CSCD58

SharkBite

The coolest network scanner of all time

Demo Link: https://www.youtube.com/watch?v=Kx1ui64Sd64

https://drive.google.com/file/d/1Sn90OB4NkY53bOvsdGixT0_GQyS6gsMO/view?usp=sharing (Alternate link. Download for good quality)

Team

Borui Li : 1003231807

Junheng (George) Wang : 1006390031

Anirudha Kanodia : 1004608392

Description And Goals

SharkBait is a multi-functional network tool that is used for network analysis and auditing. It offers a range of functionality, from host discovery and port scanning to banner grabbing, service detection and even identifying vulnerable services. The following goes over its main functionality in more detail.

Host Discovery

Most penetration tests involving networks usually start off with reconnaissance. It is often important to narrow down the targets in the network to those that are interesting and active. After identifying these nodes in the network, a penetration tester can then proceed to exploit it more actively to gain access to internal systems. Our tool can detect the hosts that are currently active in a network and reports their IP addresses as well as MAC addresses. For the scan to work, the user must be connected to the network they are scanning. Please refer to the examples section for examples.

Port Scanning

After a penetration tester has identified the targets, he/she begins to perform further active reconnaissance against the target machine. One of the major steps in this process involves determining the open ports. To assist with this process, our tool has various types of handy scans that are suited for different situations to determine the open ports on a target, which can be abused to gain access to the target. The following are the types of scans we offer:

1. **TCP SYN Scan (-sS)**: This type of scan, also sometimes known as “Stealth” Scan, does not establish a connection with the target but rather just sends the SYN packet in the 3-way handshake and awaits for the response. The ACK packet is never sent and hence a connection is not established.. This type of scan is relatively stealthy and fast.

2. **TCP Connect Scan (-sT):** Performs a TCP Connect scan on the given target. This scan establishes a TCP connection with the target by completing the 3-way handshake. This scan is more time consuming and more intrusive than the TCP SYN scan.
3. **TCP ACK Scan (-sA):** Performs a TCP ACK scan on the given target. This scan does not establish a TCP connection with the target but rather just sends the ACK packet in the 3-way handshake. It works differently compared to the TCP SYN scan and TCP Connect scan and is particularly useful when there is a firewall involved, to determine if the firewall is stateful or not and identifying filtered ports. Open and closed ports both send back a RST response.
4. **UDP Scan (-sU):** Perform a UDP scan on the given target. Assigns the state of the port based on the response received. These scans are often overlooked while testing, however are important while conducting any tests as UDP ports can reveal hidden information that would otherwise be lost.

Service Scanning And Vulnerability Detection

Next, a tester would like to determine the services running behind the open ports on the target machine, as this is their path to break into the target machine. It's quite common to find a vulnerable version of a service such as a web server running behind a port. One of the coolest features of our tool is that after identifying the service running behind a port, we check for associated CVEs and report any that we find. This can greatly assist the penetration tester, as now they have a promising way into the system. We cover some examples of the same in the demo as well as in the examples section of this report.

