National College of Ireland

Toward Automated Penetration Testing Intelligently with Reinforcement Learning

MSc Research Project
Cybersecurity

Kar Chun Goh
Student ID: 20102062

School of Computing
National College of Ireland

Supervisor: Niall Heffernan

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Kar Chun Goh
Student ID:	20102062
Programme:	Cybersecurity
Year:	2021
Module:	MSc Research Project
Supervisor:	Niall Heffernan
Submission Due Date:	16/8/2021
Project Title:	Toward Automated Penetration Testing Intelligently with Reinforcement Learning
Word Count:	6088
Page Count:	21

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Goh Kar Chun
Date:	16th August 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Toward Automated Penetration Testing Intelligently with Reinforcement Learning

Kar Chun Goh
20102062
MSc Research Project in Cybersecurity

Abstract

As the world was moving into the generation of Artificial Intelligent (AI), many sectors utilise AI for various types of tasks. In this research, an intelligent automated penetration testing system is introduced in this thesis. The research is about implementing a machine learning technique called reinforcement learning to predict the use of Metasploit Framework's module and achieve the best performance and result while conducting automated penetration testing with the Metasploit framework. In this research, two learning algorithms have been implemented, Q-learning and Deep Q-learning. In this research, Q-learning has achieved a notable result and also discovered a flaw of the purposed method in this research.

1 Introduction

Automation is a goal of daily production in every industry, information security or cybersecurity industry has no exception. Although, automated security tools was not something new in the industry. For instance, the Intrusion Detection System/Intrusion Prevention System (IDS/IPS) is one of the famous instances, it automated filtering or blocking network packets by a set of rules and machine learning techniques. Although, the industry of penetration testing does not has many such toolkits. Mostly are semi-automated penetration toolkit will require a high-level interaction. A few famous instances, such as Metasploit Framework, Nmap, Brupsuite and OWASP ZAP.

Cybersecurity becoming one of the hottest fields in the industry of Information Technology as the pandemic brings most of the services and products online. The demand for information security is much higher before the pandemic. However, the information security expert is still not enough to fulfil the requirement of the information security industry, especially the expert in the field of penetration testing. According to the Varonic Statistics and Trends for 2021 by ROB SOBERS (2021), 68 per cent of business feels their business cybersecurity risk are increasing; 36 billion records exposed in the first half of 2020; 45 per cent of breach featured by hacking. In addition, the average IBM spent 3.86 million on data breaches; a data breach requires an average of 207 days to identify a data breach; 280 days from identifying to containment. FBI reported, 300 per cent of cybercrime increase since pandemic and worldwide cybercrime will hit 6 trillion dollars annually in 2021.

As above paragraph pointed out the importance of cybersecurity mean to an organization. However, not every organization has the resource to spend on security. This research is investigating the potential use of machine learning in offensive security for lowering the cost of cybersecurity and increase the security level for organizations. This research proposes a methodology that utilizes the Metasploit framework to implement an automated penetration testing system. On top of the automated penetration testing system, a sub-branch of machine learning techniques is implemented for making the decision while the automated penetration testing program is running. The decision made by the reinforcement learning agent mainly predicts the best approach which is the Metasploit Framword's modules to exploit the target system. The reinforcement learning agent will require a certain level of training, before making predictions. However, the primary objective of this research is to investigate the potential use of machine learning in offensive security and evaluate it. The paper is organised by the literature review, methodology description, implementation of the proposed method and evaluation of the methodology.

2 Related Work

2.1 Offensive Security and Penetration testing

The research did by Wigmore (n.d.) describe the conventional information security can be considered defensive security. It usually statically discover software vulnerability and patch new version of the software. In contrast, offensive security typically simulates as an offender to disable or disrupt the operation of a system in order to discover the vulnerability of a system. Penetration testing is one of the offensive information security techniques used to discover potentially exploitable vulnerabilities of a system in an organisation. Penetration testing is a proactive method to secure a system. Typically, penetration testing conduct through simulate the attacker and try to exploit the system. Along with the penetration testing, the penetration tester will discover the vulnerabilities of the system and submit a report to other security teams for design and implement mitigation strategies. A penetration testing life-cycle can be defined into four phases , which reconnaissance (information gathering), vulnerabilities discover, exploitation and post-exploitation, stated by GoCertify (2020) (CEHv9), Abu-Dabaseh and Alshammari (2018) and Kennedy (2011), Yaroslav Stefinko, Andrian Piskozub (2016). Some literature defining the penetration testing life-cycle into more specific by adding the phase of pre-engagement, reporting, access maintaining, fingerprint covering or threading modelling.

The research did by Abu-Dabaseh and Alshammari (2018) summarised the comparison of automated penetration testing and manual penetration testing. The research stated attributes of automated penetration testing such as lower cost, easily adaptation to repetitive tasks, faster performance and standard process. Beside, conducting manual penetration testing is time and money consuming, as well as training an expert of a penetration tester. From the research can conclude the necessity and importance of automated penetration testing. However, there are still very few fully automated penetration testing tools in the current market and industry. The research did by Ghanem and Chen (2020) pointed out the recent automated penetration tools or programs are more automated vulnerabilities discovery rather than automated exploitation. For instance, several reputable software such as an open-source penetration testing software OpenVas or Tenable

commercial product Nessus. In addition, those products are covering certain purposes and tasks or single purpose and task in the penetration testing life cycle. Because the automation process of penetration testing hardly adapts to the various circumstance and uncertainty of the system environment.

2.2 Machine Learning and Reinforcement Learning

Machine Learning (ML) is one of the branches of Artificial Intelligent (AI), the main purpose of machine learning allows a computer to make predictions and decisions base on suitable data and algorithms. Machine learning has three main branches are Supervised Learning, Unsupervised Learning and Reinforcement Learning. Each of the branches has its advantage in solving various circumstances. Supervised learning use labelled data to train and mapping by algorithms to make predictions; unsupervised learning learns structure or distribution of input data only, mentioned by Dalton (2017).

