Project 2: Exploitation

Miguel Archundia
Jacobo Sanchez
Derek Hernandez
Salvador Romero Mondragon

Section I (Introduction)

Summary:

The main point of Project two was to not only setup the network and root permissions but test whether the custom attack scripts will work in the task of retrieving the files from a remote computer. This task presents the strategy of manipulating stack memory. After the attack is successful we then test what kind of defenses are used to protect memory manipulation.

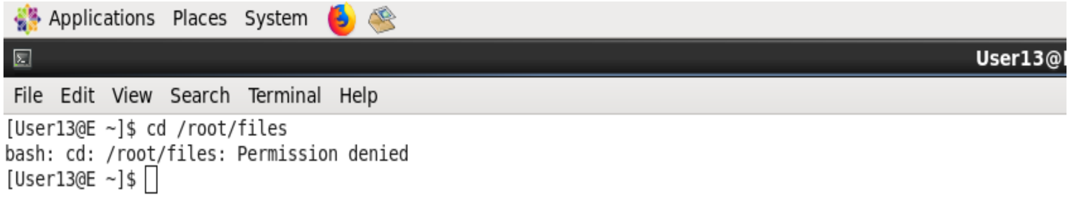
Responsibilities:

Project: Task I and II were performed by Jacob and Miguel with support from Salvador. Task III and IV were performed by Jacob and Miguel with support from Salvador and Derek.

Report: Sections I-II were done by Miguel, Jacob edited the final report and wrote section IV with a suggestion from Derek, section I was done by Derek, sections II-III were done by Salvador.

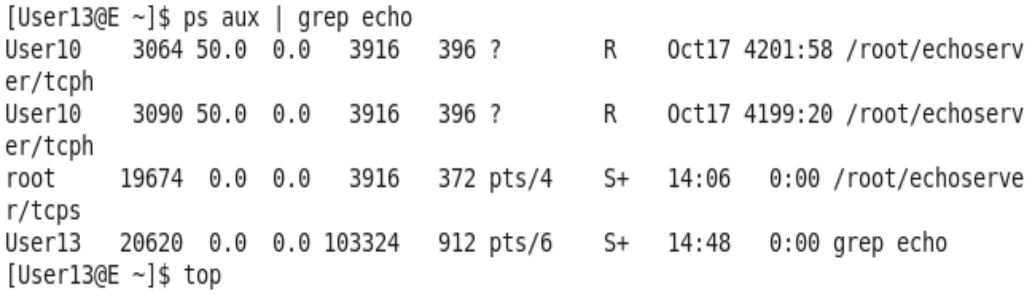
Section II (Task II)

1. .
    a. Not able to access /root/files without root password


Applications Places System
User13@
File Edit View Search Terminal Help

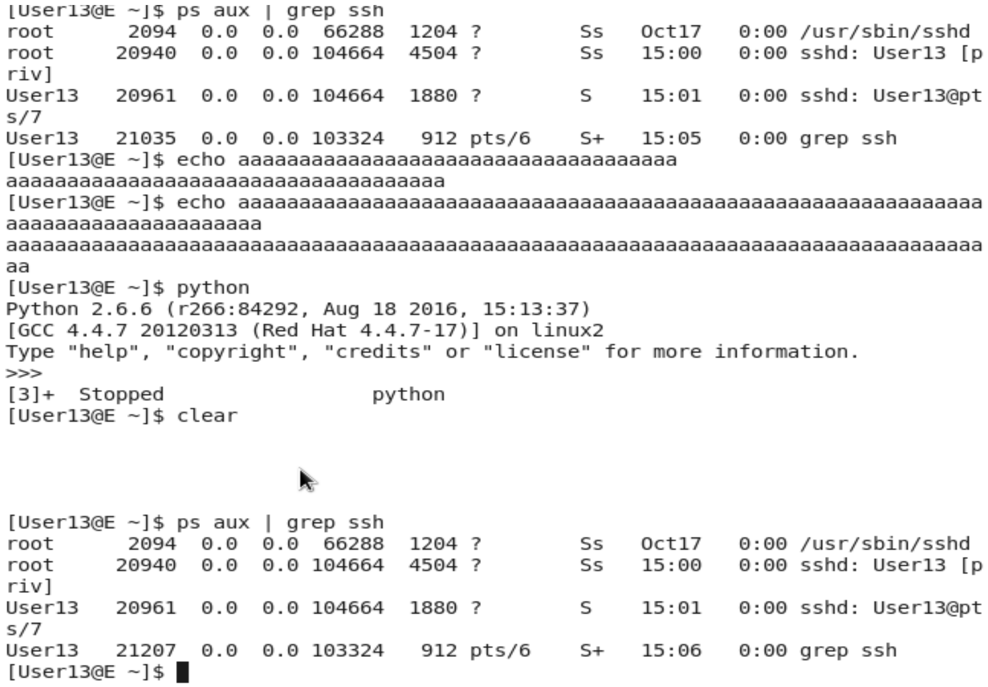
```
[User13@E ~]$ cd /root/files
bash: cd: /root/files: Permission denied
[User13@E ~]$ 
```

