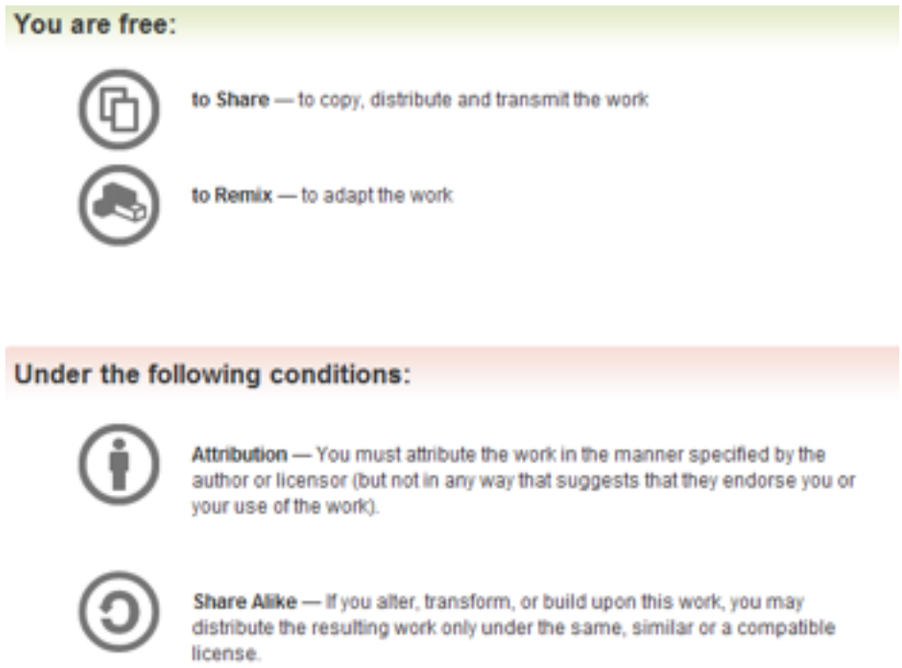
Introduction to Intel x86-64 Assembly, Architecture, Applications, & Alliteration

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"

Prepared VM

- student/student
- root/toor (but you shouldn’t need this account, student has sudo)

If you are using some other Linux system, just put bomb-x64 on it (from the “TheseGoInVMs” class material directory), and make sure you have gdb installed

Bomb lab

- From CMU architecture class - http://csapp.cs.cmu.edu/public/labs.html
- Thanks to Randal E. Bryant & David R. O'Hallaron for providing the source code so it could be ported to x86-64 (and Windows in the Intro RE class, and ARM in the Intro ARM class)
- The textbook for the class which the bomb lab is a part of is "Computer Systems: A Programmer's Perspective, 2nd Edition, Prentice Hall, 2011; Bryant and O'Hallaron"

Bomb lab 2

- Goal is to reverse engineer multiple phases to determine the program's desired input
- Create a text file with answers, one per line, named "answers"
- gdb -x myCmds bomb-x64
- run with "r answers"
- Should add/remove breakpoints on the different phases as you go along

Bomb lab - EXPERT MODE!

- If you already know a thing or two about asm (and were just here for the 64 bit update), let's see how far you can get how fast if you play it on expert mode, without symbol information. Execute the following command in the directory where the bomb resides:
  - strip bomb-x64
- This is more like what you will actually see with malware. You're not going to get symbols in that case.
- *Now* go ahead and see how fast you can go through the rounds ;)

Phase_2 hint

- sscanf() is defined as follows:
- `int sscanf(const char *str, const char *format, ...);`
- So if it was e.g. “sscanf(foo, “%d %d”, &a, &b)”
- It would take whatever string was pointed to by the first argument, *parse* it according to the second format string argument, and then store the parsed out values in the variables which were given by the subsequent n arguments (for n = 2 in this case)
- “On success, the function returns the number of variables filled"

Phase_2 hint

- sscanf() is defined as follows:
- `int sscanf(const char *str, const char *format, ...);`
- So if it was e.g. “sscanf(guess, “%d %d %d %d %d %d”, &var1, &var2, &var3, &var4, &var5, &var6)”… then whatever numbers were given in the guess string, would be placed into the respective variables
- And sscanf() returns the number of successfully parsed format elements

Break glass in case of emergency

- For purposes of doing the lab, you should not use this
  - Do it the rigorous way the first time, then do it the faster way the next times
- But in the real world, if you're REing something in gdb, you can

- To set a register to a specific value
  - set $rax = 0x1234
- To set a memory location to a specific value
  - set {int}0x7FFFFFFFE80 = 0x1

