Lab 13: Virtual Private Network the Heartbleed Bug and Attack
Md Rony

IPtables

I followed the instruction from this website to add a firewall
.https://www.digitalocean.com/community/tutorials/how-to-set-up-a-firewall-with-ufw-on-ubuntu-14-04

First i install UFW with apt-get on my kali linux terminal. My server embedded IPv6, ensure that UFW is configured to support IPv6 so that it managed firewall rules for IPv6 in addition to IPv4. To do this, open the UFW configuration with your favorite editor. I used nano.

```
File  Edit  View  Search  Terminal  Help
  GNU nano 2.9.8                    /etc/default/ufw

# /etc/default/ufw
#

# Set to yes to apply rules to support IPv6 (no means only IPv6 on loopback
# accepted). You will need to 'disable' and then 'enable' the firewall for
# the changes to take affect.
IPV6=yes

# Set the default input policy to ACCEPT, DROP, or REJECT. Please note that if
# you change this you will most likely want to adjust your rules.
DEFAULT_INPUT_POLICY="DROP"

# Set the default output policy to ACCEPT, DROP, or REJECT. Please note that if
# you change this you will most likely want to adjust your rules.
DEFAULT_OUTPUT_POLICY="ACCEPT"

# Set the default forward policy to ACCEPT, DROP or REJECT.  Please note that
# if you change this you will most likely want to adjust your rules
DEFAULT_FORWARD_POLICY="DROP"
                              [ Read 45 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line
```

I Set Up Default Policies,Allow SSH Connections, Enabled UFW policies and enable other connection rules.

```
File  Edit  View  Search  Terminal  Help
Creating config file /etc/ufw/after6.rules with new version
Created symlink /etc/systemd/system/multi-user.target.wants/ufw.service → /lib/systemd
/system/ufw.service.
update-rc.d: We have no instructions for the ufw init script.
update-rc.d: It looks like a non-network service, we enable it.
Processing triggers for systemd (239-7) ...
Processing triggers for man-db (2.8.3-2) ...
Processing triggers for rsyslog (8.36.0-1) ...
root@kali:~# sudo nano /etc/default/ufw
root@kali:~# sudo ufw status verbose
Status: inactive
root@kali:~# sudo ufw default deny incoming
Default incoming policy changed to 'deny'
(be sure to update your rules accordingly)
root@kali:~# sudo ufw default allow outgoing
Default outgoing policy changed to 'allow'
(be sure to update your rules accordingly)
root@kali:~# sudo ufw allow ssh
Rules updated
Rules updated (v6)
root@kali:~# sudo ufw allow 22
Rules updated
Rules updated (v6)
root@kali:~#
```

```
File  Edit  View  Search  Terminal  Help
root@kali:~# sudo nano /etc/default/ufw
root@kali:~# sudo ufw status verbose
Status: inactive
root@kali:~# sudo ufw default deny incoming
Default incoming policy changed to 'deny'
(be sure to update your rules accordingly)
root@kali:~# sudo ufw default allow outgoing
Default outgoing policy changed to 'allow'
(be sure to update your rules accordingly)
root@kali:~# sudo ufw allow ssh
Rules updated
Rules updated (v6)
root@kali:~# sudo ufw allow 22
Rules updated
Rules updated (v6)
root@kali:~# sudo ufw enable
Firewall is active and enabled on system startup
root@kali:~# sudo ufw allow http
Rule added
Rule added (v6)
root@kali:~# sudo ufw allow 80
Rule added
Rule added (v6)
root@kali:~#
```

```
File  Edit  View  Search  Terminal  Help
Rule added (v6)
root@kali:~# sudo ufw allow https
Rule added
Rule added (v6)
root@kali:~# sudo ufw allow 443
Rule added
Rule added (v6)
root@kali:~# sudo ufw allow ftp
Rule added
Rule added (v6)
root@kali:~# sudo ufw allow 21/tcp
Skipping adding existing rule
Skipping adding existing rule (v6)
root@kali:~# sudo ufw allow 6000:6007/tcp
Rule added
Rule added (v6)
root@kali:~# sudo ufw allow 6000:6007/udp
Rule added
Rule added (v6)
root@kali:~# sudo ufw allow from 15.15.15.51
Rule added
root@kali:~# sudo ufw allow from 15.15.15.51 to any port 22
Rule added
root@kali:~#
```

```
File  Edit  View  Search  Terminal  Help
root@kali:~# sudo ufw allow from 15.15.15.0/24 to any port 22
Rule added
root@kali:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen
 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group de
fault qlen 1000
    link/ether 08:00:27:74:17:d4 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0
       valid_lft 82693sec preferred_lft 82693sec
    inet6 fe80::a00:27ff:fe74:17d4/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
root@kali:~# sudo ufw allow in on eth0 to any port 80
Rule added
Rule added (v6)
root@kali:~# sudo ufw allow in on eth1 to any port 3306
Rule added
Rule added (v6)
root@kali:~#
```

```
File  Edit  View  Search  Terminal  Help
 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group de
fault qlen 1000
    link/ether 08:00:27:74:17:d4 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0
       valid_lft 82693sec preferred_lft 82693sec
    inet6 fe80::a00:27ff:fe74:17d4/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
root@kali:~# sudo ufw allow in on eth0 to any port 80
Rule added
Rule added (v6)
root@kali:~# sudo ufw allow in on eth1 to any port 3306
Rule added
Rule added (v6)
root@kali:~# sudo ufw deny http
Rule updated
Rule updated (v6)
root@kali:~# sudo ufw deny from 15.15.15.51
Rule updated
```

