CMSC389R

Binaries II

COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

recap

HW10

--

Questions?

Itinerary

- Review
- Reverse Engineering
  - Static analysis
  - Dynamic analysis
- Tools
- Exercises

Review

- x86 Assembly
  - Registers, instructions, conventions
- Tools
  - objdump, yasm, gdb

Essence of Analysis

- Jump into completely unknown code
- Get high-level idea of operation
  - *Which* parts do computation?
  - *Which* parts do checking?
- Dig into “interesting” parts
  - *What* does this compute?
  - *Why* is there a check?
- Build complete mental map

Static Analysis

- “lacking in movement, action, or change”
- Analyzing a binary without running it
- Useful for certain circumventions
  - Malware
  - Network access
  - System modifications

Dynamic Analysis

- “stimulates change or progress”
- Analyzing a binary by running it
  - May be too complex to comprehend statically
  - May exhibit unique behavior based on environment in which it executes
- Behavioral Analysis

Tools

- *xxd <file>* - make a hexdump or do the reverse
  - *-b* print bits instead of hex
  - *-c* change column width
  - *-p* plain hex output (no line numbers/ASCII)
  - *-r* transform hexdump to binary format
    - Manually patch binaries!

Tools

- *xxd* - make a hexdump or do the reverse
  - *-b* print bits instead of hex
  - *-c* change column width
  - *-p* plain hex output (no line numbers/ASCII)
  - *-i* output as array of bytes in C
  - *-r* transform hexdump to binary format
    - Manually patch binaries!

Tools

```
[j@b0x:~][130]$ xxd /bin/ls
00000000: 7f45 4c46 0201 0100 0000 0000 0000 0000  .ELF............
00000010: 0300 3e00 0100 0000 3054 0000 0000 0000  ..>.....0T......
00000020: 4000 0000 0000 0000 30f7 0100 0000 0000  @.......0.......
00000030: 0000 0000 4000 3800 0900 4000 1e00 1d00  ....@.8...@.....
00000040: 0600 0000 0500 0000 4000 0000 0000 0000  ........@.......
00000050: 4000 0000 0000 0000 4000 0000 0000 0000  @.......@.......
00000060: f801 0000 0000 0000 f801 0000 0000 0000  ................
00000070: 0800 0000 0000 0000 0300 0000 0400 0000  ................
00000080: 3802 0000 0000 0000 3802 0000 0000 0000  8.......8.......
00000090: 3802 0000 0000 0000 1c00 0000 0000 0000  8...............
000000a0: 1c00 0000 0000 0000 0100 0000 0000 0000  ................
000000b0: 0100 0000 0500 0000 0000 0000 0000 0000  ................
```

Tools

- *strings* - print strings of ASCII in file
  - *-e* to change encoding
  - Only prints strings >4 in length
  - Useful for hardcoded values in binaries
  - Quickly search files for known ASCII values
    - *strings memory_dump | grep “FLAG-{“*

Tools

```
[j@b0x:~][130]$ strings /bin/ls
/lib64/ld-linux-x86-64.so.2
libselinux.so.1
_ITM_deregisterTMCloneTable
__gmon_start__
_Jv_RegisterClasses
_ITM_registerTMCloneTable
_init
fgetfilecon
freecon
lgetfilecon
_fini
libc.so.6
fflush
strcpy
gmtime_r
__printf_chk
fnmatch
readdir
```

Tools

- *readelf* - information on ELF files
  - “Executable and Linkable Format”
  - Extracts metadata from binary based on ELF format
  - see ELF.png in git repo

Tools

- *file* - determines the type of a file
  - helpful to determine type of binary
    - Windows? Linux? macOS? ARM?
  - If magic bytes are corrupted this may not work

