Lecture Outline

- Notion of binding time
- Object lifetime and storage management

- An aside: Stack Smashing 101
 - Slides courtesy of RPISEC/MBE
- Scoping
 - Static scoping
 - Dynamic scoping

Stack Frames

In x86-64 RBP is fp and RSP is sp. Define the stack frame for the currently executing function

- local variables
- pointer to previous frame
- return address

09/09/2019

```
void foo() {
  long long x = 0x1337;
  char str[16];
  strcpy(str, "ABCDEFGH0123456");
```

```
mov
                   eax, [rax+128h]
             test
                   eax, eax
                   short loc 3AFR
            note: for 64bit,
            each 'slot' is 8 bytes
             add
                   edx, eax
                   rax, [rbp+var_28]
                   [rax+OCh], edx
                         🗓 <- local var
     0x1337
                           <- local var
   "ABCDEFGH
                         [cSt11char_traitsIcESaIc
  0123456\0"
                          <- caller
0x7fff10203040
                           frame RBP
    0x400134
                          <- return
```

short loc 30 Cress rsi, aWannaCheatYes1 rax, cs:_ZSt4cout_ptr_

rax. [rbp+var 14]

__ZStlsISt11char_traitsIcEERSt1

rdi, rax

call

= qword ptr -30h

= qword ptr -28h = dword ptr -14h

rbp

rbx rsp, 28h

rbp, rsp

[rbp+var_28], rdi [rbp+var_30], rsi rax, [rbp+var_28]

push

push

mov

var 30

var_28

var_14

loc 30FB:

MBE - Overview

RSP ->

RBP ->

What is corruption?

```
var_30
                 = qword ptr -30h
var_28
                = qword ptr -28h
var_14
                 = dword ptr -14h
```

```
; __unwind {
                 push
                          rbp
                 mov
                          rbp, rsp
                 push
                          rbx
```

sub

loc_30FB:

[rbp+var_28], rdi mov [rbp+var_30], rsi mov rax, [rbp+var_28] mov eax, [rax+128h] test eax, eax

rsp, 28h

short loc_30FB rx, [rbp+var_28] edx, [rax+0Ch]

> mov rax, [rbp+var_28] ecx, [rax+14h] mov rax, [rbp+var_30] eax, [rax+10h] mov ebx, ecx

ebx, eax sub eax, ebx add edx, eax

rax, [rbp+var_28] mov mov [rax+0Ch], edx

rax, [rbp+var_28] eax, [rax+0Ch] mov

eax, eax test jns loc_31C4 rax, [rbp+var_28]

add rax, 18h rsi, rax

rax, cs:_ZSt4cout_ptr mov rdi, rax call __ZStlsIcSt11char_traitsIcESaIc

rsi, aIsDead lea rdi, rax mov call __ZStlsISt11char_traitsIcEERSt13

rdx, cs:_ZSt4endlIcSt11char_tra: rsi, rdx mov rdi, rax call __ZNSolsEPFRSoS_E ; std::ostrea rax, [rbp+var_28]

eax, [rax+8] mov eax, eax test short loc 31BD

mov

call

l ea

lea rsi, aWannaCheatYes1; "wanna cl rax, cs:_ZSt4cout_ptr mov mov rdi, rax

rax. [rbp+var 14]

__ZStlsISt11char_traitsIcEERSt13

So what happens if a programer makes a simp mistake:

```
char foo[64];
```

int money = 0;

gets(foo);

gets()?

```
var_30
                = qword ptr -30h
var_28
                = qword ptr -28h
                = dword ptr -14h
var_14
; __unwind {
                push
                         rbp
                mov
                         rbp, rsp
                push
                         rbx
                         rsp, 28h
                sub
                         [rbp+var_28], rdi
                mov
```

jns

add

mov

call

loc_31C4

rax, 18h rsi, rax

rdi, rax

rax, [rbp+var_28]

rax, cs:_ZSt4cout_ptr

__ZStlsIcSt11char_traitsIcESaIc

```
NAME
    gets - get a string from standard input (DEPRECATED)

SYNOPSIS
    #include <stdio.h>
    char *gets(char *s);

DESCRIPTION
    Never use this function.

gets() reads a line from stdin into the buffer pointed to by s until either a terminating new-line or EOF, which it replaces with a null byte ('\0'). No check for buffer overrun is performed (see BUGS below).

— DO NOT EVER USE

test eax, eax
```

