic attackers using attack graph games. Computers & Security, 87, p.101578.
[67] Z. Hu, R. Beuran, and Y. Tan. Automated penetration testing using deep reinforcement learning. In 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS PW), page 2, (2020).
[68] Chu, G., 2021. Automation of Penetration Testing. Ph.D Thesis, University of Liverpool.
[69] Montuno., D, 2018. Machine Learning in Vulnerability Assessment. Defence Research and Development Canada – Ottawa Research Centre.
[70] Niculae, S., Dichiu, D., Yang, K. and Bäck, T., Automating penetration testing using reinforcement learning. (2020).
[71] Maeda, R. and Mimura, M., 2021. Automating post-exploitation with deep reinforcement learning. Computers & Security, 100, p.102108.
[72] Dayan, P. and Daw, N. D. (2008). Decision theory, reinforcement learning, and the brain. Cognitive, Affective, & Behavioral Neuroscience 2008, 8 (4), 429-453 doi:10.3758/CABN.8.4.429.
[73] Dejmal, S., Fern, A. and Nguyen, T. (2008). Reinforcement Learning for Vulnerability Assessment in Peer-to-Peer Networks. IAAI'08 Proceedings of the 20th national conference on Innovative applications of artificial intelligence V3. Pages 1655-1662, Chicago, Illinois — July 13 - 17, 2008, AAAI Press ISBN: 978-1-57735-368-3.
[74] Durkota, K, Lisy, V., Bosansk, B. and Kiekintveld, C. (2015). Optimal Network Security Hardening Using Attack Graph Games. IJCAI2015 Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence.
[75] Ghosh, N. and Ghosh, S. (2009). An Intelligent Technique for Generating Minimal Attack Graph, August 6, 2009
[76] Chowdary., A, Huang., D, Mahendran.,J, Romo.,D, Deng.,Y. and Sabur., A. 2020. Autonomous Security Analysis and Penetration Testing. Arizona State University.
[77] McKinnel, D., Dargahi, T., Dehghantanha, A. and Choo, K., 2019. A systematic literature review and meta-analysis on artificial intelligence in penetration testing and vulnerability assessment. Computers and Electrical Engineering, 75, pp.175-188.
[78] Speicher, P., Steinmetz, M., Hoffmann, J., Backes, M. and Künnemann, R., 2019. Towards automated network mitigation analysis. Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing.
[79] Spaan, M. (2012). Partially Observable Markov Decision Processes, Reinforcement Learning: State of the Art, Springer Verlag, 2012.
[80] Taylor, M., Whiteson, S. and Stone, P. (2007). Temporal Difference and Policy Search Methods for Reinforcement Learning: An Empirical Comparison. In Proceedings of the Twenty-Second National Conference on Artificial Intelligence (AAAI 07), pp. 1675-1678, Vancouver, Canada, July 2007.
[81] Tangkaratt, V., Van Hoof, H., Parisi, S., Neumann, G., Peters, J. and Sugiyama, M. 2016. Policy Search