This is delete ,disable and resate rules

```
File  Edit  View  Search  Terminal  Help

root@kali:~# sudo ufw delete 2
Deleting:
 allow 22
Proceed with operation (y|n)? y
Rule deleted
root@kali:~# sudo ufw delete allow http
Could not delete non-existent rule
Could not delete non-existent rule (v6)
root@kali:~# sudo ufw delete allow 80
Rule deleted
Rule deleted (v6)
root@kali:~# sudo ufw disable
Firewall stopped and disabled on system startup
root@kali:~# sudo ufw reset
Resetting all rules to installed defaults. Proceed with operation (y|n)? y
Backing up 'user.rules' to '/etc/ufw/user.rules.20181024_120834'
Backing up 'before.rules' to '/etc/ufw/before.rules.20181024_120834'
Backing up 'after.rules' to '/etc/ufw/after.rules.20181024_120834'
Backing up 'user6.rules' to '/etc/ufw/user6.rules.20181024_120834'
Backing up 'before6.rules' to '/etc/ufw/before6.rules.20181024_120834'
Backing up 'after6.rules' to '/etc/ufw/after6.rules.20181024_120834'

root@kali:~#
```

This was my command i used to configured and managed firewall rules on IPv6 and IPv4

```
File  Edit  View  Search  Terminal  Help
root@kali:~# sudo ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 22/tcp                     ALLOW IN    Anywhere
[ 2] 22                         ALLOW IN    Anywhere
[ 3] 80/tcp                     DENY IN     Anywhere
[ 4] 80                         ALLOW IN    Anywhere
[ 5] 443/tcp                    ALLOW IN    Anywhere
[ 6] 443                        ALLOW IN    Anywhere
[ 7] 21/tcp                     ALLOW IN    Anywhere
[ 8] 6000:6007/tcp              ALLOW IN    Anywhere
[ 9] 6000:6007/udp              ALLOW IN    Anywhere
[10] Anywhere                   DENY IN     15.15.15.51
[11] 22                         ALLOW IN    15.15.15.51
[12] Anywhere                   ALLOW IN    15.15.15.0/24
[13] 22                         ALLOW IN    15.15.15.0/24
[14] 80 on eth0                 ALLOW IN    Anywhere
[15] 3306 on eth1               ALLOW IN    Anywhere
[16] 22/tcp (v6)                ALLOW IN    Anywhere (v6)
[17] 22 (v6)                    ALLOW IN    Anywhere (v6)
[18] 80/tcp (v6)                DENY IN     Anywhere (v6)
[19] 80 (v6)                    ALLOW IN    Anywhere (v6)
```

pf

https://www.cyberciti.biz/faq/how-to-set-up-a-firewall-with-pf-on-freebsd-to-protect-a-web-server/

First i turned on the fire wall and create a firewall rules in /usr/local/etc/pf.conf but i couldn't start pf firewall . i checked for syntax error but i couldn't detect my error.

```
File  Edit  View  Search  Terminal  Help
root@kali:~# echo 'pf_enable="YES"' >> /etc/rc.conf
root@kali:~# echo 'pf_rules="/usr/local/etc/pf.conf"' >> /etc/rc.conf
root@kali:~# echo 'pflog_enable="YES"' >> /etc/rc.conf
root@kali:~# echo 'pflog_logfile="/var/log/pflog"' >> /etc/rc.conf
root@kali:~# vi /usr/local/etc/pf.conf
```

```
File  Edit  View  Search  Terminal  Help
## Blocking spoofed packets
antispoof quick for $ext_if

# Open SSH port which is listening on port 22 from VPN 139.xx.yy.zz Ip only
# I do not allow or accept ssh traffic from ALL for security reasons
pass in quick on $ext_if inet proto tcp from 139.xxx.yyy.zzz to $ext_if_ip port
= ssh flags S/SA keep state label "USER RULE: Allow SSH from 139.xxx.yyy.zzz"
## Use the following rule to enable ssh for ALL users from any IP address #
## pass in inet proto tcp to $ext_if port ssh
### [ OR ] ###
## pass in inet proto tcp to $ext_if port 22

# Allow Ping-Pong stuff. Be a good sysadmin
pass inet proto icmp icmp-type echoreq

# All access to our Nginx/Apache/Lighttpd Webserver ports
pass proto tcp from any to $ext_if port $webports

# Allow essential outgoing traffic
pass out quick on $ext_if proto tcp to any port $int_tcp_services
pass out quick on $ext_if proto udp to any port $int_udp_services

# Add custom rules below
                                                          60,24          Bot
```

Here is my unsuccessful attempt

```
File  Edit  View  Search  Terminal  Tabs  Help
root@kali: ~                    root@kali: ~
root@kali:~# /etc/rc.d/pf check
bash: /etc/rc.d/pf: No such file or directory
root@kali:~# pfctl -n -f /usr/local/etc/pf.conf
bash: pfctl: command not found
root@kali:~# vim /usr/local/etc/pf.conf
root@kali:~# pfctl -n -f /usr/local/etc/pf.conf
bash: pfctl: command not found
root@kali:~# service pf check
pf: unrecognized service
root@kali:~# service pf check
pf: unrecognized service
root@kali:~# service pf start
Failed to start pf.service: Unit pf.service not found.
root@kali:~# service pf stop
Failed to stop pf.service: Unit pf.service not loaded.
root@kali:~# service pf check
pf: unrecognized service
root@kali:~# service pf restart
Failed to restart pf.service: Unit pf.service not found.
root@kali:~# service pf status
Unit pf.service could not be found.
root@kali:~# pfctl -s rules
bash: pfctl: command not found
root@kali:~#
```

4.2 TEAM 2: Creating the hiring exam

4.2.1 FrobozzCo Firewall Test

To begin configuring the Firewall we begin by using the command followed as mentioned in the lab after using ssh to access the server node.

