CMSC389R

Binaries II

COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

recap

HW5

--

Questions?

Itinerary

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

Review

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

Calling Conventions

- Arguably one of the more important aspects when starting to learn reverse engineering
- Argument passing
  - rdi, rsi, rdx, rcx, r8, r9
- Function setup
  - Adjusting base/stack pointer
  - Where is return address stored?
  - How do we return?

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
	0x00405fd0	
	0x00405fd8	
	0x00405fe0	
	0x00405fe8	
rsp	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabcd
rbp	0x00406000	1234
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
	0x00405fd0	
	0x00405fd8	
	0x00405fe0	
	0x00405fe8	
rsp	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabcd
rbp	0x00406000	1234
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
	0x00405fd0	
	0x00405fd8	
	0x00405fe0	
rsp	0x00405fe8	
	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabcd
rbp	0x00406000	1234
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
	0x00405fd0	
	0x00405fd8	
	0x00405fe0	
rsp	0x00405fe8	0x00406000
	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabcd
rbp	0x00406000	1234
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
	0x00405fd0	
	0x00405fd8	
	0x00405fe0	
rsp	0x00405fe8	0x00406000
	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabcd
rbp	0x00406000	1234
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
	0x00405fd0	
	0x00405fd8	
	0x00405fe0	
rbp rsp	0x00405fe8	0x00406000
	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabcd
	0x00406000	1234
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
	0x00405fd0	
	0x00405fd8	
	0x00405fe0	
rbp rsp	0x00405fe8	0x00406000
	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabcd
	0x00406000	1234
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
rsp	0x00405fd0	
	0x00405fd8	
	0x00405fe0	
rbp	0x00405fe8	0x00406000
	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabcd
	0x00406000	1234
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
rsp	0x00405fd0	
	0x00405fd8	
	0x00405fe0	
rbp	0x00405fe8	0x00406000
	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabcd
	0x00406000	1234
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
rsp	0x00405fd0	123
	0x00405fd8	456
	0x00405fe0	789
rbp	0x00405fe8	0x00406000
	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabce
	0x00406000	1235
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
rsp	0x00405fd0	123
	0x00405fd8	456
	0x00405fe0	789
rbp	0x00405fe8	0x00406000
	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabce
	0x00406000	1235
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
	0x00405fd0	123
	0x00405fd8	456
	0x00405fe0	789
rsp rbp	0x00405fe8	0x00406000
	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabce
	0x00406000	1235
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
	0x00405fd0	123
	0x00405fd8	456
	0x00405fe0	789
rsp rbp	0x00405fe8	0x00406000
	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabce
	0x00406000	1235
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
	0x00405fd0	123
	0x00405fd8	456
	0x00405fe0	789
rsp rbp	0x00405fe8	0x00406000
	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabce
	0x00406000	1235
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
	0x00405fd0	123
	0x00405fd8	456
	0x00405fe0	789
rsp	0x00405fe8	0x00406000
	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabce
rbp	0x00406000	1235
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
	0x00405fd0	123
	0x00405fd8	456
	0x00405fe0	789
	0x00405fe8	0x00406000
rsp	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabce
rbp	0x00406000	1235
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
	0x00405fd0	123
	0x00405fd8	456
	0x00405fe0	789
	0x00405fe8	0x00406000
rsp	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabce
rbp	0x00406000	1235
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```


```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

rip	MY RETURN ADDRESS

Static Analysis

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

Static Analysis Tools

- objdump: disassembles binaries
  - *-D* will disassemble EVERYTHING
  - *-M intel* will output with Intel syntax

```
0000000000001200 <my_memset>:
    1200:       55                          push   rbp
    1201:       48 89 e5                    mov    rbp,rsp
    1204:       c9                          leave
    1205:       c3                          ret

0000000000001206 <my_strncpy>:
    1206:       55                          push   rbp
    1207:       48 89 e5                    mov    rbp,rsp
    120a:       c9                          leave
    120b:       c3                          ret
    120c:       0f 1f 40 00                 nop    DWORD PTR [rax+0x0]
```

Static Analysis Tools

- objdump: disassembles binaries
  - *-D* will disassemble EVERYTHING
  - *-M intel* will output with Intel syntax

```
0000000000001200 <my_memset>:
    1200:       55                          push   rbp
    1201:       48 89 e5                    mov    rbp,rsp
    1204:       c9                          leave
    1205:       c3                          ret

0000000000001206 <my_strncpy>:
    1206:       55                          push   rbp
    1207:       48 89 e5                    mov    rbp,rsp
    120a:       c9                          leave
    120b:       c3                          ret
    120c:       0f 1f 40 00                 nop    DWORD PTR [rax+0x0]
```

Static Analysis Tools

- nm: list symbols from binaries
- *man nm* for details

Static A

- nm: list symbols fr
- *man nm* for details

