Jonathan Ho

jlh5360@rit.edu

Team Bravo

CSEC 473.02

2235 Spring

HW4 Red Team Tool

GitHub Repository:

- https://github.com/jlh5360/CSEC-473
- https://github.com/jlh5360/CSEC-473/tree/main/HW4

HW4 Installation/Usage Instructions

1. On test1 (this Linux instance | 192.168.0.133), ensure you are the user ubuntu:

   - If not, then run this command in the terminal:

```
su ubuntu
```

2. Python3 is already installed

   - Run the "python3 --version" command to verify

3. Ensure you are in this path: /home/ubuntu/hw4

   - If not, then run this command in the terminal:

```
cd ~/hw4
```

4. After running the "ls" command, you should see these files:

```
port_knocker.py
README.txt
```

5. Test your connection/connectivity with Linux (192.168.2.73) by pinging:

```
ubuntu@test1:~/hw4$ ping 192.168.2.73
PING 192.168.2.73 (192.168.2.73) 56(84) bytes of data.
64 bytes from 192.168.2.73: icmp_seq=1 ttl=64 time=2.90 ms
```

```
64 bytes from 192.168.2.73: icmp_seq=2 ttl=64 time=0.973 ms
64 bytes from 192.168.2.73: icmp_seq=3 ttl=64 time=0.363 ms
64 bytes from 192.168.2.73: icmp_seq=4 ttl=64 time=0.436 ms
^C
--- 192.168.2.73 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3026ms
rtt min/avg/max/mdev = 0.363/1.168/2.900/1.027 ms
```

6. On target (Linux instance | 192.168.2.73), ensure you are the user ubuntu:
   - If not, then run this command in the terminal:

```
su ubuntu
```

7. Apache is already installed:
   - Run the "apachectl -v" command to verify
8. Ensure you are in this path: /var/www/html
   - If not, then run this command in the terminal:

```
cd /var/www/html
```

9. After running the "ls" command, you should see this file:

```
index.html
```

10. After catting the index.html file, you should see this result:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=devide-width, inital-scale=1.0">
        <title>Red Team Target</title>
    </head>
    <body>
        <h1>Welcome to the Red Team Target!</h1>
    </body>
```

```
</html>
```

11. Open Firefox Web Browser (or any other available web browser), go to the URL and enter "localhost" or "192.18.2.73" (which is localhost/the IPv4 address), and you should see this webpage (by simply being to "localhost", "192.18.2.73", "192.18.2.73:80", or "192.18.2.73:8080" (since I manually added that port in the /etc/apache2/ports.conf and /etc/apache2/sites-enabled/000-default.conf files):

```
Welcome to the Red Team Target!
```

12. Going back to target (this instance | the developer node/instance), open Firefox Web Browser (or any other available web browser), go to the URL and enter "192.18.2.73" (which is IPv4 address of target), and you should see this webpage:

```
Welcome to the Red Team Target!
```

13. In the terminal, go to this path if you are not already:

```
/home/ubuntu/hw4
```

13. Run the Python file "port_knocker.py" by running this command in the terminal and should see a similar result (the only difference is the date and time when the Python file ran or was executed):

```
ubuntu@test1:~/hw4$ python3 port_knocker.py
2024-02-14 20:44:40.102594  |  Port 22 is already open
2024-02-14 20:44:41.104659  |  Port 80 is already open
2024-02-14 20:44:42.107085  |  Knocking on port 443
2024-02-14 20:44:42.107085  |  Failed to connect to 192.168.2.73:443 --> [Errno 103] Software caused connection abort
2024-02-14 20:44:42.108586  |  Knocking on port 1000
2024-02-14 20:44:42.108586  |  Failed to connect to 192.168.2.73:1000 --> [Errno 103] Software caused connection abort
2024-02-14 20:44:42.110235  |  Knocking on port 2000
2024-02-14 20:44:42.110235  |  Failed to connect to 192.168.2.73:2000 --> [Errno 103] Software caused connection abort
2024-02-14 20:44:42.111095  |  Knocking on port 3000
```