According to the research of Jeerige et al. (2019), reinforcement learning has been existing in the industry of machine learning, it is also a subsection of machine learning. Reinforcement learning is based on the mathematical algorithm called Markov Decision Process (MDP). Markov Decision Process is defined by a tuple $\langle S, A, R, T, \gamma \rangle$. Which a set of state S; a set of action A; a reward function R; a transition function T; and discount factor γ stated by Hester et al. (2018). Reinforcement learning is based on a concept about machine study and learning through interaction with the environment and feedback from the environment. Several differences between machine learning and reinforcement learning. For instance, the research published by the Abhishek and Biswas (2012) describe reinforcement learning does not train by data; reinforcement learning does not interact with the dataset but the environment; reinforcement learning is based on the environment and the parameter of the environment; the objective of reinforcement learning is based on goal; reinforcement learning obtains reward from the environment. Based on the attributes of reinforcement learning, it can tell that reinforcement learning is suitable to interact with an unknown and complex environment. Moreover, in the case of predicting without a dataset, Reinforcement learning has its advantage compared with other types of machine learning techniques.

After concluded several literature, such as Morales and Zaragoza (2011); Jeerige et al. (2019); Dalton (2017); Lavet (2018); Tan and Karakose (2020), reinforcement learning has five main components which Agent, Environment, State, Action and Reward. An agent is an entity that can explore the environment and interact with the environment. An agent learns from the feedback of the environment without any supervision. An agent can be considered as the learning program itself as well. An environment is a situation that an agent could interact with and explore. For instance, if a board game implemented reinforcement learning. The board game would be the environment of the reinforcement learning system. Action is the movement of the agent interact with the environment. A state is a stage or a place that where the agent is present in the current environment. A reward is a feedback from an environment to evaluate the action of the agent. There are two primary two types of reward, positive reward and negative reward system. A positive reward reinforcement learning system is about giving the reward to encourage the system behaviours to happen again. A negative reward-based reinforcement learning is the opposite of a positive reward system. The concept of the negative reward is to

discourage the system try not to trigger the event anymore by giving a punishment. The ultimate goal of the agent will try to accumulate rewards as much as possible.

The workflow of the standard reinforcement learning is as simple as a continuous loop and passing the data between the agent and environment. As the agent start, the environment passing a state to the agent of the reinforcement learning. At each step loop, the agent gives an action that interacting with the environment. Then, the environment returns with the updated state and reward for the action provided by the agent, the principal is giving the reward and punishment based on the action for the desired outcome and result through the algorithm.

In the current research of reinforcement learning, reinforcement learning has three primary approaches, which include model-based reinforcement learning, policy-based reinforcement learning and value-based reinforcement learning, mentioned by an article by Lavet (2018). As the name implied, the value-based reinforcement learning approach tries to build the most optimal value function for the system.

The simplest and most popular value-based algorithm is Q-learning. Q-learning is an off-policy and off-model reinforcement learning algorithm, it keeps lookup the Q-table for making the prediction. The approach can be setting up an initial Q-table, then enter a loop of select action, measure reward and update Q-table. The Q-learning is based on the Bellman equation (Bellman and Dreyus 1962) as well, mentioned by Lavet (2018). Besides, according to Pandey and Pandey (2010) a Q-value could be as simple as $Q(s, a)$ in the Q-table, which s represent a particular state of the environment and a represent a particular action. In the Q-learning, the agent will experience a sequence of states to update the Q value. The research did by Watkins and Dayan (1992) clearly describe the workflow of the Q-learning, the process starts from observing the state s from the environment; select and perform the action a; observing the next state s'; receiving the reward r; update the Q-learning factor of the learning factor. Temporal difference learning is an algorithm that looks at the differentiation of the current estimate state and action pair and the discounted value of the next state and the reward. A realistic problem (such as the environment of penetration testing), the agent does not always have the perfect knowledge about the environment. Hence, the following mathematics equation is the example of the Q-learning algorithm and the Q-learning with temporal different learning that many researchers are using such as Pandey and Pandey (2010); Rodrigues and Vieira (2020). The equation includes the standard Q-learning and component of the Q-learning function. The equation clearly describes how the Q-value has been calculated and updated.

$$\underbrace{\text{New}Q(s,a)}_{\text{New Q-Value}} = Q(s,a) + \underbrace{\alpha}_{\substack{\text{New Q-Value}}}[\underbrace{R(s,a)}_{\text{Reward}} + \underbrace{\gamma}_{\substack{\text{Discount rate}}} \overbrace{\max Q'(s',a')}^{\substack{\text{Maximum predicted reward, given} \\ \text{new state and all possible actions}}} - Q(s,a)] \tag{1}$$

In short, Q-learning is relying on a Q-table with the action and state of Q-value. The agent keeps lookup the Q-table, selects the appropriate action, then update the Q-value. The standard Tabular Q-learning works great if the environment has small scale action and state space. However, the limitation of standard Q-learning, became less useful if facing large state and action space,this drawback was point-out in the research did by Ge et al. (2019).

Another Q-learning approach is Deep Q-learning, which was first introduced in 2015 by Mnih et al. Deep Q-learning using an artificial neural network instead of the Q-table. An artificial neural network or neural network was inspired by the neural network of the animal brain. Deep Q-learning builds a mapping network by the state and estimates the Q-value for every single action through the neural network. A research Ge et al. (2019) describe the advantage and disadvantage of deep Q-learning. First, it is able to extract and processing high dimensional data, which mean it is effective while processing large state space problem. However, using a neural network to approximate the Q function, the Q-learning algorithm is not stable. The strategies to resolve such problems are experience replay and the target network.

In research did by Long and He (2020) describe deep Q-learning as Equation 2. In the process of deep Q-learning, s_i represent the current state of the environment, a_i represent the selected action by the agent, θ represents the parameter of neural network. Unlike Q-learning, Deep Q-learning choosing action through the Neural Network, which calculate the Q-value for every single action. For instance, if an environment gave the state to a deep Q-learning agent with four actions (a_0, a_1, a_2, a_3), then the deep neural network will give each action with Q-value and select the most optimal action. As mentioned in the previous section, to solve the unstable of the neural network, a target network has been introduced. The research did by Rodrigues and Vieira (2020) describe the Target Network as Equation 3. The $Q^*(s, a_i)$ is replaced by the Bellman equation where the Target network T. T is the sum of the reward r and the maximum discounted Q-value obtain through the T-network with the parameters θ_f fixed in a previous time-step. In another research did by Mnih et al. (2013) stated that Experience replay is another methodology of solving the unstable of the deep neural network. Experience replay allows the reinforcement learning agent to remember the experience of the agent and store it in the sample pool. The experience replay is able to reuse those data samples to stabilise the deep neural network. The research defines the loss function for update the sample pool in Equation 4.

$$Q^*(s, a_i) \approx Q(s, a_i, \theta) \tag{2}$$

$$T = r + \gamma \max_{a'} Q(s',\ a',\ \theta_f) \tag{3}$$

$$L_i(\theta_i) = E[(r + \gamma \max_{a'} \hat{Q}(s', a'; \theta_i{}^-) - Q(s, a; \theta_i))^2] \tag{4}$$

Besides, as reinforcement learning distinct from other machine learning approach (supervised and unsupervised) is because reinforcement learning agent needs to explore the environment and try new action. Exploration and Exploitation is a concept that balances between choosing the highest possibility of the action and explores new action, depend on the scenario the solution may variant state by Leslie Pack Kaelbling, Andrew W. Moore (1996).

