Coding for Exploit Dev (try putting this in browser quest)

1. Coding Basics
   1. Primitives
   2. Pointers
   3. Calling Conventions
   4. x86 vs ARM vs x64
      1. x86 Registers
      2. ARM Registers
      3. x64 Registers
   5. Memory (stack vs heap)
      1. What happens when I enter a function (push ebp)?
      2. What happens on memory allocation?
         1. VitrualAlloc
         2. Malloc
         3. mmap
2. Building your dev environment
   1. KVM Setup (use prebuilt VMs)
   2. using virt-manager create x86/x64/arm VMs (5 total)
3. Protection Mechanisms
   1. DEP (Software)
   2. NX/XN
   3. ASLR/PIE
4. Vulnerable Conditions (SAMATE examples)
   1. Buffer Overflow
   2. Memory Corruption
   3. Race Conditions TOCTOU (Time of Check Time of Use)
   4. Use after free
   5. Double free
   6. Heap Overflow
   7. Integer wrap
   8. Information Leaks
5. Crashing Programs (SAMATE examples)
   1. Too much data in the pants (BoF)
   2. Unexpected input (memory corruption)
   3. Dirty Input (format strings)
   4. Too busy for you (create item, long operation, check item in loop - race condition)
   5. Forget to scratch it off the list (Use after free, removing references)
6. Visualizing the Program
   1. IDA Pro

Exercise ideas:

1. Use touchscreen to do the instructional games
   1. push the args to the function in the right order
   2. assemble an exploit by dropping bits of as into a bucket

Running BrowserQuest

1. Use this repo for Ubuntu 14.10 <https://github.com/stvrbbns/BrowserQuest>
2. Install nodejs-legacy using apt-get

Coding Class

1. Coding basics
   1. OOO v Procedural
   2. C
   3. C++
   4. C#
   5. Java
   6. D
2. Scripting Basics
   1. Python
   2. Bash
   3. Batch
3. Intro to assembly
   1. x86
   2. x64
   3. ARM
4. Intro to Visual Studio
   1. C Build
   2. C++ Build
   3. C# Build
   4. D Build
5. Intro to VS Debugging
   1. Debug program (switch to single step)
6. Intro to GDB
   1. Loading a program
   2. Apropos, and searching for help
   3. ptype, gdb scripting, etc
7. Intro to Binutils
   1. objdump
   2. readelf (not part of binutils but whatever)
8. Intro to VS Tools (dumpbin, etc)
   1. vs command prompt
   2. dumpbin
9. Intro to SQL
   1. MySQL
   2. SQLite
   3. SQL Injection
10. Building a dev environment with Qemu/KVM
    1. Bring Ubuntu 14.04 machine, and install kvm environment
11. Intro to protocol sniffing and filtering
    1. Wireshark
    2. netmon
    3. filtering
    4. tcpdump
12. Bus devices (USB, UART, FTDI, SPI)
    1. USB protocol
    2. USB Sniffing
    3. FTDI/UART and Salea? or Buspirate
13. Buffer overflows
    1. Create examples from Juliet suite in SAMATE
14. Memory Corruption
    1. Create examples from Juliet suite in SAMATE
15. Use after-free
    1. Create examples from Juliet suite in SAMATE
16. NOPsleds and Heapsprays
    1. Create examples from Juliet suite in SAMATE

HELPFUL LINKS:

<http://corelabs.coresecurity.com/index.php?module=Wiki&action=attachment&type=area&page=Vulnerability_Research&file=publication%2FAbout_Exploits_Writing%2F2002.gera.About_Exploits_Writing.pdf>

<http://community.coresecurity.com/~gera/InsecureProgramming/>

USING FEDORA 21 VM

sudo qemu-system-arm -M versatilepb -kernel vmlinuz-3.2.0-4-versatile -initrd initrd.img-3.2.0-4-versatile -hda debian\_wheezy\_armel\_desktop.qcow2 -append "root=/dev/sda1" -redir tcp:9833::3389 -redir tcp:2222::22 -usb -device usb-host,hostbus=1,hostaddr=1 <- starts up with GUI and allows use of keyboard, see this link for the usb keyboard stuff <http://stackoverflow.com/questions/19665412/mouse-and-keyboard-not-working-in-qemu-emulator>

Progress (4/20/2015)

Made qemu raspbian vm on alienware - zip up qemu dir in Downloads and distribute once samate and build tools are on there along with gdb and gdb server

Using Jenkins CI for building cases (SAMATE examples):

<http://software-novotny.de/raspberry-pi-remote-compilation-with-jenkins> <- use the Fedora env described above, on a CloudAtCost box (need to test this)