```
week/5 [ nm main
0000000000004038 B __bss_start
0000000000004038 b completed.7286
                 w __cxa_finalize@@GLIBC_2.2.5
0000000000004028 D __data_start
0000000000004028 W data_start
0000000000001080 t deregister_tm_clones
00000000000010f0 t __do_global_dtors_aux
0000000000003df0 t __do_global_dtors_aux_fini_array_entry
0000000000004030 D __dso_handle
0000000000003df8 d _DYNAMIC
0000000000004038 D _edata
0000000000004040 B _end
0000000000001288 T _fini
0000000000001140 t frame_dummy
0000000000003de8 t __frame_dummy_init_array_entry
000000000000210c r __FRAME_END__
0000000000004000 d _GLOBAL_OFFSET_TABLE_
                 w __gmon_start__
0000000000002004 r __GNU_EH_FRAME_HDR
0000000000001000 T _init
0000000000003df0 t __init_array_end
0000000000003de8 t __init_array_start
0000000000002000 R _IO_stdin_used
                 w _ITM_deregisterTMCloneTable
                 w _ITM_registerTMCloneTable
0000000000001280 T __libc_csu_fini
0000000000001210 T __libc_csu_init
                 U __libc_start_main@@GLIBC_2.2.5
0000000000001149 T main
0000000000001200 T my_memset
0000000000001206 T my_strncpy
```

Static Analysis Tools

- readelf: view information about ELF files
  - ELF header
  - Section headers (.text .data .comment etc)
- *man readelf* for more details

Static Analysis Tools

- rea
  -
  -
- *man*

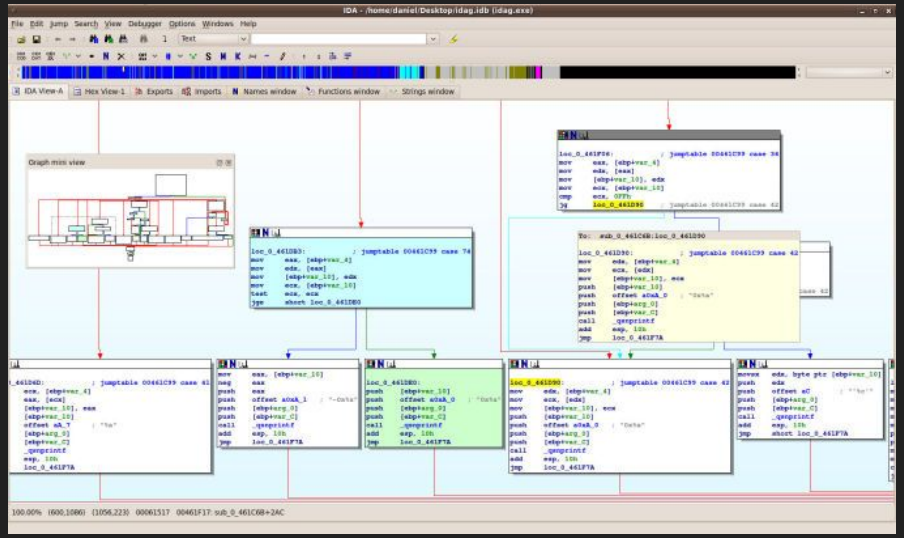
```
week/5 [ readelf -a main
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              DYN (Shared object file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x1050
  Start of program headers:          64 (bytes into file)
  Start of section headers:          14864 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         11
  Size of section headers:           64 (bytes)
  Number of section headers:         29
  Section header string table index: 28
```

tc)

Static Analysis Tools

- strings: outputs human-readable strings

```
week/5 [ strings main
/lib64/ld-linux-x86-64.so.2
)H,=[
&~:e
libc.so.6
puts
__stack_chk_fail
__cxa_finalize
__libc_start_main
GLIBC_2.4
GLIBC_2.2.5
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
u3UH
Hello WoH
rld!
rld!
```



```
[0x00003c9c 255 /usr/bin/r2]> pd $r @ sym..L94+4869 # 0x3c9c
            0x00003c9c    e970efffff   jmp 0x100002c11 ; (fcn.00002390) ;[1]
            0x00003ca1    8bbba4010000 mov edi, [ebx+0x1a4]
            0x00003ca7    8b74247c     mov esi, [esp+0x7c]
            0x00003cab    8b842494000. mov eax, [esp+0x94]
            0x00003cb2    c7442404000. mov dword [esp+0x4], 0x0
            0x00003cba    890424       mov [esp], eax
            0x00003cbd    e81ee2ffff   call 0x100001ee0 ; (sym.imp.r_core_prompt) ;[2]
               sym.imp.r_core_prompt()
            0x00003cc2    85c0         test eax, eax
            0x00003cc4    0f8eaa000000 jle 0x3d74 ;[3]
            0x00003cca    85f6         test esi, esi
            0x00003ccc    7408         jz 0x3cd6 ;[4]
            0x00003cce    893424       mov [esp], esi
            0x00003cd1    e84ae4ffff   call 0x100002120 ; (sym.imp.r_th_lock_enter) ;[5]
               sym.imp.r_th_lock_enter()
            0x00003cd6    8b942494000. mov edx, [esp+0x94]
            0x00003cdd    891424       mov [esp], edx
            0x00003ce0    e80be4ffff   call 0x1000020f0 ; (sym.imp.r_core_prompt_exec) ;[6]
               sym.imp.r_core_prompt_exec()
            0x00003ce5    8984249c000. mov [esp+0x9c], eax
            0x00003cec    83c001       add eax, 0x1
            0x00003cef    0f8424010000 jz 0x3e19 ;[7]
            0x00003cf5    85f6         test esi, esi
            0x00003cf7    7408         jz 0x3d01 ;[8]
```

```
4_t sub_40232c()
004023b4  mov     edx, 0x2
004023b9  mov     esi, 0x1
004023be  mov     edi, eax
004023c0  call    setsockopt
004023c5  cmp     eax, 0xffffffff
004023c8  jne     0x4023ec

004023ca  call    __errno_location
004023cf  mov     eax, dword [rax]
004023d1  mov     esi, eax
004023d3  mov     edi, 0x402d40  {"[ERROR] setsockopt() failed %x\n…"}
004023d8  mov     eax, 0x0
004023dd  call    printf
004023e2  mov     edi, 0xffffffff
004023e7

Define Name
Enter symbol name:
maybe_bind_socket
Cancel   OK

                  x, dword [rbp-0x14]
                  x, dword [rbp-0x18]
                  i, dword [rbp-0x4]
                  x, qword [rbp-0x20]
004023f9  mov     rdi, rax
004023fc  call    sub_40245e
00402401  test    eax, eax
00402403  je      0x40240f

00402405  mov     edi, 0xffffffff
0040240a  call    exit

0040240f  mov     eax, dword [rbp-0x4]
00402412  mov     esi, 0x14
00402417  mov     edi, eax
00402419  call    listen
0040241e  test    eax, eax
00402420  je      0x40242c

