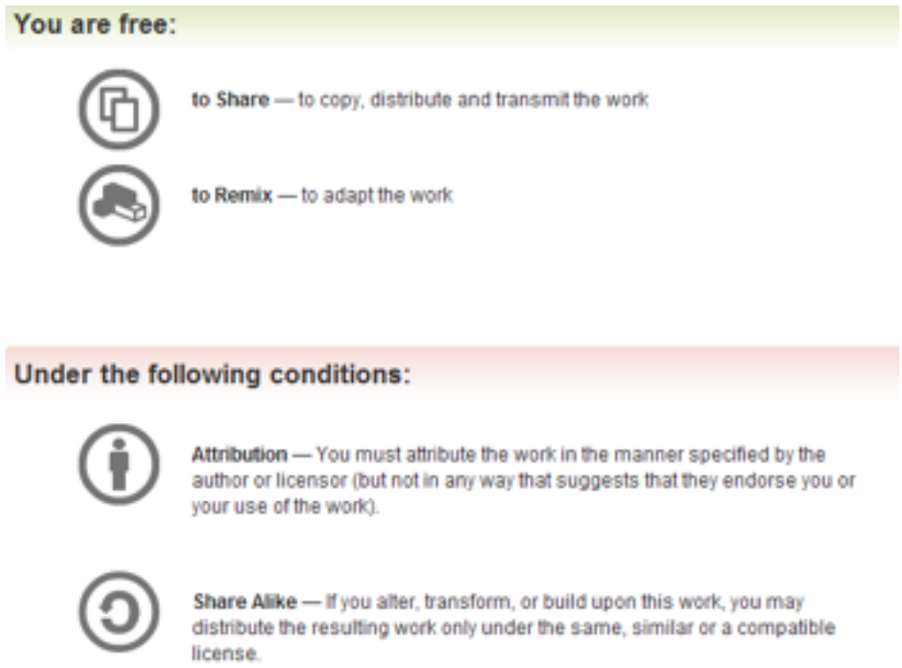
Introduction to Intel x86-64 Assembly, Architecture, Applications, & Alliteration

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"

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"

Effects of Compiler Options

Our standard build

```
//Example8.c
int main(){
        char buf[40];
        buf[39] = 42;
        return 0xb100d;
}
```

```
main:
140001000  sub         rsp,38h
140001004  mov         eax,1
140001009  imul        rax,rax,27h
14000100D  mov         byte ptr [rsp+rax],2Ah
140001011  mov         eax,0B100Dh
140001016  add         rsp,38h
14000101A  ret
```

Effects of Compiler Options 2

/O1 (minimum size) or
/O2 (maximum speed)

```
main:
140001000  mov         eax,0B100Dh
140001005  ret
```

Debug information format
Disabled (viewed from WinDbg) or
/Z7 (C7 Compatible)
(no change)

```
main:
140001000  sub         rsp,38h
140001004  mov         eax,1
140001009  imul        rax,rax,27h
14000100D  mov         byte ptr [rsp+rax],2Ah
140001011  mov         eax,0B100Dh
140001016  add         rsp,38h
14000101A  ret
```

Effects of Compiler Options 3

/GS - Buffer Security Check (default enabled nowadays)
aka “stack cookies” (MS term)
aka “stack canaries” (original research term)

```
main:
140001000  sub         rsp,38h
140001004  mov         rax,qword ptr [__security_cookie (0140004000h)]
14000100B  xor         rax,rsp
14000100E  mov         qword ptr [rsp+28h],rax
140001013  mov         eax,1
140001018  imul        rax,rax,27h
14000101C  mov         byte ptr [rsp+rax],2Ah
140001020  mov         eax,0B100Dh
140001025  mov         rcx,qword ptr [rsp+28h]
14000102A  xor         rcx,rsp
14000102D  call        __security_check_cookie (0140001190h)
140001032  add         rsp,38h
140001036  ret
```

Effects of source options

/O1 optimization when the volatile keyword is present

```
int main(){
        volatile char buf[40];
        buf[39] = 42;
        return 0xb100d;
}
```

```
main:
140001000  sub         rsp,38h
140001004  mov         eax,1
140001009  imul        rax,rax,27h
14000100D  mov         byte ptr [rsp+rax],2Ah
140001011  mov         eax,0B100Dh
140001016  add         rsp,38h
14000101A  ret
```

```
main:
140001000  sub         rsp,38h
140001004  mov         byte ptr [rsp+27h],2Ah
140001009  mov         eax,0B100Dh
14000100E  add         rsp,38h
140001012  ret
```

This is a trick I picked up from a 2009 Defcon presentation http://www.defcon.org/images/defcon-17/dc-17-presentations/defcon-17-sean_taylor-binary_obfuscation.pdf

He also talked a little bit about control flow flattening which is covered in an academic paper in the “Messing with the disassembler” section.