Tools

```
[j@b0x:~]$ file /bin/ls                                                    (04-27 13:59)
/bin/ls: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Lin
ux 2.6.32, BuildID[sha1]=3c233e12c466a83aa9b2094b07dbfaa5bd10eccd, stripped
[j@b0x:~]$ file Pictures/2017-08-19-220516_656x673_scrot.png               (04-27 13:59)
Pictures/2017-08-19-220516_656x673_scrot.png: PNG image data, 656 x 673, 8-bit/color RGB, non-interlaced
[j@b0x:~]$ file /etc/shadow                                                (04-27 13:59)
/etc/shadow: regular file, no read permission
[j@b0x:~]$                                                                 (04-27 14:00)
```

Tools

- *radare2* - reverse engineering framework
  - Suite of unix-like tools
  - Includes a hex editor/debugger, assembler, hash tool, diff tool, and many more
  - main tool is called *radare2*
  - https://github.com/radare/radare2

Tools

- *radare2 <file>* opens a shell
  - *aa* to analyze binary
    - can do *aaa* and *aaaa* for more analysis
  - *s* to seek to address/symbol
  - *pd N* to print *N* number of instructions
  - can use *?* after most commands or command prefixes to get help
  - an insane amount of more features here https://radare.gitbooks.io/radare2book/

```
[0x00404890 16% 120 /bin/ls]> pd $r @ entry0
/ (fcn) entry0 42
|         ;-- entry0:
|         0x00404890    31ed           xor ebp, ebp
|         0x00404892    4989d1         mov r9, rdx
|         0x00404895    5e             pop rsi
|         0x00404896    4889e2         mov rdx, rsp
|         0x00404899    4883e4f0       and rsp, 0xfffffffffffffff0
|         0x0040489d    50             push rax
|         0x0040489e    54             push rsp
|         0x0040489f    49c7c0d01e41.  mov r8, 0x411ed0
|         0x004048a6    48c7c1601e41.  mov rcx, 0x411e60
|         0x004048ad    48c7c7c02840.  mov rdi, main ; "AWAVAUATUH..S..H..
|         0x004048b4    e837dcffff     call sym.imp.__libc_start_main ;[1]
|            sym.imp.__libc_start_main(unk, unk, unk)
\         0x004048b9    f4             hlt
          0x004048ba    660f1f440000   nop word [rax + rax]
/ (fcn) fcn.004048c0 41
|         ; CALL XREF from 0x0040493d (fcn.00404930)
|         0x004048c0    b8ffa56100     mov eax, 0x61a5ff ; "hstrtab" @ 0x6
|         0x004048c5    55             push rbp
|         0x004048c6    482df8a56100   sub rax, 0x61a5f8
|         0x004048cc    4883f80e       cmp rax, 0xe
|         0x004048d0    4889e5         mov rbp, rsp
```

```
[0x00404890 16% 120 /bin/ls]> pc @ entry0
#define _BUFFER_SIZE 120
unsigned char buffer[120] = {
  0x31, 0xed, 0x49, 0x89, 0xd1, 0x5e, 0x48, 0x89, 0xe2, 0x48, 0x83,
  0xe4, 0xf0, 0x50, 0x54, 0x49, 0xc7, 0xc0, 0xd0, 0x1e, 0x41, 0x00,
  0x48, 0xc7, 0xc1, 0x60, 0x1e, 0x41, 0x00, 0x48, 0xc7, 0xc7, 0xc0,
  0x28, 0x40, 0x00, 0xe8, 0x37, 0xdc, 0xff, 0xff, 0xf4, 0x66, 0x0f,
  0x1f, 0x44, 0x00, 0x00, 0xb8, 0xff, 0xa5, 0x61, 0x00, 0x55, 0x48,
  0x2d, 0xf8, 0xa5, 0x61, 0x00, 0x48, 0x83, 0xf8, 0x0e, 0x48, 0x89,
  0xe5, 0x77, 0x02, 0x5d, 0xc3, 0xb8, 0x00, 0x00, 0x00, 0x00, 0x48,
  0x85, 0xc0, 0x74, 0xf4, 0x5d, 0xbf, 0xf8, 0xa5, 0x61, 0x00, 0xff,
  0xe0, 0x0f, 0x1f, 0x80, 0x00, 0x00, 0x00, 0x00, 0xb8, 0xf8, 0xa5,
  0x61, 0x00, 0x55, 0x48, 0x2d, 0xf8, 0xa5, 0x61, 0x00, 0x48, 0xc1,
  0xf8, 0x03, 0x48, 0x89, 0xe5, 0x48, 0x89, 0xc2, 0x48, 0xc1, };
```

```
0x00404890 16% 368 /bin/ls]> x @ entry0
 offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
x00404890  31ed 4989 d15e 4889 e248 83e4 f050 5449  1.I..^H..H...PTI
x004048a0  c7c0 d01e 4100 48c7 c160 1e41 0048 c7c7  ....A.H..`.A.H..
x004048b0  c028 4000 e837 dcff fff4 660f 1f44 0000  .(@..7....f..D..
x004048c0  b8ff a561 0055 482d f8a5 6100 4883 f80e  ...a.UH-..a.H...
x004048d0  4889 e577 025d c3b8 0000 0000 4885 c074  H..w.]......H..t
x004048e0  f45d bff8 a561 00ff e00f 1f80 0000 0000  .]...a..........
x004048f0  b8f8 a561 0055 482d f8a5 6100 48c1 f803  ...a.UH-..a.H...
x00404900  4889 e548 89c2 48c1 ea3f 4801 d048 d1f8  H..H..H..?H..H..
x00404910  7502 5dc3 ba00 0000 0048 85d2 74f4 5d48  u.]......H..t.]H
x00404920  89c6 bff8 a561 00ff e20f 1f80 0000 0000  .....a..........
x00404930  803d 215d 2100 0075 1155 4889 e5e8 7eff  .=!]!..u.UH...~.
x00404940  ffff 5dc6 050e 5d21 0001 f3c3 0f1f 4000  ..]...]!......@.
x00404950  4883 3da8 5421 0000 741e b800 0000 0048  H.=.T!..t......H
x00404960  85c0 7414 55bf 009e 6100 4889 e5ff d05d  ..t.U...a.H....]
x00404970  e97b ffff ff0f 1f00 e973 ffff ff0f 1f00  .{.......s......
x00404980  488b 0731 d248 f7f6 4889 d0c3 0f1f 4000  H..1.H..H.....@.
x00404990  31c0 488b 1648 3917 7406 f3c3 0f1f 4000  1.H..H9.t.....@.
x004049a0  488b 4608 4839 4708 0f94 c0c3 0f1f 4000  H.F.H9G.......@.
x004049b0  8b05 8266 2100 85c0 7506 893d 7866 2100  ...f!...u..=xf!.
x004049c0  f3c3 6666 6666 662e 0f1f 8400 0000 0000  ..fffff.........
x004049d0  e91b d8ff ff66 662e 0f1f 8400 0000 0000  .....ff.........
```