    b. The maximum allowable size of the data allowable in an echo command is 65,507 bytes. Anything more than 8 bytes.
    c. Showing user ID for echo services in E.2



```
[User13@E ~]$ ps aux | grep echo
User10    3064 50.0  0.0   3916   396 ?       R    Oct17 4201:58 /root/echoserv
er/tcph
User10    3090 50.0  0.0   3916   396 ?       R    Oct17 4199:20 /root/echoserv
er/tcph
root     19674  0.0  0.0   3916   372 pts/4   S+   14:06   0:00 /root/echoserve
r/tcps
User13   20620  0.0  0.0 103324   912 pts/6   S+   14:48   0:00 grep echo
[User13@E ~]$ top
```

    d. Showing user ID that is running SSH services on computer E.2.

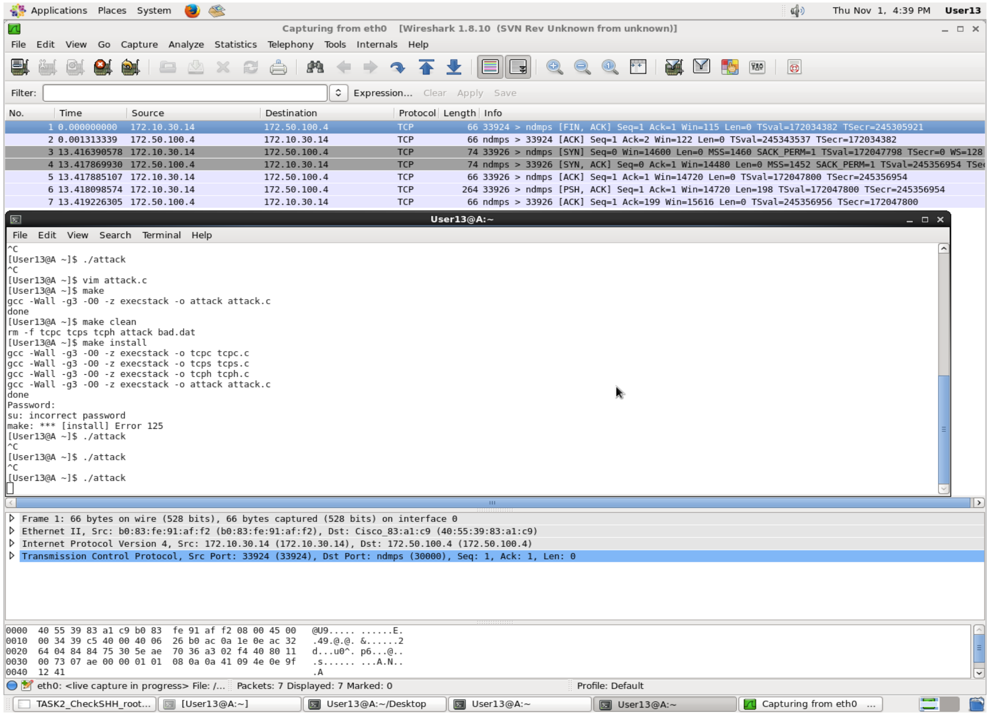


```
[User13@E ~]$ ps aux | grep ssh
root      2094  0.0  0.0  66288  1204 ?        Ss   Oct17   0:00 /usr/sbin/sshd
root     20940  0.0  0.0 104664  4504 ?        Ss   15:00   0:00 sshd: User13 [p
riv]
User13   20961  0.0  0.0 104664  1880 ?        S    15:01   0:00 sshd: User13@pt
s/7
User13   21035  0.0  0.0 103324   912 pts/6    S+   15:05   0:00 grep ssh
[User13@E ~]$ echo aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
[User13@E ~]$ echo aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aa
[User13@E ~]$ python
Python 2.6.6 (r266:84292, Aug 18 2016, 15:13:37)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-17)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
[3]+  Stopped                 python
[User13@E ~]$ clear



[User13@E ~]$ ps aux | grep ssh
root      2094  0.0  0.0  66288  1204 ?        Ss   Oct17   0:00 /usr/sbin/sshd
root     20940  0.0  0.0 104664  4504 ?        Ss   15:00   0:00 sshd: User13 [p
riv]
User13   20961  0.0  0.0 104664  1880 ?        S    15:01   0:00 sshd: User13@pt
s/7
User13   21207  0.0  0.0 103324   912 pts/6    S+   15:06   0:00 grep ssh
[User13@E ~]$ █
```