Options   Selection: 0x4023fc to 0x402401 (0x5 bytes)
```

Static Analysis Tools

- Disassemblers
  - IDA: “Interactive Disassembler”
    - Very expensive
    - State of the art, industry standard
  - Binary Ninja
    - Much cheaper
    - Fewer features than IDA, but fine if you’re only doing x86
  - radare2
    - Open-source (read: free)
    - Impressive features for a free product

Static Analysis Tools

- radare2: reverse engineering framework
- Suite of tools for useful analysis
  - rabin2: dumps info on binary
    - -I shows binary info (similar to *file*)
    - -s shows symbols (similar to *nm*)
    - -i shows imported functions
  - rasm2: assembler
  - rahash2: computes various hashing algs
  - radiff2: finds differences in files

radare2

- Command-line only interface
- *VERY* steep learning curve
  - *vim*-like hotkeys/commands (sorry emacs)
  - Most commands are only a few letters
  - https://radare.gitbooks.io/radare2book/content/
- There is a GUI... cutter

cutter

- Nice, fully-featured GUI
- Comparable to IDA, Binary Ninja
  - But freeeeeeeee
- Demo

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
  - Flag obfuscation? No worries!
  - Breakpoint at strcmp, examine memory

Dynamic Analysis Tools

- gdb: your C debugger
  - Surprise! Most reverse engineers use this
  - Very powerful if you know what you’re looking for
  - Scriptable
- angr: programmatically interact with binaries
  - Symbolically execute binaries
  - Override function behavior at runtime
  - Many more things to do!

Buffer Overflow

- In C, several functions fail to check bounds
  - strcpy, strcmp, gets, etc
- If we find boundaries aren’t checked, we can fill buffers (i.e. arrays) with more data than expected
- If buffers are local variables, we can overwrite information on the stack
- ...can we change the return address?

	0x7fffffb0	
	0x7fffffb8	
	0x7fffffc0	
	0x7fffffc8	
	0x7fffffd0	
rsp (buf)	0x7fffffd8	
	0x7fffffe0	
rbp	0x7fffffe8	OLD BASE POINTER
	0x7ffffff0	MY RETURN ADDRESS
	.....	
	0xffffffff	

```c
#include <stdio.h>

int main(void) {
     char buf[16];

     printf("hey plz gib data\n");
     gets(buf);
     printf("thx\n");

     return 0;
}
```

	0x7fffffb0	
	0x7fffffb8	
	0x7fffffc0	
	0x7fffffc8	
	0x7fffffd0	
rsp (buf)	0x7fffffd8	
	0x7fffffe0	
rbp	0x7fffffe8	OLD BASE POINTER
	0x7ffffff0	MY RETURN ADDRESS
	.....	
	0xffffffff	

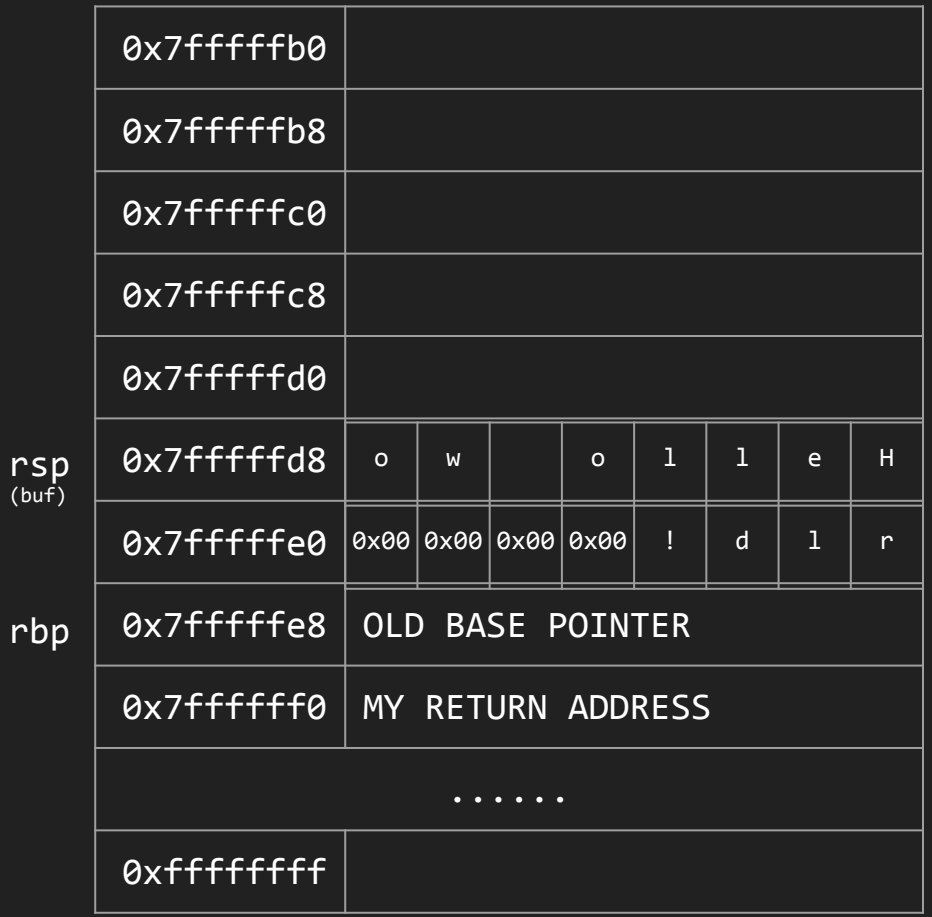
```c
#include <stdio.h>

int main(void) {
    char buf[16];

    printf("hey plz gib data\n");
    gets(buf);
    printf("thx\n");

    return 0;
}
```

Output
hey plz gib data ▮

	0x7fffffb0	
	0x7fffffb8	
	0x7fffffc0	
	0x7fffffc8	
	0x7fffffd0	
rsp (buf)	0x7fffffd8	
	0x7fffffe0	
rbp	0x7fffffe8	OLD BASE POINTER
	0x7ffffff0	MY RETURN ADDRESS
	.....	
	0xffffffff	

```c
#include <stdio.h>

int main(void) {
    char buf[16];

    printf("hey plz gib data\n");
    gets(buf);
    printf("thx\n");