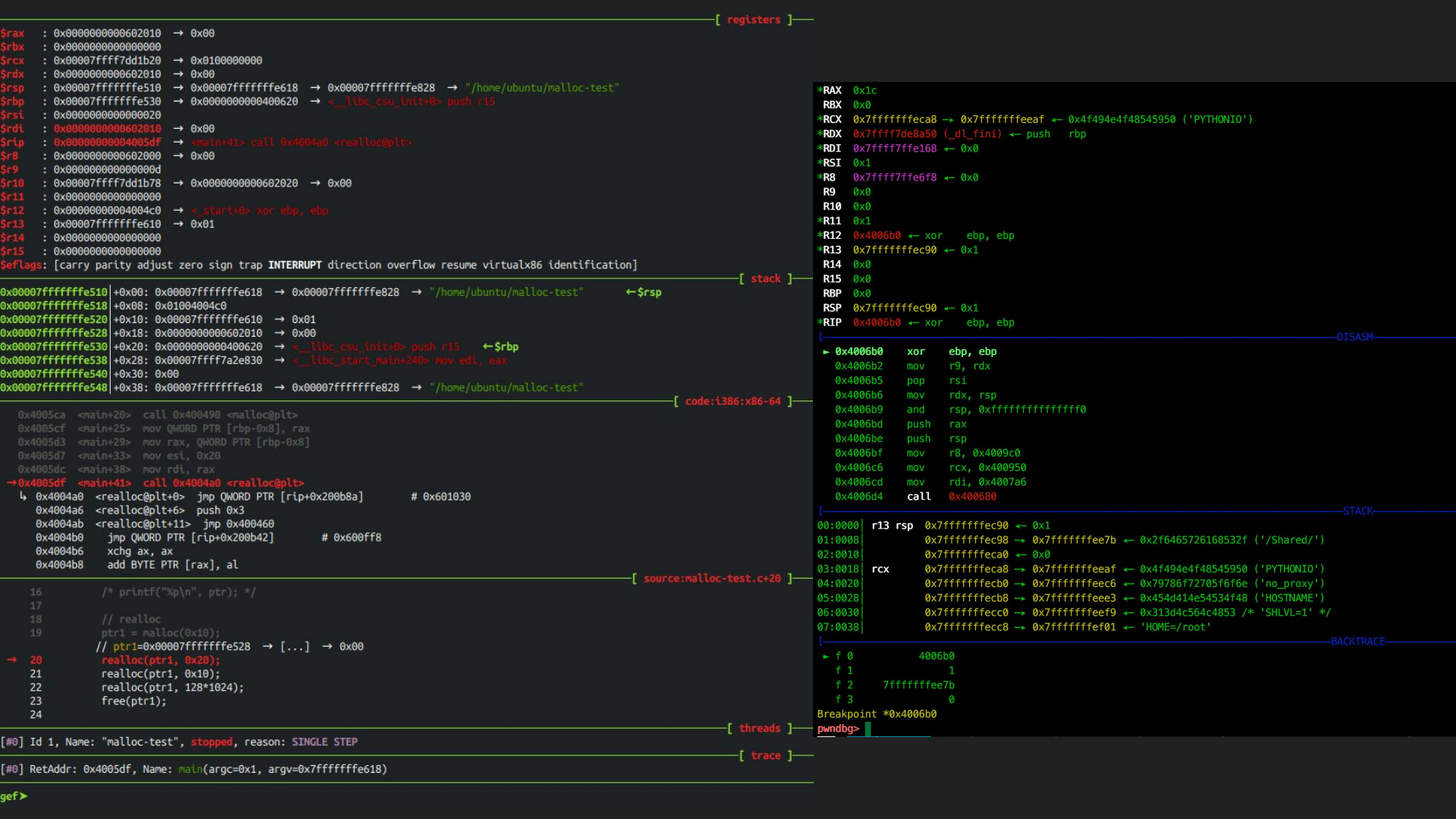
```
[0x00404890 16% 115 /bin/ls]> f tmp;sr s.. @ entry0
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x00000000  7f45 4c46 0201 0100 0000 0000 0000 0000  .ELF............
0x00000010  0200 3e00 0100 0000 9048 4000 0000 0000  ..>.....H@.....
0x00000020  4000 0000 0000 0000 00a7 0100 0000 0000  @...............
0x00000030  0000 0000 4000 3800 0900 4000 1c00 1b00  ....@.8...@.....
 r15 0x00000000         r14 0x00000000         r13 0x00000000
 r12 0x00000000         rbp 0x00000000         rbx 0x00000000
 r11 0x00000000         r10 0x00000000          r9 0x00000000
  r8 0x00000000         rax 0x00000000         rcx 0x00000000
 rdx 0x00000000         rsi 0x00000000         rdi 0x00000000
orax 0x00000000         rip 0x00000000         rflags =
 rsp 0x00000000
/ (fcn) entry0 42
|         ;-- entry0:
|         0x00404890    31ed           xor ebp, ebp
|         0x00404892    4989d1         mov r9, rdx
|         0x00404895    5e             pop rsi
|         0x00404896    4889e2         mov rdx, rsp
|         0x00404899    4883e4f0       and rsp, 0xfffffffffffffff0
|         0x0040489d    50             push rax
|         0x0040489e    54             push rsp
|         0x0040489f    49c7c0d01e41.  mov r8, 0x411ed0
```