Sudo /root/firewall/firewall.sh

Sudo /root/firewall/ extingui.sh

We then move forward and check the if configuration to verify the eth connections as well as the IP addresses used by the machine.

ifconfig

```
jjc400ca@server:~$ sudo /root/firewall/firewall.sh
Starting firewall: done.
jjc400ca@server:~$ sudo /root/firewall/extingui.sh
jjc400ca@server:~$ ifconfig
eth3      Link encap:Ethernet  HWaddr 00:04:23:c7:a5:c7
          inet addr:10.1.1.3  Bcast:10.1.1.255  Mask:255.255.255.0
          inet6 addr: fe80::204:23ff:fec7:a5c7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:1408 (1.4 KB)

eth4      Link encap:Ethernet  HWaddr 00:14:22:23:8a:fa
          inet addr:192.168.1.201  Bcast:192.168.3.255  Mask:255.255.252
.0
          inet6 addr: fe80::214:22ff:fe23:8afa/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:19750 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2156 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:28464104 (28.4 MB)  TX bytes:236330 (236.3 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:63 errors:0 dropped:0 overruns:0 frame:0
          TX packets:63 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:11746 (11.7 KB)  TX bytes:11746 (11.7 KB)
```

We then move forward and edit the firewall.sh script by using: **sudo vim firewall.s**

The question given were answered by the IPTABLES rules following the resources provided in the zip folder.:

1. Passively ignore any traffic inbound to the interface that says it's coming from the server itself (obvious spoof attempt). The server uses the localhost loopback device lo for internal traffic, so it should never see incoming traffic from its own IP on the experimental network interface. (See test case 11, below)

$IPTABLES -A INPUT -i $ETH -s 10.1.1.3 -j DROP → 10.1.1.3 (server IP address)

```
#1. Passively ignore
$IPTABLES -A INPUT -i $ETH -s 10.1.1.3 -j DROP

#2. Established traffic
$IPTABLES -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

2. Allow all established traffic on the experimental network interface. Established or related traffic is traffic that is part of previously accepted new connections.

$IPTABLES -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

```
#2. Established traffic
$IPTABLES -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

3. Accept new connections on the experimental network (10.1.x.x) of the types listed below:
    a. Inbound TCP connections to the OpenSSH, Apache, and MySQL servers on their standard ports. (Test cases 1, 3, 5)

$IPTABLES -A INPUT -i $ETH -m state --state NEW -p tcp --dport 22 -j ACCEPT → Port 22 OpenSSH

$IPTABLES -A INPUT -i $ETH -m state --state NEW -p tcp --dport 80 -j ACCEPT → Port 80 Apache (Web)

```
# (a-b) allow inbound traffic to the OpenSSH, Apache2, and MySQL servers. (MySQL traffic only allowed fr
om client.)
$IPTABLES -A INPUT -i $ETH -m state --state NEW -p tcp --dport 22 -j ACCEPT
$IPTABLES -A INPUT -i $ETH -m state --state NEW -p tcp --dport 80 -j ACCEPT
$IPTABLES -A INPUT -i $ETH -s 10.1.1.2 -m state --state NEW -p tcp --dport 3306 -j ACCEPT
```

    b. The MySQL server should only accept connections from the client host.

$IPTABLES -A INPUT -i $ETH -s 10.1.1.2 -m state --state NEW -p tcp --dport 3306 -j ACCEPT → -s (source port → 10.1.1.2 (Client IP address)); Port 3306 → MySQL

```
$IPTABLES -A INPUT -i $ETH -s 10.1.1.2 -m state --state NEW -p tcp --dport 3306 -j ACCEPT
```

    c. Inbound UDP connections to the server ports 10000 to 10005 from the host client. (Test case 8)

$IPTABLES -A INPUT -i $ETH -p udp -s 10.1.1.2 -m multiport --dport 10000 -j ACCEPT
$IPTABLES -A INPUT -i $ETH -p udp -s 10.1.1.2 -m multiport --dport 10001 -j ACCEPT
$IPTABLES -A INPUT -i $ETH -p udp -s 10.1.1.2 -m multiport --dport 10002 -j ACCEPT
$IPTABLES -A INPUT -i $ETH -p udp -s 10.1.1.2 -m multiport --dport 10003 -j ACCEPT
$IPTABLES -A INPUT -i $ETH -p udp -s 10.1.1.2 -m multiport --dport 10004 -j ACCEPT
$IPTABLES -A INPUT -i $ETH -p udp -s 10.1.1.2 -m multiport --dport 10005 -j ACCEPT

```
# (C)  allow new inbound udp traffic to ports 10000-10005, and new outbound
$IPTABLES -A INPUT -i $ETH -p udp -s 10.1.1.2 -m multiport --dport 10000 -j ACCEPT
$IPTABLES -A INPUT -i $ETH -p udp -s 10.1.1.2 -m multiport --dport 10001 -j ACCEPT
$IPTABLES -A INPUT -i $ETH -p udp -s 10.1.1.2 -m multiport --dport 10002 -j ACCEPT
$IPTABLES -A INPUT -i $ETH -p udp -s 10.1.1.2 -m multiport --dport 10003 -j ACCEPT
$IPTABLES -A INPUT -i $ETH -p udp -s 10.1.1.2 -m multiport --dport 10004 -j ACCEPT
$IPTABLES -A INPUT -i $ETH -p udp -s 10.1.1.2 -m multiport --dport 10005 -j ACCEPT
```

    d. Inbound ICMP ping requests. (Test case 6)

e. Outbound ICMP ping replies. (Test case 7)