2.3 Related Research

Along with the literature review, Artificial Intelligent (AI) has been deployed and implemented in many industries. For instance, a research Rodrigues and Vieira (2020) utilise Reinforcement learning for self-driving; and another research Long and He (2020) utilise reinforcement learning for robot path planning. As well as the cybersecurity industry, for instance, the researchers of a company BitDefender and Technical University of Cluj-Napoca Valea and Oprisa (2020) published a paper about the automation of penetration testing using the Metasploit Framework. The paper is about utilising machine learning to determine vulnerability and choose the best approach to exploit the system based on the data of system characteristics. The research used machine data gathering from the Hack The Box Platform. They used the decision tree as the machine learning model to train data. As result, the accuracy was 33.3 per cent. Another paper by Pandelea and Chiroiu (2019) describes using the matrix collect from the wearable device to crack the password. The guessing is targeting the pin of the pad-based device. However, the result is not high. The result gets 20 per cent from 310 pins with a 1240 digit set. Besides, the research only using a classification machine learning model. If the research has a larger sample with other machine learning models, the result might be better.

There are few papers and resources that investigated the possibility of using reinforcement learning in offensive security. Researchers of Microsoft and Technion Godefroid et al. (2017) have published a paper about using machine learning to generate a fuzzing list to test the software. The paper describes using a learn and fuzz algorithm to target the software. The paper also describes using that algorithm (recurrent neural network (RNN)) has found a stack-overflow bug in the Edge PDF parser. Two researchers of the University of Oslo, Norway publish their paper Erdodi and Zennaro (2020) about the approach of utilising reinforcement learning for Web hacking at the Capture The Flag (CTF) platform. The paper formalises the CTF as a game, then formalise the game for reinforcement learning. Besides, the paper defines seven layers of web models that agent can interact with. The layers include link layer, hidden link layer, dynamic content layer, web method layer, HTTP header layer, server structure layer, server modification layer. The agent has a certain purpose and is able to conduct action at each layer. However, the paper only defines their approach and modelling it without implementation. Hence, the performance of the approach is still unknown.

3 Methodology

For understanding the proposed methodology of this research from the big picture, the phases of penetration testing include information gathering, vulnerabilities discovery, exploitation and post-exploitation. This research is covering the phase of vulnerabilities discovery and exploitation phase. However, the phase of information gathering will not cover in this research. Hence the information gathering is outsourced to the Nmap (information of Nmap at appendix A.1). The strategy can be considered as utilise Nmap to gathering information about the target system, then send the information to the reinforcement learning agent to make the best prediction to exploit the system. The following step shows the ideal workflow of this methodology.

1. Accepting the parameters of the target system from the user, such as the IP address of the target system.
2. Nmap gathering target system's information, and save it in the reinforcement learning environment.
3. Reinforcement learning environment processes the information, create states for the target system and load Q-table (or other components).
4. Reinforcement learning environment creates an action set based on the current state of the reinforcement learning environment, and send it to the reinforcement learning agent.
5. Depending on the exploration and exploitation of configuration. If the reinforcement learning agent chooses exploration, the reinforcement learning agent will randomly choose an action from the action set. Else, the reinforcement learning agent chooses the best action base on the state and Q-table.
6. Reinforcement learning environment takes the action chosen by the reinforcement learning agent and sends it to Metasploit for execution.
7. The Metasploit execute the action given by the reinforcement learning environment and interact with the target system.
8. Metasploit updates the status of the target machine.
9. Reinforcement learning environment receives the feedback from Metasploit, and determine the reward based on the feedback from Metasploit.
10. Reinforcement learning environment returns the reward, state and action set to the reinforcement learning agent.
11. Reinforcement learning agent updates the Q-value (depending on the algorithm) base on the feedback from the reinforcement environment.
12. Loop steps fourth to step eleventh until the finish of testing or training.

The research will implement two reinforcement learning algorithms which Q-learning and Deep Q-learning. The algorithm of the Q-learning is mainly relying on the Q-table. The Q-table will store Q-value, which each state and action pair. At each time loop, the result and performance of the action will update the Q-table and Q-value through the learning algorithm. Therefore the Q-learning algorithm consists of the following algorithms 1. The Deep Q-learning consist of the following algorithm 2. As mentioned in the section 2.2, the Deep Q-learning is relying on the Neural network θ with the experience replay D. Both the learning algorithm require certain training before conduct the actual testing. The meaning of training in this research is about letting the agent experiencing each action and state of the environment.

Algorithm 1 Q-learning: Learning function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$

Require:

Sates $\mathcal{S} = \{1, \ldots, n_x\}$
Actions $\mathcal{A} = \{1, \ldots, n_a\}$, $A : \mathcal{S} \Rightarrow \mathcal{A}$
Reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
Black-box (probabilistic) transition function $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$
Learning rate $\alpha \in [0, 1]$, typically $\alpha = 0.1$
Discounting factor $\gamma \in [0, 1]$
procedure Q-Learning($\mathcal{S}$, A, R, T, α, γ)
    Initialize $Q - table : \mathcal{S} \times \mathcal{A}$
    while Q is not converged **do**
        Start in state $s \in \mathcal{S}$
        while s is not terminal **do**
            Calculate π according to Q and exploration strategy (e.g. $\pi(x) \leftarrow_a Q(x, a)$)
            $a \leftarrow \pi(s)$ ▷ Receive the state
            $r \leftarrow R(s, a)$ ▷ Receive the reward
            $s' \leftarrow T(s, a)$ ▷ Receive the new state
            $Q(s, a) \leftarrow \cdot Q(s, a)$ ▷ Learning algorithms with no next state
        end while
    end whilereturn Q
end procedure