    return 0;
}
```

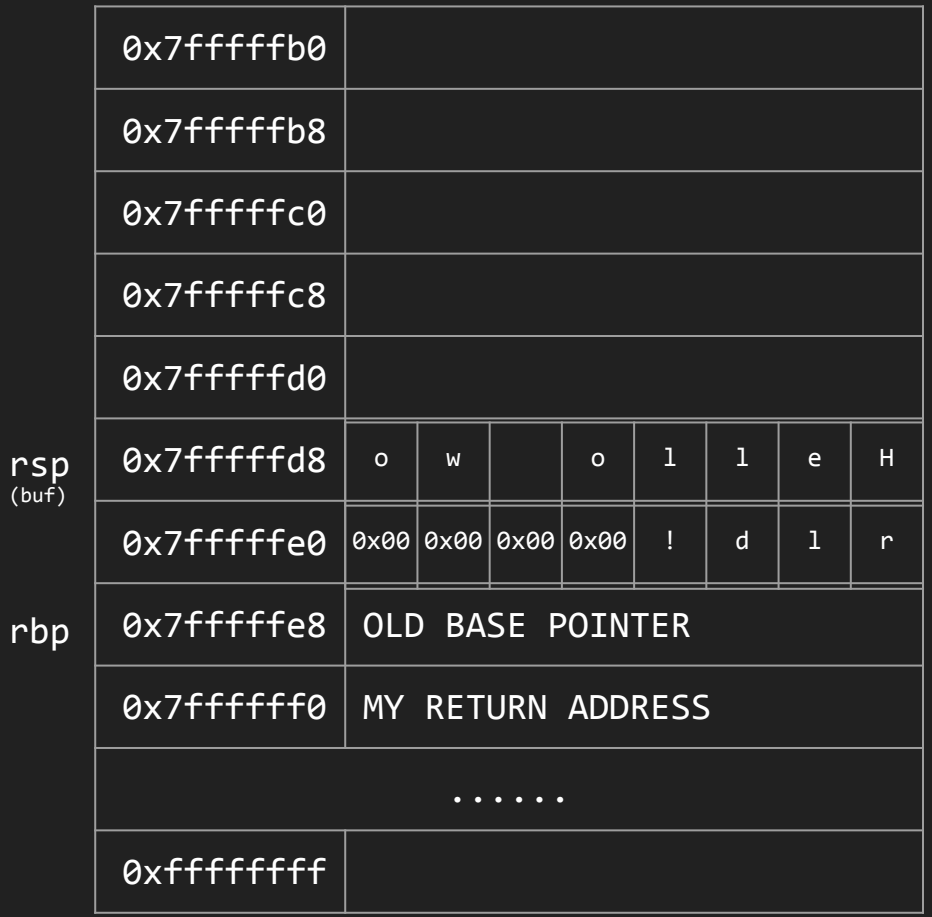
Output
hey plz gib data Hello world! ▮




```c
#include <stdio.h>

int main(void) {
    char buf[16];

    printf("hey plz gib data\n");
    gets(buf);
    printf("thx\n");

    return 0;
}
```

Output

```
hey plz gib data
Hello world!
```




```c
#include <stdio.h>

int main(void) {
    char buf[16];

    printf("hey plz gib data\n");
    gets(buf);
    printf("thx\n");

    return 0;
}
```

Output

```
hey plz gib data
Hello world!
thx
```

	0x7fffffb0	
	0x7fffffb8	
	0x7fffffc0	
	0x7fffffc8	
	0x7fffffd0	
rsp (buf)	0x7fffffd8	
	0x7fffffe0	
rbp	0x7fffffe8	OLD BASE POINTER
	0x7ffffff0	MY RETURN ADDRESS
	.....	
	0xffffffff	

```c
#include <stdio.h>

int main(void) {
     char buf[16];

     printf("hey plz gib data\n");
     gets(buf);
     printf("thx\n");

     return 0;
}
```

	0x7fffffb0	
	0x7fffffb8	
	0x7fffffc0	
	0x7fffffc8	
	0x7fffffd0	
rsp (buf)	0x7fffffd8	
	0x7fffffe0	
rbp	0x7fffffe8	OLD BASE POINTER
	0x7ffffff0	MY RETURN ADDRESS
	.....	
	0xffffffff	

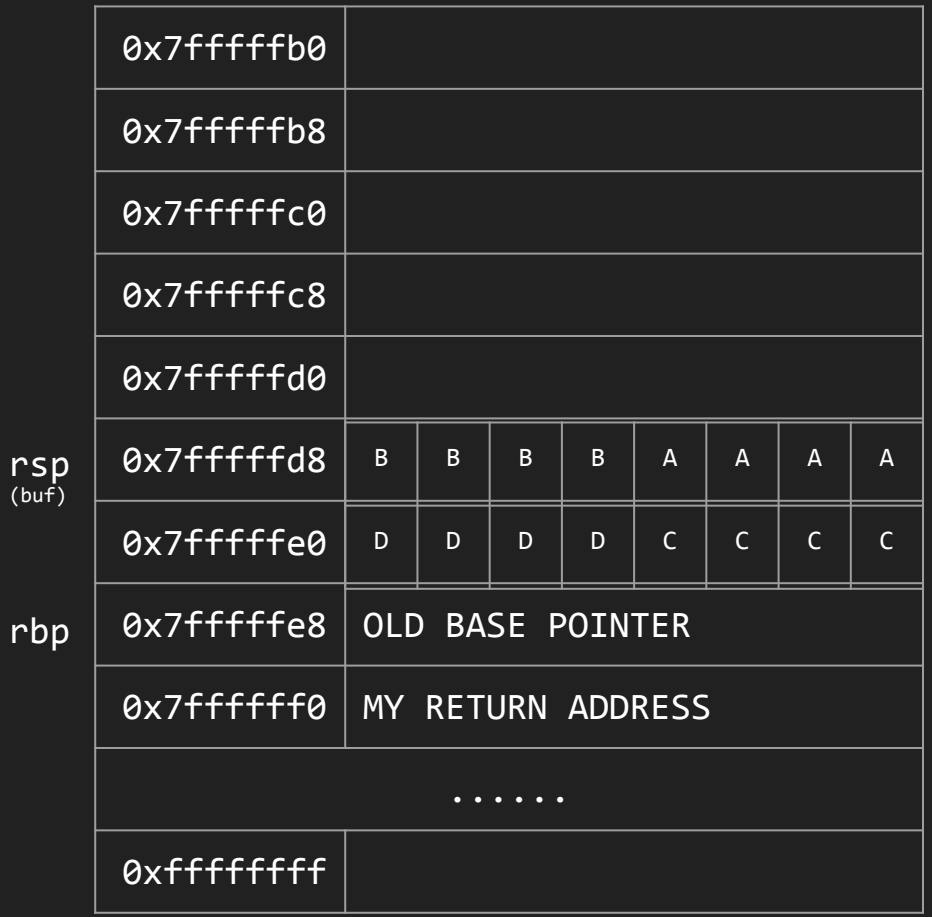
```c
#include <stdio.h>

int main(void) {
    char buf[16];

    printf("hey plz gib data\n");
    gets(buf);
    printf("thx\n");

    return 0;
}
```

Output
hey plz gib data ▮

	0x7fffffb0	
	0x7fffffb8	
	0x7fffffc0	
	0x7fffffc8	
	0x7fffffd0	
rsp (buf)	0x7fffffd8	
	0x7fffffe0	
rbp	0x7fffffe8	OLD BASE POINTER
	0x7ffffff0	MY RETURN ADDRESS
	.....	
	0xffffffff	