Implementation Details and Documentation

The following covers the main functions and their documentation. ***There are many more functions but for the sake of keeping this document short, we have not included them but they can be found in the sharkbite.py file itself.***

```
def TCP_SYN_SCAN(dst_ip, ports, show, ss=False):
'''
    Arguments:
       dst_ip: Destination IP to scan
       ports: Ports that need to be scanned
    Return:
       ScanResult object containing the results of the scans

    Perform a TCP SYN scan on the given target. This scan does not
establish a TCP connection with the
    target but rather just sends the SYN packet in the 3-way handshake.
This type of scan is relatively
    stealthy and fast.
    '''
```

```
def TCP_CON_SCAN(dst_ip, ports, show, ss=False):
    '''
    Arguments:
       dst_ip: Destination IP to scan
       ports: Ports that need to be scanned
    Return:
       ScanResult object containing the results of the scans

    Perform a TCP Connect scan on the given target. This scan establishes a
TCP connection with the
    target by completing the 3-way handshake. This scan is more time
consuming and more intrusive than
    the TCP SYN scan.
    '''
```

```
def TCP_ACK_SCAN(dst_ip, ports, show):
    '''
    Arguments:
       dst_ip: Destination IP to scan
       ports: Ports that need to be scanned
    Return:
       ScanResult object containing the results of the scans

    Perform a TCP ACK scan on the given target. This scan does not
establish a TCP connection with the
    target but rather just sends the ACK packet in the 3-way handshake. It
works differently compared to the
    TCP SYN scan and TCP Connect scan and is particularly useful when there
is a firewall involved, to determine
    if the firewall is stateful or not and identifying filtered ports. Open
and closed ports both send back a
    RST response.
    '''
```

```
def UDP_SCAN(dst_ip, ports, verbose):
    '''
    Arguments:
       dst_ip: Destination IP to scan
       ports: Ports that need to be scanned
       verbose: Whether unopen ports need to be shown
    Return:
       ScanResult object containing the results of the scans

    Perform a UDP scan on the given target. Assigns the state of the port
based
    on the response received. These scans are often overlooked while
testing,
    however are important while conducting any tests.
    '''
```

```python
def discover_hosts(ip_range, interface):
    '''
    Arguments:
      ip_range: The IP or IP range to perform an arp scan on
      interface: The target interface
    Find the IP and MAC address(es) for the given IP/IP range that are
currently active.
    '''
```

```python
def service_scan(ip, ports):
    '''
    Arguments:
       ip: The IP address to perform the scan on
       ports: The ports to conduct the scan on

    Performs a service scan to determine the services running behind
    ports. If a service is identified, the program additionally checks
    whether the service running has a CVE assigned to it (indicating it's
vulnerable).
    and if so, reports the same.
    '''
```

Contributions

All of the work was distributed evenly among all the members. The following describes what each did.

Borui Li

Primarily responsible for service scans to determine the services running behind any port. Handled banner grabbing and its parsing as well as assisting with rendering the output of the program. Also helped with some scans.

Junheng (George) Wang

Primarily responsible for CVE detection in a vulnerable service and some parts of service scans. Additionally helped with cleaning up the code and documentation. Also assisted with formatting the output of the program.

Anirudha Kanodia

Primarily response for host discovery and some scans. Assisted on user input validation and error checking. Additionally helped with improving accuracy of some scans and formatting output.

Getting Started for Testing

1. Ensure you are running Linux

Ensure that you are running this program on a linux operating system (best on a linux vm). It does not run on Windows.

2. Download requirements

Run the following command to download all the required libraries

```
pip3 install -r requirements.txt
```

3. Setting up TryHackMe

We need a target machine to test our program. Visit https://tryhackme.com/ and create an account. Then, download the VPN configuration file and run it to connect to tryhackme's network. Next, navigate to the room: CVE-2021-41773/42013 (https://tryhackme.com/room/cve202141773) and start up the machine by clicking on "Task 4: Practical Exam" and then clicking on "Start machine". Once you have done so, you will need to wait 1 minute before the target IP is displayed. We are now good to go!

4. Important note before starting

Some commands will need root permissions to run so you will need to use sudo for them. Most importantly, try specifying the "-p <ports>" wherever appropriate to save time. Scans that don't have ports specified will take up to 10 minutes since it is scanning through 1000 ports. Additionally, it was noticed that sometimes due to network connectivity with tryhackme, it's possible the scan says the host is down in which case try again. Additionally, sometimes you'll notice the program won't stop even if you try Ctrl+C. Please find the process id using "ps aux | grep sharkbite" and then kill it using "kill <id>".

