CS527 Software Security

Introduction

Mathias Payer

Purdue University, Spring 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
- Founded b01lers CTF team in 2014.
- Homepage: `http://nebelwelt.net`

Support

- TA: Bader AlBassam
- Research area: software security
- CTF player
- B01lers team leader

Course outline

- Secure software lifecycle
- Security policies
- Attack vectors
- Defense strategies
- Case studies: browser/web/mobile security

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

Best practices?

- Always lock your screen (on mobile/desktop)
- Unique password for each service
- Two-factor authentication
- Encrypt your transport layer (TLS)
- Encrypt your messages (GPG)
- Encrypt your filesystem (DM-Crypt)
- Disable password login on SSH
- Open (unkown) executables/documents in an isolated environment

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.

Learning outcomes

- Understand causes of common weaknesses.
- Identify security threats, risks, and attack vector.
- Reason how such problems can be avoided.
- Evaluate and assess current security best practices and defense mechanisms for current systems.
- Become aware of limitations of existing defense mechanisms and how to avoid them.
- Identify security problems in source code and binaries, assess the associated risks, and reason about severity and exploitability.
- Assess the security of given source code.

Syllabus: Basics

- **Secure software lifecycle:** Design; Implementation; Testing; Updates and patching
- **Basic security principles:** Threat model; Confidentiality, Integrity, Availability; Least privileges; Privilege separation; Privileged execution; Process abstraction; Containers; Capabilities
- **Reverse engineering:** From source to binary; Process memory layout; Assembly programming; Binary format (ELF)

Syllabus: Policies and Attacks

- **Security policies:** Compartmentalization; Isolation; Memory safety; Type safety
- **Bug, a violation of a security policy:** Arbitrary read; Arbitrary write; Buffer overflow; Format string bug; TOCTTOU
- **Attack vectors:** Confused deputy; Control-flow hijacking; Code injection; Code reuse; Information leakage;

Syllabus: Defenses

- **Mitigations:** Address Space Layout Randomization; Data Execution Prevention; Stack canaries; Shadow stacks; Control-Flow Integrity; Sandboxing; Software-based fault isolation
- **Testing:** Test-driven development; Beta testing; Unit tests; Static analysis; Fuzz testing; Symbolic execution; Formal verification
- **Sanitizer:** Address Sanitizer; Valgrind memory checker; Undefined Behavior Sanitizer; Type Sanitization (HexType)

Syllabus: Case studies

- **Browser security:** Browser security model; Adversarial computation; Protecting JIT code; Browser testing
- **Web security:** Web frameworks; Command injection; Cross-site scripting; SQL injection
- **Mobile security:** Android market; Permission model; Update mechanism

Course material

- Software security is rapidly evolving
- There are no *standard* text books
    - Research papers
    - Articles and tutorials
    - Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau. Operating Systems: Three Easy Pieces
    - Trent Jaeger, Operating System Security
- Labs and exercises

Capture-The-Flag!

- Security awareness is an acquired skill. This class heavily involves programming and security exercises.
- A semester long Capture-The-Flag (CTF) to train security skills:
    - Binary analysis
    - Reverse engineering
    - Exploitation techniques
    - Web challenges

Course project (1/2)

- Design and implementation of a project in C
- Security evaluation of your peers' applications
- Fixing any reported security vulnerabilities
- Teams of up to 3 people allowed

Course project (2/2)

- Use a source repository to check in solutions,
- Organize your project according to a design document,
- Peer review and comment the code of other students,
- Work with a large code base, develop extensions.

Grading

- Lab assignments (CTF): 25%
- Programming project: 25%
- Midterm: 20%
- Final: 30%
- The grade will be curved.

Academic Integrity

All work that you submit in this course must be your own.
Unauthorized group efforts are considered *academic dishonesty*.
You are allowed to discuss the problem with your peers but you may not copy or reuse any part of an existing solution.
We will use automatic tools to compare your solution to those of other current and past students. The risk of getting caught is too high!

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!