Mason Competitive Cyber

Introduction to reverse engineering

Reverse engineering

: to disassemble and examine or analyze in detail (a product or device) to discover the concepts involved in manufacture usually in order to produce something similar

- Merriam Webster

FIVE STAR

Presents...

Space Station...

Cracked By:

THE ATOM

Thanx To:

The Cracksmith

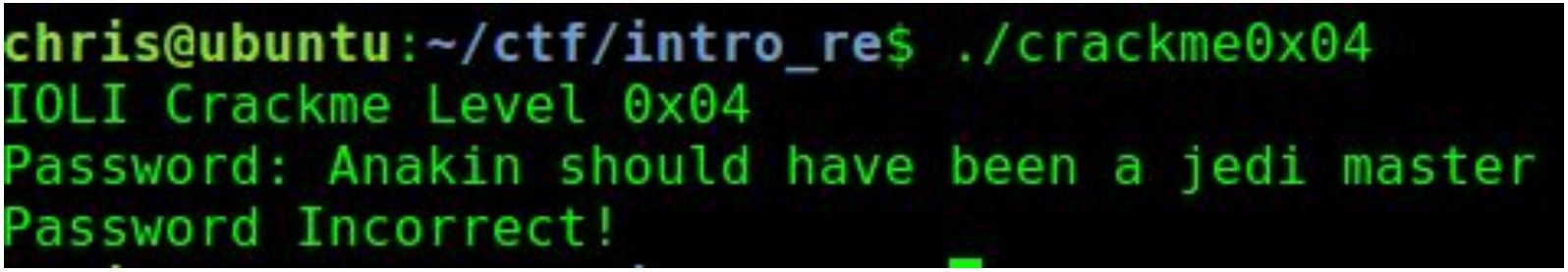
Yellowbeard

Title Page By: The Triton

Early Days

When they couldn't crack the software itself.