Algorithm 2 Deep Q-learning with experience replay

Initialise replay memory D to capacity N
Initialise action-value function Q with random weights θ
Initialise target action-value function Q with weights $\theta^{-} = \theta$
procedure DEEP Q-LEARNING($\mathcal{S}$, A, R, θ)
    Initialise sequence $s_1 = \{x_1\}$ and prepossessed sequence $\phi_1 = \phi(s_1)$
    while True **do**
        With probability ε select a random action a_t
        Otherwise select $a_t = argmax_a Q(\phi(s_t), a; \theta)$
        Execute action a_i in emulator and observe reward r_t and image $x_t + 1$
        Set $s_{t+1} = s_a, a_t, x_t + 1$ and preprocess $\phi_{t+1} = \phi(s_t + 1)$
        Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
        Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
        Update random weights θ by the random sample minibatch $(\phi_j, a_j, r_j$
    end while
end procedure

4 Design Specification

4.1 Architecture

Figure 1 is a high level abstract of the architecture of the proposed method in this research. As the figure can tell, the left part of the figure is the components of reinforcement learning. As explained in the section 3 the program will utilise Nmap to gathering the information about the target system and save the information in the reinforcement learning environment which is the "Metasploit Environment" in the figure. Once the Nmap gathered enough information from the target machine, the Metasploit Environment will send the target machine's information as states (which the services on the target machine), action set (which Metasploit's module) and reward. The reinforcement learning agent process those state and action set through the chosen learning algorithms. Then return an action to the reinforcement learning environment. After that, the reinforcement learning passes the action to the Metasploit console for execution.

As mentioned in the section of 2.2, a reinforcement learning environment is an environment that the agent can interact with. Hence the Target Machine is the reinforcement learning environment, and the Metasploit Console and Metasploit Environment can be considered as a bridge between the agent and the target machine. Based on the nature of the architecture of this system, the definition of reinforcement learning environment includes the components of the Metasploit Environment, Metasploit Console, and the Target Machine.

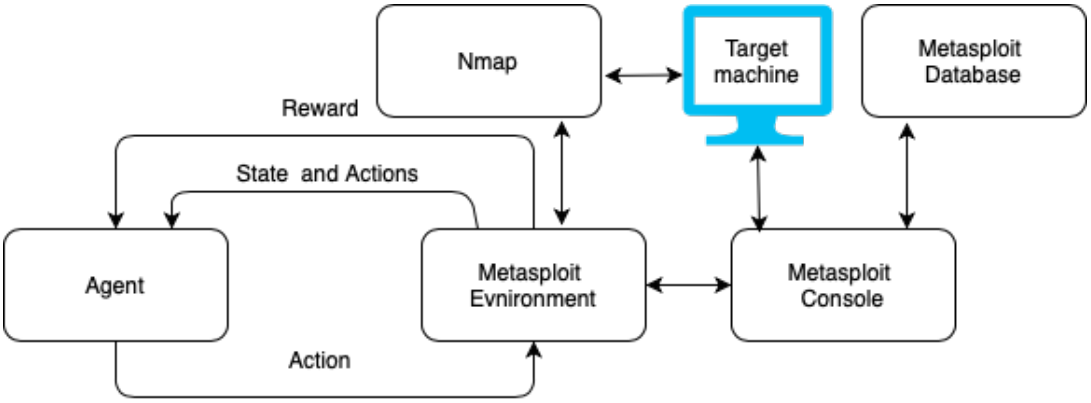

Figure 1: Conceptual of reinforcement learning implementation

4.2 Environment

This research does not have any open-source reinforcement learning environment on the internet, because of its uniqueness. Hence, this research will require to develops a reinforcement learning environment that is suitable for this project. However, the reinforcement learning environment is more like an adaptor for connecting to the Metasploit Framework and the Nmap. Through the Metasploit Framework and Nmap, the reinforcement learning agent become able to interact with the real environment, which is the target machine in this research.

The research develops two reinforcement learning environments. One for the purpose of training the reinforcement learning agent, and another for the purpose of conduct actual penetration testing. Since the workflow of the training environment and testing environment has different. The training and actual testing environment workflow show below:

Workflow of Training Environment:

1. Gathering necessary information from the target system.
2. Transform information into state set.
3. Entry Loop:
   (a) Randomly choose a state from state set.
   (b) Create action set base on current state.
   (c) Reduce action set to maximum twenty-five of action
   (d) Execute action chosen action
      i. Fill requirement.
      ii. Load action payload
      iii. Brute force payload
   (e) Remove action from action set
   (f) Check is the action success to exploit target

(g) If success, enter new state, and new load action set

(h) Return action set, state, reward

4. Loop desire round from third step for the purpose of training

Workflow of Testing Environment:

1. Gathering necessary information from the target system.
2. Transform information into state set.
3. Entry Loop:
   (a) Randomly choose a state from state set.
   (b) Create action set for current state.
   (c) Reduce action set to maximum twenty-five of action
   (d) Execute action chosen action
      i. Fill requirement.
      ii. Load action payload
      iii. Brute force payload
   (e) Record number of time that action perform, set maximum of testing (five in this research)
   (f) Check is action success to exploit target
   (g) If success, enter new state, and new load action set
   (h) Return action set, state, reward
4. Show all the modules that able to exploit the target machine.

The main reason for designing two workflows is because the purpose of training is to let the agent experiencing every possible action for a particular state; the purpose of testing is to choose the action that has the best experience in the particular state. However, during the testing, if the particular state has no corresponding exploit module to exploit the service, the agent might enter an infinite loop caused by the natural design of this system.

4.3 Q-learning

Q-table could be one of the core components that make Q-learning workable in reinforcement learning. In this research, the state and action are used to form the Q-table, as Table 1 shows. All the Matesploit's modules will be defined as the action and store in the Q-table row. The attributes of the target machine service are defined as the state in this research. The state includes the target machine services attribute of the port number, services name, product, and product version. Each of the states are store as a string

since the Q-table not able to store high dimensional values in a single column. However, Q-learning does not good at handling large action sets and state sets.

The action set has modules total of 3875 (2142 exploit modules, 1140 auxiliaries and 592 payloads). But the number of the state remains unknown. So, for each step loop, the reinforcement learning agent will require to check the existence of a state in the Q-table. If the state does not exist in the Q-table, the reinforcement learning agent will need to put it in the Q-table.