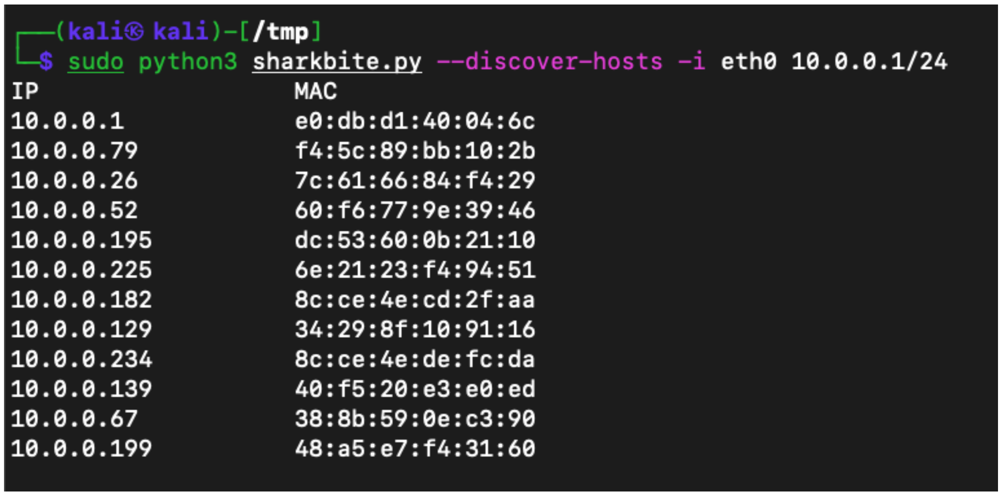
Testing Examples

The documentation for these can be found earlier in this report and in the python file itself. As a result, please refer to those for details about what each command does. Here, we will just be giving examples.

Host Discovery

You can use –discover-hosts with the -i interface and target subnet to find the active IPs and their corresponding MAC addresses for a given IP range.

```
sudo python3 sharkbite.py --discover-hosts -i eth0 10.0.0.1/24
```



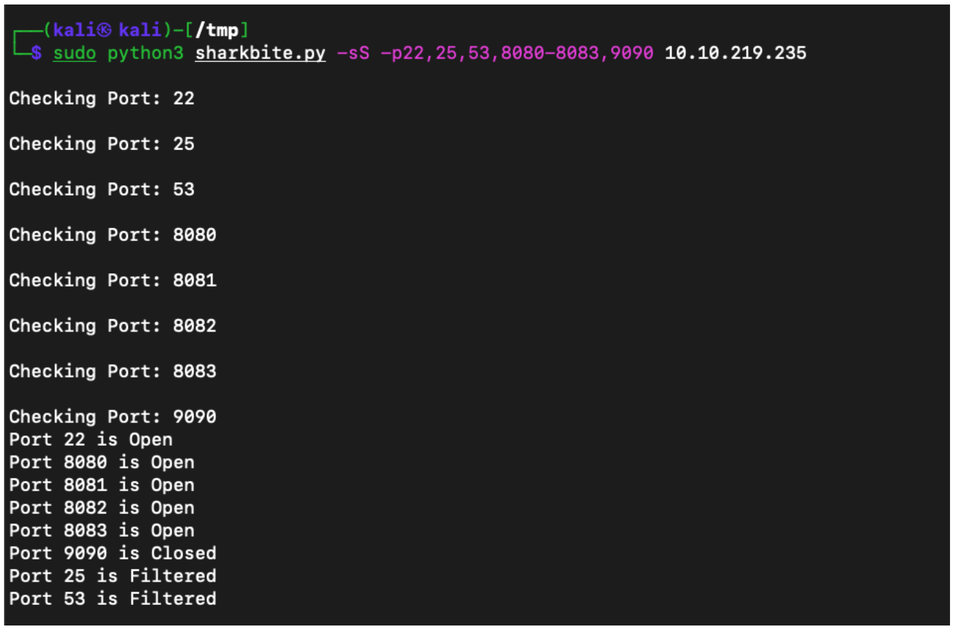
```
┌──(kali㉿kali)-[/tmp]
└─$ sudo python3 sharkbite.py --discover-hosts -i eth0 10.0.0.1/24
IP                  MAC
10.0.0.1            e0:db:d1:40:04:6c
10.0.0.79           f4:5c:89:bb:10:2b
10.0.0.26           7c:61:66:84:f4:29
10.0.0.52           60:f6:77:9e:39:46
10.0.0.195          dc:53:60:0b:21:10
10.0.0.225          6e:21:23:f4:94:51
10.0.0.182          8c:ce:4e:cd:2f:aa
10.0.0.129          34:29:8f:10:91:16
10.0.0.234          8c:ce:4e:de:fc:da
10.0.0.139          40:f5:20:e3:e0:ed
10.0.0.67           38:8b:59:0e:c3:90
10.0.0.199          48:a5:e7:f4:31:60
```