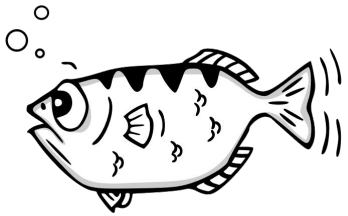
Translating to CTFs



Modern cracking/reverseing

The tools for modern cracking/reverseing

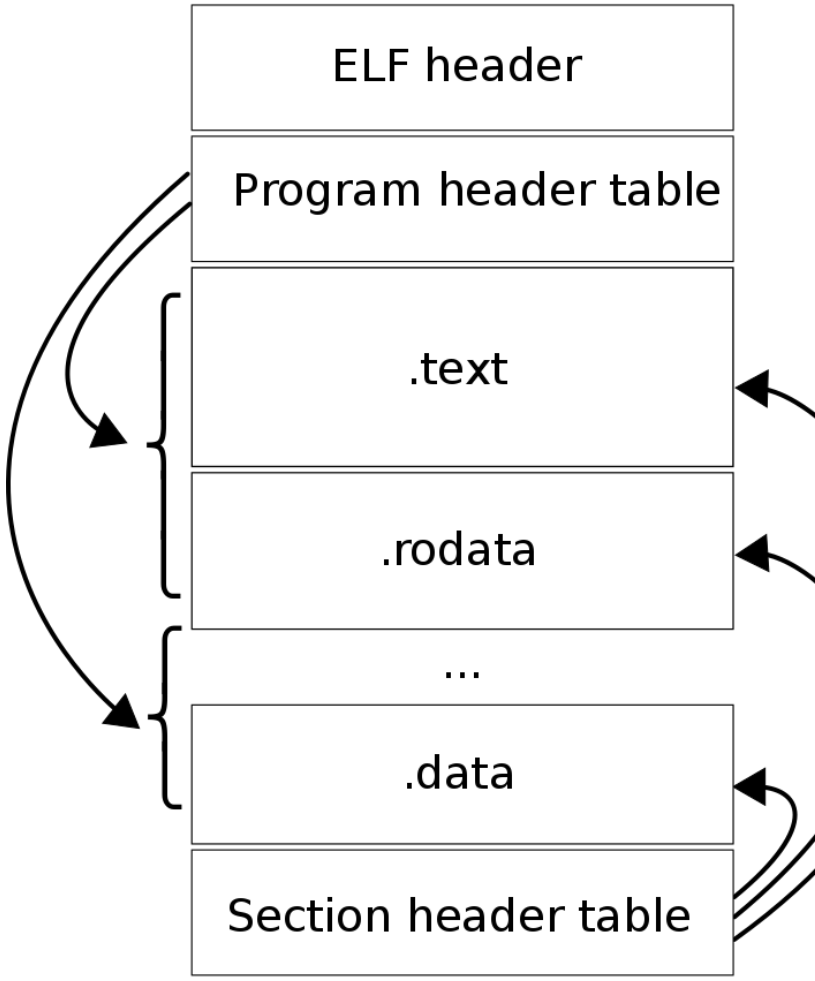
- Strings
- Disassembler
- Debugger



Strings

Strings

- Compiled programs need to place runtime constants somewhere
- Strings are used everywhere and are placed into the “read-only” data section of our binaries.
- In the standard linux bin-utils you will have the command **strings** available to dump out these values.

