CMPSC 447
Software Security

Trent Jaeger
Systems and Internet Infrastructure Security (SIIS) Lab
Computer Science and Engineering Department
Pennsylvania State University

About Me

- *Trent Jaeger* (PhD, University of Michigan)
- Professor in CSE at PSU since 2005 -- after 9 years at IBM Research
- Worked in industry between BS and MS/PhD (that company now part of HP)
- Research: Systems and Software Security
- Example Software Security Work
  - Linux – Several practical contributions to the Linux kernel
  - Program analysis techniques to find/patch flaws and prevent exploitation of flaws
  - Automated exploit generation techniques
- I hope to help give you some useful tools and techniques to improve the security of the software you will be developing

This course….

- Is a programming course that teaches the causes of software vulnerabilities and how to prevent them through safe programming and testing
- Caveat: We are still trying to figure out both
- Topics: What are … Program flaws and how to they become vulnerabilities? … Safe and usable programming techniques to avoid vulnerabilities? … Tools and techniques to detect vulnerabilities? … Efficient security mechanisms and how to add them to your programs?

Background

- Required:
  - CMPSC 443, which requires CMPSC 473 concurrently
  - In theory, you should have passed CMPSC 473 too
  - In practice, you will hate this course if you have had trouble with CMPSC 473
- Expected:
  - Solid Programming Skill in C
- Additional background that would be helpful:
  - Programming Languages (CMPSC 461)
    - We will learn some program analysis techniques to detect vulnerabilities and prevent exploitation

Course Materials

- Canvas
  - For assignments and online quizzes
  - Link to website and other information
- Website
  - http://www.cse.psu.edu/~trj1/cmpsc447-s22/
  - Course schedule is referenced here
    - Check back often -- I may change some of the assignments
- Course Readings
  - Available on the course schedule

Main Facts

- CMPSC 447 held on Tu-Th 3:05-4:20pm
- Professor: Trent Jaeger
- Email: trj1@psu.edu
- Office Hour: Tu 1-2 at W359 Westgate Bldg and W 3-4 virtual
- Location: Leonhard 102
- TA: Kaiming Huang, office hours TBD (virtual to start)
- Piazza: TBD

Course Calendar

- The course calendar has…
- Lecture schedule, assignment due dates, and slides
- Links to online readings
- Please check the calendar frequently
  - It's the real-time state of the course

course calendar

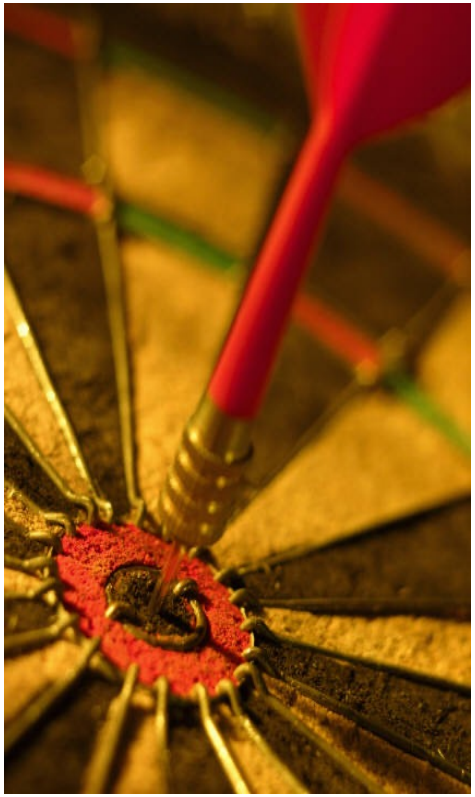
Home Schedule

Below is the calendar for this semester course. This is the preliminary schedule, which will be altered as the semester progresses. It is the *responsibility of the students* to frequently check this web-page for schedule, readings, and assignment changes. As the professor, I will attempt to announce any change to the class, but this web-page should be viewed as authoritative. If you have any questions, please contact me (contact information is available at the course homepage).