Section III (Task III)

1. .
    a. Showing that echo services can be exploited by the provided shell code.(shown in the image below)

b. Showing the exploited packets captured in Computer E.2 (shown in the image below)



Applications Places System
Thu Nov 1, 4:39 PM User13
Capturing from eth0 [Wireshark 1.8.10 (SVN Rev Unknown from unknown)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save
User13@A:~
File Edit View Search Terminal Help
TASK2_CheckSHH_root...
[User13@A:~]
User13@A:~/Desktop
User13@A:~
User13@A:~
Capturing from eth0 ...

```
No.     Time          Source         Destination    Protocol Length Info
      1 0.000000000   172.10.30.14   172.50.100.4   TCP       66 33924 > ndmps [FIN, ACK] Seq=1 Ack=1 Win=115 Len=0 TSval=172034382 TSecr=245305921
      2 0.001313339   172.50.100.4   172.10.30.14   TCP       66 ndmps > 33924 [ACK] Seq=1 Ack=2 Win=122 Len=0 TSval=245343537 TSecr=172034382
      3 13.416390578  172.10.30.14   172.50.100.4   TCP       74 33926 > ndmps [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=172047798 TSecr=0 WS=128
      4 13.417869930  172.50.100.4   172.10.30.14   TCP       74 ndmps > 33926 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1452 SACK_PERM=1 TSval=245356954 TSe
      5 13.417885107  172.10.30.14   172.50.100.4   TCP       66 33926 > ndmps [ACK] Seq=1 Ack=1 Win=14720 Len=0 TSval=172047800 TSecr=245356954
      6 13.418098574  172.10.30.14   172.50.100.4   TCP      264 33926 > ndmps [PSH, ACK] Seq=1 Ack=1 Win=14720 Len=198 TSval=172047800 TSecr=245356954
      7 13.419226305  172.50.100.4   172.10.30.14   TCP       66 ndmps > 33926 [ACK] Seq=1 Ack=199 Win=15616 Len=0 TSval=245356956 TSecr=172047800
```

```
^C
[User13@A ~]$ ./attack
^C
[User13@A ~]$ vim attack.c
[User13@A ~]$ make
gcc -Wall -g3 -O0 -z execstack -o attack attack.c
done
[User13@A ~]$ make clean
rm -f tcpc tcps tcph attack bad.dat
[User13@A ~]$ make install
gcc -Wall -g3 -O0 -z execstack -o tcpc tcpc.c
gcc -Wall -g3 -O0 -z execstack -o tcps tcps.c
gcc -Wall -g3 -O0 -z execstack -o tcph tcph.c
gcc -Wall -g3 -O0 -z execstack -o attack attack.c
done
Password:
su: incorrect password
make: *** [install] Error 125
[User13@A ~]$ ./attack
^C
[User13@A ~]$ ./attack
^C
[User13@A ~]$ ./attack
```