Action/State	**A(0)**	**A(1)**	**A(...)**	**A(n)**
S(0)	Q(A0, S0)	Q(A0, S1)	Q(A0, S..)	Q(A0, Sn)
S(1)	Q(A1, S0)	Q(A1, S1)	Q(A1, S..)	Q(A1, Sn)
S(...)	Q(..., S0)	Q(..., S1)	Q(..., ...)	Q(A.., Sn)
S(n)	Q(An, S0)	Q(An, S1)	Q(An, S..)	Q(An, Sn)

Table 1: Q table

4.4 UML Diagram

Figure 2 is the conceptual UML diagram of this research. As Figure 2 show, the Agent and the Metasploit Environment is the main class of this system in this research. The Agent and the Metasploit Environment component also represent the Agent and Metasploit Environment in the Figure 1. In this research, the Agent class has two learning function which the Q-learning and the Deep Q-learning. Both learning function will have the function such as store and load, in the case of Q-learning the store and load will be the Q-table; Deep Q-learning will be the model of neural network. In addition, the learning function will have the function of choosing the action, storing the experience, training the experience.

As mentioned in the previous section, the role of Metasploit Environment represents a bridge to connect the Metasploit Console. Therefore, the class require a client function to establish the connection to the Metasploit Console. The client function also extends to others function, such as loading the Metasploit module, execute the module and management of the session. In addition, the Metasploit Environment will require to handle the communication with the agent as well. In Figure 2, the function of "Loop Step" is used to handling and manage the communication with the Agent, such as updating the State, Action, target machine information and reward function. Besides, this class is responsible for handling the information gathering as well. So the function used to connect with Nmap is necessary as well.

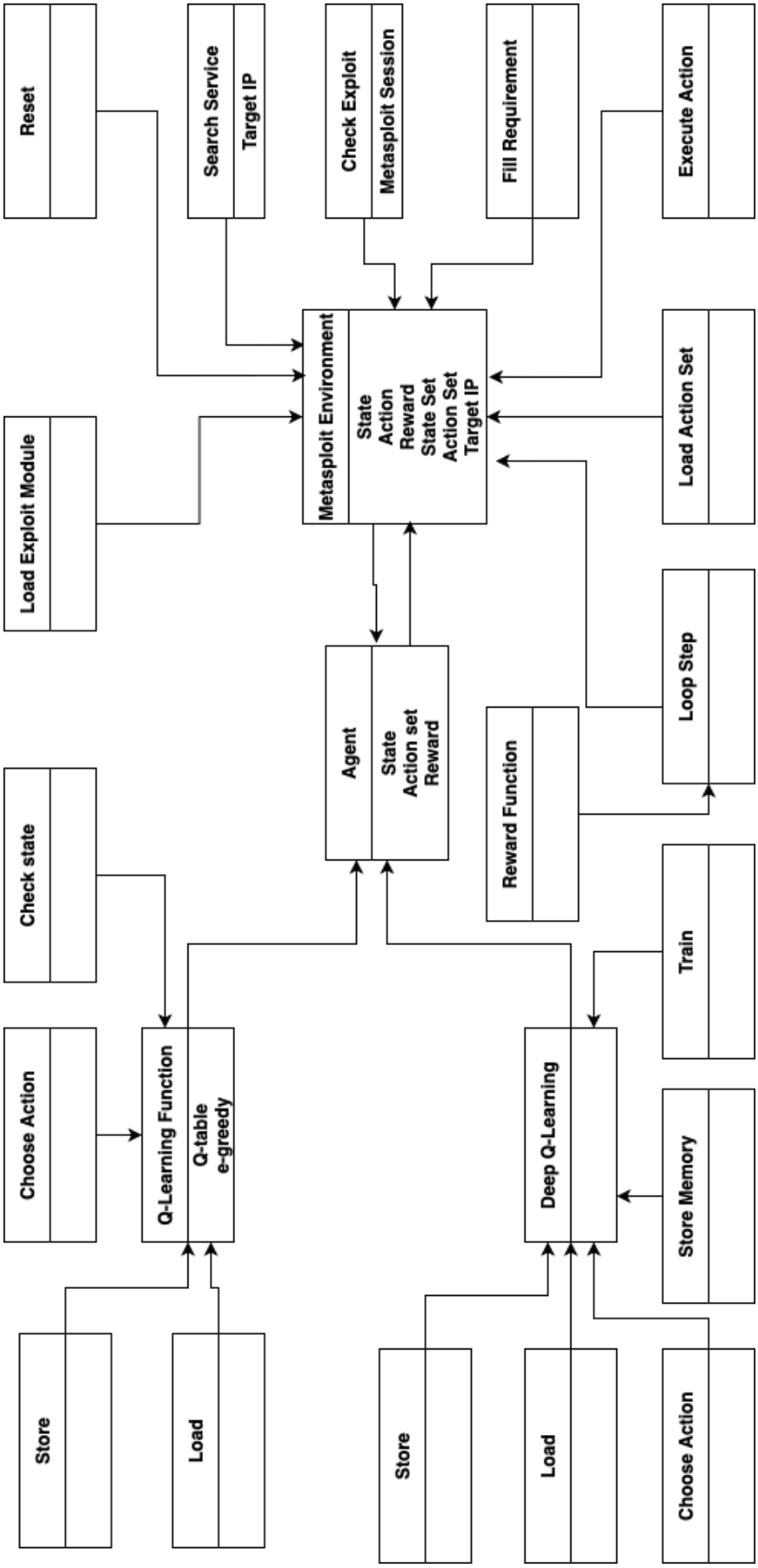

Figure 2: System UML

5 Implementation

The technology involves in this research include Anaconda, Python, TensorFlow, Pandas, Metaasplotable2, Pymetasploit3, Nmap and Python-Nmap3, Docker, Metasploit Framework. The detail and functions of the technologies at the appendix A.1. From the technology can tell that the research is conduct in the Anaconda environment with Python. As mentioned in the section of 4.2, the Internet does not have an environment suitable for this research. Therefore, the first step is the development of the Reinforcement learning Environment. Reinforcement learning relies on multiple libraries. First, the Nmap3 is a library for python3 interacting with the Nmap application, it is used to gathering information from the target machine. Then, create the function that communicates with the Metasploit console, this function would require the library of PyMetasploit3. Then, a function that used to manage the communication with the agent. The sub-function refer to section 4.4. As mentioned in the previous section, this research will not cover the phase of information gathering. Hence, the Nmap became the role of gathering information in this research. The primary objective of Nmap is to scan the service detail of the target machine system such as the port number, service type, service name and service version.