with High-Dimensional Context Variables.
[82] Veeramachaneni, K., Arnaldo, I., Cuesta-Infante, A., Korrapati, V., Bassias, C. and Li, K. 2016. AI2: Training a big data machine to defend. CSAIL, MIT Cambridge. Our work
[83] Al-Emran, M.: Hierarchical reinforcement learning: A survey. International Journal of Computing and Digital Systems 4, 2210–142 (2015). https://doi.org/10.12785/ijcds/040207
[84] Moerland, T., Broekens, J., Jonker, C.: Model-based Reinforcement Learning: A Survey, (2020). https://doi.org/10.48550/arXiv.2006.16712.
[85] Roijers, D.M., Vamplew, P., Whiteson, S., Dazeley, R.: A survey of multiobjective sequential decision-making. Journal of Artificial Intelligence Research 48, 67–113 (2013).
[86] Spaan, M.T.: Partially observable markov decision processes. In: Reinforcement Learning, pp. 387–414. Springer, (2012)
[87] Jain, A., Niekum, S.: Efficient Hierarchical Robot Motion Planning Under Uncertainty and Hybrid Dynamics (2018).
[89] Stock, S.: Hierarchical hybrid planning for mobile robots. KI-K¨unstliche Intelligenz 31(4), 373–376 (2017)
[90] Joglekar, N. Hierarchical planning under uncertainty: Real options and heuristics, page 291–313. https://doi.org/10.1016/B978-0-7506-8552-8.50014-1. (2008).
[91] Babenko, L., Kirillov, A.: Development of automated malware detection system. izvestiya SFedU. Engineering sciences, 153–167 (2022). https://doi.org/10.18522/2311-3103-2021-7-153-167
[92] Zhou, R., Pan, J., Tan, X., Xi, H.: Application of clips expert system to malware detection system, vol. 1, pp. 309–314. https://doi.org/10.1109/CIS. (2008).
[93] Pineau, J., Gordon, G.: Point-based value iteration: An anytime algorithm for POMDP. proceedings international joint conference of artificial intelligence., pp. 1025–1032 (2003)
[94] Spaan, M., Vlassis, N.: Perseus: Randomized point-based value iteration for POMDP. Journal Artificial Intelligence Research. (JAIR) 24, pages 195–220 (2005).
[95] Walraven, E., Spaan, M.T.J.: Accelerated vector pruning for optimal POMDP solvers. In: AAAI (2017).
[96] Zhang, W., Nevin, D., Zhang, N., Supervisor, T., Golin, G.: Algorithms for partially observable markov decision processes. doi.org/10.14288/1.0098252 (2003)
[97] Walraven, E., Spaan, M.: Accelerated vector pruning for optimal POMDP solvers. Proceedings of the AAAI Conference on Artificial Intelligence 31(1) (2017).
[98] Valea,. O and Oprisa,. C. 2020. Towards Pentesting Automation Using the Metasploit Framework. Technical University of Cluj-Napoca and Bitdefender.
[99] Andrew, Y. and Jordan, M. PEGASUS: A policy search method for large MDPs and POMDPs. Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (2013).
[100] Cassandra, A.R., Littman, M.L., Zhang, N.L.: Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes. arXiv preprint arXiv:1302.1525 (2013)
[101] Hans, E.W., Herroelen, W., Leus, R. and Wullink, G., A hierarchical approach to multi-project planning under uncertainty. Omega, 35(5), page 563-577. (2007).
[102] Kim, S. and Lee, H., Software systems at risk: An empirical study of cloned vulnerabilities in practice. Computers & Security, 77, pp.720-736. (2018).
[103] Gatti, C. (2015). Design of Experiments for Reinforcement Learning, Springer International Publishing Switzerland 2015.Springer Theses, DOI 10.1007/978-3-319-12197-0_2.
[104] Meuleau, N., Kim, K., Kaelbling, L., Cassandra, A. 2013. Solving POMDPs by Searching the Space of Finite Policies. Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI1999).
[105] Heinl, C. H. 2016. Artificial (Intelligent) Agents and Active Cyber Defense: Policy Implications. 6th International Conference on Cyber Conflict. NATO CCD COE Publications, Tallinn.
[106] MITRE, C., 2019. Common vulnerabilities and exposures (cve). Retrieved from https://cve.mitre. org/about/index. html.
[107] Heinrich, J. and Silver, D. 2016. Deep Reinforcement Learning from Self-Play in Imperfect-Information Games, University College London.
[108] Kervinen, A. and Virolainen, P. (2005). Heuristics for Faster Error Detection With Automated Black Box Testing. Electronic Notes in Theoretical Computer Science 111 (2005) 53–71.
[109] McKinnel, D.R., Dargahi, T., Dehghantanha, A. and Choo, K.K.R., A systematic literature review and meta-analysis on artificial intelligence in penetration testing and vulnerability assessment. Computers & Electrical Engineering, 75, pages 175-188. (2019).
[110] Bryce, D. and Kambhampati, S. (2007). A Tutorial on Planning Graph–Based Reachability Heuristics. Association for the Advancement of Artificial Intelligence. All rights reserved. ISSN 0738-4602, SPRING 2007
[111] Li, Z., Zou, D., Xu, S., Jin, H., Qi, H. and Hu, J., December. VULPECKER: an automated vulnerability