```
        2024-02-14 20:44:42.111095  |  Failed to connect to 192.168.2.73:3000 --> [Errno
103] Software caused connection abort
        2024-02-14 20:44:42.111627  |  Knocking on port 4000
        2024-02-14 20:44:42.111627  |  Failed to connect to 192.168.2.73:4000 --> [Errno
103] Software caused connection abort
        2024-02-14 20:44:42.112145  |  Knocking on port 5000
        2024-02-14 20:44:42.112145  |  Failed to connect to 192.168.2.73:5000 --> [Errno
103] Software caused connection abort
        2024-02-14 20:44:42.112742  |  Knocking on port 6000
        2024-02-14 20:44:42.112742  |  Failed to connect to 192.168.2.73:6000 --> [Errno
103] Software caused connection abort
        2024-02-14 20:44:42.113215  |  Knocking on port 7000
        2024-02-14 20:44:42.113215  |  Failed to connect to 192.168.2.73:7000 --> [Errno
103] Software caused connection abort
        2024-02-14 20:44:42.113614  |  Knocking on port 8000
        2024-02-14 20:44:42.113614  |  Failed to connect to 192.168.2.73:8000 --> [Errno
103] Software caused connection abort
        2024-02-14 20:44:42.114122  |  Port 8080 is already open
        2024-02-14 20:44:43.115762  |  Knocking on port 9000
        2024-02-14 20:44:43.115762  |  Failed to connect to 192.168.2.73:9000 --> [Errno
103] Software caused connection abort
```

14. Then in the terminal, you can telnet the target IPv4 address using the ports previously used above by using this command format:

```
telnet [target IPv4 address] [port number]


ex: telnet 192.168.2.73 22
    telnet 192.168.2.73 80
    etc...
```

You should/will see the same results where only ports 22, 80, and 8080 can and are connected, but not the others ports

15. This concludes the end of port knocking where I am externally checking if ports are open and send/attempt a connection request or knocking on a port

- Port knocking is a method of externally opening ports on a firewall by generating a connection attempt on a set of closed ports

Questions

Answer the following questions about your tool. Your answers should be thorough and fully answer the question. Answers should be grammatically correct and be full sentences.

1. **(8 points) What is the goal of this tool? What purpose does it bring to the competitions?**

   The goal of the port knocking tool is to provide a means for the red team to stealthily open ports on a target system by sending a sequence of connection attempts to specific ports. This allows the red team to access services behind the target's firewall without directly triggering alerts or detections. The tool enhances the red team's capabilities by providing a covert method of gaining access to restricted services during security competitions. By automating the port knocking process, the tool improves efficiency and reduces the likelihood of detection during reconnaissance and exploitation phases.

2. **(8 points) Did other tools influence your tool? If so, what are they? If not, what was your inspiration for the tool?**

   While the port knocking tool is primarily inspired by the concept of port knocking itself and not my other red team tools, its implementation and development is influenced by various resources. Commonly used libraries and modules in Python for network communication, such as the socket library, have guided the development of the tool. Additionally, the design principles of simplicity, reliability, and stealthiness, often emphasized in red teaming scenarios, have been considered in the development of the tool's functionality and approach. Although no specific red team tools were directly referenced, the development process drew insights from existing port knocking implementations and related network security tools with these resources I referenced and utilized to understand and develop a port knocking tool following common practices:
   - https://resources.infosecinstitute.com/topics/mitre-attck/mitre-attck-port-knocking/
   - https://www.youtube.com/watch?v=IBR3oLqGBj4
   - https://github.com/sidpalas/devops-directive/tree/master/2020-10-05-port-knocking
   - https://www.tecmint.com/port-knocking-to-secure-ssh/
   - https://goteleport.com/blog/ssh-port-knocking/

- https://www.howtogeek.com/442733/how-to-use-port-knocking-on-linux-and-why-you-shouldnt/

3. **(8 points) What is the feasibility of another team member quickly learning to use or contribute to your tool? What makes it easy or difficult to learn?**

   Another team member can quickly learn to use or contribute to the port knocking tool due to several factors. Firstly, the tool is implemented in Python, a widely used and beginner-friendly programming language known for its readability and simplicity. The code is well-commented and structured, making it easy for team members to understand its logic and functionality. Additionally, the provided README.txt file that I wrote contains comprehensive installation and usage instructions, guiding users through the setup and execution process step by step. Moreover, the modular nature of the tool allows for easy customization or extension to adapt to different environments or requirements, further enhancing its usability for team members with varying levels of expertise. Overall, the combination of clear documentation, readable code, and flexibility ensures that other team members can quickly grasp and utilize the port knocking tool effectively in red teaming exercises.