Port Scans

TCP SYN Scan (-sS)

Perform a TCP SYN scan on the given target. This scan does not establish a TCP connection with the target but rather just sends the SYN packet in the 3-way handshake. This type of scan is relatively stealthy and fast. The -p ports parameter is optional but highly recommended if you know the ports. If not, it will scan the top 1000 ports which can take up to 10 minutes., depending upon the connection with the target server.

```
sudo python3 sharkbite.py -sS -p22,25,53,8080-8083 10.10.219.235
```

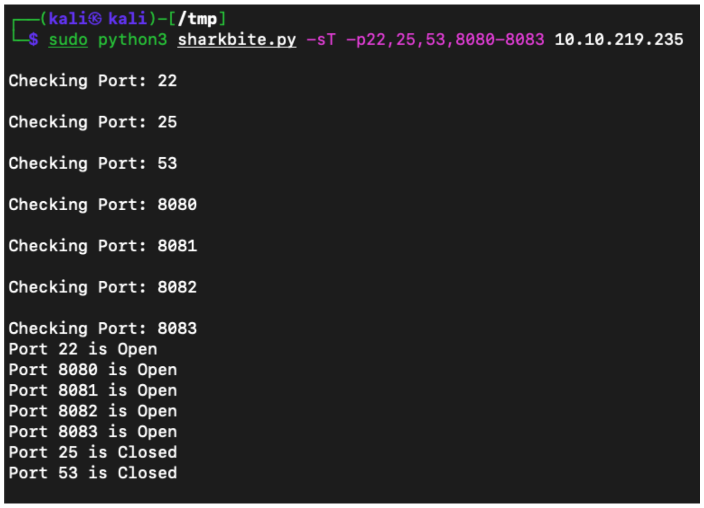


```
┌──(kali㉿ kali)-[/tmp]
└─$ sudo python3 sharkbite.py -sS -p22,25,53,8080-8083,9090 10.10.219.235

Checking Port: 22

Checking Port: 25

Checking Port: 53

Checking Port: 8080

Checking Port: 8081

Checking Port: 8082

Checking Port: 8083

Checking Port: 9090
Port 22 is Open
Port 8080 is Open
Port 8081 is Open
Port 8082 is Open
Port 8083 is Open
Port 9090 is Closed
Port 25 is Filtered
Port 53 is Filtered
```

TCP Connect Scan (-sT)

Perform a TCP Connect scan on the given target. This scan establishes a TCP connection with the target by completing the 3-way handshake. This scan is more time consuming and more intrusive than the TCP SYN scan.

```
sudo python3 sharkbite.py -sT -p22,25,53,8080-8083 10.10.219.235
```



```
┌──(kali㉿kali)-[/tmp]
└─$ sudo python3 sharkbite.py -sT -p22,25,53,8080-8083 10.10.219.235

Checking Port: 22

Checking Port: 25

Checking Port: 53

Checking Port: 8080

Checking Port: 8081

Checking Port: 8082

Checking Port: 8083
Port 22 is Open
Port 8080 is Open
Port 8081 is Open
Port 8082 is Open
Port 8083 is Open
Port 25 is Closed
Port 53 is Closed
```

TCP ACK Scan (-sA)

