15-213 Recitation: Data Lab

January 29, 2024

Agenda

- Office Hours (Reminder)
- Data Lab
  - Getting started
  - Running your code
  - Reminders
- Looking ahead to Bomblab

Office Hours

- Queue link: https://ohq.eberly.cmu.edu/#/
- Please locate the TA in the specified location!
- OH Logistics will be on post @74

Day	Location	Time
Monday	GHC 6501	7pm - 10pm
Tuesday	GHC 8102	7pm - 10pm
Wednesday	GHC 8102	7pm - 10pm
Thursday	TBD	7pm - 10pm
Friday	TBD	7pm - 10pm
Saturday	GHC 6501	2pm - 4pm
Sunday	GHC 8102	2pm - 4pm

OH Etiquette

- Office hours are for getting ideas on how to debug or better approach your homework!
- Please try to narrow down your problem area as much as possible to help TAs help you!
- **Write a description!** If you don't have a description, you may be frozen/removed from the queue. Make sure to use the tags!
- TAs will only spend 10 minutes per student and then you can rejoin the queue.
- We will close the queue early so everyone can be helped so please keep this in mind!

Data Lab: Getting Started

- Download the handout from autolab
  - Method 1:
    - `scp <path to datalab.tar> <andrewid>@shark.ics.cs.cmu.edu:<my course directory>`
    - `ssh <andrewid>@shark.ics.cs.cmu.edu`
    - `cd to the datalab.tar file`
    - `tar -xf datalab.tar`
  - Method 2:
    - `autolab download 15213-s24:datalab`
    - `cd into the datalab folder`
    - `tar -xf datalab.tar`

Data Lab: Getting Started

- Upload `bits.c` file to Autolab for submission
  - `make submit`

Data Lab: Running your code

- `dlc`: a modified C compiler
- `btest`: runs your solutions on random values
- `bddcheck`: exhaustively tests your solutions
  - Checks all values, formally verifying the solution
- `driver.pl`: Runs both dlc and bddcheck
  - Exactly matches Autolab's grading script
  - You will likely only need to submit once
- For more information, **read the writeup**
  - Available under autolab as "**View writeup**"
  - **Read the writeup please!**

Data Lab: Reminders

- Casting between **int** and **long** is ok, in either direction
- Be aware of operations and their types!
  - ! returns an **int** *even if the argument is a long*
- Good idea to append "L" suffix to every integer constant
  - `(1L << 63)` is not the same as `1 << 63`
  - `(!x << 63)` is not the same as `((long)!x) << 63`

Form Groups of 3 - 4

- Series of exercises
  - Operators
  - Puzzle

Questions?

- Remember, DataLab is due this Thursday (2/1)!
  - You really should have started already!
- Read the lab writeup!

Looking Ahead… Bomblab!!

What is Bomb Lab?

- An exercise in reading x86-64 assembly code.
- A chance to practice using GDB (a debugger).
- Why?
  - x86 assembly is low level machine code. Useful for understanding security exploits or tuning performance.
  - GDB can save you days of work in future labs **(Malloc)** and can be helpful long after you finish this class.

Downloading Your Bomb

- Here are some highlights of the write-up:
  - Bombs can only run on the shark machines. They fail if you run them locally or on another CMU server.
  - Each bomb is unique - if you download a second bomb, bad things can happen! Stick to only one bomb.
  - Bombs have six phases which get progressively harder.
  - Make sure to read the writeup for more tips and common mistakes you might make.

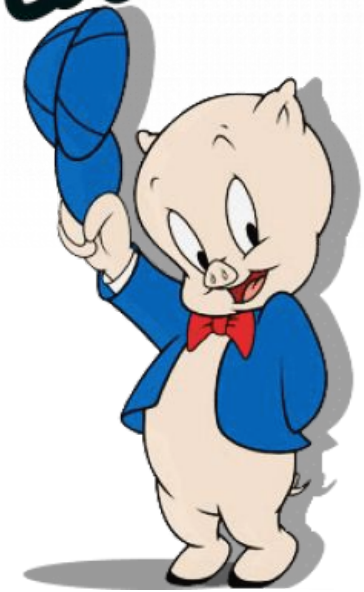
Detonating Your Bomb

- Blowing up your bomb automatically notifies Autolab
  - ***Dr. Evil*** deducts 0.5 points each time the bomb explodes.
  - It's very easy to prevent explosions using **break points** in GDB. More information on that soon.
- Inputting the correct string moves you to the next phase.
- Don't tamper with the bomb. Skipping or jumping between phases detonates the bomb.
- You have to solve the phases in order they are given. Finishing a phase also notifies Autolab automatically.

Bomb Hints

- ***Dr. Evil*** may be evil, but he isn't cruel. You may assume that functions do what their name implies
  - i.e. phase_1() is most likely the first phase. printf() is just printf(). If there is an explode_bomb() function, it would probably help to set a breakpoint there!
- Use the man pages for library functions.
  - Although you can examine the assembly for snprintf(), we assure you that it's easier to use the man pages ($ man snprintf) than to decipher assembly code for system calls.
- Most cryptic function calls you'll see (e.g. callq … <_exit@plt>) are also calls to C library functions.
  - You can safely ignore the @plt as that refers to dynamic linking.

"That's all Folks!"



Bomb lab Slides

https://docs.google.com/presentation/d/1OjLAjPxhEXsElWsvmNnMGD-fzNCeGXip/edit#slide=id.p1