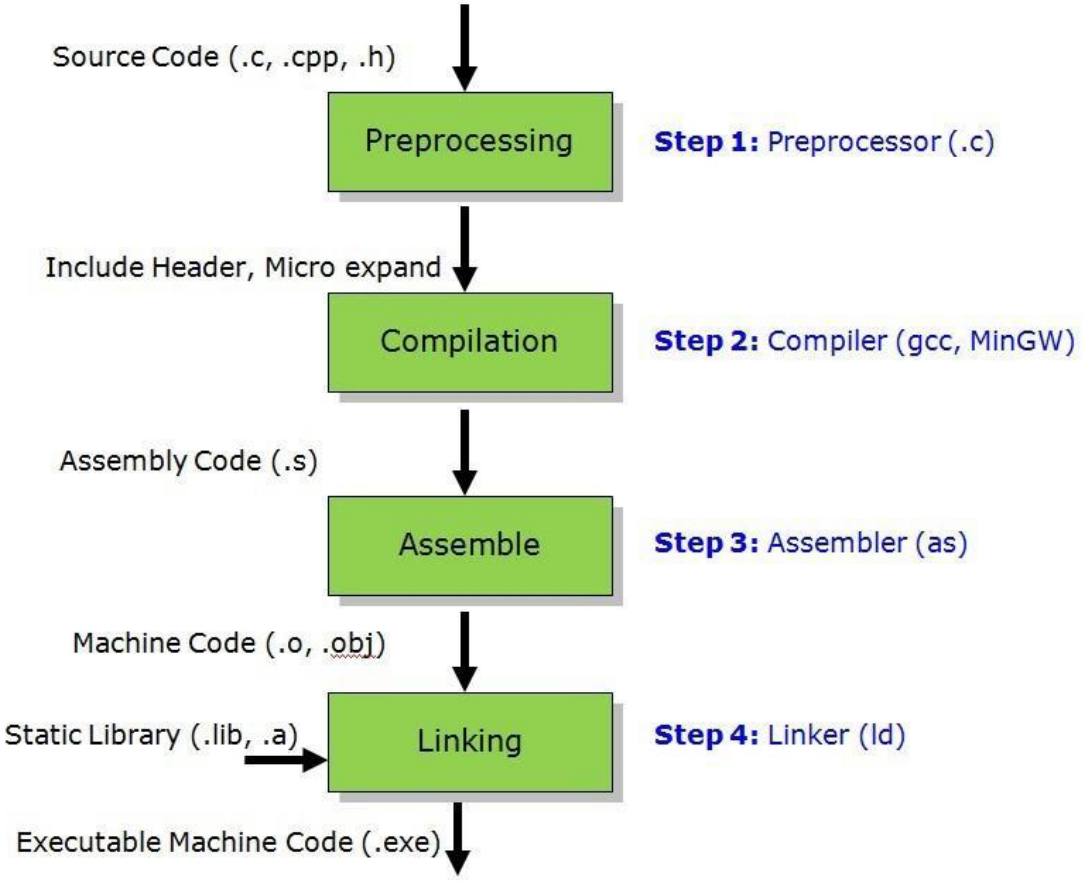


Why don't we just see source

Disassembly

- Most reverse engineering problems are written in C or C++
- These compiled languages produce machine code packed into an executable file format: "ELF"

```
pop eax
pop ebx
pop ebp
ret
```

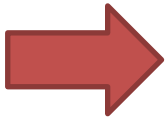


Disassembly

Disassembly

- With the release of Ghidra, this part of reverse engineering has become a lot easier.
- Ghidra is a disassembler released by NSA designed to reverse engineer software.
- We can take compiled code and see pseudocode produced from our assembly

```
                     *************************************************************
                     *                          FUNCTION                         *
                     *************************************************************
                     undefined main()
     undefined         AL:1           <RETURN>
     undefined1        Stack[-0x7c]:1 local_7c                          XREF[2]:     08048540(*),
                                                                                      08048553(*)
     undefined4        Stack[-0x9c]:4 local_9c                          XREF[1]:     08048543(W)
     undefined4        Stack[-0xa0]:4 local_a0                          XREF[4]:     08048528(*),
                                                                                      08048534(*),
                                                                                      08048547(*),
                                                                                      08048556(*)
                     main                                          XREF[2]:     Entry Point(*),
                                                                                _start:080483e7(*)
08048509 55              PUSH    EBP
0804850a 89 e5           MOV     EBP,ESP
0804850c 81 ec 88        SUB     ESP,0x88
         00 00 00
08048512 83 e4 f0        AND     ESP,0xfffffff0
08048515 b8 00 00        MOV     EAX,0x0
         00 00
0804851a 83 c0 0f        ADD     EAX,0xf
0804851d 83 c0 0f        ADD     EAX,0xf
08048520 c1 e8 04        SHR     EAX,0x4
08048523 c1 e0 04        SHL     EAX,0x4
08048526 29 c4           SUB     ESP,EAX
08048528 c7 04 24        MOV     dword ptr [ESP]=>local_a0,s_IOLI_Crackme_Level... = "IOLI Crackme Le
         5e 86 04 08
0804852f e8 60 fe        CALL    printf                                   int printf(char * _
         ff ff
08048534 c7 04 24        MOV     dword ptr [ESP]=>local_a0,s_Password:_08048677   = "Password: "
         77 86 04 08
0804853b e8 54 fe        CALL    printf                                   int printf(char * _
         ff ff
08048540 8d 45 88        LEA     EAX=>local_7c,[EBP + -0x78]
08048543 89 44 24 04     MOV     dword ptr [ESP + local_9c],EAX
08048547 c7 04 24        MOV     dword ptr [ESP]=>local_a0,DAT_08048682           = 25h    %
         82 86 04 08
0804854e e8 21 fe        CALL    scanf                                    int scanf(char * __format, ...)
         ff ff
08048553 8d 45 88        LEA     EAX=>local_7c,[EBP + -0x78]
08048556 89 04 24        MOV     dword ptr [ESP]=>local_a0,EAX
08048559 e8 26 ff        CALL    check                                    undefined check(undefined4 param...
         ff ff
0804855e b8 00 00        MOV     EAX,0x0
         00 00
08048563 c9              LEAVE
08048564 c3              RET
```