5.1 Q-learning and Deep Q-learning

The next step would develop the reinforcement learning agent, which the Q-learning and Deep Q-learning. Since the core component of the Q-learning is the Q-table and the Q-learning algorithm. The Q-learning algorithm can refer to the Algorithm 1. The Q-table is using the Padas supported table for storing the Q-value $Q(S, A)$. Besides, the Q-table is storing in a PKI format. In addition, the table will check the existence of a state, when encountering a new state s. However, there is one difference between the ideal algorithms and implementation, which is the reinforcement learning environment and learning algorithm have no transitioning function and next state s' in the implementation.

The implementation of Deep Q-learning is similar to Q-learning, it replaces the Q-table with the neural network. In this research, the implementation of the neural network is referring to aamrani-dev Github repository with the modification to suit this research, such as remove the state transitioning.

5.2 States and Actions

States and actions are the core data transfer between the Reinforcement learning agent and environment. Each learning algorithm and handling states and actions differently. Since the Q-learning is not great at handling high dimensional state, hence the state is concatenated as a single string. For some reason, the neural network can only process numeric data, hence the state has been converted into numeric hash data for Deep Q-learning.

5.3 Experiment Environment

After necessary components (Reinforcement learning environment and reinforcement learning agent) have been developed. The research entering into the experience phase. The

experience environment is fully deployed and conduct in the container under Docker containerisation. The purpose of deploying the research in containerisation is to achieve a lightweight and platform free system. The experiment environment consists of three containers, which a container for reinforcement learning; a container for Metasploit Framework; a container for the target system (it could be a vulnerable machine for training or the machine for actual testing).

To deploy the Metasploit Framework, two options can be considered. First, deploy the Metasploit framework by the container image provided by the Rapid 7; second, deploy Kali Linux images and install the Metasploit Framework. This research is using the second approach. Metasploit provides two options for remote access and interaction with Metasploit, which are Msfrpcd a daemon server for Metasploit framework and Msgrpc a plugin for Msfconsole. This research deploys the Kali Linux container images and runs the Msfrpcd. Since Msgrpc only allows the loopback IP address to connect the Msgrpc.

The second container is the container for deploy anaconda and reinforcement learning components. This research chooses to deploy the Mini-conda a light version of anaconda. In addition, the necessary toolkits that need to built reinforcement learning such as Python3, Numpy, Tensorflow will be installed in the anaconda environment. Besides, the Nmap will install in this anaconda environment as well, for reinforcement learning to access and interact with it.

To easier manage all the containers in this research, a sub-net has been created. All the containers will be deployed under this sub-net and assign a specific IP address and name for identity and management as wellåå.

5.4 Training and testing

As mentioned multiple times throughout the thesis, training is a requirement for reinforcement learning. Every agent and learning algorithm requires a certain level of training. This research will utilise the Metasploitable2 for training both the Q-learning agent and the Deep Q-learning agent. The Metasploitable2 is deployed as the third container for the purpose of training and testing the performance of reinforcement learning.

In this research, two training strategies have been used to train Q-learning and Deep Q-learning. The first strategy is about brute-forcing every action and state for the agent to experience. Which mean every single state will require experience every single action. This method is implemented for Deep Q-learning. The second method is about filtering the action based on the particular state. For instance, if the state is related to SQL, it only loads the module related to SQL as an action set for the agent to experience instead of experience every single action even irrelevant modules. This method is implemented for the Q-learning agent. Both methodologies have their advantages and disadvantages.

6 Evaluation

The evaluation consists of four case studies, one without and learning algorithm, one q-learning for training, one q-learning for testing, and one Deep Q-learning for testing and training. Each case study consisted of two cases sample from the experiment. Each of the

plots has x as the number of steps or episodes; and y as the accumulated reward. In this research, every agent has starting reward of 50, when successful to exploit a service, the agent gains 100 rewards; fail to exploit minus 1 reward, and other results obtain reward 0.

6.1 Without Learning Algorithm

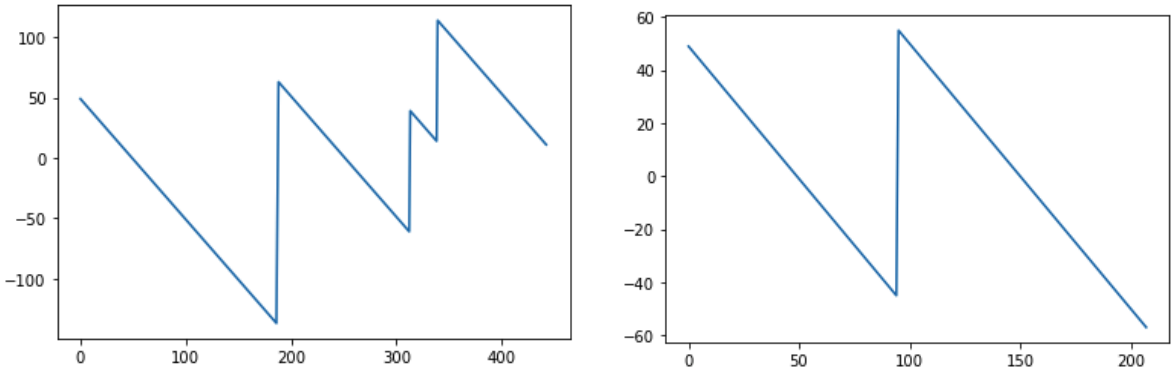

Figure 3: No Algorithms Assist

Figure 3 are experiment about automated penetration testing without any algorithm assists. The system has been set to fully random choosing the module from all the exploit modules at each step loop. In the figure can tell, average hundred steps to exploit a service. The result is seen relatively high. After investigated the case, it found out an exploit module was compromised the target machine two out of four in the left case.