Tools

- *gdb* - good ol' GNU debugger
  - *si, ni* for stepping in/over *instructions*
    - Binaries compiled w/o -d flag won't have source code available
  - *p* to print values
    - *p $eax* to print registers
    - *p *($ebx)* to print value pointed by register
      - May need to cast pointers like in C
  - *b* to set breakpoints
    - Useful to skip computations and go to results

Tools

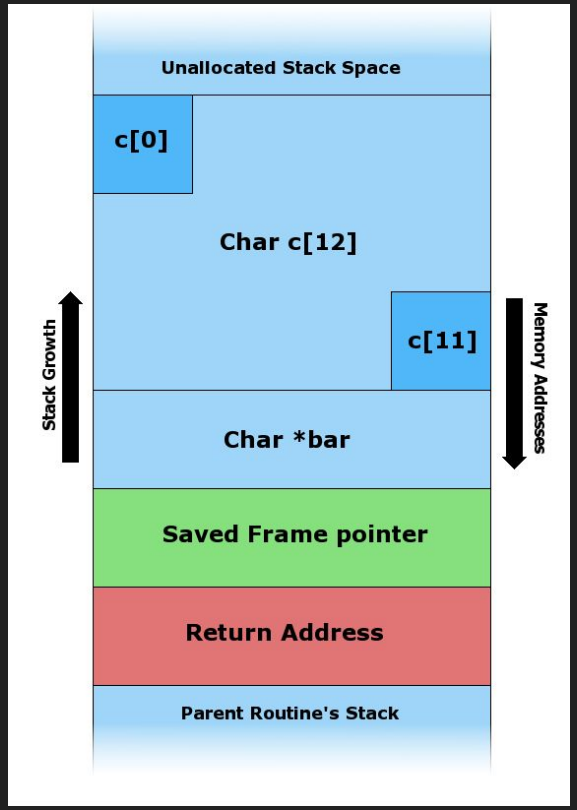
- *gdb*
  - Lots of useful plugins to aid in debugging
  - pwndbg - https://github.com/pwndbg/pwndbg
  - PEDA - https://github.com/longld/peda
  - GEF - https://github.com/hugsy/gef
  - gdbinit - https://github.com/gdbinit/gdbinit
  - Most add stack/register/instruction viewing windows, syntactic sugar, or architecture compatibility