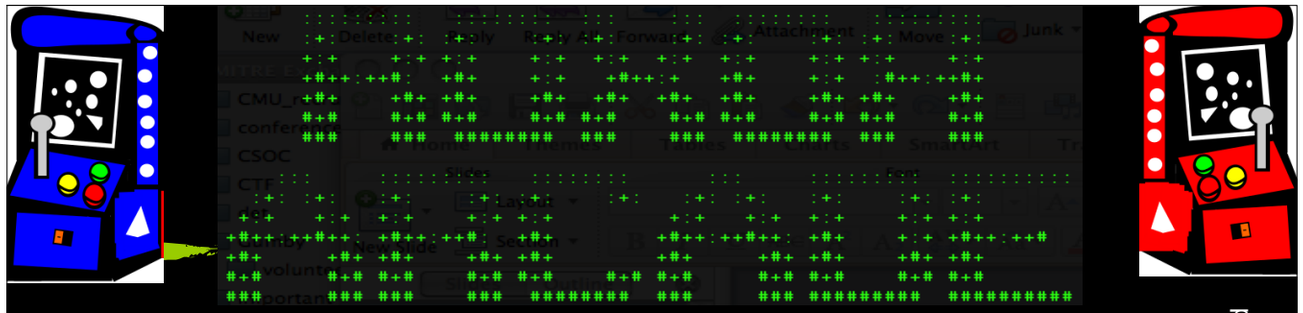
GDB/Bomb Lab Cheat Sheet

- Christian Arllen found this, and it has many more example of gdb syntax, as well as some help for if you get stuck on the lab

- http://condor.depaul.edu/~jriely/csc373fall2010/extras/mygdbnotes.txt
- (get it on google cache while you can, because it's gone now)

Optional Labs

- perl r0x0r-arcade-read-only/start.pl
- Prototype games that I haven’t had time to work on since 2012, let alone update for x86-64…



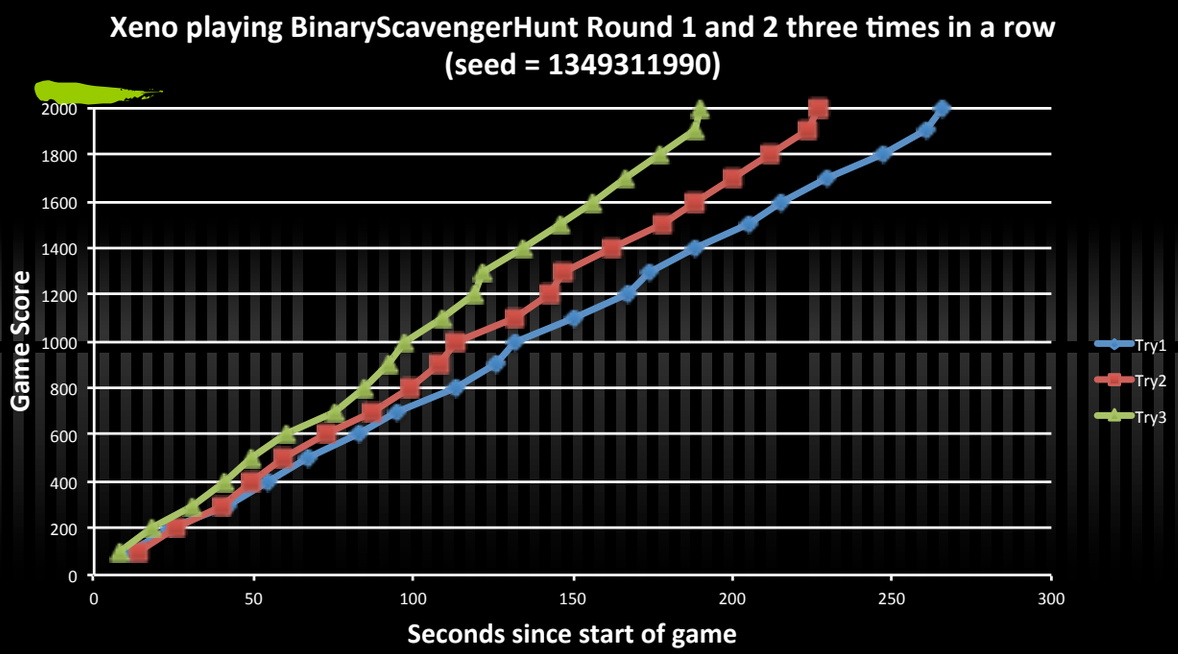
- Some learning requires grinding!
- But it can still be accelerated!

- http://code.google.com/p/roxor-arcade
  - Such timeless classics as:
  - BinDeciHex
  - The *other* ESP game
  - 1 step forward 3 steps back
  - BinaryScavengerHunt
  - May your buffer overfloweth

- Quick demo of "BinaryScavengerHunt" if I'm using my own laptop to present.

From "TrainingSecurityExpertsAtScale" presented by Xeno at the NIST-NICE conference 2014

From “Training Security Experts At Scale” presented by Xeno at the NIST-NICE conference 2014



From “Training Security Experts At Scale” presented by Xeno at the NIST-NICE conference 2014