Perform a TCP ACK scan on the given target. This scan does not establish a TCP connection with the target but rather just sends the ACK packet in the 3-way handshake. It works differently compared to the TCP SYN scan and TCP Connect scan and is particularly useful when there is a firewall involved, to determine if the firewall is stateful or not and identifying filtered ports. Open and closed ports both send back a RST response.

```
sudo python3 sharkbite.py -sA -p22,25,53,8080-8083 10.10.219.235
```

UDP Scan (-sU)

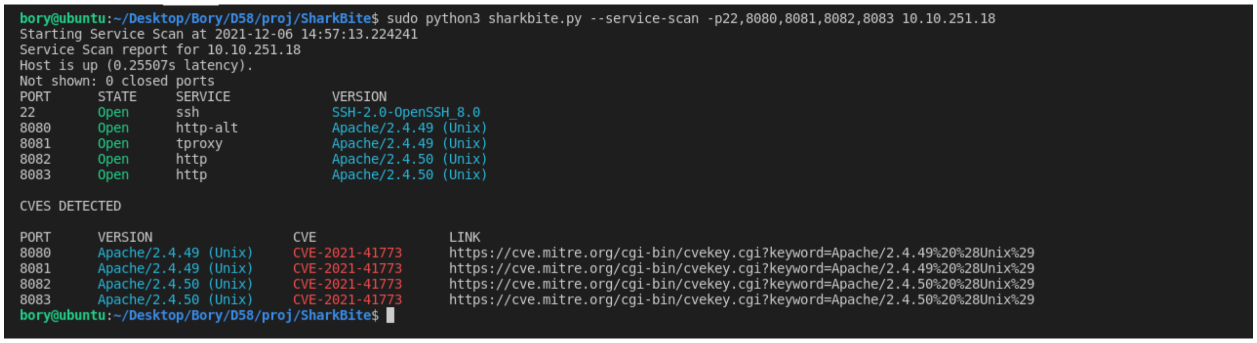
Perform a UDP scan on the given target. Assigns the state of the port based on the response received. These scans are often overlooked while testing, however are important while conducting any tests.

```
sudo python3 sharkbite.py -sU -p22,25,53,8080-8083 10.10.219.235
```

Service detection and CVE detection

To detect the services running behind ports, we can use service scan on a target IP address as shown below. If the program identifies a vulnerable service, it will report the same. Here, we know the ports from a previous scan so we specify them to save time. Otherwise the program scans the top 1000 ports. This however, is very slow and takes up to 10 minutes to run.

```
sudo python3 sharkbite.py --service-scan -p22,8080-8083 10.10.251.18
```



```
bory@ubuntu:~/Desktop/Bory/D58/proj/SharkBite$ sudo python3 sharkbite.py --service-scan -p22,8080,8081,8082,8083 10.10.251.18
Starting Service Scan at 2021-12-06 14:57:13.224241
Service Scan report for 10.10.251.18
Host is up (0.25507s latency).
Not shown: 0 closed ports
PORT      STATE     SERVICE             VERSION
22        Open      ssh                 SSH-2.0-OpenSSH_8.0
8080      Open      http-alt            Apache/2.4.49 (Unix)
8081      Open      tproxy              Apache/2.4.49 (Unix)
8082      Open      http                Apache/2.4.50 (Unix)
8083      Open      http                Apache/2.4.50 (Unix)

CVES DETECTED

PORT      VERSION                  CVE                 LINK
8080      Apache/2.4.49 (Unix)     CVE-2021-41773      https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=Apache/2.4.49%20%28Unix%29
8081      Apache/2.4.49 (Unix)     CVE-2021-41773      https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=Apache/2.4.49%20%28Unix%29
8082      Apache/2.4.50 (Unix)     CVE-2021-41773      https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=Apache/2.4.50%20%28Unix%29
8083      Apache/2.4.50 (Unix)     CVE-2021-41773      https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=Apache/2.4.50%20%28Unix%29
bory@ubuntu:~/Desktop/Bory/D58/proj/SharkBite$ 
```