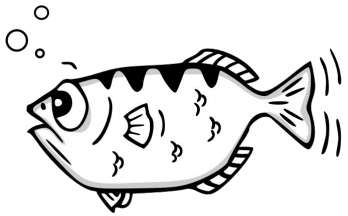
Decompile: main - (crackme0x04)

```
1
2  undefined4 main(void)
3
4  {
5    undefined local_7c [120];
6
7    printf("IOLI Crackme Level 0x04\n");
8    printf("Password: ");
9    scanf("%s",local_7c);
10   check(local_7c);
11   return 0;
12 }
13
```

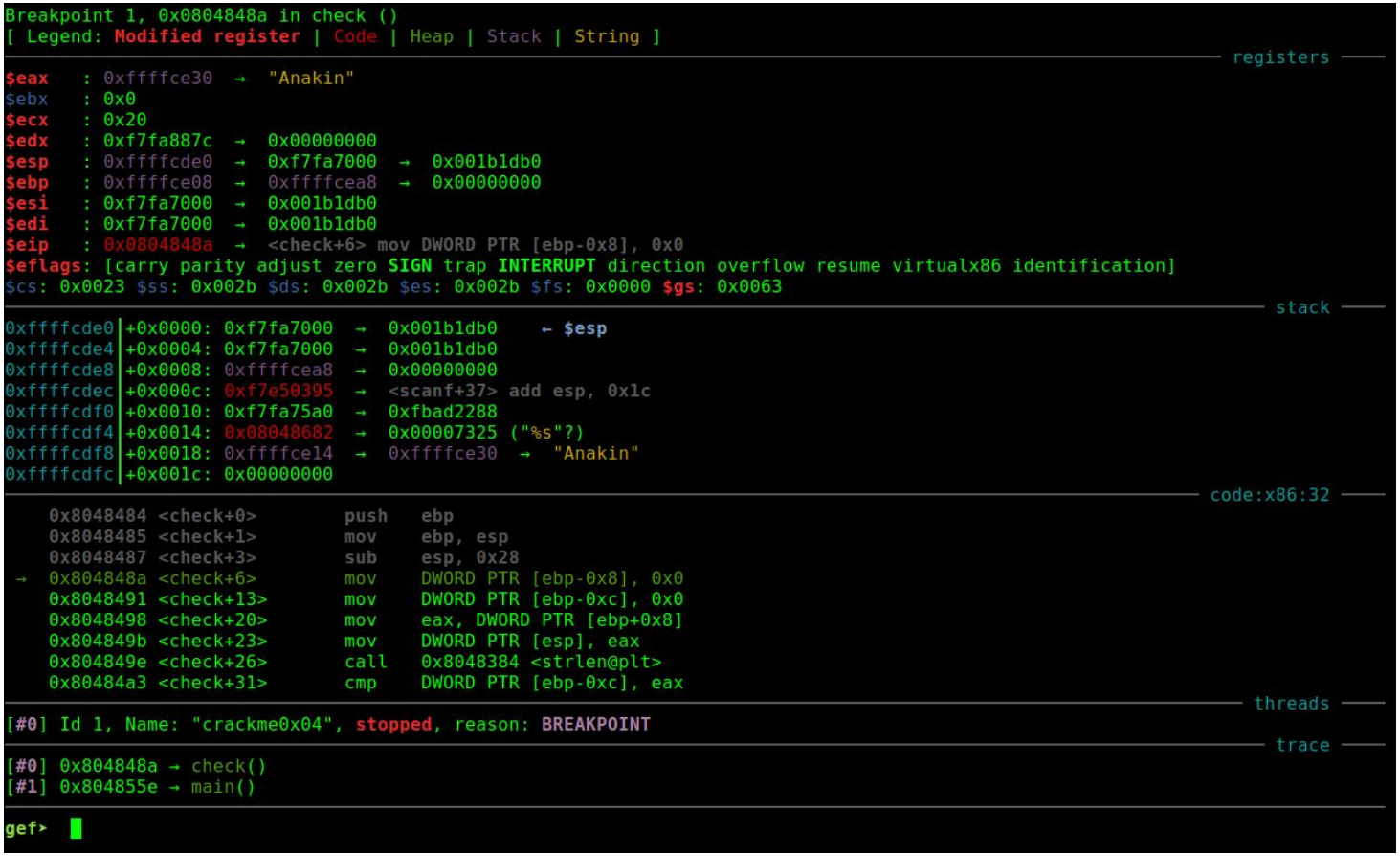
Debugging

GDB

- GDB has been the bread and butter for debugging programs since 1986.
- Run-time level analysis vs static analysis
- Can view what the contents of variables are after a complicated function or routine
- In CTFs you will generally set breakpoints and “dump memory”



Debugging


Breakpoint 1, 0x0804848a in check ()
[Legend: Modified register | Code | Heap | Stack | String]
registers
$eax : 0xffffce30 → "Anakin"
$ebx : 0x0
$ecx : 0x20
$edx : 0xf7fa887c → 0x00000000
$esp : 0xffffcde0 → 0xf7fa7000 → 0x001b1db0
$ebp : 0xffffce08 → 0xffffcea8 → 0x00000000
$esi : 0xf7fa7000 → 0x001b1db0
$edi : 0xf7fa7000 → 0x001b1db0
$eip : 0x0804848a → <check+6> mov DWORD PTR [ebp-0x8], 0x0
$eflags: [carry parity adjust zero SIGN trap INTERRUPT direction overflow resume virtualx86 identification]
$cs: 0x0023 $ss: 0x002b $ds: 0x002b $es: 0x002b $fs: 0x0000 $gs: 0x0063
stack
