Language-based Security: Course Summary & Conclusions

Dr. Kevin Hamlen
Louis A. Beecherl, Jr. Distinguished Professor
Computer Science Department
The University of Texas at Dallas

Supported* in part by:
AFOSR Awards FA9550-08-1-0044 (YIP), FA9550-10-1-0088, FA9550-12-1-0077, and FA9550-14-1-0173;
NSF Awards #1065216, #1054629 (CAREER), and #1513704;
ONR Awards N00014-14-1-0030 and N00014-17-1-2995;
NSA Award H98230-15-1-0271;
DARPA Award FA8750-19-C-0006;
and NSF I/UCRC Awards from Lockheed-Martin, Raytheon, and I-A-I.

*Any recommendations, opinions, or conclusions expressed are those of the author and not necessarily of the above supporters.
Questions/Feedback?

• Questions about any topics we covered?
• Questions about cybersecurity in general? Big picture questions?
• Course suggestions
  • Did you find the course useful? intellectually stimulating? challenging?
  • Topics you wish had been covered but weren’t?
  • Alternative diagnostic approaches?
    • exams? different assignments? alternative quiz format?
• More/fewer readings?
  • two papers per class?
  • optional extra papers?
• Content per lecture
  • too much / too little?
  • higher / lower level of detail
  • slides vs. whiteboard presentation style
Topics Covered in this Course

• Secure software design from the ground up
  • Coq, functional programming, automated theorem proving

• Advanced bug-finding
  • ISA modeling, software model-checking

• In-lined Reference Monitoring
  • safety/liveness, aspect-oriented IRMs, automated certification

• Native code security and code-reuse attacks/defenses
  • ROP, ASLR, SFI, CFI, Reins, STIR
  • code injection, return-to-libc, implementation-aware code-reuse

• Web Security
  • end-to-end security, scripting security (JavaScript), malvertising

• Information Flow (confidentiality enforcement)
  • type-based, mostly static info flow control, covert channels

• Software Cyber-deception
  • honeypotting, honey-patching, taint-tracking, secret redaction
Advanced Topics, High Reward

• High reward
  • learn extremely cutting-edge material
    • too new to be found in textbooks
    • actively developed by industry movers (Microsoft, Google, …)
  • learn to read research *from the source*
    • extremely valuable long-term skill (essential for managers)
    • quizzes & lectures teach how to extract essential points
  • applied projects give deeper insight into one topic
    • Few software developers have ever formally verified *any* software!

• High-difficulty, low-volume workload
  • Homework (Coq programming): short + challenging!
  • Quizzes: Consider hard questions supervisors/clients would ask
  • No exams, no in-class time-pressure situations
Software (In)security

WikiLeaks Releases 8,200 New Emails From DNC Hack

Equifax Breach
- 143 Million Americans
- Names, Addresses
- Social Security Numbers

Yahoo!
- Over 1 Billion Accounts
  May Have Been HACKED

CapitalOne

Target

Marriott Data Breach

PlayStation Network HACKED

SuperFish

Ashley Madison
- Life is short. Have an affair.
Why is software so insecure?

• Huge and constantly evolving
  • Windows XP has 40 million lines of code
  • Microsoft Office had 30 million lines in 2006
  • Debian 5.0 has a staggering 324 million lines!
    • contrast: Space shuttle has only 2.5 million moving parts!

• Often written in unsafe languages
  • C, C++, VC++, Visual Basic, scripting languages, ...

• Increasingly sophisticated attacks
  • code-injection – attacker injects malicious code into vulnerable program, taking complete control
  • buffer-overrun
  • return-to-libc
  • return-oriented programming (RoP)
  • implementation-aware code-reuse attacks
Cybersecurity is UNINTUITIVE

Non-scientific approaches to software security today are largely *ad hoc*. They are based on intuition, guesswork, tradition, and a “check-the-boxes” mentality.
Philosophy of Security

- Good security is a *science*
  - minimal trusted computing base
  - maximal threat model
  - machine-checkable proofs of safety

- Security by obscurity *does not work!*
  - most obscurity is an illusion
  - invites disaster (e.g., current critical infrastructure security crisis)

- Probabilistic fault tolerance guarantees *are not secure*
  - security based on random testing doesn’t work against non-random adversaries!

- Desperate need for a more educated workforce who can recommend and apply scientific security practices at every stage of the software lifecycle
Needed: A *Science of Security*

- Scientific approaches to software security use **systematic, methodological, and mathematical reasoning** to offer *provable, quantitative security guarantees*.