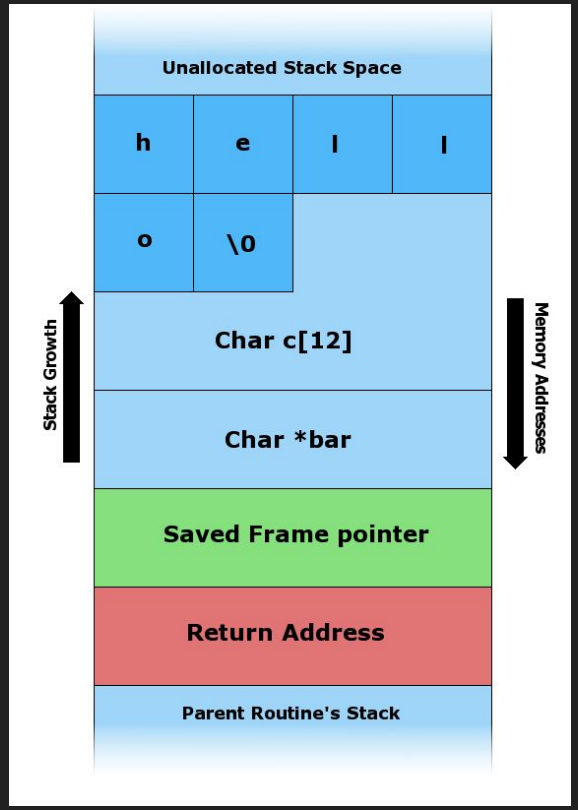

[registers]
$rax : 0x0000000000602010 → 0x00
$rbx : 0x0000000000000000
$rcx : 0x00007ffff7dd1b20 → 0x0100000000
$rdx : 0x0000000000602010 → 0x00
$rsp : 0x00007fffffffe510 → 0x00007fffffffe618 → 0x00007fffffffe828 → "/home/ubuntu/malloc-test"
$rbp : 0x00007fffffffe530 → 0x0000000000400620 → <__libc_csu_init+0> push r15
$rsi : 0x0000000000000020
$rdi : 0x0000000000602010 → 0x00
$rip : 0x00000000004005df → <main+41> call 0x4004a0 <realloc@plt>
$r8 : 0x0000000000602000 → 0x00
$r9 : 0x000000000000000d
$r10 : 0x00007ffff7dd1b78 → 0x0000000000602020 → 0x00
$r11 : 0x0000000000000000
$r12 : 0x00000000004004c0 → <_start+0> xor ebp, ebp
$r13 : 0x00007fffffffe610 → 0x01
$r14 : 0x0000000000000000
$r15 : 0x0000000000000000
$eflags: [carry parity adjust zero sign trap INTERRUPT direction overflow resume virtualx86 identification]
[stack]
0x00007fffffffe510 +0x00: 0x00007fffffffe618 → 0x00007fffffffe828 → "/home/ubuntu/malloc-test" ←$rsp
0x00007fffffffe518 +0x08: 0x01004004c0
0x00007fffffffe520 +0x10: 0x00007fffffffe610 → 0x01
0x00007fffffffe528 +0x18: 0x0000000000602010 → 0x00
0x00007fffffffe530 +0x20: 0x0000000000400620 → <__libc_csu_init+0> push r15 ←$rbp
0x00007fffffffe538 +0x28: 0x00007ffff7a2e830 → <__libc_start_main+240> mov edi, eax
0x00007fffffffe540 +0x30: 0x00
0x00007fffffffe548 +0x38: 0x00007fffffffe618 → 0x00007fffffffe828 → "/home/ubuntu/malloc-test"
[code:i386:x86-64]
0x4005ca <main+20> call 0x400490 <malloc@plt>
0x4005cf <main+25> mov QWORD PTR [rbp-0x8], rax
0x4005d3 <main+29> mov rax, QWORD PTR [rbp-0x8]
0x4005d7 <main+33> mov esi, 0x20
0x4005dc <main+38> mov rdi, rax
→0x4005df <main+41> call 0x4004a0 <realloc@plt>
↳ 0x4004a0 <realloc@plt+0> jmp QWORD PTR [rip+0x200b8a] # 0x601030