scanf("%s", ...) as well

• So what happens if we give this program a punch so fice of the program a pounch so fice of the program a p

mov eax, [rax+8]
test eax, eax
inz short loc_31BD
lea rsi, aWannaCheatYes1; "wanna ch
mov rax, cs:_ZSt4cout_ptr
mov rdi, rax
call __ZStlsISt11char_traitsIcEERSt13
lea rax, [rbp+var 14]

var_30 = gword ptr -30h var_28 = qword ptr -28h var_14 = dword ptr -14h

push

mov rbp, rsp push rbx rsp, 28h sub

rbp

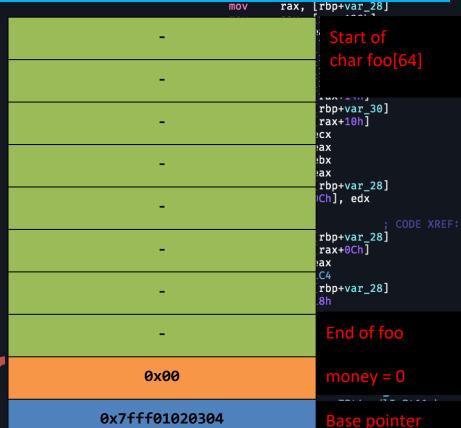
[rbp+var_28], rdi mov

[rbp+var_30], rsi

main() has a

stack frame

- Contains local variables
- Pointer to previous frame
- Return address



Not supposed to touch

Higher Memory

Lower Memory

0x40051f **RETURN ADDRESS**

short loc 31BD

```
var 30
                = gword ptr -30h
var_28
                = qword ptr -28h
var_14
                = dword ptr -14h
```

; __unwind {

push rbp mov rbp, rsp push rbx rsp, 28h sub

[rbp+var_28], rdi [rbp+var_30], rsi

rax, [rbp+var_28]

As gets() continues to Lower Memory read input, we fill up the 64 bytes allocated for buffer foo

0x4141414141414141

0x4141414141414141

0x4141414141414141

0x4141414141414141

0x4141414141414141

0x4141414141414141

0x4141414141414141

0x4141414141414141

0x00

0x7fff01020304

0x40051f

Start of char foo[64]

ŀbх rbp+var_28] Ch], edx

rbp+var_30]

rax+10h] :CX

ax

rbp+var 28]

rax+0Ch] ax .C4 rbp+var_28]

End of foo

Base pointer

RETURN ADDRESS

Not supposed to touch

Higher Memory

```
var 30
                = gword ptr -30h
var_28
                = qword ptr -28h
var_14
                = dword ptr -14h
```

; __unwind {

0x4141414141414141

push rbp mov rbp, rsp push rbx rsp, 28h sub [rbp+var_28], rdi [rbp+var_30], rsi rax, [rbp+var_28]

Start of

rbp+var_28] Ch], edx

rbp+var 28]

rbp+var_28]

End of foo

rax+0Ch]

char foo[64]

As gets() continues to Lower Memory read input, we fill up the 64 bytes allocated for foo

Go far enough, it corrupts important data!

0x4141414141414141

rbp+var_30] 0x4141414141414141 rax+10h] :CX ax bx 0x4141414141414141

0x4141414141414141

0x4141414141414141

0x4141414141414141

0x4141414141414141

ax

0x7fff01020304

0x41

0x40051f

Base pointer

RETURN ADDRESS

Not supposed to touch

Higher Memory

09/09/2019

Memory

short loc 31BD lea rsi, aWannaCheatYes1; "wanna cl rax, cs:_ZSt4cout_ptr_ mov rdi, rax call __ZStlsISt11char_traitsIcEERSt13 rax. [rbp+var 14]