Date	Topic	Assignments Due	Readings for Discussion (do readings before class)
01/09/18	Introduction (Slides)		Course syllabus link
01/11/18	Security Basics (Slides)		Reflections on Trusting Trust. K. Thompson, Turing Award Lecture, 1983. link
01/16/18	Passwords (Slides)		Linux Password and Shadow File Formatlink
01/18/18	Programming Mistakes (Info Flow) (Slides)	Base Object Serverlink	SQL Injection Cheat Sheet and Tutoriallink The Risks Digest link
01/23/18	Programming Flaws (Buffer Overflows) (Slides)		Smashing the Stack for Fun and Profit, Aleph One. Phrack 7(49), 1996link Common Vulnerabilities and Exposures link
01/25/18	Programming Flaws (Memory Errors) (Slides)		Hacker's Hut: Exploiting the Heap (11-11.2)link Security Focus: BugTraq link
01/30/18	Programming Flaws (Memory Errors) (Slides)		
02/01/18	Confused Deputy (Slides)		The Confused Deputy (or why capabilities might have been invented). Norm Hardy. Operating Systems Review, pp. 36-38, Oct. 1988. link
02/06/18	Defensive Programming (Slides)		Secure Programming HOWTO (Chapter 6)link
02/08/18	Defensive Programming (Slides)		Secure Programming HOWTO (Chapter 5)link
02/13/18	Penetration Testing (Slides)		Penetration Testing Tutoriallink
02/15/18	Fuzz Testing (Slides)		American Fuzzy Loplink
02/20/18	Static Analysis (Slides)		TBD - Static Analysis Tutoriallink
02/22/18	Static Analysis (Slides)		TBD - Static Analysis Tutoriallink
02/27/18	Symbolic Execution (Slides)		KLEE: Unassisted and Automatic Generation of High-Coverage Tests for Complex Systems. Cristian Cadar, Daniel Dunbar, Dawson Engler, in Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation, 2008. link
03/01/18	**Midterm**		
03/06/18	**Spring Break - No class**		
03/08/18	**Spring Break - No class**		
03/13/18	Security Mechanisms (Slides)		
03/15/18	Privilege Separation (Slides)		PtrSplit: Supporting General Pointers in Automatic Program Partitioning. S. Liu, G. Tan, and T. Jaeger. In 24th ACM Conference on Computer and Communications Security (CCS), 2017.link
03/20/18	Execution Integrity (Slides)		Control-Flow Integrity: Precision, Security, and Performance (Section 2.1)link
03/22/18	Execution Integrity (Slides)		Securing Software by Enforcing Data-Flow Integrity (Section 2.1)link
03/27/18	Comparing Java to C (Slides)		

Grading

- Exams (60%)
  - Midterm (25%)
    - In class
  - Final (35%)
- Projects (30%)
  - Programming Projects
- Participation (10%)
  - Quizzes and Assignments
    - Ungraded and graded
  - Be prepared with readings



Lateness Policy

- Assignments and project milestones are assessed a 20% per-day late penalty, up to a maximum of 4 days. Unless the problem is apocalyptic, don't give me excuses. Students with legitimate reasons who contact the professor before the deadline may apply for an extension.
- You decide what you turn in

Academic Integrity

- See Computer Science and Engineering Department's Policy on Academic Integrity Standards
  - http://www.eecs.psu.edu/students/resources/EECS-CSE-Academic-Integrity.aspx
- The course projects are to be carried out individually. Students are explicitly not allowed to share information, source code, or even discuss the contents of the projects. Any violation of this policy will be considered cheating and will result in the student receiving an 'F' grade for the project and a full letter grade off the final grade for the course.

Ethics Statement

- This course considers topics involving personal and public privacy and security. As part of this investigation, we will cover technologies whose abuse may infringe on the rights of others. As an instructor, I rely on the ethical use of these technologies. Unethical use may include circumvention of existing security or privacy measurements for any purpose, or the dissemination, promotion, or exploitation of vulnerabilities of these services. Exceptions to these guidelines may occur in the process of reporting vulnerabilities through public and authoritative channels. Any activity outside the letter or spirit of these guidelines will be reported to the proper authorities and may result in dismissal from the class.
- When in doubt, please contact the instructor for advice. **Do not** undertake any action which could be perceived as technology misuse anywhere and/or under any circumstances unless you have received explicit permission from Professor Jaeger.