```c
#include <stdio.h>

int main(void) {
    char buf[16];

    printf("hey plz gib data\n");
    gets(buf);
    printf("thx\n");

    return 0;
}
```

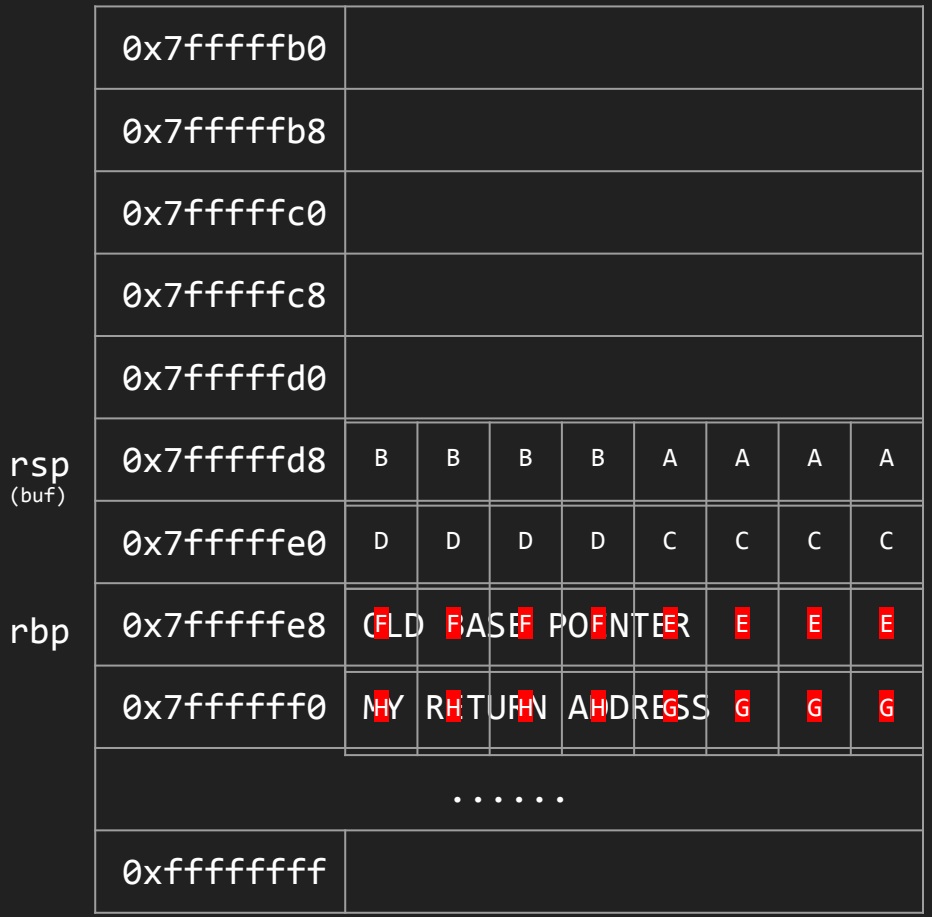
Output
hey plz gib data AAAABBBBCCCCDDDD ▮




```c
#include <stdio.h>

int main(void) {
     char buf[16];

     printf("hey plz gib data\n");
     gets(buf);
     printf("thx\n");

     return 0;
}
```

Output

```
hey plz gib data
AAAABBBBCCCCDDDD
```




```c
#include <stdio.h>

int main(void) {
     char buf[16];

     printf("hey plz gib data\n");
     gets(buf);
     printf("thx\n");

     return 0;
}
```

Output

```
hey plz gib data
AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHH
```

Buffer Overflow

- We can overflow badly checked buffers to change variables on the stack!
  - Including the return address
- If there’s a function we want to call that we can’t normally access, we can now access it!
- Demo

Buffer Overflow

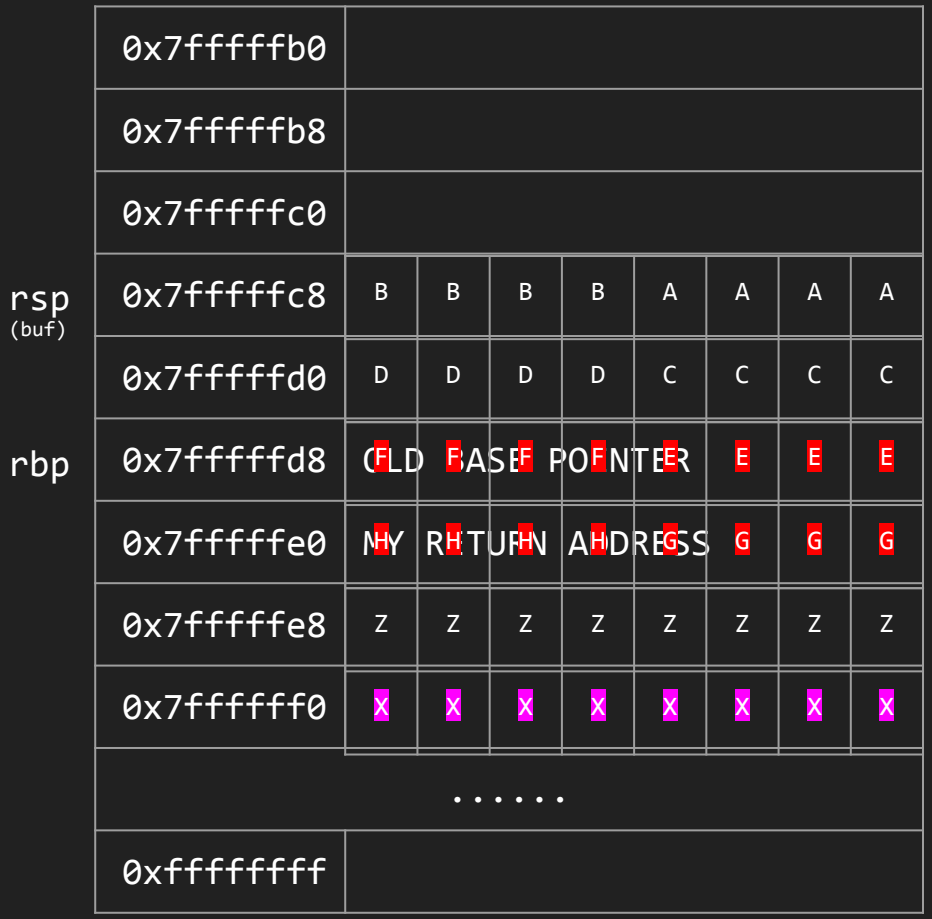
- We can overflow badly checked buffers to change variables on the stack!
  - Including the return address
- If there’s a function we want to call that we can’t normally access, we can now access it!
- Demo
- ...can we chain together function calls?

Return Oriented Programming

- ...yes we can!
- We can overflow the stack to change the return address for one function
- If we know how that other function sets up its stack, we can put another function call after!
- Most functions will just have a stored base pointer, then a return address
- We can just add 8 bytes of junk, then our next function address!

	0x7fffffb0	
	0x7fffffb8	
	0x7fffffc0	
rsp (buf)	0x7fffffc8	
	0x7fffffd0	
rbp	0x7fffffd8	OLD BASE POINTER
	0x7fffffe0	MY RETURN ADDRESS
	0x7fffffe8	
	0x7ffffff0	
	.....	
	0xffffffff	

