Exercise 1:
Service Fingeprinting

Contents

1 Tasks

1.1 HTTP Service Fingerprinting

Because the ports for this task were not given, the first step that we need to do is find one open HTTP port. This can easily be done by using the following Python script:

```
from socket import socket, AF_INET, SOCK_STREAM
from threading import Thread

def main() -> None:
    for port in range(1, 65535):
        thread = Thread(target=check_open, args=('10.0.23.15', port))
        thread.start()

def check_open(ip: str, port: str) -> None:
    connection = socket(AF_INET, SOCK_STREAM)
    status = connection.connect_ex((ip, port))

    if status == 0:
        try:
            connection.send(b'GET / HTTP/1.1\r\nHost: test\r\n\r\n')
            response = connection.recv(4096)
        except Exception:
            connection.close()
            return

        print(f'[*] {port}')
        print(response)

    connection.close()
```

Listing 1: Finding open HTTP ports

The first open HTTP port that we got was **12080** (response starts with **'HTTP/1.1 200 OK'**). By further analyzing the response, it can be seen that the server returns the **Server** header attribute with the value of **webfs/1.21**, which in fact identifies the service that is running. The **Server** header attribute can easily be extracted by using the following function, which utilizes regular expressions:

```
def identify_httpd(response: str) -> str:
    server = search(r'Server:.*', response)

    if server:
        return server.group(0).replace('Server:', '').strip()

    return None
```

Listing 2: Regular expression for extracting the Server header attribute

The above code will start at the first instance of the string **'Server:'** that it finds in the response and match everything until the end of the line. And because every header attribute is given in a new line, this is exactly what we need.

If the regular expression finds a match, the *identify_httpd()* function will also remove the string **'Server:'**, and then remove all spaces on the left and right side of the string.

The full, working code can be found inside the file **http_scanner.py**. As specified in the exercise, the program expects one parameter - the IP address of the target (**-i/--ip**). An example command for the IP address given in the exercise would be:

python3 http_server.py $-i$ 10.0.23.15

Note: If the script shows *OSError (Errno 24): Too many open files*, execute the following command in the terminal:

```
ulimit -Sn 65535
```

Listing 3: Increase file size limit

The results of this command are given in Table 1.

1.1.1 Results

Port	Service	Name	Version
12080	http	webfs	1.21
12346	http	micro_httpd	/
20863	http	lighttpd	1.4.33
31711	http	Apache	2.4.7 (Ubuntu)
40305	http	mini_httpd	1.19 19dec2003

Table 1: Identified HTTP services

1.2 FTP Service Fingerprinting

This task was a lot easier because we were given specific ports and FTP services, and the only thing needed to do was map them to each other. Because of this, there was no need for concurrency.

By picking the first two ports, we can see how each of them responds to a certain request. By using *telnet*, we can connect to each of the ports:

telnet 10.0.23.15 210

telnet 10.0.23.15 2100

On both servers, we are greeted with the message **220 Secret FTPd ;)**. When we try to log in by using the command **USER anonymous**, we are greeted with two different messages:

$$\begin{array}{lcl} 10.0.23.15:210 & \rightarrow & 331\ Password\ required\ for\ anonymous \\ 10.0.23.15:2100 & \rightarrow & 331\ User\ anonymous\ OK.\ Password\ required \end{array}$$

By using the same logic, we can also test the other two FTP servers. Both of these responses were also different:

$$\begin{array}{lcl} 10.0.23.15:2121 & \rightarrow & 331\ Give\ me\ password \\ 10.0.23.15:21000 & \rightarrow & 331\ Please\ specify\ the\ password. \end{array}$$

Because all of the services provided a different response to that command, we can use this knowledge to identify which port corresponds to which FTP service. A quick Google search for the different service names given in the exercise will provide us with enough knowledge to map each port to each FTP service.

For example, the following post gives us enough information in order to map port **21000** to **vsftpd**: https://www.linuxquestions.org/questions/linux-server-73/ftp-login-failure-4175503091. We can therefore create a dictionary in Python that maps each of there responses to a particular service:

```
FTP_FINGERPRINTS = {
    '331 Password required for anonymous': 'pro-ftpd',
    '331 User anonymous OK. Password required': 'pure-ftpd',
    '331 Give me password': 'py-ftpd',
    '331 Please specify the password.': 'vs-ftpd'
}
```

Listing 4: Python dictionary to map responses to services

The full, working code can be found inside the file **ftp_scanner.py**. As specified in the exercise, the program expects one parameter - the IP address of the target (**-i/--ip**). An example command for the IP given in the exercise would be:

$$python3\ ftp\_server.py\ -i\ 10.0.23.15$$

The results of this command are given in Table 2.

1.2.1 Results

Port	Service	Name	Version
210	ftp	pro-ftpd	/
2100	ftp	pure-ftpd	/
2121	ftp	py-ftpd	/
21000	ftp	vs-ftpd	/

Table 2: Identified FTP services

2 Requirements

No requirements needed, everything should already be installed on *Ubuntu 20.04 LTS*.