CS 676 Advanced Topics in Systems Security Syllabus

Department of Computer Science
Spring 2019

Instructor: Georgios Portokalidis, Office Hours: by appointment (NB307A)
Canvas: https://sit.instructure.com/courses/29677

COURSE DESCRIPTION

This course covers a wide range of advanced topics in the area of Systems Security. A computer system is composed by software, hardware, policies, and practices. Systems security involves both designing and building secure systems, as well as improving and evaluating the security of existing systems. During this course, students will study and present in the classroom recent papers in the area of systems security, write a literature survey on a particular topic, and work on a semester-long project, which will involve designing, implementing, and evaluating a system. Those who take the class should be skilled programmers and should already have some knowledge in the area of systems security.

LEARNING OBJECTIVES

After the completion of this course students will (a) be able to study scientific articles, (b) be able to give presentations on recent topics in systems security, (c) be familiar with directions in the area, and (d) have demonstrated their ability to design and build a secure system.

Describing how recent attacks against systems work and defining their impact	[BS-CyS B analyze] [BS-CyS G impact] [BS-CyS I currency] (MS-CyS B security threats) (MS-CyS B security practice)
Explaining how recent prevention proposals improve the security and privacy of systems	[BS-CyS B analyze] [BS-CyS I currency] (MS-CyS B security practice)
Designing and implementing a system that improves security and/or privacy	[BS-CyS C design] [BS-CyS K construction] [BS-CyS J tradeoffs] [BS-CyS F communicate] (MS-CyS B security practice) (MS-CyS D privacy)
Organizing and delivering presentations on modern threats and defenses	[BS-CyS F communicate]

FORMAT AND STRUCTURE

- Presentations given by students based on a research paper selected from the list of papers below. The paper/topics have already been tied to particular dates. Students can select papers on a first-come, first-serve basis.
- Q&A sessions, following each presentation, between the student presenting, other students, and the instructor.
- Before each lecture, students will write and submit a short (1-2 paragraph) constructive critique of that week's papers.
- A semester-long project, which must involve significant programming effort, aiming to replicate the results of a paper or conduct original research. Students need to select an appropriate topic, which needs to be approved by the instructor, conduct a literature survey on it, and present the proposed project in class. At the end, students need to deliver the code they developed, a technical report, and a presentation in class.

COURSE MATERIALS

Mandatory readings include the papers in the schedule and the following:

- SoK:(State of) The Art of War: Offensive Techniques in Binary Analysis
- SoK: Automated Software Diversity
- SoK: Eternal War in Memory
- How to Read a Paper

Suggested reading includes the papers below:

Isolation
How to Run POSIX Apps in a Minimal Picoprocess
Practical and Effective Sandboxing for Non-root Users
Information flow
HDFI: Hardware-Assisted Data-flow Isolation
Enabling Refinable Cross-Host Attack Investigation with Efficient Data Flow Tagging and Tracking
Control-flow Integrity
Control-Flow Integrity - Principles, Implementations, and Applications
Practical Context-Sensitive CFI
Per-Input Control-Flow Integrity
CCFI: Cryptographically Enforced Control Flow Integrity
Enforcing Forward-Edge Control-Flow Integrity in GCC & LLVM
A Tough call: Mitigating Advanced Code-Reuse Attacks at the Binary Level
Memory & type safety
HeapHopper: Bringing Bounded Model Checking to Heap Implementation Security
Stack Bounds Protection with Low Fat Pointers
Delta Pointers: Buffer Overflow Checks Without the Checks

Attacks
Q: Exploit Hardening Made Easy
Just-In-Time Code Reuse: On the Effectiveness of Fine-Grained Address Space Layout Randomization
Out Of Control: Overcoming Control-Flow Integrity
Stitching the Gadgets: On the Ineffectiveness of Coarse-Grained Control-Flow Integrity Protection
Framing Signals - A Return to Portable Shellcode
Position-independent Code Reuse: On the Effectiveness of ASLR in the Absence of Information Disclosure
Dedup Est Machina: Memory Deduplication as an Advanced Exploitation Vector
Undermining Information Hiding (and What to Do about It)
Missing the Point(er): On the Effectiveness of Code Pointer Integrity
Moving target defenses
Countering code-injection attacks with instruction-set randomization
On the effectiveness of address-space randomizatione effectiveness of address-space randomization
kR^X: Comprehensive Kernel Protection against Just-In-Time Code Reuse
On the Effectiveness of Address-Space Randomization
Architectural-level attacks
Inferring Fine-grained Control Flow Inside SGX Enclaves with Branch Shadowing
Drammer: Deterministic Rowhammer Attacks on Mobile Platforms
Other defenses
CAn't Touch This: Software-only Mitigation against Rowhammer Attacks targeting Kernel Memory

COURSE REQUIREMENTS

Project A research project related to the material covered in the course that involves significant programming effort completed by small groups of students (1-3), including:

- a 5-6 pages project proposal that includes a literature survey
- a 5-6 pages technical report on the software project
- software
- proposal presentation
- final project presentation

Although the project may rely on replicating existing papers, in that case a more thorough analysis of the original work's strengths and weaknesses must be undertaken. Investigating new ideas is encouraged.