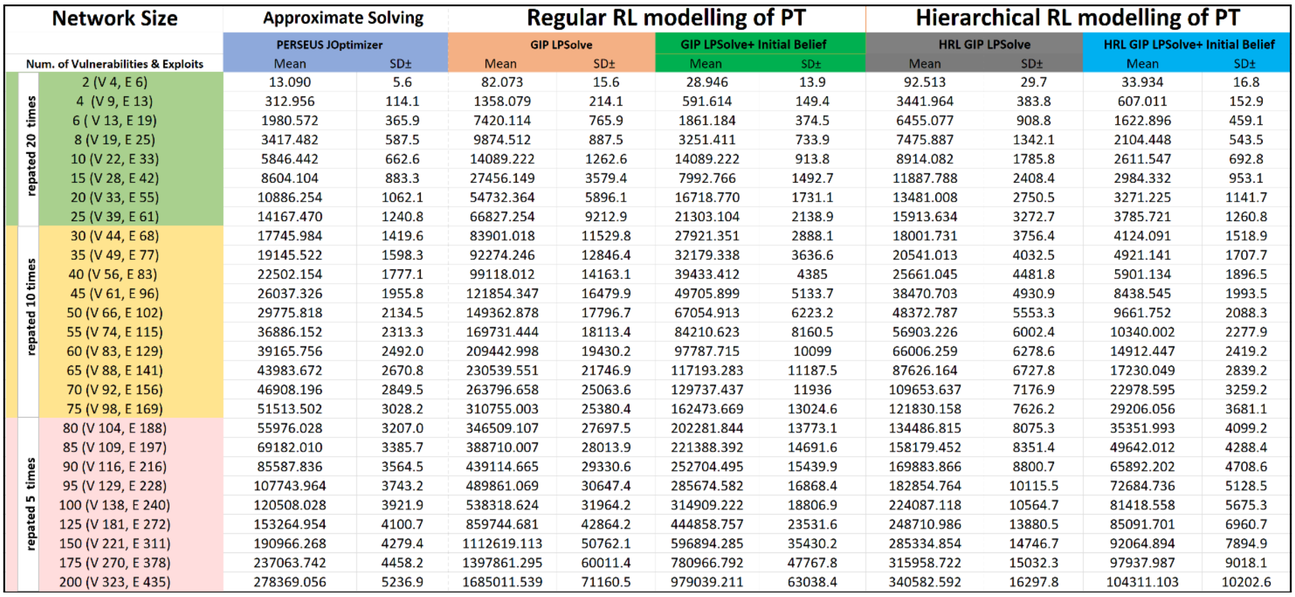
detection system based on code similarity analysis. In Proceedings of the 32nd Annual Conference on Computer Security Applications, page 201-213, (2016).
[112] Giarratano, J., Ph.D. CLIPS User's Guide Version Quicksilver Beta,December 31st 2007. http://clipsrules.sourceforge.net/OnlineDocs.html.
[113] CLIPS Reference Manua Volume I Basic Programming Guide Quicksilver Beta, December 31st 2021.http://clipsrules.sourceforge.net/OnlineDocs.html
[114] Svacina, J., Raffety, J., Woodahl, C., Stone, B., Cerny, T., Bures, M., Shin, D., Frajtak, K. and Tisnovsky, P., On vulnerability and security log analysis: A systematic literature review on recent trends. In Proceedings of the International Conference on Research in Adaptive and Convergent Systems. pages 175-180. (2020).
[115] Jesse Hoey and Pascal Poupart. 2005. Solving POMDPs with continuous or large discrete observation spaces. In Proceedings of the 19th international joint conference on Artificial intelligence (IJCAI'05). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1332–1338.

