CS 6956-001: Software and System Security

Administrative Details and Syllabus

Spring 2023

Overview

Class Website	Canvas (available through CIS)
Lectures	Tuesdays and Thursdays 3:40–5:00PM at GC 2660
Instructor	Mu Zhang
TA	Levi Li and Raihan Ahmed
Final course grade	Lab assignments 30%, Paper presentation 15%, Paper review 15%, Course project 30%, Class participation 10%

Course Information

Description. This is a graduate-level research-oriented course, which covers both classic and cutting-edge topics in the area of software security. It provides the students with a good understanding of software problems in not only traditional binary executables but also emerging mobile applications, cyber-physical systems and Blockchain platforms.

Learning Outcomes.

- Evaluate the concept, categorization, and impact of software security problems and countermeasures
- Integrate appropriate defense mechanisms into practical systems
- Validate security choices based on impact, cost, and efficiency
- Analyze software bugs and malware activities with manual reverse engineering and automated analysis tools
- Implement custom analysis tools to enable dedicated software vulnerability analysis

Course Materials

Website. The class will use a Canvas website available through CIS. Here, the schedule, course notes, assignments and other course information will be provided, *updated weekly*. Materials for a given week will be posted by the end of the previous week so that students have plenty of time to prepare in advance of lectures.

Course notes. The instructor often makes use of slides, sample code and other materials during lecture. These items are posted on the class website following the lecture; however, such posted items may not represent completely the material covered in class. Students who must miss class are strongly encouraged to check with a classmate.

Student Evaluation

Lab Assignments. A major aspect of the learning experience for this material is attained by hands-on programming to interact directly with systems security problems and tools. Particularly, we will be working on three lab assignments:

- Lab 1: Buffer Overflow Attacks. Implement simple exploits to trigger vulnerabilities in binary code.
- Lab 2: Static Program Analysis. Develop a simple Soot transform to reveal API usage in Android apps.
- Lab 3: Symbolic Execution. Write simple angr scripts to find vulnerabilities.

Programming assignment deadlines are strict, due via Canvas submission by 11:59PM on the posted due date. Late programming assignments are accepted according to the following rules.

- Assignments are not accepted more than 3 days after the due date.
- Assignments submitted any time X days after the due date (midnight to 11:59PM) are penalized $X \times 10\%$ of the assignment grade.

It is the student's responsibility to ensure the successful and timely submission of each assignment – start early and follow the instructions carefully. Corrupted or missing files are not grounds for extensions – double-check your submissions and save a digital copy of all of your work.

Paper Review and Presentation. Each student is required to present one paper in the class for about 15 minutes and lead the discussion. Each student is required to write reviews of at least 300 words for all the papers presented by students, before the papers are presented in class. A reading list will be provided on Canvas.

Course Project. Each student is required to participate in a course project. A list of suggested projects will be provided. Students may also propose their own projects. Projects can be done individually or by groups. Each group should not exceed 2 students. In the project report, each member's contribution should be clearly stated.

Final course Grade. The final course grade will be based on three evenly-weighted lab-work assignments (30% total), four evenly-weighted paper reviews (15% total), a paper presentation (15%), a course project (30%) and course participation (10%).

Regrades. Students who wish to appeal a score on an assignment or an exam must do so within one week of receiving the score.

Getting Help

Instructor office hours. 4:00 – 5:00PM Mondays and Wednesdays, unless otherwise indicated.

Teaching assistants and consulting hours. See Canvas for the consulting schedule of the course TAs.

Communication. For questions outside of class and consulting hours, students are encouraged to post a question to Canvas Discussion, or contact the course staff directly via email. Whenever the question is a clarification on the assignment and not giving away the answers, feel free to post to the entire class. When in doubt, only send the question to the TAs and Instructor.

Course Guidelines

Students are bound by policies and guidelines from the university, College of Engineering, School of Computing, and specifically for this course, described in the documents listed below.

- CS6491 Academic Misconduct Policy, posted on the Canvas website under `Admin/policy.pdf`.
- School of Computing Academic Misconduct Policy `https://handbook.cs.utah.edu/2021-2022/Academics/misconduct.php`.
- College of Engineering Guidelines `www.coe.utah.edu/students/academic-affairs/academics/semester-guidelines`
- UofU Student code `www.regulations.utah.edu/academics/guides/students/studentRights.html`

Students should read and understand each of these documents, asking questions as needed.

