Topics in Language-based Software Security

Introduction

Mathias Payer

EPFL EDIC, CS-725, Fall 2018

About me

- Instructor: Mathias Payer
- Research area: system/software security
  - Memory/type safety
  - Mitigating control-flow hijacking
  - Compiler-based defenses
  - Binary analysis and reverse engineering
- CTF player, founding member of gnoobz, founder of b01lers.
- Homepage: `http://nebelwelt.net`

Why should you care?

- Security impacts everybody's day-to-day life
- Security impacts your day-to-day life
- User: make safe decisions
- Developer: design and build secure systems
- Researcher: identify flaws, propose mitigations

Software Engineering versus Security

Software engineering aims for

- **Dependability:** producing fault-free software
- **Productivity:** deliver on time, within budget
- **Usability:** satisfy a client's needs
- **Maintainability:** extensible when needs change

Software engineering combines aspects of PL, networking, project management, economics, etc.
Security is secondary and often limited to testing.

Definition: *Security*

Security is the application and enforcement of policies through mechanisms over data and resources.

- Policies specify what we want to enforce
- Mechanisms specify how we enforce the policy (i.e., an implementation/instance of a policy).

Definition: *Software Security*

Software Security is the area of Computer Science that focuses on (i) testing, (ii) evaluating, (iii) improving, (iv) enforcing, and (v) proving the security of software.

Why is software security difficult?

- Human factor
- Concept of weakest link
- Performance
- Usability

Definition: *Software Bug*

> *A software bug is an error, flaw, failure, or fault in a computer program or system that causes it to produce an incorrect or unexpected result, or to behave in unintended ways. Bugs arise from mistakes made by people in either a program's source code or its design, in frameworks and operating systems, and by compilers.*

Source: Wikipedia

Definition: *Software Vulnerability*

A vulnerability is a software weakness that allows an attacker to exploit a software bug. A vulnerability requires three key components (i) system is susceptible to flaw, (ii) adversary has access to the flaw (e.g., through information flow), and (iii) adversary has capability to exploit the flaw.

Course goals

Software running on current systems is exploited by attackers despite many deployed defence mechanisms and best practices for developing new software.
Goal: understand state-of-the-art software attacks/defenses across all layers of abstraction: from programming languages, compilers, runtime systems to the CPU, ISA, and operating system.
To achieve these goals, we will read and discuss foundational and current papers in systems and security top tier conferences.

Course outline

- Understanding software flaws
- Language safety and formal verification
- Software testing (fuzzing and sanitization)
- Mitigation

Course material

- Software security is rapidly evolving
- Software Security: Principles, Policies, and Protection
- Research papers
- Labs and exercises

Grading

- Paper presentations: 60%
- Course participation: 10%
- Course projects (3x): 30%
- The grade will be curved.

Summary

- Software Security is the area of Computer Science that focuses on (i) testing, (ii) evaluating, (iii) improving, (iv) enforcing, and (v) proving the security of software.
- Learn to identify common security threats, risks, and attack vectors for software systems.
- Assess current security best practices and defense mechanisms for current software systems.
- Design and evaluate secure software.
- Have fun!