$IPTABLES -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT

$IPTABLES -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT

```
# 4. allow the server to send and respond to ICMP pings.
$IPTABLES -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT
$IPTABLES -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT
```

f. Outbound TCP connections to any OpenSSH, SMTP, and Apache (on standard ports). (Test cases 2, 4)

$IPTABLES -A OUTPUT -o $ETH -m state --state NEW -p tcp --dport 22 -j ACCEPT → Port 22 → OpenSSH

$IPTABLES -A OUTPUT -o $ETH -m state --state NEW -p tcp --dport 587 -j ACCEPT → Port 587→ SMTP

$IPTABLES -A OUTPUT -o $ETH -m state --state NEW -p tcp --dport 80 -j ACCEPT → Port 80→ Apache

```
# 4. allow the server to send and respond to ICMP pings.
$IPTABLES -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT
$IPTABLES -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT

#F Outbound TCP connections Open SSH, SMTP, apache.
$IPTABLES -A OUTPUT -o $ETH -m state --state NEW -p tcp --dport 22 -j ACCEPT
$IPTABLES -A OUTPUT -o $ETH -m state --state NEW -p tcp --dport 587 -j ACCEPT
$IPTABLES -A OUTPUT -o $ETH -m state --state NEW -p tcp --dport 80 -j ACCEPT

# OTHER CONNECTIONS
# ----------------
# *IGNORE* all other traffic
$IPTABLES -A INPUT -i $ETH -s 0/0 -d 10.1.1.3 -j DROP

# No changes below this line:
# ---8<--------------------------
echo "done."

~
```

g. Outbound UDP connections to the ports 10006 to 10010 on host client from the server. (Test case 9)

$IPTABLES -A OUTPUT -o $ETH -p udp -s 10.1.1.2 -m multiport --dport 10006 -j ACCEPT

$IPTABLES -A OUTPUT -o $ETH -p udp -s 10.1.1.2 -m multiport --dport 10007 -j ACCEPT

$IPTABLES -A OUTPUT -o $ETH -p udp -s 10.1.1.2 -m multiport --dport 10008 -j ACCEPT

$IPTABLES -A OUTPUT -o $ETH -p udp -s 10.1.1.2 -m multiport --dport 10009 -j ACCEPT

$IPTABLES -A OUTPUT -o $ETH -p udp -s 10.1.1.2 -m multiport --dport 10010 -j ACCEPT

```
# udp traffic to ports 10006-10010. Inbound and outbound UDP traffic should be limited to being from cli
ent (for input) or to client (for output).
# (You can get client's address from DETERLab.)
$IPTABLES -A OUTPUT -o $ETH -p udp -s 10.1.1.2 -m multiport --dport 10006 -j ACCEPT
$IPTABLES -A OUTPUT -o $ETH -p udp -s 10.1.1.2 -m multiport --dport 10007 -j ACCEPT
$IPTABLES -A OUTPUT -o $ETH -p udp -s 10.1.1.2 -m multiport --dport 10008 -j ACCEPT
$IPTABLES -A OUTPUT -o $ETH -p udp -s 10.1.1.2 -m multiport --dport 10009 -j ACCEPT
$IPTABLES -A OUTPUT -o $ETH -p udp -s 10.1.1.2 -m multiport --dport 10010 -j ACCEPT
```

4. Passively ignore all other traffic. (Do not allow it or respond to it in any way.) (Test case 10)

$IPTABLES -A INPUT -i $ETH -s 0/0 -d 10.1.1.3 -j DROP

```
# OTHER CONNECTIONS
# -----------------
# *IGNORE* all other traffic
$IPTABLES -A INPUT -i $ETH -s 0/0 -d 10.1.1.3 -j DROP
```

To verify that the commands were activated we used the **sudo iptables -L** command:

```
jjc400ca@server:/root/firewall$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere             state NEW,RELATED,ESTABLISHED
DROP       all  --  server-link0         anywhere
ACCEPT     all  --  anywhere             anywhere             ctstate RELATED,ESTABLISHED
ACCEPT     tcp  --  anywhere             anywhere             state NEW tcp dpt:ssh
ACCEPT     tcp  --  anywhere             anywhere             state NEW tcp dpt:http
ACCEPT     tcp  --  client-link0         anywhere             state NEW tcp dpt:mysql
ACCEPT     udp  --  client-link0         anywhere             multiport dports 10000
ACCEPT     udp  --  client-link0         anywhere             multiport dports 10001
ACCEPT     udp  --  client-link0         anywhere             multiport dports 10002
ACCEPT     udp  --  client-link0         anywhere             multiport dports 10003
ACCEPT     udp  --  client-link0         anywhere             multiport dports 10004
ACCEPT     udp  --  client-link0         anywhere             multiport dports 10005
ACCEPT     icmp --  anywhere             anywhere             icmp echo-reply
DROP       all  --  anywhere             server-link0

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere             state NEW,RELATED,ESTABLISHED
ACCEPT     udp  --  client-link0         anywhere             multiport dports 10006
ACCEPT     udp  --  client-link0         anywhere             multiport dports 10007
ACCEPT     udp  --  client-link0         anywhere             multiport dports 10008
ACCEPT     udp  --  client-link0         anywhere             multiport dports 10009
ACCEPT     udp  --  client-link0         anywhere             multiport dports 10010
ACCEPT     icmp --  anywhere             anywhere             icmp echo-request
ACCEPT     tcp  --  anywhere             anywhere             state NEW tcp dpt:ssh
ACCEPT     tcp  --  anywhere             anywhere             state NEW tcp dpt:submission
ACCEPT     tcp  --  anywhere             anywhere             state NEW tcp dpt:http
jjc400ca@server:/root/firewall$
```