```c
#include <stdio.h>

int main(void) {
	char buf[16];
	printf(“hey plz gib data\n”);
	gets(buf);
	printf(“thx\n”);
	return 0;
}

void secret(void) {
  printf("you shouldn't be here\n");
}

void other_secret(void) {
  printf("you shouldn't be here either!\n");
}
```




```c
#include <stdio.h>

int main(void) {
        char buf[16];
        printf("hey plz gib data\n");
        gets(buf);
        printf("thx\n");
        return 0;
}

void secret(void) {
  printf("you shouldn't be here\n");
}

void other_secret(void) {
  printf("you shouldn't be here either!\n");
}
```

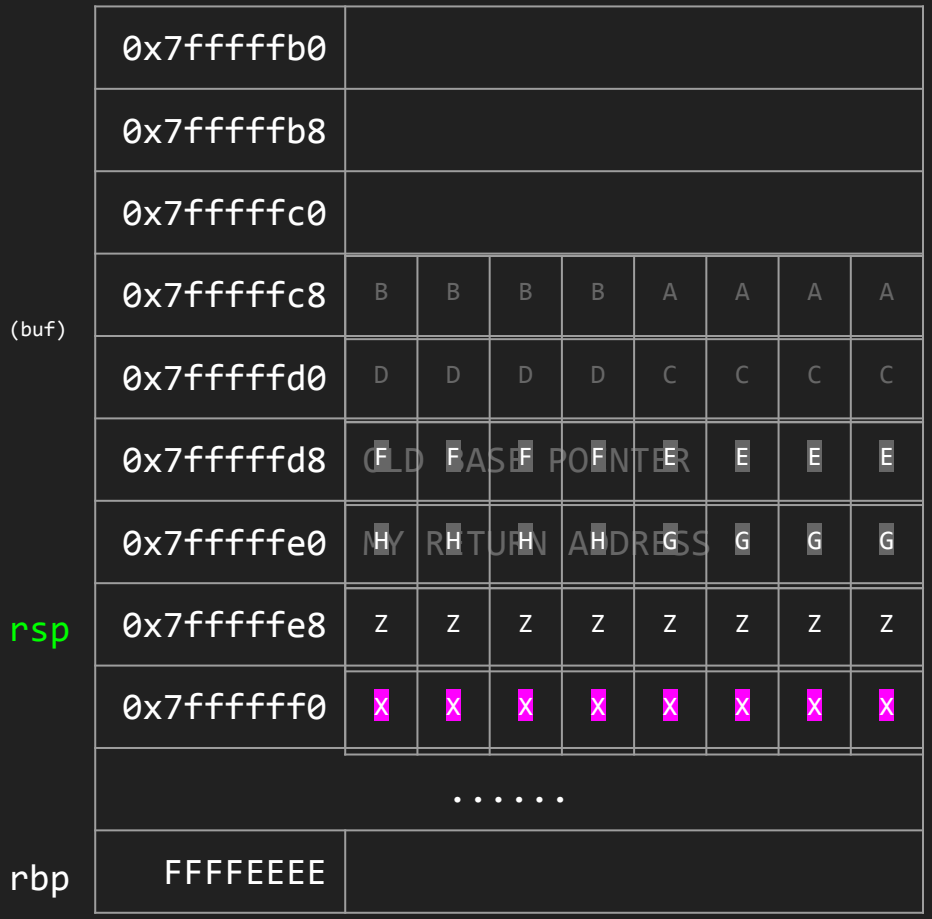
Output

```
hey plz gib data
AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHHZZZZZZZZXXXXXXXX
```


```c
#include <stdio.h>

int main(void) {
       char buf[16];
       printf("hey plz gib data\n");
       gets(buf);
       printf("thx\n");
       return 0;
}

void secret(void) {
  printf("you shouldn't be here\n");
}

void other_secret(void) {
  printf("you shouldn't be here either!\n");
}
```

Output

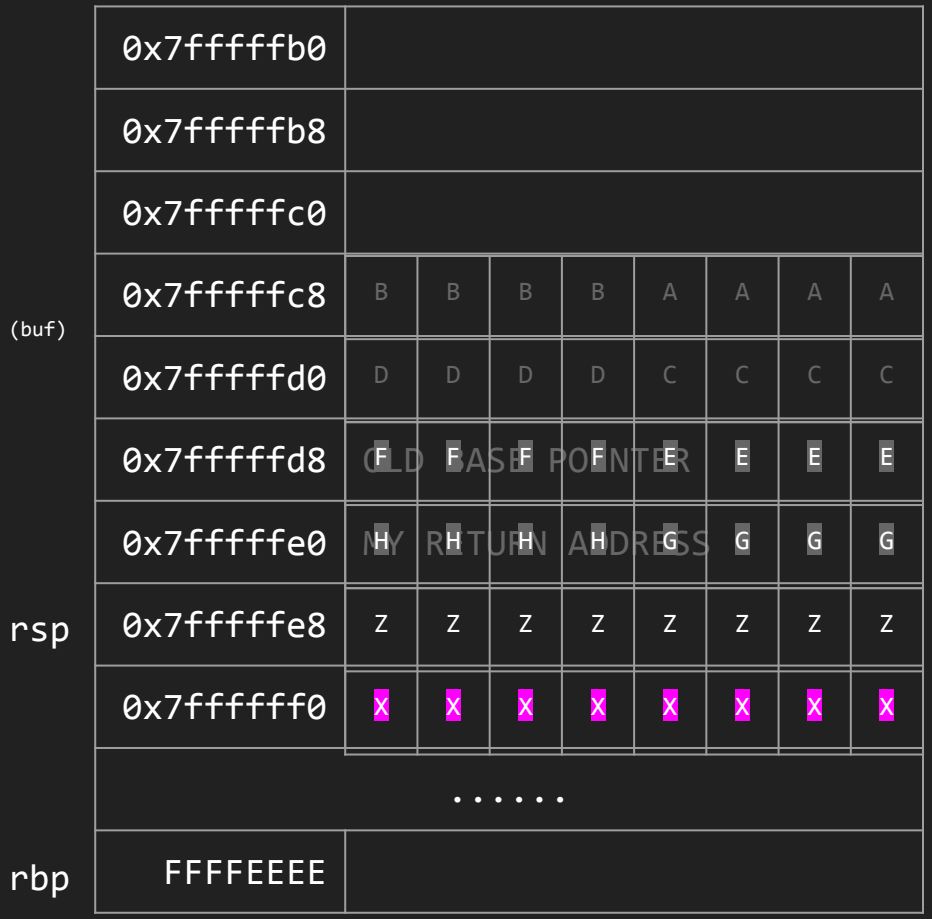
```
hey plz gib data
AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHHZZZZZZZZXXXXXXXX
```