Road Map

- Introduction
  - 1. Today
  - 2. C Review
  - 3. C Debugging
- Software Attacks
  - 1. History
  - 2. System
  - 3. Stack and Heap
- Software Flaws (and how to prevent them)
  - 1. Spatial Errors
  - 2. Type Errors
  - 3. Temporal Errors
- Security Mechanisms (to prevent exploitation of flaws)
  - 1. Current
  - 2. CFI and SFI
  - 3. Privilege Separation
- Finding Program Flaws (to remove more flaws)
  - 1. Dynamic Testing
  - 2. Static Analysis
  - 3. Symbolic Execution
- Special Topics
  - 1. New Hardware Features
  - 2. Safe Programming Techniques and Tools
  - 3. Future of Software Security
  - 4. Some Recent Research

C Language

- This course will focus mainly on the C programming language
  - Learn about
    - Classes of attacks
      - C programs can exhibit most all types of attacks
    - And techniques to prevent their success
- All projects will be in C
  - As will most examples in lecture

Why Do I Care?

- We teach you a bit about programming in C in other classes
  - CMPSC 311 and CMPSC 473
- But, mostly about achieving a functionality goal
- Do we teach you about security in C?

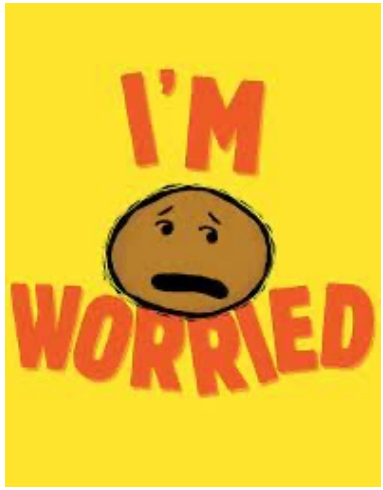
Why Do I Care?

PENNSTATE 1855

- I first programmed professionally in C in 1986
- Been programming operating system code (all in C) since 1996
  - Mostly proof-of-concept implementations at IBM
  - However, I have a little code in the Linux kernel
    - Google "Linux xfrm hooks for selinux" if you want to verify
  - I should be an expert and maybe I am
- But, I still feel uncomfortable with the code that I write

Why Worry?

- What am I worried about?



Why Worry?

- Lots of ways to attack C programs
  - Stay tuned…

Just Switch to Another Language

PENNSTATE 1855

- There are other languages, but C is …
  - Popular: Still among the top-3 languages in preference in programmer surveys
  - Useful: Easy to write high performance code
  - Lots of code: Legacy code abounds
  - Present: C code is often part of system runtimes and libraries (even for other languages)
- A good chance you will face the security issues we will discuss in this course in some C code
  - And in other languages

Will Be A Practical Course

- In the 1990s, security researchers conjectured that people would learn to program in C securely once they learned about the security issues
- Unfortunately, that has not been the case
  - Why not?

Will Be a Practical Course

PENNSTATE 1855

- Why not?
- Think about the following…
  - On one hand: it is really easy to write a C program that will compile
  - On the other hand: it often requires a significant effort to get that C program to do what you (or the professors/bosses) want
- C programs exhibit a lot of "extra" unintended functionality – adversaries can exploit this

Will Be a Practical Course

PENNSTATE 1855

- So, we are going to have to get our hands dirty
  - Write some C programs
  - Learn the causes of flaws
  - Learn how flaws can be exploited (fun, but can be hard)
  - Learn how to program safely (warning: takes more work)
  - Learn defenses that prevent exploitation (sometimes)
  - Learn how programs are tested for flaws
- We will study unintended function in C programs
  - And how to make safe intended functionality

Challenge

- Isn't studying unintended function hard?
  - It's hard enough just to understand the intended function
- Is harder in general, but is also critically important to start thinking about these issues
  - Communication is key
  - But, sometimes we don't know what we don't know
  - I will give quizzes to gauge understanding
- Some lectures will be more like "recitations" related to project tasks

Take Away

- In this class, we will focus on the methods to improve the security of C programs you write
  - Reduce flaws that lead to vulnerabilities
  - Improve defenses to prevent exploitation of flaws
- We will look at several techniques
  - How to write safe code?
  - How to detect flaws in code?
  - How to apply available security mechanisms to code?
  - What advances may enable future improvements?