Talks 2-4 40-min presentations given by each student on one of the courses' papers, evaluated on the following:

- Understanding: Does the presenter understand the material?

- Thoughtfulness: Does the presented have insights and opinions beyond what is in the paper?
- Clarity: Can the audience understand the presentation? Are there useful examples?
- Materials: Do the slides (or use of whiteboard) illustrate and support the talk? Are there diagrams to help convey the technicalities?
- Delivery: Has the presenter practiced?
- Non-regurgitation: Did the presenter do something beyond simply typing sections of the paper as bullet points? Did the presenter motivate the ideas in their own words?
- Answering questions: Can the presenter handle questions from the audience?

Reviews 1-2 paragraph paper reviews for all presented papers. The reviews should:

- discuss the pros and cons of the proposed idea, protection mechanism, or bypass technique.
- conclude with at least two thought-provoking questions regarding the material covered, along with a brief direction of future work.

Attendance Participation is mandatory

GRADING PROCEDURES

Grades will be based on:

Project software and report	45%
Project presentation	5%
Project proposal and literature survey	15%
Proposal presentation	5%
Presentations	20%
Reviews	5%
Attendance	5%

You will not need a 97% to get an A in this course. Generally, A corresponds to excellent performance, B to good, C to fair, and F to failure to understand the basics.

COMMUNICATING

Email.

ACADEMIC INTEGRITY

Graduate Student Code of Academic Integrity

All Stevens graduate students promise to be fully truthful and avoid dishonesty, fraud, misrepresentation, and deceit of any type in relation to their academic work. A student's submission of work for academic credit indicates that the work is the student's own. All outside assistance must be acknowledged. Any student who violates this code or who knowingly assists another student in violating this code shall be subject to discipline.

All graduate students are bound to the Graduate Student Code of Academic Integrity by enrollment in graduate coursework at Stevens. It is the responsibility of each graduate student to understand and adhere to the Graduate Student Code of Academic Integrity. More information including types of violations, the process for handling perceived violations, and types of sanctions can be found at www.stevens.edu/provost/graduate-academics.

TENTATIVE COURSE SCHEDULE

Week #	Date	Topic
1	Monday, January 28, 2019	Introduction
2	Monday, February 4, 2019	Dynamic Taint Analysis for Automatic Detection, Analysis, and Signature Generation of Exploits on Commodity Software
		Practical Control Flow Integrity & Randomization for Binary Executables
		ASLR-Guard: Stopping Address Space Leakage for Code Reuse Attacks
3	Monday, February 11, 2019	CCured: Type-Safe Retrofitting of Legacy Code
		Fast Byte-Granularity Software Fault Isolation
		Code-Pointer Integrity
4	Tuesday, February 19, 2019	ShadowReplica: Efficient Parallelization of Dynamic Data Flow Tracking
		Preventing memory error exploits with WIT
		Readactor: Practical Code Randomization Resilient to Memory Disclosure
5	Monday, February 25, 2019	Architecture-Independent Dynamic Information Flow Tracking
		GRIFFIN: Guarding Control Flows Using Intel Processor Trace
		SoftBound: Highly Compatible and Complete Spatial Memory Safety for C
6	Monday, March 4, 2019	Shuffler: Fast and Deployable Continuous Code Re-Randomization
		Untrusted Hosts and Confidentiality: Secure Program Partitioning
		Adapting Software Fault Isolation to Contemporary CPU Architectures
7	Monday, March 11, 2019	Proposal presentations
	Monday, March 18, 2019	Spring break

8	Monday, March 25, 2019	Low-Fat Pointers: Compact Encoding and Efficient Gate-Level Implementation of Fat Pointers for Spatial Safety and Capability-based Security
		Compiler-assisted Code Randomization
		Mcfi-modular control-flow integrity
9	Monday, April 1, 2019	Control-Flow Bending: On the Effectiveness of Control-Flow Integrity
		PtrSplit: Supporting General Pointers in Automatic Program Partitioning
		Cling: A Memory Allocator to Mitigate Dangling Pointers
10	Monday, April 8, 2019	Last-Level Cache Side-Channel Attacks are Practical
		No Need to Hide: Protecting Safe Regions on Commodity Hardware
		A Software-Hardware Architecture for Self-Protecting Data
11	Monday, April 15, 2019	Flip Feng Shui: Hammering a Needle in the Software Stack
		Counterfeit Object-oriented Programming: On the Difficulty of Preventing Code Reuse Attacks in C++ Applications
		Fides: Selectively Hardening Software Application Components against Kernel-level or Process-level Malware
12	Monday, April 22, 2019	Translation Leak-aside Buffer: Defeating Cache Side-channel Protections with TLB Attacks
		Heisenbyte: Thwarting Memory Disclosure Attacks using Destructive Code Reads
		TaintDroid: An Information-Flow Tracking System for Realtime Privacy
13	Monday, April 29, 2019	Meltdown: Reading Kernel Memory from User Space
		The Dynamics of Innocent Flesh on the Bone: Code Reuse Ten Years Later
		Data-Oriented Programming: On the Expressiveness of Non-Control Data Attacks
14	Monday, May 6, 2019	Project presentations