[116] Douglas Aberdeen and Jonathan Baxter. Scalable internal-state policy-gradient methods for POMDPs. In Proceedings of the Nineteenth International Conference on Machine Learning, pages 3–10, (2002).
[117] Hansen, E. Solving POMDPs by searching in policy space. In Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, pages 211–219, Madison, WI, 1998.
[118] Planning and acting in partially observable stochastic domains. Artificial Intelligence, 101:99–134, (1998).
[119] Leonid Peshkin, Nicolas Meuleau, and Leslie P. Kaelbling. Learning policies with external memory. In Proceedings of the Sixteenth International Conference on Machine Learning, pages 307–314, San Francisco, CA, (1999).
[120] Littman, M. Memoryless policies: Theoretical limitations and practical results. In Dave Cliff, Philip Husbands, Jean-Arcady Meyer, and Stewart W. Wilson, editors, Proceedings of the Third International Conference on Simulation of Adaptive Behavior, Cambridge, MA, 1994. The MIT Press.
[121] Madani O, Hanks O, and Condon A. On the undecidability of probabilistic planning and infinite-horizon partially observable decision problems. In Proceedings of the Sixteenth National Conference on Artificial Intelligence, pages 541–548, Orlando, 1999.
[122] Marco Wiering and Juergen Schmidhuber. HQ-learning. Adaptive Behavior, 6(2):219–246, 1997.
[123] Meuleau N, Kim K, Kaelbling L, and Cassandra A. Solving POMDPs by searching the space of finite policies. In Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, pages 417–426, Stockholm, 1999.
[124] Spaan MTJ, Vlassis N (2005a) Perseus: Randomized point-based value iteration for POMDPs. Journal of Artificial Intelligence Research 24:195–220
[125] Papadimitriou C and Tsitsiklis J. The complexity of Markov decision processes. Mathematics of Operations Research, 12(3):441–450, 1987.
[126] Varakantham P, Maheswaran R, Tambe M (2005) Exploiting belief bounds: Practical POMDPs for personal assistant agents. In: Proc. of Int. Conference on Autonomous Agents and Multi-Agent Systems
[127] Vlassis N, Toussaint M (2009) Model-free reinforcement learning as mixture learning. In: International Conference on Machine Learning, ACM, pp 1081–1088.
[128] Jain, A., Niekum, S.Efficient Hierarchical Robot Motion Planning Under Uncertainty and Hybrid Dynamics (2018)
[129] Stock, S.: Hierarchical hybrid planning for mobile robots. KI-Kunstliche Intelligenz 31(4), 373{376 (2017)
[130] Joglekar, N.: Hierarchical planning under uncertainty: Real options and heuristics, pp. 291{313 (2008). https://doi.org/10.1016/B978-0-7506-8552-8.50014-1.
[131] Babenko, L., Kirillov, A.: Development of automated malware detection system. izvestiya SFedU. Engineering sciences, 153{167 (2022). https://doi.org/10.18522/2311-3103-2021-7-153-167
[132] Zhou, R., Pan, J., Tan, X., Xi, H.: Application of clips expert system to malware detection system, vol. 1, pp. 309{314 (2008). https://doi.org/10.1109/CIS.2008.100
[133] Pineau, J., Gordon, G.: Point-based value iteration: An anytime algorithm for pomdps., proceedings international joint conference of artificial intelligence., pp. 1025{1032 (2003)
[134] Zhang, W., Nevin, D., Zhang, N., Supervisor, T., Golin, G.: Algorithms for partially observable markov decision processes. doi.org/10.14288/1.0098252 (2003)
[135] Walraven, E., Spaan, M.: Accelerated vector pruning for optimal pomdp solvers. Proceedings of the AAAI Conference on Artificial Intelligence31(1) (2017)
[136] Barreto, A., Hou, S., Borsa, D., Silver, D. and Precup, D., 2020. Fast reinforcement learning with generalized policy updates. Proceedings of the National Academy of Sciences, 117(48), pp.30079-30087.

[137] Sutton, R. and Barto, A. 2018. Reinforcement Learning: An Introduction (MIT Press, 2018).
[138] Kober, J., Bagnell, A. and Peters, J. 2013. Reinforcement learning in robotics: A survey. International Journal of Robotics Research. 32, 1238–1274 (2013)
[139] Janner, M., Fu, J., Zhang, M. and Levine, S., 2019. When to trust your model: Model-based policy optimization. Advances in Neural Information Processing Systems, 32.
[140] Dimitrakakis, C. and Ortner, R., 2018. Decision making under uncertainty and reinforcement learning. Book available at *http://www.cse.chalmers.se*.
[141] Ghanem, M., Chen, T. and Nepomuceno, E. 2022. Hierarchical Reinforcement Learning for Efficient and Effective Automated Penetration Testing of Large Networks, Journal of Intelligent Information Systems, https://doi.org/10.21203/rs.3.rs-1686285/v1.