6.2 Q-learning Training

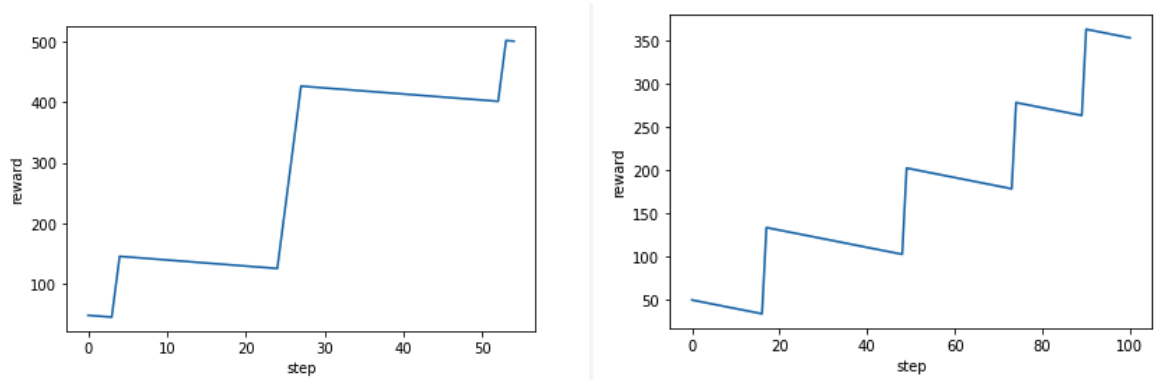

Figure 4: Q-learning Training

As mentioned in the section 5.4, the state and action pairs for the agent to experience have been filtered by the search algorithm of the Metasploit Framework. Each upslope in the figure can be considered a session been established, and updated in the Q-table. Both left and right cases can show that has 4 services have been exploited during training. As the figure can tell, most the state has experience around 25 of action until success exploits

the service. However, since the action set for each state has limited to a maximum thirty of actions. Some of the states require few more rounds to discover all the actions.

6.3 Q-learning Testing

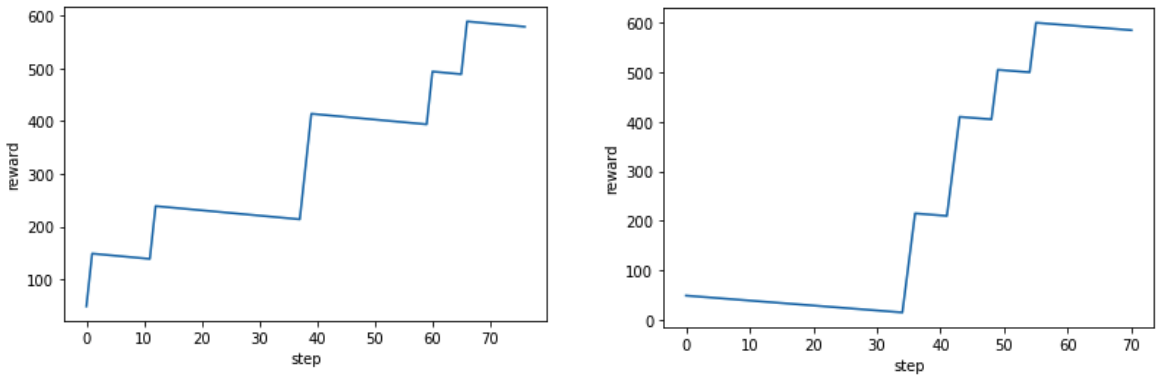

Figure 5: Q-learning Testing

As the figure 5 can tell both the case has estimate total of 70 steps to complete the testing against the Metasploitable2, and six exploits have been detected. It was not a surprising result, and consider it better than the testing without any algorithms assist. In addition, redundancies of the exploit module exist in the result. Cause by the training of Q-learning in this research.

6.4 Deep Q-learning Testing and Training

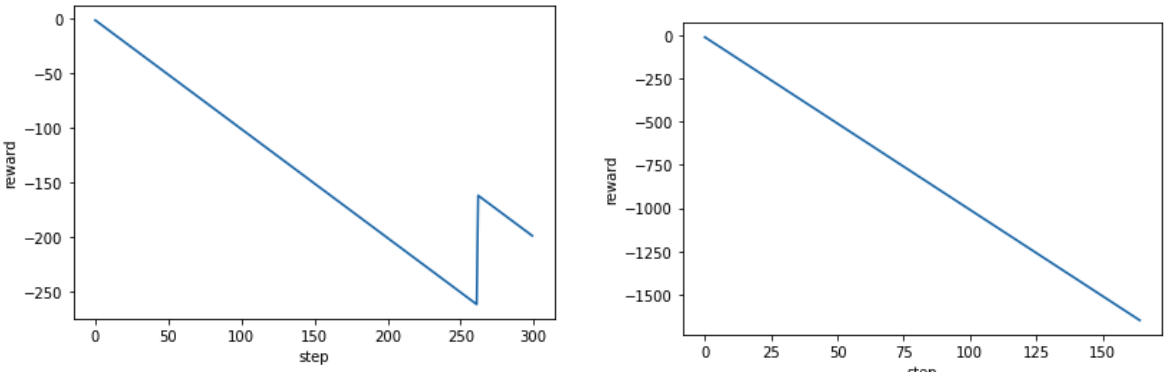

Figure 6: Deep Q-learning Training and Testing

The left case is about the result of training; the right case is about the result of testing. As both cases can tell that the Deep Q-learning's result was not great. As the figure shows, the training has three hundred steps and one exploits detected. The figure of testing has a hundred and fifty steps with no exploit detected. This mean non of the predictions are correct.

6.5 Discussion

In general, the above figures has shown that the potentiality of utilising reinforcement learning for penetration testing. The comparison of with and without the reinforcement learning algorithm can show that a huge improvement regards the time consume and performance.

However, a flaw been discovered during the experience phase. During the training of the Q-learning agent, the exploit module may map to incorrect vulnerability. Case one, the search algorithm giving the "wrong experience" to the agent. For instance, a service Linux Telnet has the keyword "Linux", the search algorithm bring all the modules related to "Linux" and Telnet modules. There is a coincidence that irrelevant modules are still able to exploit the particular service, such as Linux SQL. In another case, training of Deep Q-learning, the agent was experiencing every state for every statesåç (service), which mean it is a high potential that the mixed up with the state and particular exploit module.

Besides, Deep Q-learning is considered as the failure of implementation, as the evaluation section can tell, both the experiment result are not great. After investigation, several reasons have been summarised. First, the improper of defining state. Second, insufficient training time, since a hundred steps could take two hours of training. Third, crashed by the insufficient capacity of handling and establishing new HTTP request at the Msfrpc daemon, and lead to failure of training in a long period of time.

7 Conclusion and Future Work

After months of research, implementation of methodology and experiment, this research has achieved the objective which investigates the potentiality of using reinforcement learning for penetration testing. This research concludes that reinforcement learning has the potential to increase the performance of automated penetration testing and decrease the resource of testing. As the section of evaluation described, a notable difference between using the reinforcement learning algorithm and without using the reinforcement learning algorithm. The algorithm reduces the resource of time and increases the chance of exploitation during automated penetration testing. However, the result of this methodology proposed in this research was not fantastic and a flaw was discovered, but a simple algorithm such as Q-learning still achieve a notable result. Besides, I can assume that utilise other machine learning techniques or reinforce the framework of the methodology can achieve a better result.

