Engineering Robust Server Software

Introduction
Welcome To ERSS!

- Welcome to Engineering Robust Server Software (ERSS)
  - A brand new class [first time offered]
    - Pardon any rough edges
    - Feedback/suggestions welcomed
- Introductions:
  - I'm Drew Hilton——call me Drew
    - Many of you know me from 551 (but not all)
  - Introduce yourselves to everyone
Assumptions Going Into This Class

1. I assume you want to be a software development professional
2. I assume you are taking 650 (or have equivalent preparation)
   - You are competent C programmer (Mastery of 551 material)
   - You know basic systems concepts: caching, instructions, etc… (550)
   - If not in 650, you know or are learning:
     - Programming with pthreads
     - Networking
3. I assume you are eager to learn this material, and write a bunch of code
4. I assume you can consult documentation, try things out, etc.
What is this class about?

- **Engineering Robust Server Software**
  - **Software**: This class is all about software
    - Hardware may come up in regards to how it affects sw performance
  - **Engineering**: Designing and building systems
    - This is an engineering class, so expect to build a lot of software
    - Focus on useful things in real world
  - **Robust**: Stands up in the face of adversity
    - Badly formed user inputs, many requests at once, evil users…
  - **Server**: handles requests from clients
    - Different constraints from most programs you have written
Server Software

- Servers come in a wide range of "flavors"
- We are going to consider two major ones
  - UNIX daemons: sshd, httpd, …
    - C/C++, systems programming…
  - Web-sites: writing the server side logic for a website
    - Django, databases
- Three major themes
  - Security
  - Resilience
  - Scalability
Five Major Parts To Semester

• [1] Intro (now—~2/7)
  • Requirements/constraints/differences from other software
  •Protocols
  •Unix Daemons
  •DJango/website/AJAX basics
  •Guest lecture: Broad Systems Picture
Five Major Parts To Semester

  - Cryptography basics
  - TLS (https)
  - Common attacks/vulnerability types
    - (e.g., SQL injection, privilege escalation, …)
  - Randomness
  - Side channel attacks
  - Famous vulnerabilities: Heartbleed, Dirty COW, Apple goto
  - Guest Lecture: Tara Gu (Google, Duke ECE MEng Alumn)
Five Major Parts To Semester

  • Error handling, exception models/safety
  • Dealing with non-atomic operations
  • High-availability/disaster recovery (Tyler)
  • Guest Lecture: Melissa Fritcher (IBM, Duke ECE MEng Alumn)
    • Melissa works on the team that make sure ibm.com is always up.
Interlude

- Spring break (No class 3/14 or 3/16)
  - I will be in China, limited email
- Then midterm exam Tuesday 3/21
Five Major Parts To Semester

  • Non-blocking IO
  • C++ atomics, memory model
  • Serialization bottlenecks
    • Locking granularity
    • "hidden" locks
  • Load balancing
  • Load testing
  • IO Scalability (Tyler)
Five Major Parts To Semester

  • Tami Lehman: Intel SGX (Intel / Duke PhD Student)
  • Jim Posen: Secure Payment Systems (Coinbase / Duke ECE alumn)
  • Vlad Petric: Reliable No-SQL Systems (Hedgefund)
  • (TDB)
What Will You Do?

- 4 Homeworks:
  - Pair programming (different partner each homework)
  - Thinking about and write down "dangers"
    - Revisit as semester progresses

1. Caching Http Proxy (Unix Daemon in C)
2. Simple Website (Django)
3. Exploit programs (Attack programs I give you)
4. Load testing (Any language)
"Danger" Log

• Critical programming skill: "spidey sense"
  • As you write, internal mental warning of danger
    • "What if the user …"
    • "What if we run out of memory…"
    • "What if this fails…"
    • "What if…"

• As you code, think of these, write them down
  • Submit a text file with your thoughts
  • Particular focus on class themes (security, resilience, scalability)
"Danger" Log 2.0

- As you learn new things, revisit old assignments
  - Look at code:
    - What should you have worried about?
  - Look at danger logs:
    - What could you have done about these dangers?
- Update log with new thoughts ~weekly.
Pair Programming

- Highly recommended development model: pair programming
  - Not just "doing assignment with a partner"
- Partners work on code at same time
  - One is "driver"
  - The other "navigator"
  - Switch roles frequently/as needed
- Driver: writes code
- Navigator: watches
  - Looks for errors, danger, thinks about bigger picture..
Pair Programming

- Useful tool: screen (or tmux)
  - Multiplex terminal session
  - Can have two terminals connected to one logical terminal
    - Both of you can look at, edit code from your own laptops
  - Facilitates switching driver/navigator
- Either in same room, or on voice chat of some sort
  - Typing too slow
What Will You Do? (cont'd)

