**CSCE – 5552: Cybersecurity Essentials**

**Hack the Box Challenge: Vipere**

**Group-3**

The Hack the Box "Vipere" challenge involves exploiting a Python script to gain access credentials from a remote server. The report will give the progress made so far in the project, the tools and techniques used in the project, and a detailed plan of action.

**Current Progress:**

* **Program Analysis:** Understanding the Python script structure and a deep analysis of flow, its dependencies, and the major functional blocks in the script was carried out. The main constituent parts of the Python script include “SecureCommands,” a method in the script, and the “Station” method, which is a server/client interaction class.
* **Vulnerability Identification:** Vulnerabilities in the script, if any, were analyzed in potential areas such as command injection, insecure user input handling, possible unauthorized access to system resources, etc. In this, we found the `verify\_commands` method in the `SecureCommands` class is one of the potential vulnerabilities.

**Tools and Techniques Utilized:**

* **Burp Suite:** Used Burp Suite to test the server, including interception and modification of HTTP requests and responses, as well as to analyze and manipulate communication between the server and clients.
* **PDB (Python Debugger):** Utilized the Python Debugger (PDB) to debug the script to help better understand how it is executed, spot possible vulnerabilities, and troubleshoot any difficulties.
* **Nmap:** Used Nmap for network reconnaissance and discovery, looking for open ports and services on the target machine to gather important information about its network setup and possible attack points.
* **Netcat (NC):** Netcat was used to set up the listeners, monitor network traffic for intercepting, and analyze the communication between the server and our machine to execute payloads.

Used Nmap to do a scan on the target machine for possible open ports like FTP or SMB, with which we can try to gain the initial access.

A screenshot of a computer

Description automatically generated

We tried to access the target from the browser, however we can see it isn’t a normal website.

A screenshot of a computer

Description automatically generated

So, we used the burp suite to intercept the response and tried to look for possible vulnerabilities.

A screenshot of a computer

Description automatically generated

As we can observe, we have provided the input as {whoami}{get\_time} while making the request, and we received the response with those details.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

If we have a glance at the python script, the server is being hosted by python itself using sockets and there are predefined commands: whoami, get\_time, get\_version.

A screenshot of a computer program

Description automatically generated

We can see the debugger has been used, and we can use this {debug} as a command and do the remote debugging and execute the method get\_infected, through which we can gain access to the database.

A screen shot of a computer

Description automatically generated

We tried to implement our approach on the Python script using pdb; as we can see, we are adding the new command debug into the dispatcher list. This dispatcher list will execute the methods which are linked to it. So we are adding the method get\_infected to the command ‘debug’.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

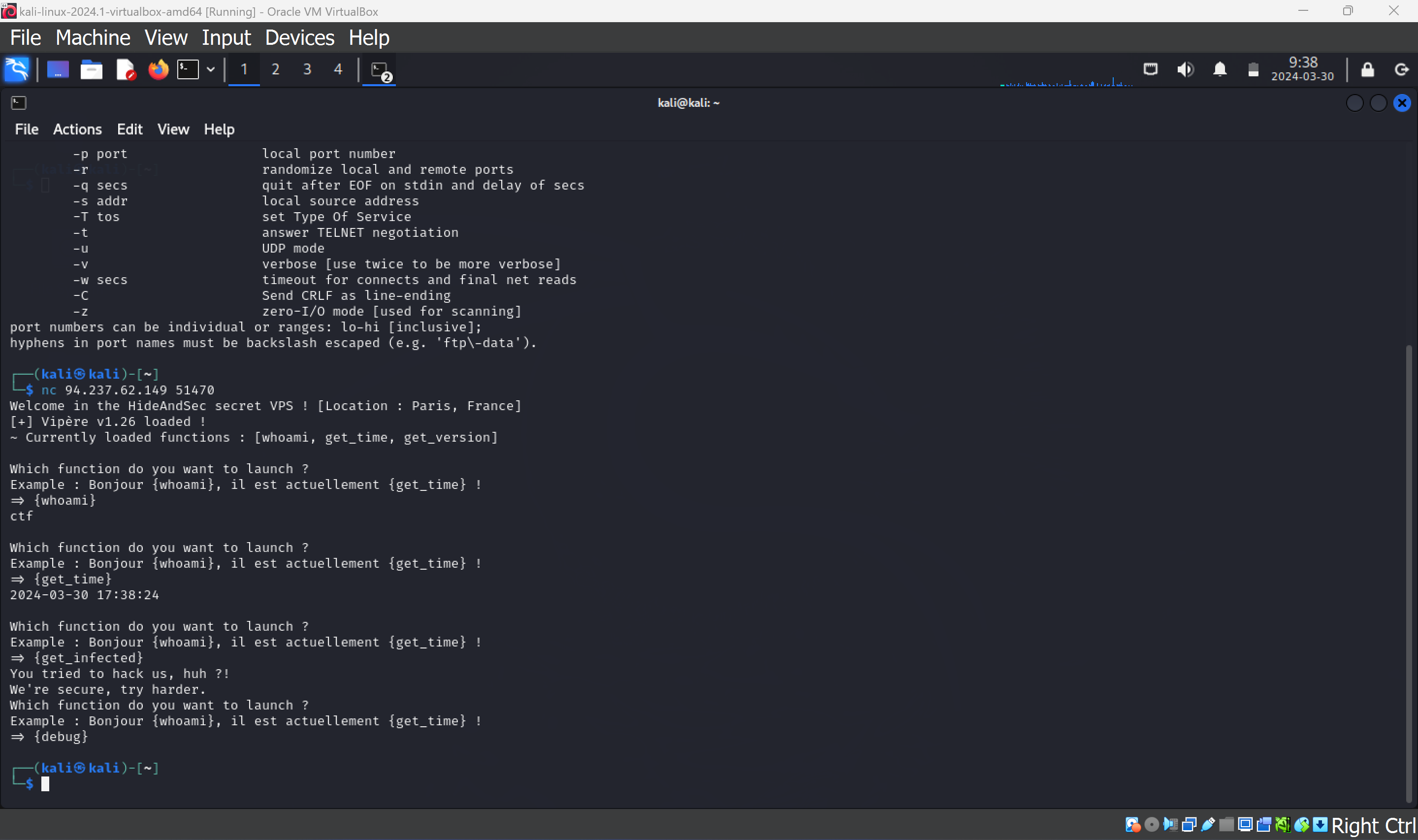
Description automatically generated

However, when we tried the {debug} command we didn’t receive any response. So we have to use other tools apart from burp suite to interact with the target system.

A screenshot of a computer

Description automatically generated

Then we used netcat, to implement this approach in a more interactive way. As we can see, we can communicate with the target. However, when we use debug command we are losing the connection.



This could be the potential rabbit hole, so we started implementing a listener on our system to implement reverse shell to gain access to the system.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Future Steps:**

**Reverse Shell Implementation:** Implement a reverse shell payload to connect the target system through our machine and gain remote access.

**Binary Analysis:** Retrieve the execution binary from the target system and analyze it using tools like Ghidra to identify vulnerabilities such as buffer overflows or other weaknesses.

**Exploit Development:** Develop and implement necessary payloads to exploit identified vulnerabilities in the binary, ensuring compatibility with the target environment and desired outcomes.

**Execution and Access:** Deploy the crafted payloads on the target system, gaining access and control over the system to retrieve the flag variable and complete the challenge.