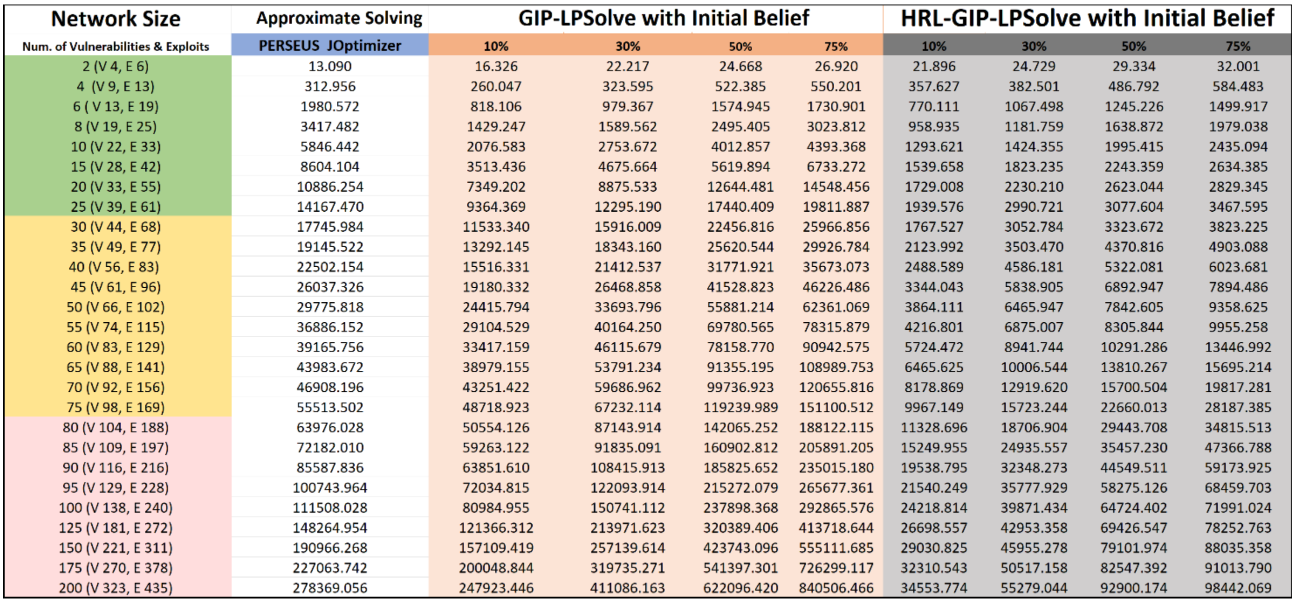
Appendix A

Table 11: Time and standard deviation in seconds consumed to solve POMDP problem for different size LANs



