M1522.000800 System Programming
Fall 2018

System Programming
Buflab Report

박승일
2016-10760

1. <lab> Buffer Lab

The goal of the buffer lab is tomake exploit fileand attack stack frame. By this, We can understand stackframe structure, and IA-32 calling convention. And how to prevent or make stackframe attack.

2. Implementation

The buffer lab consists of five phases: Smoke, Sparkler, Firecracker, Dynamite and Nitroglycerin. Each phases have different attack method, but fundamentally use buffer overflow. Push big object in buffer and overwrite some data to made stack frame for attack.

2.1 Smoke

Smoke case is just alter returnaddress to smoke call address. Smoke address found by 'break smoke'. Buffer+old%ebx, old%ebp size is 52bytes and 4bytes return address is upon old%ebp.

Perl -e 'print "61 "x52, "e8 8b 04 08"'

2.2 Sparkler

Sparkler case is also alter return address to call fizz. First, find first address offizz by 'break fizz' 'bisas'. And find how to get val1, val2. By this val1, val2 is located upon start address + 4bytes. And if val1 is 0xffffffff and val2 is 0x00000000 condition could be true, So push trash bit 52bytes and fizz start address and trashbit 4bytes and val1(0xffffffff) and val2(0x0000000).

perl -e 'print "61 "x52, "12 8c 04 08 ", "61 "x4, "FF "x4, "00 "x4 '> hex4

2.3 Firecracker

*Firecracker case is first case use instruction sequences. We make global_value to make condition true, In my case 0x0000000e by movl and push bang's start address and ret. We need to find start address of buffer, so break *getbuf + 21 and see %eax. And make this instructions binary codes. Lastly, push this binary codes and trash bits 36 bytes and start address of buffer*

Movl $0x0000000x, 0x804d120
Pushl $0x08048c81
Ret

Perl -e 'print "c7 05 20 d1 04 08 0e 00 00 00 68 81 8c 04 08 c3 ", "61 "x36, "10 3a 68 55 " '>hex5

2.4 Dynamite

*Dynamite case is not return to attack function.This return to test case but make return value is cookie not 1. We use overflow this case also, but this case we saved old %ebp, old%ebx because it have to be saved for callee. We find this by 'break *test+20','I r'. And we need to find where we have to push val value. And we know it by 'break *test+20', 'I r' and get %eax value. Now we make instructions. First movl cookies to val value address. And push test next instructions address and ret.Finally push this instructions sequence binary code and trash bits 28bytes and old %ebx, old%ebp, and start address of buffer.*

Movl $0x4396204e, 0x55683a64
Push $0x8048e65
Ret
Perl -e 'print "c7 05 64 3a 68 55 4e 20 96 43 68 55 68 65 8e 04 08 c3 " , "61 "x20, "00 00 00 00 70 3a 68 55 10 3a 68 55 " '> hex6

2.5 Nitroglycerin

*Nitroglycerin case use testn and getbufn. We will call testn 5 times and it make stack frame of getbufn different. So we will arrange instructions sequence location. It's start address not change over 240, so we put 509 nop instructions and 15 byte instruction binary codes and return address. However, return address have to be first return address + 240~269 to make it work. We find return address use break *getbufn+18 and watch %eax. So if return address change, it will be okay because it starts from over 240~269 so it always counter nop and next instructions sequence. And instruction have to repair %esp andmake val(%eax) be cookieand return next instruction of testn.*

Lea 0x28 (%esp), %ebp
Movl $0x4396204e, %eax
Pushl $0x08048dd1
Ret

Perl -e 'print "90 "x509, "8d 6c 24 28 b8 4e 20 96 43 68 d1 8d 04 08 c3 ", "38 39 68 55 " '>hex6

3. Conclusion

I really enjoy this project to attack stack frame. I could learn what is stack frame and how to use it. Also, learn there is many way to attack stack frame and prevent it. Also know make instruction sequence and make it binary to apply into bufbomb. And now can debug what is happen inside stack frame using gdb.