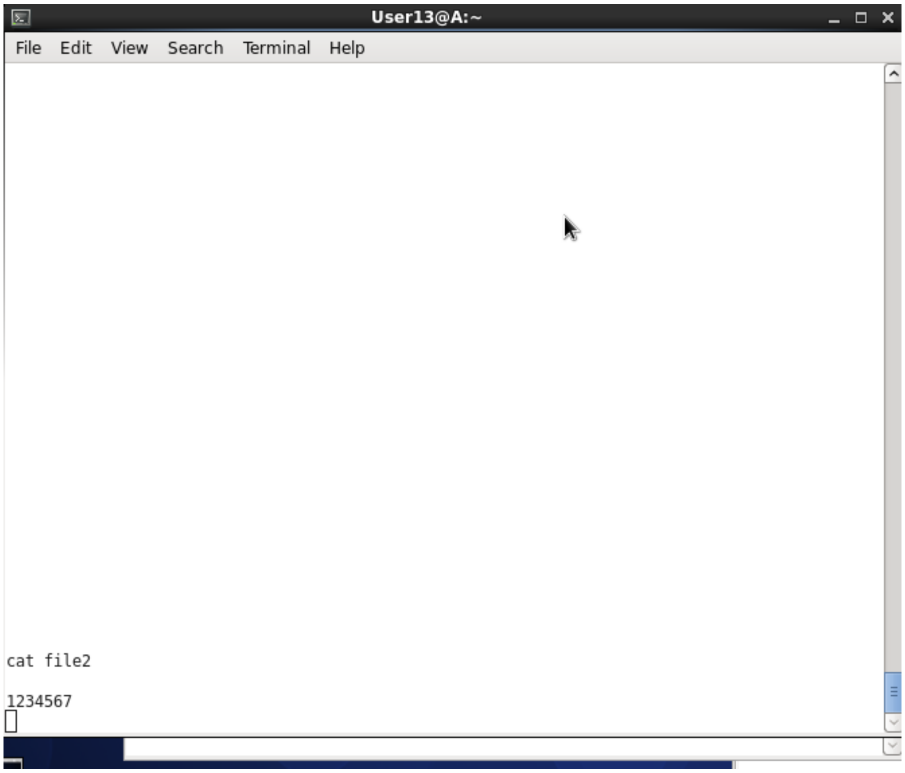
```
▷ Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
▷ Ethernet II, Src: b0:83:fe:91:af:f2 (b0:83:fe:91:af:f2), Dst: Cisco_83:a1:c9 (40:55:39:83:a1:c9)
▷ Internet Protocol Version 4, Src: 172.10.30.14 (172.10.30.14), Dst: 172.50.100.4 (172.50.100.4)
▷ Transmission Control Protocol, Src Port: 33924 (33924), Dst Port: ndmps (30000), Seq: 1, Ack: 1, Len: 0
```

```
0000  40 55 39 83 a1 c9 b0 83  fe 91 af f2 08 00 45 00   @U9..... ......E.
0010  00 34 39 c5 40 00 40 06  26 b0 ac 0a 1e 0e ac 32   .49.@.@. &......2
0020  64 04 84 84 75 30 5e ae  70 36 a3 02 f4 40 80 11   d...u0^. p6...@..
0030  00 73 07 ae 00 00 01 01  08 0a 0a 41 09 4e 0e 9f   .s...... ...A.N..
0040  12 41                                              .A
```

```
eth0: <live capture in progress> File: /...   Packets: 7 Displayed: 7 Marked: 0   Profile: Default
```

c. Report how you retrieve the files from computer E.2 to E.B. Give steps in detail
Solution provided by Jacob. After hacking into the server, target files were copied into the /var/www/html directory. The files were then downloaded to the attacking server using the wget function, i.e. attacked server ip/filename.