	Network Size	Approximate Solving		Regular RL modelling of PT				Hierarchical RL modelling of PT			
		PERSEUS JOptimizer		GIP LPSolve		GIP LPSolve+ Initial Belief		HRL GIP LPSolve		HRL GIP LPSolve+ Initial Belief	
	Num. of Vulnerabilities & Exploits	Mean	SD±	Mean	SD±	Mean	SD±	Mean	SD±	Mean	SD±
repated 20 times	2 (V 4, E 6)	13.090	5.6	82.073	15.6	28.946	13.9	92.513	29.7	33.934	16.8
	4 (V 9, E 13)	312.956	114.1	1358.079	214.1	591.614	149.4	3441.964	383.8	607.011	152.9
	6 (V 13, E 19)	1980.572	365.9	7420.114	765.9	1861.184	374.5	6455.077	908.8	1622.896	459.1
	8 (V 19, E 25)	3417.482	587.5	9874.512	887.5	3251.411	733.9	7475.887	1342.1	2104.448	543.5
	10 (V 22, E 33)	5846.442	662.6	14089.222	1262.6	14089.222	913.8	8914.082	1785.8	2611.547	692.8
	15 (V 28, E 42)	8604.104	883.3	27456.149	3579.4	7992.766	1492.7	11887.788	2408.4	2984.332	953.1
	20 (V 33, E 55)	10886.254	1062.1	54732.364	5896.1	16718.770	1731.1	13481.008	2750.5	3271.225	1141.7
	25 (V 39, E 61)	14167.470	1240.8	66827.254	9212.9	21303.104	2138.9	15913.634	3272.7	3785.721	1260.8
repated 10 times	30 (V 44, E 68)	17745.984	1419.6	83901.018	11529.8	27921.351	2888.1	18001.731	3756.4	4124.091	1518.9
	35 (V 49, E 77)	19145.522	1598.3	92274.246	12846.4	32179.338	3636.6	20541.013	4032.5	4921.141	1707.7
	40 (V 56, E 83)	22502.154	1777.1	99118.012	14163.1	39433.412	4385	25661.045	4481.8	5901.134	1896.5
	45 (V 61, E 96)	26037.326	1955.8	121854.347	16479.9	49705.899	5133.7	38470.703	4930.9	8438.545	1993.5
	50 (V 66, E 102)	29775.818	2134.5	149362.878	17796.7	67054.913	6223.2	48372.787	5553.3	9661.752	2088.3
	55 (V 74, E 115)	36886.152	2313.3	169731.444	18113.4	84210.623	8160.5	56903.226	6002.4	10340.002	2277.9
	60 (V 83, E 129)	39165.756	2492.0	209442.998	19430.2	97787.715	10099	66006.259	6278.6	14912.447	2419.2
	65 (V 88, E 141)	43983.672	2670.8	230539.551	21746.9	117193.283	11187.5	87626.164	6727.8	17230.049	2839.2
	70 (V 92, E 156)	46908.196	2849.5	263796.658	25063.6	129737.437	11936	109653.637	7176.9	22978.595	3259.2
	75 (V 98, E 169)	51513.502	3028.2	310755.003	25380.4	162473.669	13024.6	121830.158	7626.2	29206.056	3681.1
repated 5 times	80 (V 104, E 188)	55976.028	3207.0	346509.107	27697.5	202281.844	13773.1	134486.815	8075.3	35351.993	4099.2
	85 (V 109, E 197)	69182.010	3385.7	388710.007	28013.9	221388.392	14691.6	158179.452	8351.4	49642.012	4288.4
	90 (V 116, E 216)	85587.836	3564.5	439114.665	29330.6	252704.495	15439.9	169883.866	8800.7	65892.202	4708.6
	95 (V 129, E 228)	107743.964	3743.2	489861.069	30647.4	285674.582	16868.4	182854.764	10115.5	72684.736	5128.5
	100 (V 138, E 240)	120508.028	3921.9	538318.624	31964.2	314909.222	18806.9	224087.118	10564.7	81418.558	5675.3
	125 (V 181, E 272)	153264.954	4100.7	859744.681	42864.2	444858.757	23531.6	248710.986	13880.5	85091.701	6960.7
	150 (V 221, E 311)	190966.268	4279.4	1112619.113	50762.1	596894.285	35430.2	285334.854	14746.7	92064.894	7894.9
	175 (V 270, E 378)	237063.742	4458.2	1397861.295	60011.4	780966.792	47767.8	315958.722	15032.3	97937.987	9018.1
	200 (V 323, E 435)	278369.056	5236.9	1685011.539	71160.5	979039.211	63038.4	340582.592	16297.8	104311.103	10202.6

Table 12: Time in seconds consumed to solve POMDP problem for Re-testing cases after introducing a variable percentage of change and using last testing output as initial belief.