Another way of viewing if all the commands were without errors we used: **sudo iptables -S**

```
jjc400ca@server:~$ sudo iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT
-A INPUT -s 10.1.1.3/32 -i eth4 -j DROP
-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A INPUT -i eth4 -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A INPUT -i eth4 -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
-A INPUT -s 10.1.1.2/32 -i eth4 -p tcp -m state --state NEW -m tcp --dport 33
06 -j ACCEPT
-A INPUT -s 10.1.1.2/32 -i eth4 -p udp -m multiport --dports 10000 -j ACCEPT
-A INPUT -s 10.1.1.2/32 -i eth4 -p udp -m multiport --dports 10001 -j ACCEPT
-A INPUT -s 10.1.1.2/32 -i eth4 -p udp -m multiport --dports 10002 -j ACCEPT
-A INPUT -s 10.1.1.2/32 -i eth4 -p udp -m multiport --dports 10003 -j ACCEPT
-A INPUT -s 10.1.1.2/32 -i eth4 -p udp -m multiport --dports 10004 -j ACCEPT
-A INPUT -s 10.1.1.2/32 -i eth4 -p udp -m multiport --dports 10005 -j ACCEPT
-A INPUT -p icmp -m icmp --icmp-type 0 -j ACCEPT
-A INPUT -d 10.1.1.3/32 -i eth4 -j DROP
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -s 10.1.1.2/32 -o eth4 -p udp -m multiport --dports 10006 -j ACCEPT
-A OUTPUT -s 10.1.1.2/32 -o eth4 -p udp -m multiport --dports 10007 -j ACCEPT
-A OUTPUT -s 10.1.1.2/32 -o eth4 -p udp -m multiport --dports 10008 -j ACCEPT
-A OUTPUT -s 10.1.1.2/32 -o eth4 -p udp -m multiport --dports 10009 -j ACCEPT
-A OUTPUT -s 10.1.1.2/32 -o eth4 -p udp -m multiport --dports 10010 -j ACCEPT
-A OUTPUT -p icmp -m icmp --icmp-type 8 -j ACCEPT
-A OUTPUT -o eth4 -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A OUTPUT -o eth4 -p tcp -m state --state NEW -m tcp --dport 587 -j ACCEPT
-A OUTPUT -o eth4 -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
jjc400ca@server:~$
```

PS configuration :

I checked for syntax error “service pf check” but i couldn’t detached my error in 4.1.2. I am still confused what part i did wrong . I doubt i was unsuccessful creating and saved firewall rules in /usr/local/etc/pf.conf.

Firewall configuration

After successfully completing the task assigned in 4.2 we discussed the error faced with the rules and the .sh file editing and creation while fixing the bugs. The most challenging part of 4.2.1 was understanding and incorporating the script of firewall.sh file. We faced minor bugs in the code because of type errors as well as the multiport function for 3(c) and 3(g) did not accept the ports together. We later had to fix the error by separating the ports and presenting them singularly.

Another major error we faced in the beginning of the 4.2 is when accessing the firewall.sh script to edit the syntax. It always displayed the error of swapping, renaming and creating another file named: firewall.sh.swp. To overcome this error we used to sudo cp command and copied the firewall.sh file over and deleted the original firewall.sh file. This cleared the error of creating firewall.sh.swp file. Overall, the completion of section 4.2 was a success with minor errors as presented above.

Heartbleed

“During a Heartbeat test the client sends Heartbeat Request message to the server in order to see if the server is still connected and alive. The server receives the Heartbeat Request message which contains the payload and the payload length of the message, the server should then send back a copy of the received message in the message. The attacker verifies that the Heartbeat Response message is the original sent. At this time the connections is still alive, but if the message isn’t the same, the Heartbeat Request message is retransmitted for a set number of retransmissions. The Heartbleed exploit took advantage of this message transmitting due to how the Heartbeat extension not checking to see if the payload length is an appropriate number. We can take advantage of this by setting a word such as “Hello” as the payload, and instead setting “5” as the payload length we can instead set “12345” as the payload length. This would then send the payload “Hello” and the length “12345” to the receiver, who then responds back with the payload and any following data stored in the memory to fill the other 12340 bytes. It is possible for an attacker to read up to 65,635 bytes of data at one time which seems to be a little amount of data, but an attacker is able to run this attack multiple times. They can then go through the data until they find the information they want whether it be usernames, passwords or the content of a message”

Implementation

“Our team used two laptop to download the tools to see how the bug works. The goal was to show how heart bleed works without the patch extension. Then we had to analyze the risks and document our findings. The attacking laptop and the server laptop were owned by a member of the group, and said group member gave permission to attack the server laptop. To initiate a Heartbleed attack, we first obtained two Virtual Machines with one acting as the attacker and the other acting as the server from which we plan to obtain information from. The attacker Virtual Machine had Kali Linux installed, while the server Virtual Machine had Ubuntu 12.04.4 LTS installed. Kali Linux is chosen for the attacker as it has the Metasploit Framework which we will use to initiate a Heartbleed attack on the server. Ubuntu 12.04.4 LTS is chosen as the server due to it having OpenSSL 1.0.1 already installed. To start Apache, is installed onto the server Virtual Machine which is used as the web server after which the SSL module is enabled to provide SSL v3.0 and TLS support for the web server. After the setup we then found the Ip address of the server which was 10.0.255.67, by going to https://10.0.255.67 we were able to see that the web server is set up (Figure 1). From there Metasploit is started on the attack Virtual Machine, and the scanner “openssl_heartbleed “ is chosen. Using the command “set RHOSTS 10.0.255.67” we set the Ip address as the victim of our attack. We also used the command “set verbose true” since without the command we were not given the results from the attack. Finally we gave the command “run” which starts the attack on the server, and then

prints out the results. The results from our experiment have no value since there is nothing on the web server. If on the hand the web server had usernames, passwords and secret keys on it we would be able to run Heartbleed repeatedly until we get the information we want."