- 1 Midterm (Tuesday 3/21)
- 1 Final (Registrar exam schedule)
- 1 Project (Due 4/27)
  - Do in pairs (may select partner from prior hwk)
  - Half class: e-commerce site ("Amazon")
  - Half class: shipping site ("UPS")
  - Systems have to interact
Project: High-level View

"Warehouse" → "Credit Card" → E-commerce site → Shipping site → "Trucks"

web interface to user
web interface to user
Project: High-level View

• I will define these protocols/implement these parts...
  • I'll give you a protocol spec
  • ...but you should be resilient to anything
    • After all, that is a goal of this class
Project: High-level View

- You will do either the red (e-commerce) or the blue (shipping)
  - Protocol between them? Defined by your interoperability group
4 groups (8 people) = 1 interoperability group

- Both e-commerce sites must work with both/either shipping site.
- 8 of you define protocol
Where will you do it?

• You will each have your own server
  • You get root on it, you administer it
• OIT will provide VMs with a restricted network
  • Reduce security risks
  • Accessible only by Duke IPs
• Go to http://vm-manage.oit.duke.edu/
  • Login with netid
Choose New in Upper Right

- Go ahead and do now…
Use ERSS Image to Provision

Choose This One

- Bioinformatics (IPython/Anaconda)
- Debian Jessie
- Django
- Dream Factory
- Drupal
- Dune-substrate
- Engineering Robust Server Software
- GeMS Workshop Jan 2016
- Julia
- LAMP Stack
- MATLAB
- MEAN
- MediaWiki
- NGS Analysis Workshop
- Node.js
- OpenProject
- RHEL 6 Basic
- RHEL 7 Basic
- Ruby Stack
- STA 663
Virtual Machine Management Panel

General Information for ECE 590- Engineering Robust Server Software

Full Name
Bitnami Image
Initial Password
Current Status
Owner:
Requested:
Expired?

Snapshots
Snapshots are made daily at 06:00 ET. Only one snapshot will be kept. You can create a more recent snap, but it will be overwritten the next morning.

System
Base Memory 2 GB
Processors 2
Extra Info patched 7-24-2016
Next Steps

• Login to your server
  • Username bitnami
  • Password (provided on confirmation screen)
• Setup a user account w/ sudo
  • sudo adduser name
  • sudo adduser name sudo
• Now you can ssh in as name
Install Software!

- Your server: fresh image, not much software installed
  - `sudo apt-get install package`

Drew@ubuntu14-generic-template-01:~$ gcc
The program 'gcc' is currently not installed. You can install it by typing:
sudo apt-get install gcc
Drew@ubuntu14-generic-template-01:~$
Packages you probably want to install

- For C development: `gcc g++ make valgrind`
- For editing: `emacs screen`
  - Recommended `.screenrc`: escape `^oo`
- For source control: `git`
- Database: `postgresql-9.5`
- For Django: `python python3-pip`
  - Then do: `sudo pip3 install django psycopg2`
  - Then `django-admin --version` should give 1.10.4
- Libraries: `libssl-dev libxerces-c-dev libpqxx-dev`
- Documentation: `manpages-posix-dev`
Recommended Server Setup [Optional]

- Set up your "dot files"
  - ~/.emacs : emacs configuration
  - ~/.profile : commands read on login
    ```
    export EDITOR='emacs -nw'
    export VISUAL='emacs -nw'
    ```
- Setup ssh key pair(s)
  - Login without password: private key authenticates
- Pick somewhere to backup your work
  - Keep a git remote on another computer
Grading

- Grade Breakdown:
  - Homeworks: 25%
  - Project: 35%
  - Midterm: 20%
  - Final: 30%

- Letter grade:

<table>
<thead>
<tr>
<th>Grade</th>
<th>Score Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-</td>
<td>[90, 93)</td>
</tr>
<tr>
<td>A</td>
<td>[93, 97)</td>
</tr>
<tr>
<td>A+</td>
<td>[97, ∞)</td>
</tr>
<tr>
<td>B-</td>
<td>[80, 83)</td>
</tr>
<tr>
<td>B</td>
<td>[83, 87)</td>
</tr>
<tr>
<td>B+</td>
<td>[87, 90)</td>
</tr>
<tr>
<td>C-</td>
<td>[70, 73)</td>
</tr>
<tr>
<td>C</td>
<td>[73, 77)</td>
</tr>
<tr>
<td>C+</td>
<td>[77, 80)</td>
</tr>
<tr>
<td>F</td>
<td>(-∞, 70)</td>
</tr>
</tbody>
</table>
RFCs

- Many standards are in the form of RFCs
- You SHOULD spend some time reading RFCs this semester
  - ...and may effectively write one during your project
- Start with this one (describes MUST/MAY/SHOULD etc in RFCs)
Next Time..

- Wrap up for this time:
  - Questions?
  - Find partners for homework 1

- Next time:
  - Start talking about server software