0x4004a6 <realloc@plt+6> push 0x3
0x4004ab <realloc@plt+11> jmp 0x400460
0x4004b0 jmp QWORD PTR [rip+0x200b42] # 0x600ff8
0x4004b6 xchg ax, ax
0x4004b8 add BYTE PTR [rax], al
[source:malloc-test.c+20]
16 /* printf("%p\n", ptr); */
17
18 // realloc
19 ptr1 = malloc(0x10);
// ptr1=0x00007fffffffe528 → [...] → 0x00
→ 20 realloc(ptr1, 0x20);
21 realloc(ptr1, 0x10);
22 realloc(ptr1, 128*1024);
23 free(ptr1);
24
[threads]
[#0] Id 1, Name: "malloc-test", stopped, reason: SINGLE STEP
[trace]
[#0] RetAddr: 0x4005df, Name: main(argc=0x1, argv=0x7fffffffe618)
gef➤
*RAX 0x1c
RBX 0x0
*RCX 0x7fffffffeca8 —▸ 0x7fffffffeeaf ◂— 0x4f494e4f48545950 ('PYTHONIO')
*RDX 0x7ffff7de8a50 (_dl_fini) ◂— push rbp
*RDI 0x7ffff7ffe168 ◂— 0x0
*RSI 0x1
*R8 0x7ffff7ffe6f8 ◂— 0x0
R9 0x0
R10 0x0
*R11 0x1
*R12 0x4006b0 ◂— xor ebp, ebp
*R13 0x7fffffffec90 ◂— 0x1
R14 0x0
R15 0x0
RBP 0x0
RSP 0x7fffffffec90 ◂— 0x1
*RIP 0x4006b0 ◂— xor ebp, ebp
[DISASM]
► 0x4006b0 xor ebp, ebp
0x4006b2 mov r9, rdx
0x4006b5 pop rsi
0x4006b6 mov rdx, rsp
0x4006b9 and rsp, 0xfffffffffffffff0
0x4006bd push rax
0x4006be push rsp
0x4006bf mov r8, 0x4009c0
0x4006c6 mov rcx, 0x400950
0x4006cd mov rdi, 0x4007a6
0x4006d4 call 0x400680
[STACK]
00:0000 r13 rsp 0x7fffffffec90 ◂— 0x1
01:0008 0x7fffffffec98 —▸ 0x7fffffffee7b ◂— 0x2f6465726168532f ('/Shared/')
02:0010 0x7fffffffeca0 ◂— 0x0
03:0018 rcx 0x7fffffffeca8 —▸ 0x7fffffffeeaf ◂— 0x4f494e4f48545950 ('PYTHONIO')
04:0020 0x7fffffffecb0 —▸ 0x7fffffffeec6 ◂— 0x79786f72705f6f6e ('no_proxy')
05:0028 0x7fffffffecb8 —▸ 0x7fffffffeee3 ◂— 0x454d414e54534f48 ('HOSTNAME')
06:0030 0x7fffffffecc0 —▸ 0x7fffffffeef9 ◂— 0x313d4c564c4853 /* 'SHLVL=1' */
07:0038 0x7fffffffecc8 —▸ 0x7fffffffef01 ◂— 'HOME=/root'
[BACKTRACE]
► f 0 4006b0
f 1 1
f 2 7fffffffee7b
f 3 0
Breakpoint *0x4006b0
pwndbg>