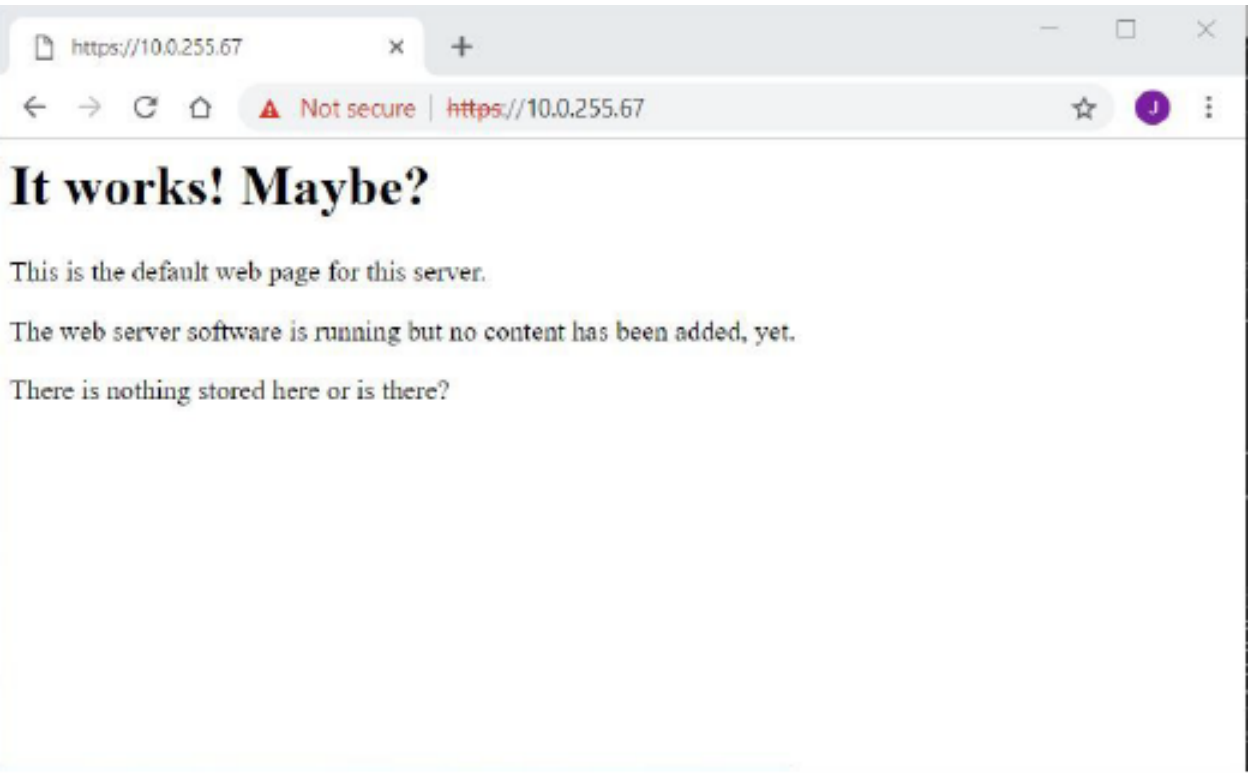

Figure : Web server is setup and webpage is accessible.