We can give ourselves money

var_30 = gword ptr -30h var_28 = qword ptr -28h var_14 = dword ptr -14h

push

; __unwind {

0x02

mov rbp, rsp push rbx sub rsp, 28h

rbp

- [rbp+var_28], rdi mov [rbp+var_30], rsi rax, [rbp+var_28]
- mov eax, [rax+128h] test eax, eax
- jnz short loc_30FB rax, [rbp+var_28] mov
- edx, [rax+0Ch] rax, [rbp+var_28]
- If we want to set money to 0x1337beef weigh
 - [rbs var_30] eax, [rax+10h] mov ebx, ecx
 - ebx. eax sub eax, ebx
 - edx, eax
 - Most x86 machines are little endian (little byte goes first)x, [rbp+var_28]
 - Meaning the byte order for numbers is "backwards" in memory 28]

0x03

0x01020304 would be

- eax, eax test jns loc_31C4 rax, [rbp+var_28] mov add rax, 18h mov rsi, rax
- rax, cs:_ZSt4cout_ptr li, rax

eax, [rax+0Ch]

- ZStlsIcSt11char_traitsIcESaIc 0x01 i, aIsDead li, rax
- call __ZStlsISt11char_traitsIcEERSt13 mov rdx, cs:_ZSt4endlIcSt11char_tra:
- rsi, rdx mov rdi, rax call __ZNSolsEPFRSoS_E ; std::ostrea rax, [rbp+var_28]
- eax, [rax+8]
- test eax, eax short loc 31BD

l ea

lea rsi, aWannaCheatYes1; "wanna cl rax, cs:_ZSt4cout_ptr mov mov rdi, rax call __ZStlsISt11char_traitsIcEERSt13

rax. [rbp+var 14]

0x04

know:

What else can we corrupt?

= dword ptr -14h var_14 ; __unwind {

push

var_30

var_28

rbp mov rbp, rsp push rbx sub

= gword ptr -30h

= qword ptr -28h

- rsp, 28h [rbp+var_28], rdi mov [rbp+var_30], rsi mov rax, [rbp+var_28]
- mov eax, [rax+128h] test eax, eax
- jnz short loc_30FB rax, [rbp+var_28] edx, [rax+0Ch]
- rax, [rbp+var_28] ecx, [rax+14h] mov
- rax, [rbp+var_30] eax, [rax+10h]
- ebx, ecx. What happens if you corrupt further? When
 - add edx, eax rax, [rbp+var_28] mov mov [rax+0Ch], edx
 - What was that about a return address?....

```
rax, [rbp+var_28]
        eax, [rax+0Ch]
        eax, eax
test
jns
        loc_31C4
        rax, [rbp+var_28]
mov
add
        rax, 18h
        rsi, rax
        rax, cs:_ZSt4cout_ptr
mov
        rdi, rax
call
         __ZStlsIcSt11char_traitsIcESaIc
        rsi, aIsDead
lea
        rdi. rax
call
         __ZStlsISt11char_traitsIcEERSt13
        rdx, cs:_ZSt4endlIcSt11char_tra:
        rsi, rdx
        rdi, rax
call
         __ZNSolsEPFRSoS_E ; std::ostrea
        rax, [rbp+var_28]
```

rsi, aWannaCheatYes1; rax, cs:_ZSt4cout_ptr

rax. [rbp+var 14]

__ZStlsISt11char_traitsIcEERSt13

eax, [rax+8]