```c
#include <stdio.h>

int main(void) {
       char buf[16];
       printf("hey plz gib data\n");
       gets(buf);
       printf("thx\n");
       return 0;
}

void secret(void) {
  printf("you shouldn't be here\n");
}

void other_secret(void) {
  printf("you shouldn't be here either!\n");
}
```

Output

```
hey plz gib data
AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHHZZZZZZZZXXXXXXXX
```




```c
#include <stdio.h>

int main(void) {
       char buf[16];
       printf("hey plz gib data\n");
       gets(buf);
       printf("thx\n");
       return 0;
}

void secret(void) {
  printf("you shouldn't be here\n");
}

void other_secret(void) {
  printf("you shouldn't be here either!\n");
}
```

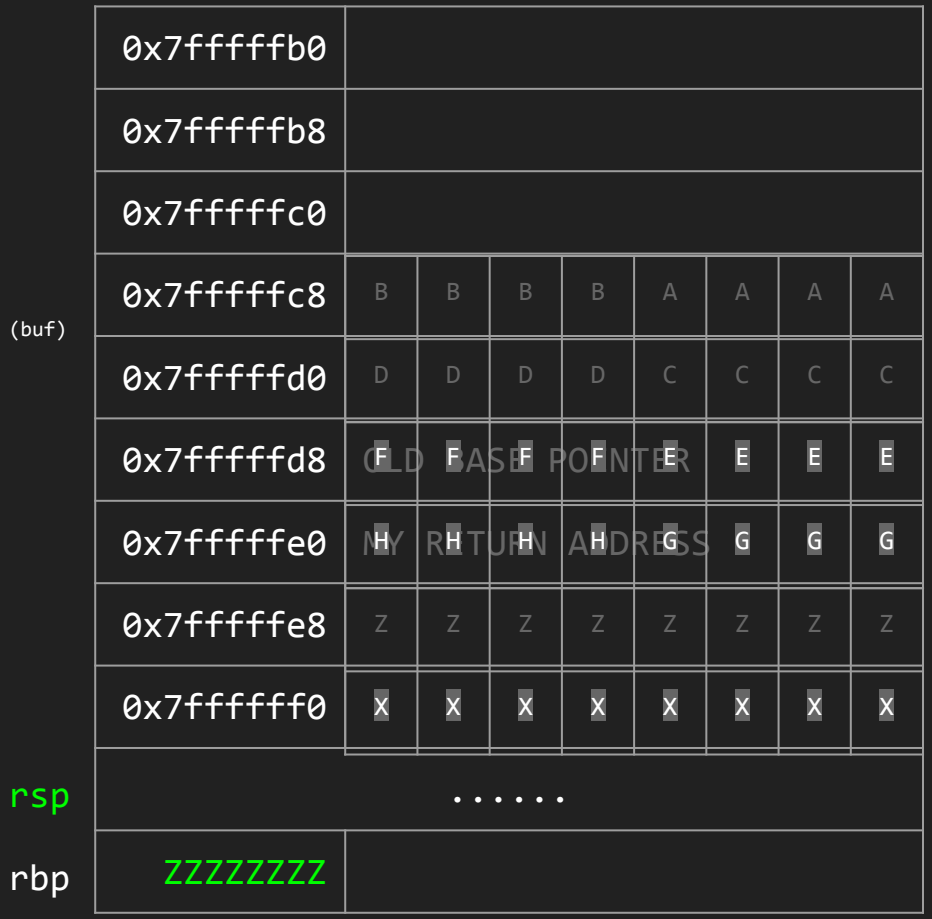
Output

```
hey plz gib data
AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHHZZZZZZZZXXXXXXXX
you shouldn't be here
```


```c
#include <stdio.h>

int main(void) {
       char buf[16];
       printf("hey plz gib data\n");
       gets(buf);
       printf("thx\n");
       return 0;
}

