CS-527 Software Security

Introduction

Asst. Prof. Mathias Payer

Department of Computer Science
Purdue University

TA: Kyriakos Ispoglou

https://nebelwelt.net/teaching/17-527-SoftSec/

Spring 2017

Table of Contents

Mathias Payer

- MSc. ETH in 2006, Dr. sc. ETH in 2012, focusing on runtime mitigations for binaries: "*Safe Loading and Efficient Runtime Confinement: A Foundation for Secure Execution*".
- Post doc in Dawn Song's BitBlaze group at UC Berkeley, focusing on memory safety errors and compiler-based mitigations.
- Faculty at Purdue since fall 2014.
- Founded b01lers CTF team in 2014.
- Homepage: `http://nebelwelt.net`

HexHive: Research Focus

- Perfect security is unachievable, software will always have bugs.
- Goal: Protect software in the presence of vulnerabilities, ensure integrity and confidentiality of the system at all times.
- Several active research projects in compiler-based and binary rewriting-based memory safety, focus on strong defenses.
- Adversarial research exploiting limitations of software.
- Contact me (with ideas) for graduate research projects.
- Group homepage: `http://hexhive.github.io`

Kyriakos Ispoglou (Ispo)

- EMail: *kispoglo@purdue.edu*
- 3rd year PhD student
- Long-time CTF player and hacker
- Will supervise and organize the labs

Table of Contents

Why should you care?

There are multiple levels of caring:

- Security impacts your day-to-day life.
- Security impacts everybody's day-to-day life.
- Security-aware user: make safe decisions.
- Security-aware developer: design and build secure systems.
- Security researcher: identify security flaws, propose mitigations.

Security

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).

Software Security

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.

Software Security

Goals

Software running on current systems is exploited by attackers despite many deployed defence mechanisms and best practices for developing new software. In this course you will learn about security threats, attack vectors, and defence mechanisms on current systems. You will work with real world problems and technical challenges of security mechanisms (both in the design and implementation of programming languages, compilers, and runtime systems).

Learning outcomes

- Understand causes of common weaknesses in software security.
- Identify security threats, risks, and attack vectors for software.
- Reason how such problems can be avoided in software.
- Evaluate and assess current security best practices and defense mechanisms for current software 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 or applications.

Syllabus

1. Introduction to software security
2. Software vulnerabilities: memory (un-)safety
3. Introduction to reverse engineering
4. Dynamic defense mechanisms
5. Static protection through bug finding
6. Finding and exploiting vulnerabilities
7. Operating system security and forensics
8. Protecting data
9. Defense in practice
10. Web security
11. Browser security
12. Android/mobile security
13. Malware analysis

Table of Contents

Software Engineering versus Security

Software engineering is a discipline whose aims are:

- **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 computer science (PL, networking, OS, databases, and many more), project management, economics, and many more.

Security is of secondary concern and often limited to testing.

Definitions (1)

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. Most bugs arise from mistakes and errors made by people in either a program's source code or its design, or in frameworks and operating systems used by such programs, and a few are caused by compilers producing incorrect code.[a]

[a] According to Wikipedia.

Definitions (2)

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.

Security Fails

- iCloud: leaked pictures
- HeartBleed: online accounts, passwords, keys
- Malware: $105B/year industry
- Stuxnet: governmentally sponsored attack against Iran's nuclear program

iCloud: The Fappening

- More than 500 private pictures leaked on 4chan on Aug-31, 14
- Obviously huge privacy invasion but what are the technical aspects of the attack?
- Initial assumption: hacker gained access to Apple's servers
- In reality: brute-forcing of targeted passwords
- Apple ID has many authentication methods and huge API
- Access to FindMyiPhone API did not enforce limits on number of authentication attempts
- This API was then likely used to brute-force passwords to well-known logins

Table of Contents

The Lab and projects

Software security is an acquired skill. We will expose you to a lot of practical security tasks:

- A semester long capture-the-flag (CTF) game. In this Jeopardy-style CTF we will release new challenges (riddles/tasks/questions) every week following the class topics. You will use your reverse engineering and hacking skills to solve these challenges. The earlier you solve the challenge, the more points you get. To discourage from "*sharing*" solutions, the amount of points is reduced with each additional person solving the challenge.
- Design and implementation of a small application in C.
- Security evaluation of your peers' applications.
- Fixing any reported security vulnerabilities.

Grading policy, projects, exams, and homework

- Lab assignments (30% of grade)
- Programming projects (20% of grade)
- Midterm (15% of grade)
- Final (35% of grade)

Submitting homework and projects

Class teaches formal aspects of software security, projects and homework allow practical experience:

- 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.

Recommended Books

- There are no comprehensive books on software security.
- Software security is defined through best practices, learning experience, and getting pwned.
- Software Security requires deep understanding of systems to reason across layers.
- Trent Jaeger, Operating System Security `http://www.morganclaypool.com/doi/abs/10.2200/S00126ED1V01Y200808SPT001`.
- Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau. Operating Systems: Three Easy Pieces. `http://pages.cs.wisc.edu/~remzi/OSTEP/`
- Randal E. Bryant and David R. O'Hallaron, Computer Systems: A Programmer's Perspective `http://csapp.cs.cmu.edu/`.

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 or work in a team.

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!*

Course Organization

- Homepage:
  `http://www.nebelwelt.net/teaching/17-527-SoftSec`
- Lecture: Monday and Wednesday 3:30p to 4:20p in LWSN B134.
- Office hours: Monday 2:30p to 3:30p in LWSN 3154M.
- Homework/projects are generally due at 11:59pm.

Table of Contents

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.
- Goal of the class: learn to identify common security threats, risks, and attack vectors for software systems.
- Evaluate and assess current security best practices and defense mechanisms for current software systems.
- Design and evaluate secure software.
- Enjoy the CTF and project.

Questions?

?