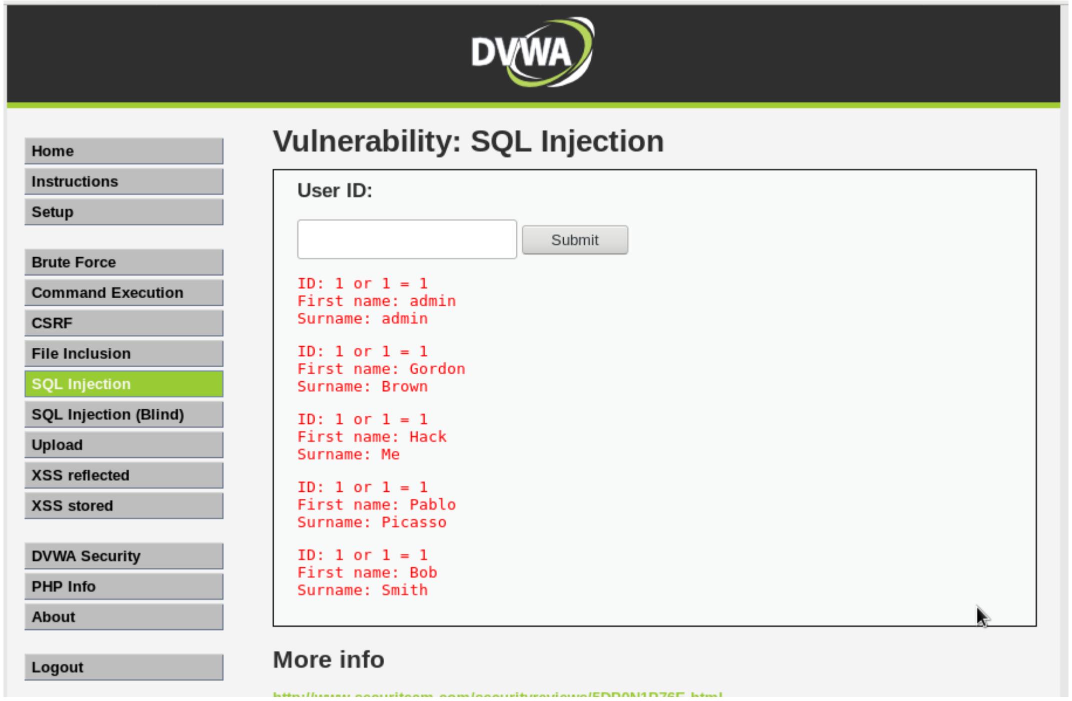
d. .



```
cat file2

1234567
```

e. The statement we used to display all of the user ID is (1 or 1 = 1)

f. Showing the screenshot of the web page that show all user ID’s, first names, and last names.



Vulnerability: SQL Injection

User ID:

```
ID: 1 or 1 = 1
First name: admin
Surname: admin

ID: 1 or 1 = 1
First name: Gordon
Surname: Brown

ID: 1 or 1 = 1
First name: Hack
Surname: Me

ID: 1 or 1 = 1
First name: Pablo
Surname: Picasso

ID: 1 or 1 = 1
First name: Bob
Surname: Smith
```

Section
IV

a) Discuss the reason that randomization can defeat the attack.

Randomization does not defeat attacks but minimizes the chances of script attack success by randomizing target addresses. This makes attacks that rely on using specific memory locations more difficult.

b) Assume only the low 16 bits of the stack address is randomized. What is the probability that an exploiting packet can compromise the server? Assume an attacker can send 10 exploiting packets every second. How long can the attacker compromise the server?

Assuming isolated guessing where address space is re-randomized after each attack attempt, the probability of server compromisation is g = 1 - (1 -2^-N)^a where a is 0 <= a => g = 1 - (1 - 2^16) ^10 = .001526*100% = .1526%

https://en.wikipedia.org/wiki/Address_space_layout_randomization

The attacker can compromise the server until the server OS reacts to program crashes due to attack failures.

c) Discuss the reason that exec-shield can defeat the attack.

Exec-shield defeats the attack because the script changes the permissions of executables in the memory.

d) Discuss if exec-shield prevents stack overflow. If not, what attack can be achieved?

Exec shield in combination with position independent executables randomization patch make heap and format string exploits almost impossible. An application that is not fully compatible is still vulnerable, however.