Inclusivity Statement

It is our intent that students from all diverse backgrounds and perspectives be well-served by this course, that students' learning needs be addressed both in and out of class, and that the diversity that the students bring to this class be viewed as a resource, strength and benefit. It is our intent to present materials and activities that are respectful of diversity: gender identity, sexuality, disability, age, socioeconomic status, ethnicity, race, nationality, religion, and culture. Your suggestions on how we can make the course more inclusive and welcoming are encouraged and appreciated. We also expect students to treat others in the class, including the teaching staff, with the same level of respect. We take incidents of discrimination, bias, and harassment seriously. We will file reports with the Office or Equal Opportunity, Affirmative Action, and Title IX (OEO) about such incidents. If you are unsure what differentiates free speech and professional behavior from discrimination, bias, and harassment we are happy to have an open, judgment-free, and confidential conversation with you, or refer you to the OEO.

The Americans with Disabilities Act

The University of Utah seeks to provide equal access to its programs, services, and activities for people with disabilities. If you will need accommodations in this class, reasonable prior notice needs to be given to the Center for Disability Services, 162 Olpin Union Building, (801) 581-5020. CDS will work with you and the instructor to make arrangements for accommodations. All written information in this course can be made available in an alternative format with prior notification to the Center for Disability Services.

University Safety Statement

The University of Utah values the safety of all campus community members. To report suspicious activity or to request a courtesy escort, call campus police at 801-585-COPS (801-585-2677). You will receive important emergency alerts and safety messages regarding campus safety via text message. For more information regarding safety and to view available training resources, including helpful videos, visit safeu.utah.edu.

Addressing Sexual Misconduct

Title IX makes it clear that violence and harassment based on sex and gender (which Includes sexual orientation and gender identity/expression) is a civil rights offense subject to the same kinds of accountability and the same kinds of support applied to offenses against other protected categories such as race, national origin, color, religion, age, status as a person with a disability, veteran's status or genetic information. If you or someone you know has been harassed or assaulted, you are encouraged to report it to the Title IX Coordinator in the Office of Equal Opportunity and Affirmative Action, 135 Park Building, 801-581-8365, or the Office of the Dean of Students, 270 Union Building, 801-581-7066. For support and confidential consultation, contact the Center for Student Wellness, 426 SSB, 801-581-7776. To report to the police, contact the Department of Public Safety, 801-585-2677(COPS).

Syllabus

The following are the key topics planned for study, and the corresponding reading materials.

Introduction

Vulnerabilities

Smashing the Stack for Fun and Profit. Aleph One. Phrack 49(14), Nov. 1996.
The Security Mindset, Bruce Schneier. 2008.

Malware

Semantics-Aware Malware Detection, Oakland 2005
Dissecting Android Malware: Characterization and Evolution, Oakland 2012

Practical Defense

StackGuard: Automatic Adaptive Detection and Prevention of Buffer-Overflow Attacks, USENIX Sec 98
Address Space Layour Randomization (ASLR)
Data Execution Prevention (DEP)
SoK: Eternal War in Memory, Oakland 2013

Advanced Defense

Control-Flow Integrity Principles, Implementations, and Applications, TISSEC 2009
Native Client: A Sandbox for Portable, Untrusted x86 Native Code, Oakland 2009
Code-Pointer Integrity, OSDI 2014
Language-Based Information-Flow Security, JSAC 2003
MoCFI: A Framework to Mitigate Control-Flow Attacks on Smartphones, NDSS 2012

Static Analysis

Finding Security Vulnerabilities in Java Applications with Static Analysis, USENIX Security 2005
FlowDroid: Precise Context, Flow, Field, Object-sensitive and Lifecycle-aware Taint Analysis for Android Apps, PLDI 2014
Soot - A framework for analyzing and transforming Java and Android applications

Dynamic Analysis

Dynamic Taint Analysis for Automatic Detection, Analysis, and Signature Generation of Exploits on Commodity Software, NDSS 2005
DroidScope: Seamlessly Reconstructing the OS and Dalvik Semantic Views for Dynamic Android Malware Analysis, USENIX Security 2012