The future work has few directions. First, investigate and resolve Deep Q-learning and Q-learning to this research, such as investigate and resolve the problem described above section and redefine the state for Deep Q-learning. Second, further development of reinforcement learning algorithms, such as adding the feature of auto-filling the user name and password or the feature of state transitions. Third, further investigate other machine learning and reinforcement learning algorithms option, such as implement other reinforcement learning algorithms or merging supervised learning. Fouth, define and design a training suitable for this research. Link to the code in here.

References

Abhishek, N. and Biswas, M. (2012). *Reinforcement learning with Open AI, TensorFlow and Keras Using Python*, Vol. 3.

Abu-Dabaseh, F. and Alshammari, E. (2018). Automated Penetration Testing : An Overview, pp. 121–129.

Dalton, W. (2017). *The silicon hat hacker : using reinforcement learning in hybrid warfare*, number Figure 1.

Erdodi, L. and Zennaro, F. M. (2020). The agent web model: modelling web hacking for reinforcement learning, *arXiv* .

Ge, H., Song, Y., Wu, C., Ren, J. and Tan, G. (2019). Cooperative deep q-learning with q-value transfer for multi-intersection signal control, *IEEE Access* **7**: 40797–40809.

Ghanem, M. C. and Chen, T. M. (2020). Reinforcement learning for efficient network penetration testing, *Information (Switzerland)* **11**(1): 185–192.

GoCertify (2020). *Certified Ethical Hacker (CEH)*, Sybex.
URL: *http://www.gocertify.com/certifications/ec-council/certified-ethical-hacker.html*

Godefroid, P., Peleg, H. and Singh, R. (2017). LearnFuzz: Machine Learning for Input Fuzzing, *arXiv* pp. 50–59.

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Osband, I. et al. (2018). Deep q-learning from demonstrations, *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

Jeerige, A., Bein, D. and Verma, A. (2019). Comparison of deep reinforcement learning approaches for intelligent game playing, *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0366–0371.

Kennedy, D. (2011). The Basics of Hacking and Penetration Testing, *Network Security* **2011**(12): 4.

Lavet, F. (2018). An Introduction to Deep Reinforcement Learning., *Foundations and trends in machine learning* **II**(3 - 4): 1–140.

Leslie Pack Kaelbling, Andrew W. Moore, M. L. L. (1996). Deep reinforcement learning: a survey, *Frontiers of Information Technology and Electronic Engineering* **21**(12): 238–285.

Long, Y. and He, H. (2020). Robot path planning based on deep reinforcement learning, *2020 IEEE Conference on Telecommunications, Optics and Computer Science (TOCS)*, pp. 151–154.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning, pp. 1–9.
URL: *http://arxiv.org/abs/1312.5602*

Morales, E. F. and Zaragoza, J. H. (2011). An introduction to reinforcement learning, *Decision Theory Models for Applications in Artificial Intelligence: Concepts and Solutions* pp. 63–80.

Pandelea, A. I. and Chiroiu, M. D. (2019). Password guessing using machine learning on wearables, *Proceedings - 2019 22nd International Conference on Control Systems and Computer Science, CSCS 2019* pp. 304–311.

Pandey, D. and Pandey, P. (2010). Approximate q-learning: An introduction, *2010 Second International Conference on Machine Learning and Computing*, pp. 317–320.

ROB SOBERS (2021). 134 Cybersecurity Statistics and Trends for 2021.
URL: *https://www.varonis.com/blog/cybersecurity-statistics/*

Rodrigues, P. and Vieira, S. (2020). Optimizing agent training with deep q-learning on a self-driving reinforcement learning environment, *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 745–752.

Tan, Z. and Karakose, M. (2020). Optimized deep reinforcement learning approach for dynamic system, *2020 IEEE International Symposium on Systems Engineering (ISSE)*, pp. 1–4.

Valea, O. and Oprisa, C. (2020). Towards Pentesting Automation Using the Metasploit Framework, *Proceedings - 2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing, ICCP 2020* pp. 171–178.

Watkins, C. J. and Dayan, P. (1992). Q-learning, *Machine learning* **8**(3-4): 279–292.

Wigmore, I. (n.d.). offensive-security.
URL: *https://whatis.techtarget.com/definition/offensive-security*

Yaroslav Stefinko, Andrian Piskozub, R. B. (2016). Manual and Automated Penetration Testing. Benefits and Drawbacks. Modern Tendency, **1**: 6–9.

A Appendix

A.1 Tool Description

Anaconda: Anaconda is a Python and R distribution software for scientific computing, such as machine learning application and data processing and predictive analysis. The function of anaconda is the research and development environment for this research.

Python: Python is interpreted high-level programming language, which used among data scientist. Because of the rich library related to data science, such as the pandas, Tensorflow, Scikit-learn, BeutifulSoap, etc. Hence, Python will be the primary language used to implement the research.

TensorFlow: TensorFlow is an open-source library built for Artificial Intelligent (AI) machine learning (ML). Besides, it also pre-built the component of building a deep neural network for the user.

Pandas: A library that is used to solve various machine learning tasks. It supports multi dimensional array.

Metesploit Framework: Metasploit framework is a computer security project for provide the information of security vulnerability and assist penetration testing for the penetration testers. Metasploit framework has pre-built plenty of exploit module and auxiliary module to assist penetration tester. Besides, Metasploit has an API that allows the penetration tester to interact with Msfconlose remotely. The Metasploit framework is one of the primary components in this research. It sits between the agent of the reinforcement learning and the target machine.

PyMetasploit3: Car Phone Holder is a function of Metasploit framework that allow user connect to the API of Metasploit framework and interact with the Msfconsole remotely. Python-msfrpc is a python library that used to interact with the Msfconsole for any python program.

Nmap and Nmap3: Nmap is a network scanning toolkit that used among the penetration industry. Since the research will not cover the phase of information gathering of penetration testing. Hence Nmap would be used to gathering necessary information of the target machine, Such as the service name, version and open port. Besides, Nmap does provide a library for any python program use Nmap.

Metasploitable2:

Docker: Docker is an open-source platform that allows users to separate infrastructures and applications. In this research, all the software components and applications will be deployed under docker.