eax, eax short loc 31BD

rdi, rax

test

lea

mov

mov call

l ea

segfault?

```
= dword ptr -14h
                                                                  var_14
                                                                  ; __unwind {
Stack Smashing 201
                                                                               push
                                                                                     rbp
                                                                               mov
                                                                                     rbp, rsp
                                                                               push
                                                                                     rbx
                                                                                     rsp, 28h
                                                                                     [rbp+var_28], rdi
                                                                               mov
                                                                                     [rbp+var_30], rsi
                                                                               mov
                                                                                     rax, [rbp+var_28]
                                                                                     eax, [rax+128h]
                                                                               test
                                                                                     eax, eax
     int func() {
                                                                               jnz
                                                                                     short loc_30FB
                                                                                     rax, [rbp+var_28]
                                                                                     edx, [rax+0Ch]
                 puts("Hello World");
                                                                                     rax, [rbp+var_28]
                                                                                     ecx, [rax+14h]
                                                                               mov
                                                                                     rax, [rbp+var_30]
                 return 17;
                                                                                     eax, [rax+10h]
                                                                                     ebx, ecx
                                                                                     ebx, eax
                                                                                     eax. ebx
                                                                               add
                                                                                     edx, eax
                                                                                     rax, [rbp+var_28]
     int main() {
                                                                                     [rax+0Ch], edx
                                                                  loc_30FB:
                 int res = func();
                                                                                     rax, [rbp+var_28]
                                                                                     eax, [rax+0Ch]
                                                                               mov
                                                                                     eax, eax
                                                                               test
                 return 0;
                                                                               jns
                                                                                     loc_31C4
                                                                                     rax, [rbp+var_28]
                                                                               add
                                                                                     rax, 18h
                                                                                     rsi, rax
                                                                                     rax, cs:_ZSt4cout_ptr
                                                                               mov
                                                                                     rdi, rax
  When func() is called, runtime stores the return
                                                                                     TO CITE S traits I CESa I CI
                                                                                     rdi, rax
    on the stack (i.e., the address of the instruction
                                                                                      ZStlsISt11char_traitsIcEERSt1
                                                                                    rdx [15] [St4endlIcSt11char_tra:
                                                                                     rsi, rdx
                                                                                     rdi, rax
    immediately follows call func in main)
                                                                               call
                                                                                     __ZNSolsEPFRSoS_E ; std::ostrea
                                                                                     rax, [rbp+var_28]
                                                                                     eax, [rax+8]
                                                                               mov
                                                                                     eax, eax
                                                                               test
                                                                                     short loc 31BD
                                                                               lea
                                                                                     rsi, aWannaCheatYes1; "wanna cl
                                                                                     rax, cs:_ZSt4cout_ptr
  09/09/2019
                                                Memory
                                                                               mov
                                                                                     rdi, rax
                                                                                     __ZStlsISt11char_traitsIcEERSt1
```

= qword ptr -30h

= qword ptr -28h

call

l ea

rax. [rbp+var 14]

var_30 var_28

Before the call:

```
=> 0x40051a <main+13>: call
                               0x4004f6 <func>
   0x40051f <main+18>:
                               DWORD PTR [rbp-0x4],eax
                        mov
  0x400522 <main+21>:
                               eax.0x0
                        mov
   0x400527 <main+26>:
                        leave
  0x400528 < main + 27 > :
No argument
0000| 0x7fffffffe0f0 --> 0x7ffffffffe1e0 --> 0x1
00081
     0x7ffffffffe0f8 --> 0x0
     0x7fffffffe100 --> 0x400530 (< libc csu init>:
00161
```

```
= dword ptr -14h
var_14
; __unwind {
                push
                         rbp
                mov
                         rbp, rsp
                push
                         rbx
                         rsp, 28h
                sub
                         [rbp+var_28], rdi
                mov
                         [rbp+var_30], rsi
                mov
                         rax, [rbp+var_28]
                mov
                         eax, [rax+128h]
                         eax, eax
                test
                 jnz
                         short loc_30FB
                         rax, [rbp+var_28]
                         edx, [rax+0Ch]
                         rax, [rbp+var_28]
                         ecx, [rax+14h]
                mov
                         rax, [rbp+var_30]
                         eax, [rax+10h]
                mov
                         ebx, ecx
                         ebx, eax
                sub
                         eax. ebx
                add
                         edx, eax
                         rax, [rbp+var_28]
                mov
                mov
                         [rax+0Ch], edx
loc_30FB:
                         rax, [rbp+var_28]
                         eax, [rax+0Ch]
                mov
                         eax, eax
                test
                 jns
                         loc_31C4
                         rax, [rbp+var_28]
                mov
                add
                         rax, 18h
                         rsi, rax
                         rax, cs:_ZSt4cout_ptr
                mov
                         rdi, rax
                mov
                call
                         __ZStlsIcSt11char_traitsIcESaIc
                         rsi, aIsDead ; " is dead!"
                lea
                         rdi, rax
                         __ZStlsISt11char_traitsIcEERSt13
                call
                         rdx, cs:_ZSt4endlIcSt11char_tra:
                         rsi, rdx
                         rdi, rax
                mov
                call
                         __ZNSolsEPFRSoS_E ; std::ostrear
                         rax, [rbp+var_28]
                         eax, [rax+8]
                mov
                test
                         eax, eax
                         short loc 31BD
                         rsi, aWannaCheatYes1; "wanna cl
                         rax, cs:_ZSt4cout_ptr
11
                mov
                mov
                         rdi, rax
                call
                         __ZStlsISt11char_traitsIcEERSt13
                         rax. [rbp+var 14]
                l ea
```