Network Size	Approximate Solving	GIP-LPSolve with Initial Belief				HRL-GIP-LPSolve with Initial Belief			
Num. of Vulnerabilities & Exploits	PERSEUS JOptimizer	10%	30%	50%	75%	10%	30%	50%	75%
2 (V 4, E 6)	13.090	16.326	22.217	24.668	26.920	21.896	24.729	29.334	32.001
4 (V 9, E 13)	312.956	260.047	323.595	522.385	550.201	357.627	382.501	486.792	584.483
6 (V 13, E 19)	1980.572	818.106	979.367	1574.945	1730.901	770.111	1067.498	1245.226	1499.917
8 (V 19, E 25)	3417.482	1429.247	1589.562	2495.405	3023.812	958.935	1181.759	1638.872	1979.038
10 (V 22, E 33)	5846.442	2076.583	2753.672	4012.857	4393.368	1293.621	1424.355	1995.415	2435.094
15 (V 28, E 42)	8604.104	3513.436	4675.664	5619.894	6733.272	1539.658	1823.235	2243.359	2634.385
20 (V 33, E 55)	10886.254	7349.202	8875.533	12644.481	14548.456	1729.008	2230.210	2623.044	2829.345
25 (V 39, E 61)	14167.470	9364.369	12295.190	17440.409	19811.887	1939.576	2990.721	3077.604	3467.595
30 (V 44, E 68)	17745.984	11533.340	15916.009	22456.816	25966.856	1767.527	3052.784	3323.672	3823.225
35 (V 49, E 77)	19145.522	13292.145	18343.160	25620.544	29926.784	2123.992	3503.470	4370.816	4903.088
40 (V 56, E 83)	22502.154	15516.331	21412.537	31771.921	35673.073	2488.589	4586.181	5322.081	6023.681
45 (V 61, E 96)	26037.326	19180.332	26468.858	41528.823	46226.486	3344.043	5838.905	6892.947	7894.486
50 (V 66, E 102)	29775.818	24415.794	33693.796	55881.214	62361.069	3864.111	6465.947	7842.605	9358.625
55 (V 74, E 115)	36886.152	29104.529	40164.250	69780.565	78315.879	4216.801	6875.007	8305.844	9955.258
60 (V 83, E 129)	39165.756	33417.159	46115.679	78158.770	90942.575	5724.472	8941.744	10291.286	13446.992
65 (V 88, E 141)	43983.672	38979.155	53791.234	91355.195	108989.753	6465.625	10006.544	13810.267	15695.214
70 (V 92, E 156)	46908.196	43251.422	59686.962	99736.923	120655.816	8178.869	12919.620	15700.504	19817.281
75 (V 98, E 169)	55513.502	48718.923	67232.114	119239.989	151100.512	9967.149	15723.244	22660.013	28187.385
80 (V 104, E 188)	63976.028	50554.126	87143.914	142065.252	188122.115	11328.696	18706.904	29443.708	34815.513
85 (V 109, E 197)	72182.010	59263.122	91835.091	160902.812	205891.205	15249.955	24935.557	35457.230	47366.788
90 (V 116, E 216)	85587.836	63851.610	108415.913	185825.652	235015.180	19538.795	32348.273	44549.511	59173.925
95 (V 129, E 228)	100743.964	72034.815	122093.914	215272.079	265677.361	21540.249	35777.929	58275.126	68459.703
100 (V 138, E 240)	111508.028	80984.955	150741.112	237898.368	292865.576	24218.814	39871.434	64724.402	71991.024
125 (V 181, E 272)	148264.954	121366.312	213971.623	320389.406	413718.644	26698.557	42953.358	69426.547	78252.763
150 (V 221, E 311)	190966.268	157109.419	257139.614	423743.096	555111.685	29030.825	45955.278	79101.974	88035.358
175 (V 270, E 378)	227063.742	200048.844	319735.271	541397.301	726299.117	32310.543	50517.158	82547.392	91013.790
200 (V 323, E 435)	278369.056	247923.446	411086.163	622096.420	840506.466	34553.774	55279.044	92900.174	98442.069

Appendix B: List of publications

Journal Papers

- Ghanem, M.C.; Chen, T.M.; Nepomuceno, E.G. Hierarchical Reinforcement Learning for Efficient and Effective Automated Penetration Testing of Large Networks. Journal of Intelligent Information Systems. 2022. https://doi.org/10.21203/rs.3.rs-1686285/v1
- Ghanem, M.C.; Chen, T.M. Reinforcement Learning for Efficient Network Penetration Testing. *Information.* 2020, *11*, 6. https://doi.org/10.3390/info11010006

Conference Papers

- Ghanem, M.C.; Chen, T.M. Reinforcement Learning for Intelligent Penetration Testing. Second World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), 2018, pp. 185-192, doi: 10.1109/WorldS4.2018.8611595.