void secret(void) {
  printf("you shouldn't be here\n");
}

void other_secret(void) {
  printf("you shouldn't be here either!\n");
}
```

Output

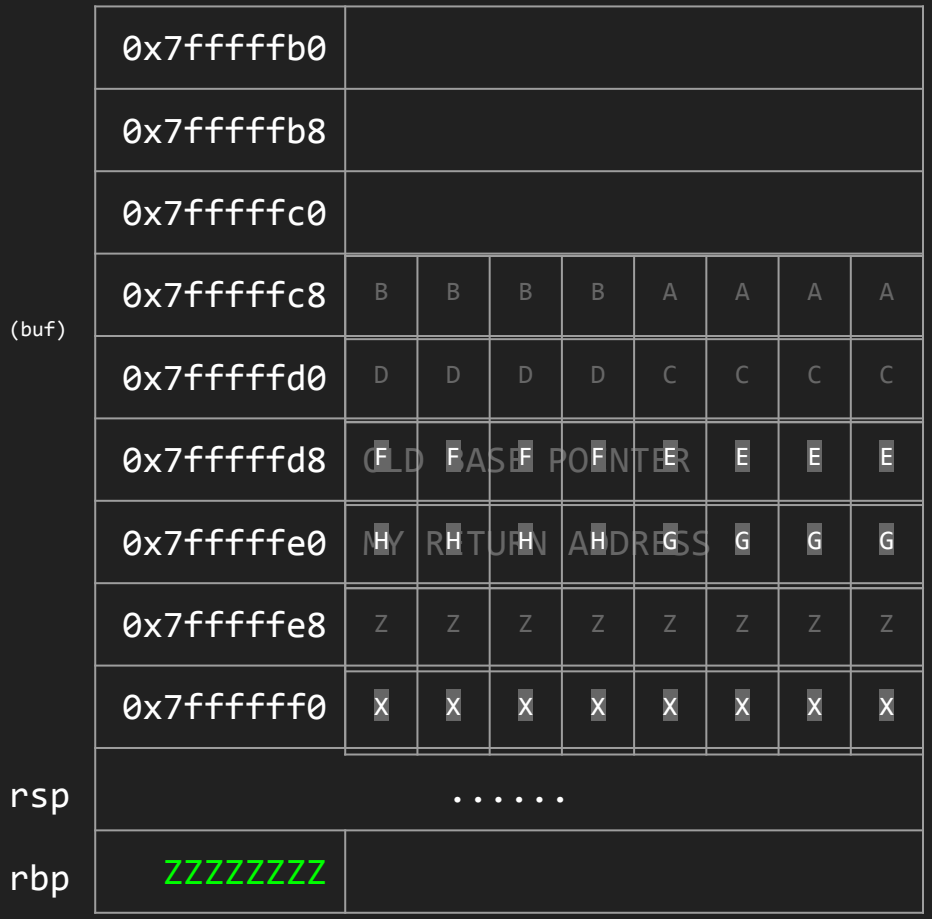
```
hey plz gib data
AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHHZZZZZZZZXXXXXXXX
you shouldn't be here
```




```c
#include <stdio.h>

int main(void) {
       char buf[16];
       printf("hey plz gib data\n");
       gets(buf);
       printf("thx\n");
       return 0;
}

void secret(void) {
  printf("you shouldn't be here\n");
}

void other_secret(void) {
  printf("you shouldn't be here either!\n");
}
```

Output

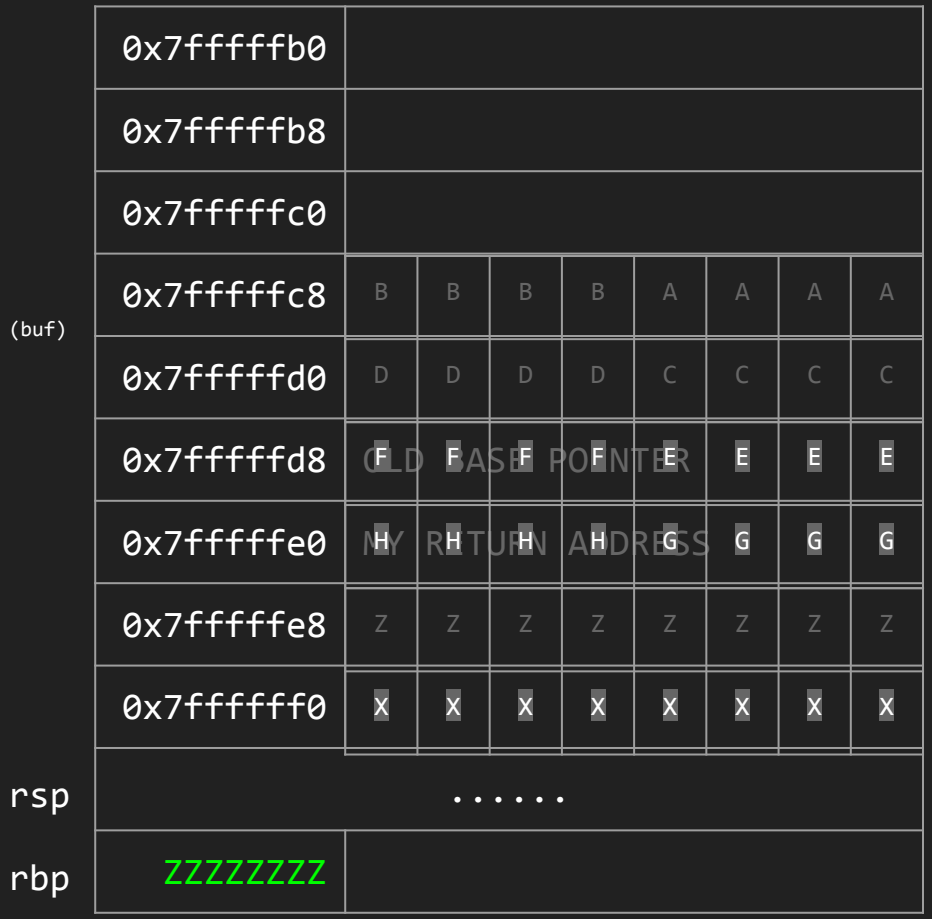
```
hey plz gib data
AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHHZZZZZZZZXXXXXXXX
you shouldn't be here
```




```c
#include <stdio.h>

int main(void) {
       char buf[16];
       printf("hey plz gib data\n");
       gets(buf);
       printf("thx\n");
       return 0;
}

void secret(void) {
  printf("you shouldn't be here\n");
}

void other_secret(void) {
  printf("you shouldn't be here either!\n");
}
```

Output

```
hey plz gib data
AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHHZZZZZZZZXXXXXXXX
you shouldn't be here
you shouldn't be here either!
```




```c
#include <stdio.h>

int main(void) {
       char buf[16];
       printf("hey plz gib data\n");
       gets(buf);
       printf("thx\n");
       return 0;
}

void secret(void) {
  printf("you shouldn't be here\n");
}

void other_secret(void) {
  printf("you shouldn't be here either!\n");
}
```

Output

```
hey plz gib data
AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHHZZZZZZZZXXXXXXXX
you shouldn't be here
you shouldn't be here either!
Segmentation fault (core dumped)
```

Return Oriented Programming

- We can theoretically keep chaining these functions, so long as our stack is big enough
- We can’t input hex data through standard in, so we redirect output into our program
  - printf “hey\x00\x10” | ./rop
  - python3 -c “print(b’hey\x00\x10’)” | ./rop
- Demo

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!