= qword ptr -30h

= qword ptr -28h

var_30

var_28

```
Before the call:
```

```
0x4004f6 <func>
=> 0x40051a <main+13>:
                        call
   0x40051f <main+18>:
                               DWORD PTR [rbp-0x4],eax
                        mov
  0x400522 <main+21>:
                               eax.0x0
                        mov
   0x400527 <main+26>:
                        leave
  0x400528 <main+27>:
No argument
     0x7fffffffe0f0 --> 0x7fffffffele0 --> 0x1
0008
     0x7ffffffffe0f8 --> 0x0
     0x7fffffffe100 --> 0x400530 (< libc csu init>:
00161
```

```
var_14
                                 = dword ptr -14h
                                 push
                                         rbp
                                 mov
                                         rbp, rsp
                                 push
                                         rbx
                                         rsp, 28h
                                 sub
                                         [rbp+var_28], rdi
                                         [rbp+var_30], rsi
                                         rax, [rbp+var_28]
                                         eax, [rax+128h]
                                 mov
                                         eax, eax
                                 test
                                      Practa by 1968
                       Afterith
                                         edx, [rax+0Ch]
                                         rax, [rbp+var_28]
                                         acv [rav+14h]
                                 may
=> 0x4004f6 <func>:
                              rbp
                       push
                                                   /ar_30]
  0x4004f7 <func+1>:
                              rbp,rsp
                       mov
                                                   .0hl
                              rdi,[rip+0xb3]
  0x4004fa <func+4>:
                        lea
  0x400501 <func+11>:
   %x400506 <func+16>:
                              eax,0x11
                        0x40051f (<main+18>:
                                               mov 'ar_28]
     0x7ff ffffe0f0 --> 0x/ffffffffe0 --> 0x1
                                         rax, [rbp+var_28]
                                         eax, [rax+0Ch]
                                 mov
                                         eax, eax
                                 test
                                  jns
                                         loc_31C4
                                         rax, [rbp+var_28]
                                         rax, 18h
                                         rsi, rax
                                         rax, cs:_ZSt4cout_ptr
                                         rdi, rax
                                      pollics baback dead!"
     Return address
                                          __ZStlsISt11char_traitsIcEERSt1:
                                       off in Main 1st 1char_train
       to where it left
                                         rdi, rax
                                         __ZNSolsEPFRSoS_E ; std::ostrear
                                 call
                                         rax, [rbp+var_28]
                                         eax, [rax+8]
                                 test
                                         eax, eax
                                         short loc 31BD
```

mov

mov

call

l ea

rsi, aWannaCheatYes1; "wanna cl

__ZStlsISt11char_traitsIcEERSt1

rax, cs:_ZSt4cout_ptr

rax. [rbp+var 14]

rdi, rax

= qword ptr -30h

= qword ptr -28h

var_30

var_28

```
var_30 = qword ptr -30h
var_28 = qword ptr -28h
var_14 = dword ptr -14h
```

push

test

test

lea

mov

mov call

l ea

; __unwind {

```
mov rbp, rsp
push rbx
sub rsp, 28h
mov [rbp+var_28], rdi
mov [rbp+var_30], rsi
mov rax, [rbp+var_28]
mov eax, [rax+128h]
```

eax, eax

rbp

Returning just takes whatever is on the top of the long of the lon

and jumps there, equivalently: pop rip

About to return:

```
=> 0x40050c <func+22>:
   0x40050d <main>:
                         push
                                 rbp
   0x40050e < main+1>:
                         mov
                                 rbp,rsp
   0x400511 <main+4>:
                         sub
                                 rsp,0x10
   0x400515 <main+8>:
                         mov
                                eax,0x0
                                    (<main+18>:
      0x7fffffffe0f0 --> ux/IIIIIIIIe1eu --> ux
0016| 0x7ffffffffe0f8 --> 0x0
```

```
ierstack,
        edx, [rax+0Ch]
mov
        rax, [rbp+var_28]
        ecx, [rax+14h]
mov
mov
        rax, [rbp+var_30]
        eax, [rax+10h]
mov
mov
        ebx, ecx
sub
        ebx, eax
        eax, ebx
add
        edx, eax
        rax, [rbp+var_28]
mov
mov
        [rax+0Ch], edx
```