0xffffcde0 +0x0000: 0xf7fa7000 → 0x001b1db0 ← $esp
0xffffcde4 +0x0004: 0xf7fa7000 → 0x001b1db0
0xffffcde8 +0x0008: 0xffffcea8 → 0x00000000
0xffffcdec +0x000c: 0xf7e50395 → <scanf+37> add esp, 0x1c
0xffffcdf0 +0x0010: 0xf7fa75a0 → 0xfbad2288
0xffffcdf4 +0x0014: 0x08048682 → 0x00007325 ("%s"?)
0xffffcdf8 +0x0018: 0xffffce14 → 0xffffce30 → "Anakin"
0xffffcdfc +0x001c: 0x00000000
code:x86:32
0x8048484 <check+0> push ebp
0x8048485 <check+1> mov ebp, esp
0x8048487 <check+3> sub esp, 0x28
→ 0x804848a <check+6> mov DWORD PTR [ebp-0x8], 0x0
0x8048491 <check+13> mov DWORD PTR [ebp-0xc], 0x0
0x8048498 <check+20> mov eax, DWORD PTR [ebp+0x8]
0x804849b <check+23> mov DWORD PTR [esp], eax
0x804849e <check+26> call 0x8048384 <strlen@plt>
0x80484a3 <check+31> cmp DWORD PTR [ebp-0xc], eax
threads
[#0] Id 1, Name: "crackme0x04", stopped, reason: BREAKPOINT
trace
[#0] 0x804848a → check()
[#1] 0x804855e → main()
gef➤

Demos

MetaCTF 2018

- Stringy McStringFace (150 pts)
- Line by line, life's a flag (275 pts)

On your own

I'll also throw this link on slack.

https://github.com/angr/angr-doc/tree/master/examples/CSCI-4968-MBE/challenges

- crackme0x00a
- crackme0x01
- crackme0x02
- crackme0x03
- crackme0x04
- crackme0x05

