Introduction to Intel x86-64 Assembly, Architecture, Applications, & Alliteration

Xeno Kovah – 2014-2015
xeno@legbacore.com
All materials is licensed under a Creative Commons “Share Alike” license.

- http://creativecommons.org/licenses/by-sa/3.0/

Attribution condition: You must indicate that derivative work
"Is derived from Xeno Kovah’s ‘Intro x86-64’ class, available at http://OpenSecurityTraining.info/IntroX86-64.html"
LetErRIP.c
RIP-relative Addressing

```c
//LetErRIP.c
char someGlobal = 0;

short main()
{
    char a = 1;
    someGlobal = a;
    return 22;
}
```

```assembly
main:
0000000140001000 48 83 EC 18            sub       rsp,18h
0000000140001004 C6 04 24 01            mov       byte ptr [rsp],1
0000000140001008 0F B6 04 24            movzx     eax,byte ptr [rsp]
000000014000100C 88 05 EE 44 06 00      mov       byte ptr [40065500h],al
0000000140001012 B8 16 00 00 00         mov       eax,16h
0000000140001017 48 83 C4 18            add       rsp,18h
000000014000101B C3                      ret
```
//LetErRIP.c
char someGlobal = 0;

short main()
{
    char a = 1;
    someGlobal = a;
    return 22;
}

Takeaways:
Visual Studio 2012 displays RIP-relative addresses _misleadingly_!
64 bit bug I think. _View with symbols to see the accurate address_
(Some students said this was fixed in VS 2015)
char someGlobal = 0;

short main(){
    char a = 1;
    someGlobal = a;
    return 22;
}

On Linux:

RIP-relative Addressing

More clearly RIP-relative

Helpful math of next instruction (0x4004FF) + displacement (0x200B3A) = 0x601039