loc_30FB:

```
rax, [rbp+var_28]
        eax, [rax+0Ch]
mov
        eax, eax
test
jns
        loc_31C4
        rax, [rbp+var_28]
add
        rax, 18h
        rsi, rax
        rax, cs:_ZSt4cout_ptr
mov
        rdi, rax
mov
call
         __ZStlsIcSt11char_traitsIcESaIc
lea
        rsi, aIsDead
        rdi. rax
call
         __ZStlsISt11char_traitsIcEERSt13
        rdx, cs:_ZSt4endlIcSt11char_tra:
        rsi, rdx
        rdi, rax
call
         __ZNSolsEPFRSoS_E ; std::ostrea
```

rax, [rbp+var_28] eax, [rax+8]

rax. [rbp+var 14]

rax, cs:_ZSt4cout_ptr

rsi, aWannaCheatYes1; "wanna cl

__ZStlsISt11char_traitsIcEERSt13

eax, eax

rdi, rax

```
var_30 = qword ptr -30h
var_28 = qword ptr -28h
var_14 = dword ptr -14h
```

push

mov

test

; __unwind {

```
mov rbp, rsp
push rbx
sub rsp, 28h
mov [rbp+var_28], rdi
mov [rbp+var_30], rsi
mov rax, [rbp+var_28]
```

eax, eax

eax, [rax+128h]

rbp

Returning just takes whatever is on the top

and jumps there, equivalently: pop rip

```
mov edx, [rax+0Ch]
mov rax, [rbp+var_28]
mov ecx, [rax+14h]
mov rax, [rbp+var_30]
mov eax, [rax+10h]
mov ebx, ecx
sub ebx, eax
mov eax, ebx
```

rdi, rax

eax, eax

rdi, rax

rax, [rbp+var_28] eax, [rax+8]

rax. [rbp+var 14]

rsi, aWannaCheatYes1; rax, cs:_ZSt4cout_ptr

__ZNSolsEPFRSoS_E ; std::ostrea

__ZStlsISt11char_traitsIcEERSt13

About to return:

```
Returned back to main max, [rbp+var_28]
```

loc_30FB:

```
=> 0x40050c <func+22>:
   0x40050d <main>:
                         push
                                rbp
   0x40050e < main+1>:
                                rbp,rsp
                         mov
   0x400511 <main+4>:
                         sub
                                rsp,0x10
   0x400515 <main+8>:
                         mov
                                eax,0x0
                                    (<main+18>:
     0x7fffffffe0f0 --> 0x7ffffffffele0 --> 0
0016| 0x7ffffffffe0f8 --> 0x0
```

```
rax, [rbp+var_28]
                                                eax, [rax+0Ch]
                                       mov
=> 0x40051f <main+18>:
                                             [rbp-0x4],eax
                                  DWORD PTR
    0x400522 <main+21>:
                          mov
                                  eax,0x0
                                                           ar_28]
    0x400527 <main+26>:
                           leave
    0x400528 <main+27>:
    0x400529:
                         DWORD PTR [rax+0x0]
                                                           t4cout ptr
      0x7fffffffe0f0 --> 0x7ffffffffele0 --> 0x1
                                                           l1char_traitsIcESaIc
                                     (< libc csu init>:
      0x7ffffffffe100 -->
                                                __ZStlsISt11char_traitsIcEERSt13
                                                rdx, cs:_ZSt4endlIcSt11char_tra:
                                                rsi, rdx
                                       mov
```

mov call

test

lea

mov

mov

call

l ea

```
var_30 = qword ptr -30h
var_28 = qword ptr -28h
var_14 = dword ptr -14h
```

mov mov

mov

sub

add

mov

mov

; __unwind {

loc_30FB:

 push
 rbp

 mov
 rbp, rsp

 push
 rbx

 sub
 rsp, 28h

 mov
 [rbp+var_28], rdi

 mov
 [rbp+var_30], rsi

mov rax, [rbp+var_28] mov eax, [rax+128h] test eax, eax

ebx, ecx

ebx, eax eax, ebx

edx, eax

rdi, rax

rdi, rax

eax, [rax+8]

rax, [rbp+var_28]

rsi, aWannaCheatYes1

rax. [rbp+var 14]

rax, cs:_ZSt4cout_ptr

__ZStlsISt11char_traitsIcEERSt1

edx, [rax+0Ch] rax. [rbp+var 28]

[rax+14h]

rax, [rbp+var_30] eax, [rax+10h]

rax, [rbp+var_28]

rax, [rbp+var_28]

[rax+0Ch], edx

Returning just takes whatever is on the top and jumps there, equivalently: pop rip

About What we change this???

Returned back to main.

```
eax, [rax+0Ch]
                                        mov
     <del><4005</del>1f <main+18≯:
                                  DWORD PTR
                                              [rbp-0x4],eax
   ∅x400522 <main+2½>:
                          mov
                                  eax,0x0
                                                            ar_28]
   0x400527 <main+26>:
                          leave
   0x400528 <main 27>:
   0 400529:
                         DWORD PTR [rax+0x0]
                                                            t4cout ptr
                                                            l1char_traitsIcESaIc
0016| 0x7ffffffffe100 --
                                      (< libc csu init>:
                                                 __ZStlsISt11char_traitsIcEERSt1:
                                        call
                                        mov
                                                 rdx, cs:_ZSt4endlIcSt11char_tra:
                                                 rsi, rdx
```

mov.

call

mov

test

Lea

mov

mov

call

1ea

```
var 30
                = gword ptr -30h
var_28
                = qword ptr -28h
var_14
                = dword ptr -14h
```

push

; __unwind {

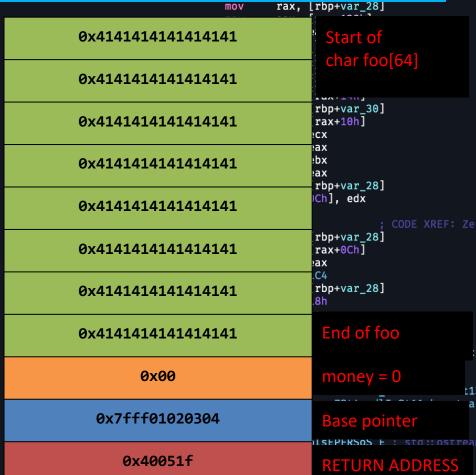
mov rbp, rsp push rbx rsp, 28h sub [rbp+var_28], rdi mov [rbp+var_30], rsi

rbp

Without corruption:

- At the end of the function, it returns
- 0x40051f is popped off the stack and stored in rip
- Control goes to that address

We want to change this



Higher Memory

Lower Memory

```
var_30 = qword ptr -30h
var_28 = qword ptr -28h
var_14 = dword ptr -14h
```

; __unwind {

 push
 rbp

 mov
 rbp, rsp

 push
 rbx

 sub
 rsp, 28h

 mov
 [rbp+var_28], rdi

 mov
 [rbp+var_30], rsi

 mov
 rax, [rbp+var_28]

Corrupted:

- At the end of the function, it returns
- Ox41414141414141 is popped off the stack and stored in rip
- Control goes to that address
- but it's invalid memory...

Segmentation fault

[rbp+var_30], rsi rax, [rbp+var_28] 0x4141414141414141 Start of char foo[64] 0x4141414141414141 rbp+var_30] 0x4141414141414141 [rax+10h] CX ax ŀbх 0x4141414141414141 rbp+var_28] Ch], edx 0x4141414141414141 rbp+var 28] 0x4141414141414141 rax+0Ch] ax .C4 rbp+var_28] 0x4141414141414141 End of foo 0x4141414141414141 0x4141414141414141 0x4141414141414141 Base pointer 0x4141414141414141 **RETURN ADDRESS**

Higher Memory

Memory

Lower Memory