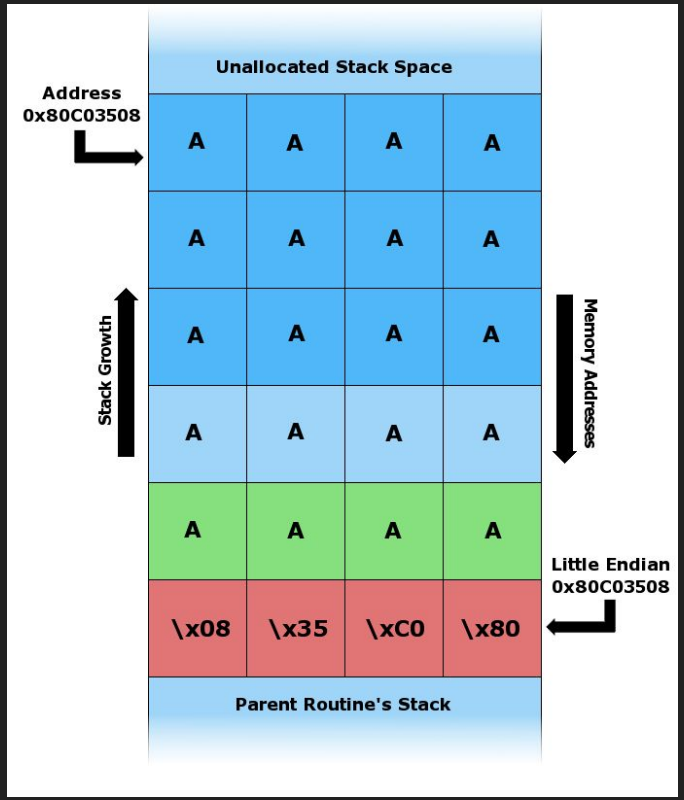
Buffer Overflow

- Dynamic analysis technique
- When user input is not handled properly
- Accomplish things from variable modifications to arbitrary code execution

Buffer Overflow







Exercises

- Download files from git
  - Week 13, under *exercises/0x0**
- Mess around with them, get them to work!

homework #11

will be posted soon.

Let us know if you have any questions!

This assignment has 2 parts.

It is due by 5/5 at 11:59PM.

Next week’s class is our final meeting!