Proud Sponsors

Thank you to these organizations who give us their support

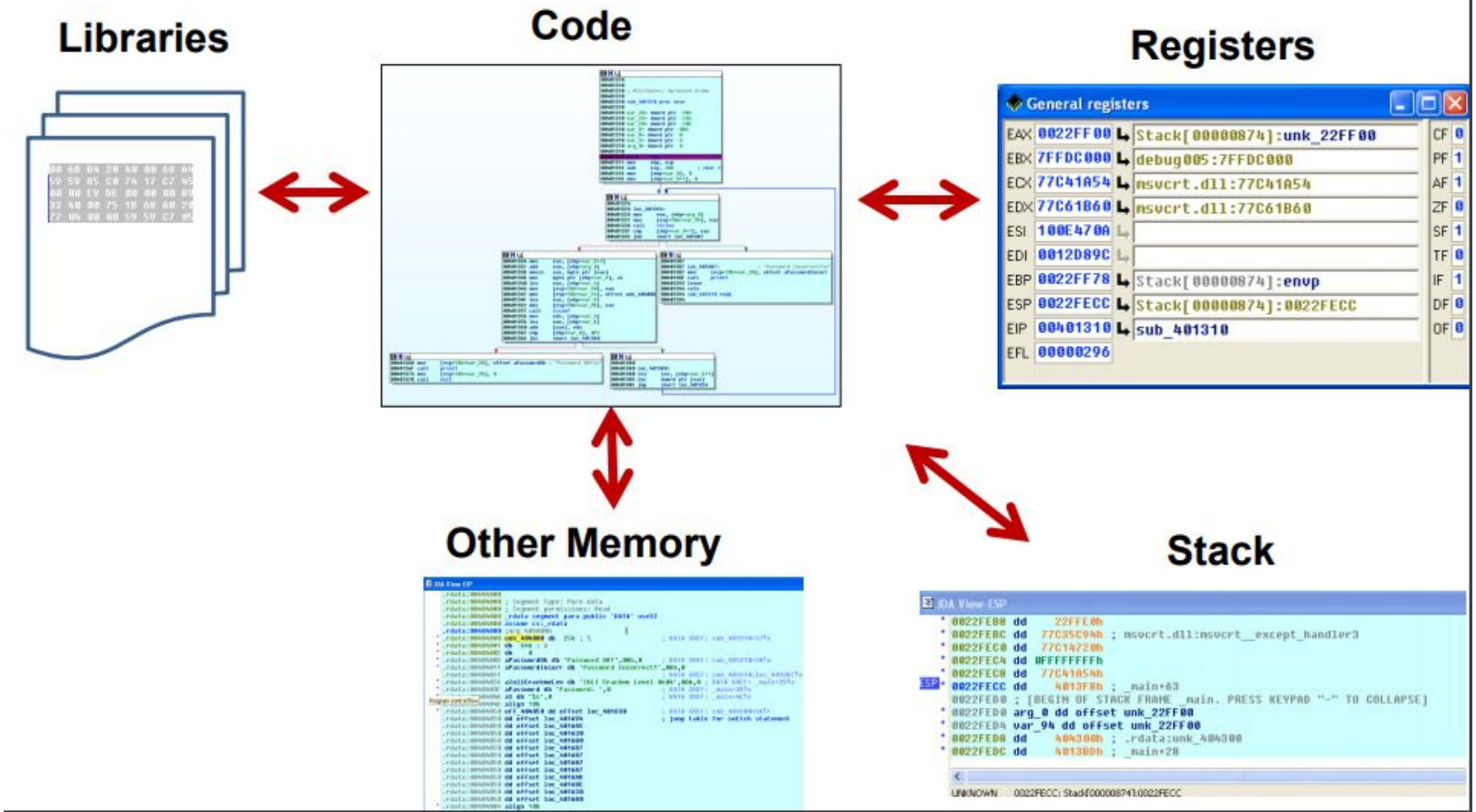
They're hiring and have hired us before

BATTELLE

It can be done™

CRYPSIS™

RE Domain


Libraries
Code
Registers
General registers
EAX 0022FF00 Stack[00000874]:unk_22FF00
EBX 7FFDC000 debug005:7FFDC000
ECX 77C41A54 msvcrt.dll:77C41A54
EDX 77C61B60 msvcrt.dll:77C61B60
ESI 100E470A
EDI 0012D89C
EBP 0022FF78 Stack[00000874]:envp
ESP 0022FECC Stack[00000874]:0022FECC
EIP 00401310 sub_401310
EFL 00000296
CF 0
PF 1
AF 1
ZF 0
SF 1
TF 0
IF 1
DF 0
OF 0
Other Memory
Stack
IDA View-ESP
0022FEB8 dd 22FFE0h
0022FEBC dd 77C35C94h ; msvcrt.dll:msvcrt__except_handler3
0022FEC0 dd 77C1A720h
0022FEC4 dd 0FFFFFFFFh