Symbolic Execution

KLEE: Unassisted and Automatic Generation of High-Coverage Tests for Complex Systems Programs, OSDI 2008
AppIntent: Analyzing Sensitive Data Transmission in Android for Privacy Leakage Detection, CCS 2013
angr: a python framework for analyzing binaries

Fuzzing

DART: Directed Automated Random Testing, PLDI 2005
AFL: american fuzzy lop
IOTFUZZER: Discovering Memory Corruptions in IoT Through App-based Fuzzing, NDSS 2018

Automated Patching

Vulnerability-Specific Execution Filtering for Exploit Prevention on Commodity Software, NDSS 2006

AppSealer: Automatic Generation of Vulnerability-Specific Patches for Preventing Component Hijacking Attacks in Android Applications, NDSS 2014
Automatic Patch Generation by Learning Correct Code, POPL 2016

Code Search

CP-Miner: A Tool for Finding Copy-paste and Related Bugs in Operating System Code, OSDI 2004
Tracelet-Based Code Search in Executables, PLDI 2014
BinDiff: zynamics BinDiff uses a unique graph-theoretical approach to compare executables
Neural network-based graph embedding for cross-platform binary code similarity detection, CCS 2017

Android Security

Android Permissions Demystified, CCS 2011
A Study of Android Application Security, USENIX Security 2011
TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones, OSDI 2010
Semantics-Aware Android Malware Classification Using Weighted Contextual API Dependency Graphs, CCS 2014
Things You May Not Know About Android (Un)Packers: A Systematic Study based on Whole-System Emulation, NDSS 2018

IoT Security

SmartAuth: User-Centered Authorization for the Internet of Things, USENIX Security 2017
Sensitive Information Tracking in Commodity IoT, USENIX Security 2018
IOTGUARD: Dynamic Enforcement of Security and Safety Policy in Commodity IoT, NDSS 2019
SoK: Security Evaluation of Home-Based IoT Deployments, Oakland 2019
Security Analysis of Emerging Smart Home Applications, Oakland 2016
ContexIoT: Towards Providing Contextual Integrity to Appified IoT Platforms, NDSS 2017
FlowFence: Practical Data Protection for Emerging IoT Application Frameworks, USENIX Security 2016

Industrial Control Systems Security

SABOT: Specification-based Payload Generation for Programmable Logic Controllers, CCS 2012
A Trusted Safety Verifier for Process Controller Code, NDSS 2014
Hey, My Malware Knows Physics! Attacking PLCs with Physical Model Aware Rootkit, NDSS 2017
Towards Automated Safety Vetting of PLC Code in Real-World Plants, Oakland 2019
ICSREF: A Framework for Automated Reverse Engineering of Industrial Control Systems Binaries, NDSS 2019

Smart Contract Security

Making Smart Contracts Smarter, CCS 2016
ZEUS: Analyzing Safety of Smart Contracts, NDSS 2018
Securify: Practical Security Analysis of Smart Contracts, CCS 2018
Sereum: Protecting Existing Smart Contracts Against Re-Entrancy Attacks, NDSS 2019
Towards Automated Safety Vetting of Smart Contracts in Decentralized Applications, CCS 2022

Security Audit

Backtracking Intrusions, SOSP 2003
Enriching Intrusion Alerts Through Multi-Host Causality, NDSS 2005

ProTracer: Towards Practical Provenance Tracing by Alternating Between Logging and Tainting, NDSS 2016

Towards a Timely Causality Analysis for Enterprise Security, NDSS 2018

HOLMES: Real-time APT Detection through Correlation of Suspicious Information Flows, Oakland 2019

Firmware Analysis

FIE on Firmware: Finding Vulnerabilities in Embedded Systems using Symbolic Execution, USENIX Security 2013

AVATAR: A Framework for Dynamic Security Analysis of Embedded Systems' Firmwares, NDSS 2014

A Large Scale Analysis of the Security of Embedded Firmwares, USENIX Security 2014

Firmalice - Automatic Detection of Authentication Bypass Vulnerabilities in Binary Firmware,NDSS 2015

Towards Automated Dynamic Analysis for Linux-based Embedded Firmware, NDSS 2016

FIRM-AFL: High-Throughput Greybox Fuzzing of IoT Firmware via Augmented Process Emulation, USENIX Security 2019