Lab Project #2: Bomblab

Professor Hugh C. Lauer
CS-2011, Machine Organization and Assembly Language

Binary Bomb

- A binary program compiled from *C*
- Six phases
- Each phase expects a particular string on `stdin`
- Incorrect string causes bomb to "explode"
- Correct string "defuses" that phase, allows you to move on to next phase
- Increasing difficulty with each phase

Your Bomb

- **Bomblab server:–**
  - http://cs2011.cs.wpi.edu:15213/ to download a new bomb
  - http://cs2011.cs.wpi.edu:15213/scoreboard to view progress
- **Server generates a different bomb for each student!**
  - Similar phases
  - Different strings
- **Grading:–**
  - 10 points each for phases 1-4
  - 15 points each for phase 5
  - 20 points for phase 6
  - –1 point *each time* you explode bomb!

Each bomb records its own progress with server!

Pre-generating Bombs

- **Server does not have enough power to respond to 25 “new bomb” requests at one time**
  - E.g., start of Recitation section
- **Therefore please, please, please …**
  - Download one *or more* bombs *before* Wednesday!
- **Each bomb has *your* loginID and e-mail address embedded in it**
  - When grading, credit goes to whose name is on bomb!

Bomblab – B-term 2017

- **Must run on Ubuntu virtual machine**
  - Checks for identity of machine!

- **Does not currently check for identity of Virtual Machine**
  - Abuse of this characteristic is a violation of WPI's "Acceptable Use Policy"!

- **Unsuccessful in making it run on CCC Linux, Eclipse on Windows, MacOS, etc.**
  - In previous terms

How to Defuse a Bomb

Use the debugging tools!

Tools

Any other tools you can find!

- **`gdb` — the Gnu Debugger**
  - Introduced in CS-2301 & CS-2303
- **`Eclipse` —installed on course Virtual machine**
  - Easy to install if not already there
  - See notes on how to debug existing binary
- **`ddd` — Data Display Debugger**
  - A GUI front-end for **`gdb`**
  - Need to install on your virtual machine
  - No longer supported!

`Nemiver` — discovered by a student two years ago. Suitable successor to `DDD`.

- **`strings` — Linux/Unix "strings" utility**
  - Prints out all ASCII strings in a file

`PEDA`— introduced by TA last year

- **`objdump` — Linux/Unix "object file dump" utility**
  - Displays lots of useful information about a binary "object" file

Using gdb with 64-bit assembly code

```
Dump of assembler code for function phase_1:
   0x0000000000400f90 <+0>:     sub    $0x8,%rsp
   0x0000000000400f94 <+4>:     mov    $0x402730,%esi
   0x0000000000400f99 <+9>:     callq  0x401468 <strings_not_equal>
   0x0000000000400f9e <+14>:    test   %eax,%eax
   0x0000000000400fa0 <+16>:    je     0x400fa7 <phase_1+23>
   0x0000000000400fa2 <+18>:    callq  0x401741 <explode_bomb>
   0x0000000000400fa7 <+23>:    add    $0x8,%rsp
   0x0000000000400fab <+27>:    retq
End of assembler dump.
```

Setup

if

Finish

Function calls

A learning exercise

- **Study the assembly code**
- **Bryant & O'Hallaron — Chapter 3**
  - Reverse engineer the C code!
  - A little bit of each kind of C statement
- **Setting breakpoints**
- **Single step through the assembly code**
  - **`stepi`** — one *machine* instruction
  - **`nexti`** — same as **`stepi`** but skips over function calls
  - **`disassemble`** — dump out a fragment of machine code
  - **`examine`** — look at an area of memory (pointed to be a register)

Strongly recommend

- **Debugger with graphic user interface**
- **Can see multiple pieces of information at the same time in separate windows**
  - Automatically updates at every breakpoint or pause
- **Eclipse — open-source IDE**
  - Special facilities for debugging existing binaries
  - Register and disassembly windows
  - Already installed on course virtual machine

Both are front ends to `gdb`!

- **ddd — The Data Display Debugger**
  - *Not* installed on Virtual Machine
  - Easy to install

Not updated since 2007!

Optionally available with modern Linux distributions

Strongly recommend (continued)

- **Nemiver — successor to DDD**
  - Professor has never used it seriously
  - Full GUI debugging

- **PEDA — *<u>P</u>ython* <u>E</u>xploit <u>D</u>evelopment**
  - Introduced by Nilesh Patel, TA from 2015
  - Plugin for GDB — friendlier display of registers, memory, disassembly of binary, etc.

“Old” *versus* “New” Bomblab

- **In spring 2016, WPI student created static solver for CMU’s bomblab**
  - I.e., the “old” bomblab
  - Could read and print out solution for any bomb without ever executing it

- **In 2016-17, MQP team created new version that forces execution of bomb**
  - I.e., you *must* use a debugger to solve bomb
  - Successfully used in D-term 2017 under management of team
  - Nicer server and web page

Operationally, still very new!

Questions?