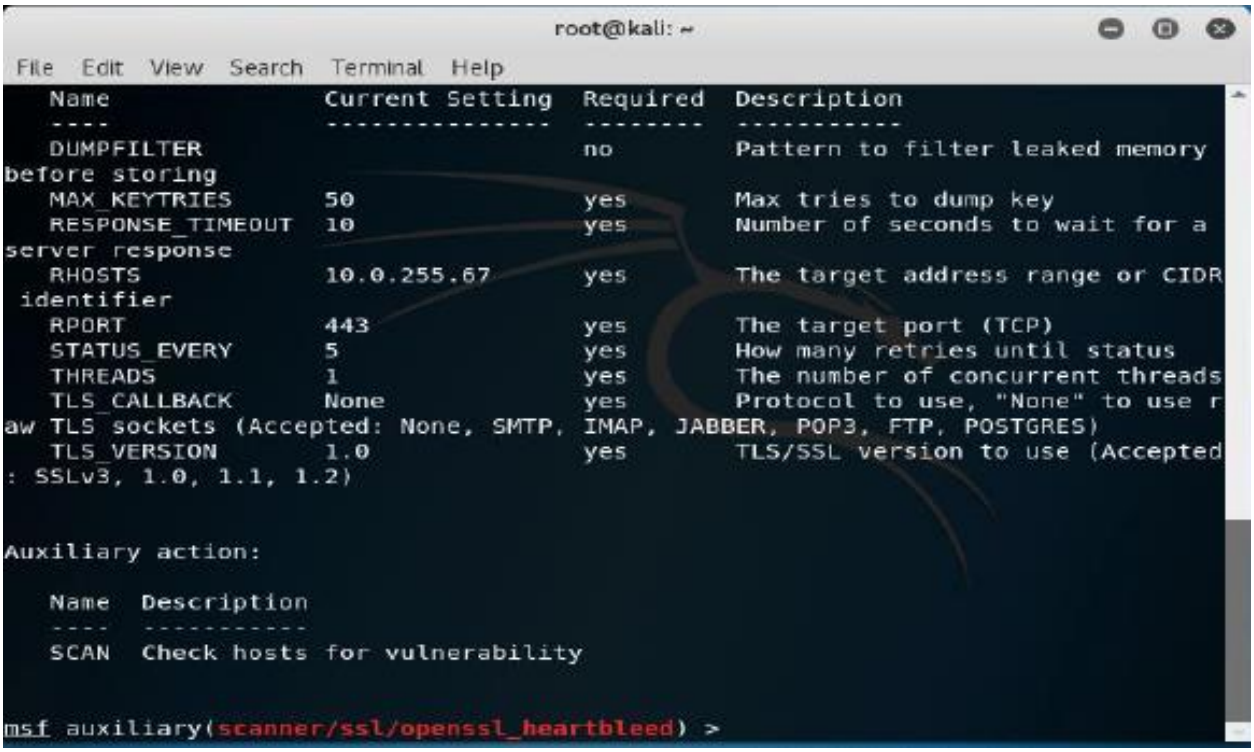

Figure: Heartbleed test

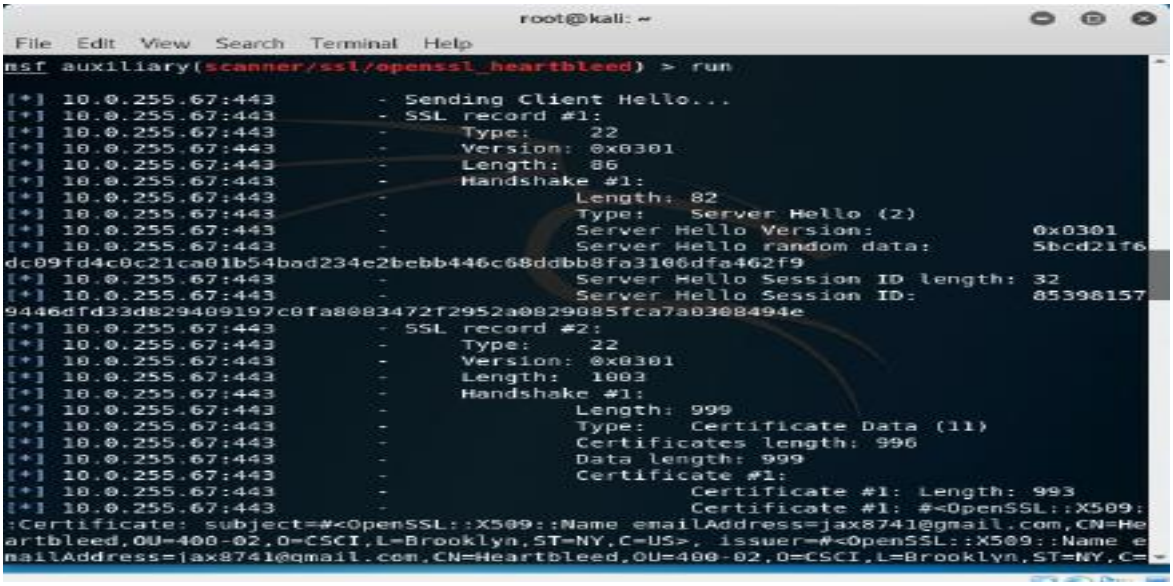

Figure: Heartbleed attack's Results