0022FEC8 dd 77C41A54h
ESP 0022FECC dd 4013F8h ; _main+63
0022FED0 ; [BEGIN OF STACK FRAME _main. PRESS KEYPAD "-" TO COLLAPSE]
0022FED0 arg_0 dd offset unk_22FF00
0022FED4 var_94 dd offset unk_22FF00
0022FED8 dd 404300h ; .rdata:unk_404300
0022FEDC dd 401380h ; _main+20
UNKNOWN 0022FECC: Stack[00000874]:0022FECC

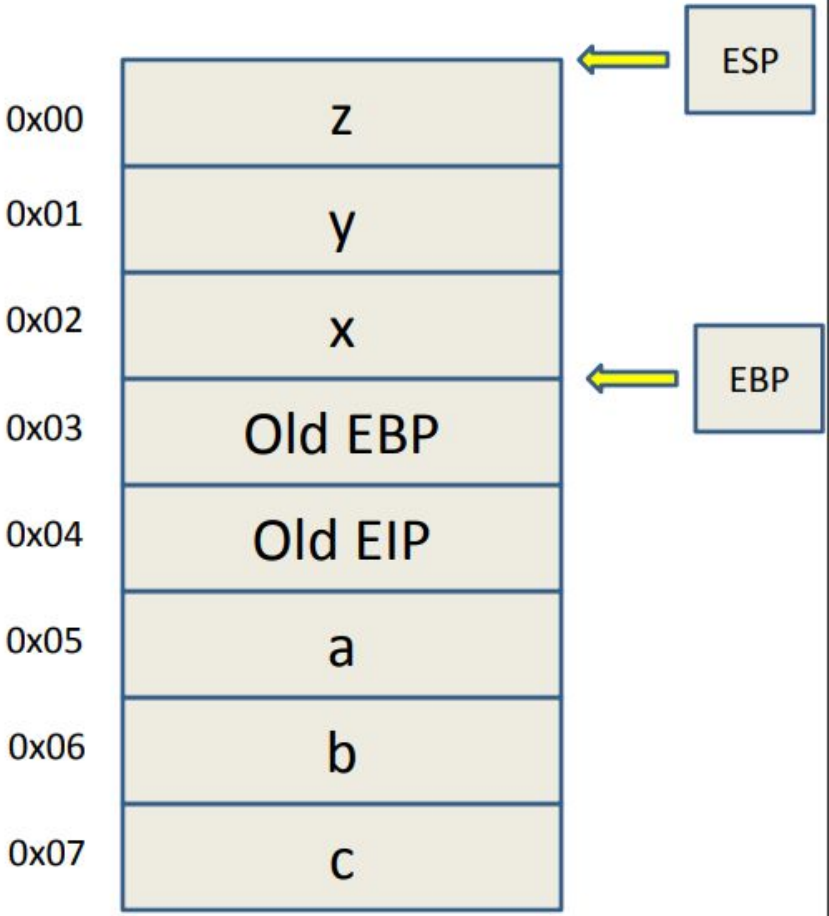
Stack

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

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


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


        foo(1,2,3);
}
```



More intro resources

- http://security.cs.rpi.edu/courses/binexp-spring2015/lectures/2/02_lecture.pdf
- http://security.cs.rpi.edu/courses/binexp-spring2015/lectures/3/03_lecture.pdf