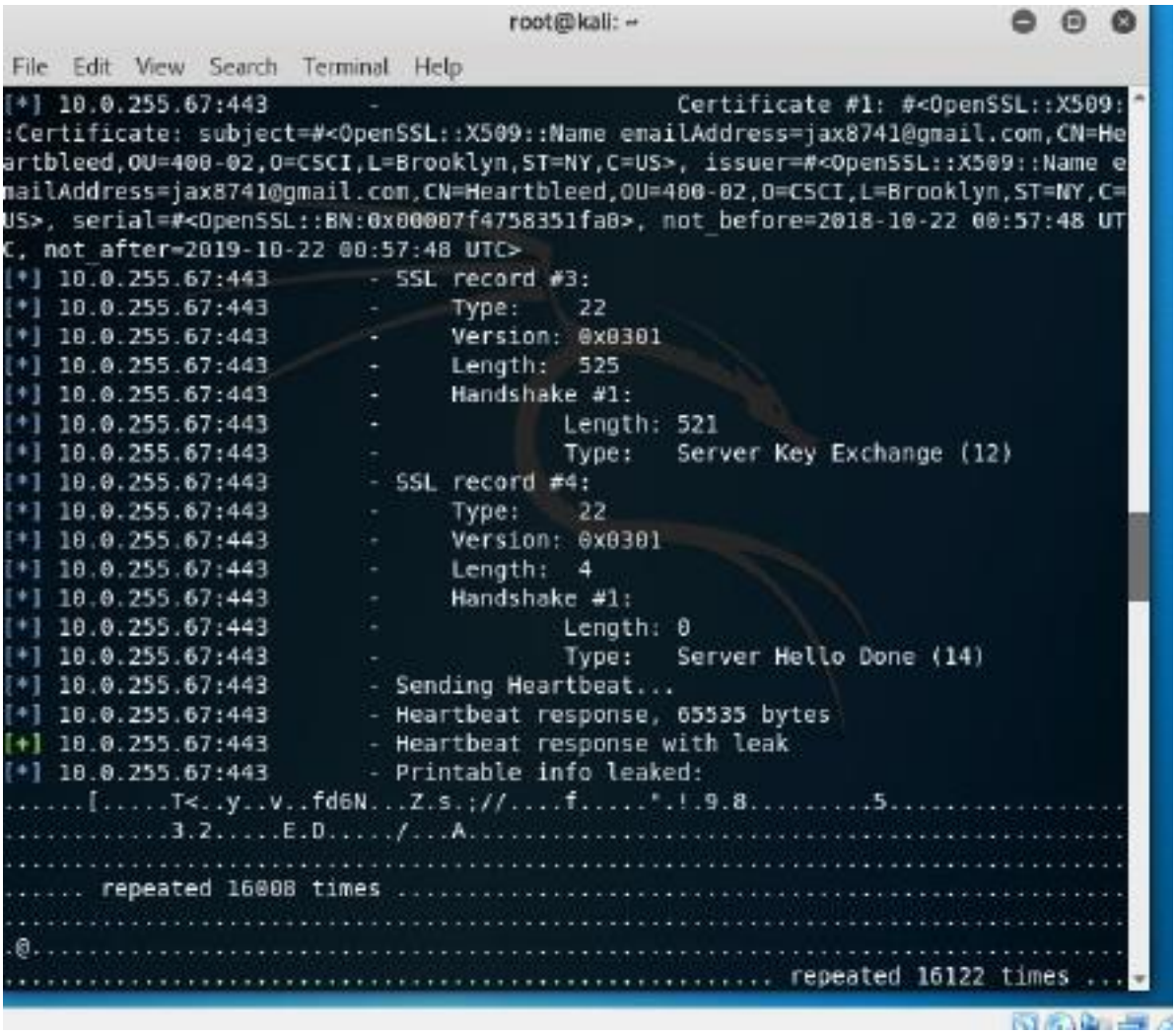

Figure: Heartbleed attack's Results

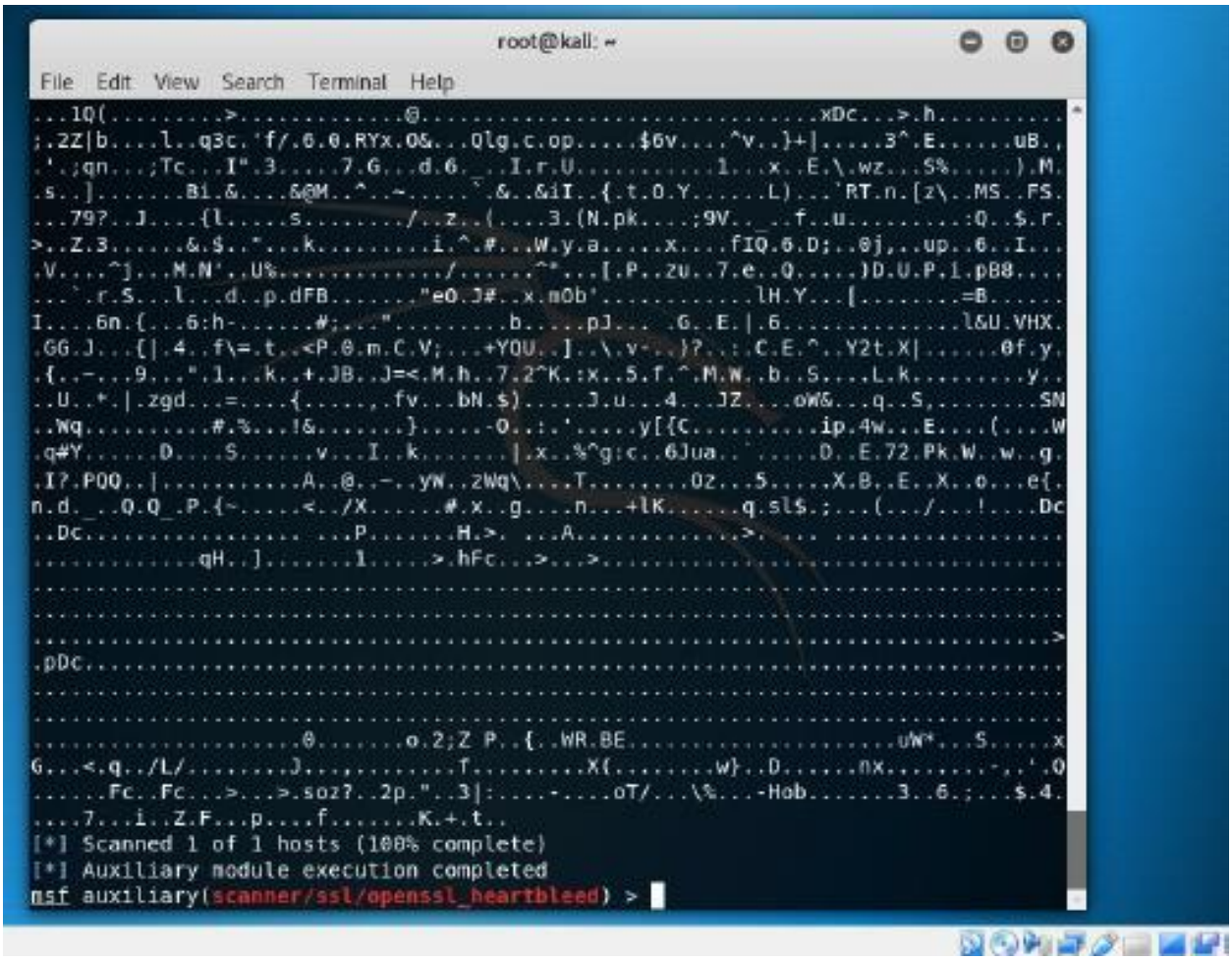

Figure: Heartbleed attack's Results"