- Examples:
  - formal, machine-checked software validation
  - software model-checking
  - science of active defense
  - science of cyber-deception
Scientific Advances from UTD

• **Opaque Control-flow Integrity** [NDSS 2015]
  • Thwarts implementation-aware code-reuse attacks by relocating security-sensitive control-data to a hidden, protected data segment, and randomizes the code
  • NSF Technology Breakthrough Award 2016

• **Object Flow Integrity** (OFI) [CSS 2017]
  • Fast control-flow Integrity with return address protection for *component-driven software* (finally!)
  • Supports the entire Windows Component Object Model
  • Example: Can fully automatically transform and secure Windows PowerPoint without sources

• **Superset Disassembly** [NDSS 2018]
  • Fully static CISC disassembly without heuristics
  • Now a mainstay of binary code hardening approaches

• **Binary Control-flow Trimming** [CCS 2019]
  • Automated binary code “debloating”: Find and remove hidden features (e.g., Bash Shellshock) without source code or any formal specification
  • Combines trace-based machine learning with a new, more powerful form of context-sensitive control-flow integrity
  • Accuracies of >99% and runtime overheads <2% when trained on ~200 sample traces
Cyberwarfare is ASYMMETRIC

Research Question: Can we make it as difficult for attackers to find and exploit these software vulnerabilities as it currently is for defenders to find and fix them?
Scalability Problems in Cyber Defense

[sources: taxpayer.net, statistica.com]
The Cyber Kill Chain

[credit: Netskope’19]

1. RECON
2. WEAPONIZE
3. DELIVERY
4. EXPLOIT
5. INSTALL
6. CALLBACK
7. PERSIST

OPPORTUNISTIC
TARGETED
SECONDS
MONTHS

HOURS TO MONTHS

a missed opportunity for thwarting attacks?
Honey-patching Heartbleed

2014

Google discovers Heartbleed

Google patch

OpenSSL notified

Codenomicon Discovers Heartbleed

OpenSSL patch

Honey-patch for Heartbleed created

THE ECONOMIC TIMES

Software

New technique Red Herring fights 'Heartbleed' virus

PTI Apr 15, 2014, 05.17PM IST

WASHINGTON: US cybersecurity researchers have developed a technique that fights the 'Heartbleed', and detects and entraps hackers who might be using it to steal sensitive data.

The Heartbleed bug, which became public last week, has set alarm bells ringing across the globe, including in India, for fear of exposing millions of passwords, credit card numbers and other sensitive information to hackers.
SignaC: A Honey-patching C Compiler

New Line of Work: “Crook-Sourcing” (will appear next month at ACSAC 2019!)

- Problem statement: Machine learning is great for training IDS defenses, but it’s very hard to get good data.
  - Severely unbalanced: Only a tiny fraction of network data is malicious.
  - Low training quality: An even tinier fraction is from advanced threats.
  - Unlabeled: We aren’t even sure which interactions are the malicious ones.
  - Privacy/Ownership: Rare good datasets cannot be released due to privacy concerns.

- Solution: Let’s “crook-source” the problem!
  - Trick attackers into thinking they’re winning, so they perform free penetration tests.
  - Goal: Keep attackers talking for as long as possible!
  - Resulting data is pre-labeled (always known to be malicious), high quality (only skillful adversaries are given deceptions), and concept-relevant (attacks are against real assets, not dedicated honeypots).
A Deception-enhanced Firewall

Adaptive Deception Example

Attacker obfuscation creates fuzzy decision boundary

Deceptions steer attacker toward unlearned portions of feature space

New data allows IDS to learn the obfuscation
Science of Software Cyber Deception

• Cyber Deception Science is a vital, understudied ingredient of future cyberdefense
  • addresses critical scalability problems of conventional defense approaches
  • levels attacker-defender asymmetries, which are growing beyond control

• Scientific Foundations for Cyber Deception
  • data-driven (machine learning, game theory, risk analysis, etc.)
  • highly interdisciplinary (cognitive sciences, economics, policy, etc.)

• Vectors of Scientific Progress
  • Assimilation – make everything valuable potentially a deceptive trap
  • Automation – remove humans from the loop to make deceptions fast
  • Adaptation – dynamically tailor deceptions to acquire specific data
  • Assurance – reliable evaluation methodologies, provable security
Recommendations and Career Advice
Recommendation: Keep Reading!

• Make it a habit: Read at least one recent (published in past 12 months) research paper in your area of study/expertise every week (minimum).
  • If some sections seem incomprehensible, don’t skip/avoid them! Read background material to fill in your knowledge gaps.

• Track the top conferences: IEEE S&P, ACM CCS, USENIX Security, NDSS, ACSAC
  • What are they publishing?
  • What are top scientists working on?
Advanced Topics, High Reward

• High reward
  • learn extremely cutting-edge material
    • too new to be found in textbooks
    • actively developed by industry movers (Microsoft, Google, ...)
  • learn to read research from the source
    • extremely valuable long-term skill (essential for managers)
    • quizzes & lectures teach how to extract essential points
  • applied projects give deeper insight into one topic
    • Few students have ever formally verified any software!

• High-challenge, low-volume workload
  • Homework (Coq programming): short + challenging!
  • Quizzes: Consider hard questions supervisors/clients would ask
  • No exams, no in-class time-pressure situations
Quizzes: A Model of How to Read

• Learn to identify and focus on the big questions...
  • “What’s the main point?”
  • “What’s the threat model?”
  • “What does the notation actually mean?”

• Extract key design motivations and implications...
  • Force you to think deeply and critically about cutting-edge research discoveries.
  • Mirror design choices and debates likely to arise in business and research meetings.
  • Motivate you to look back and reevaluate what you’ve read.
  • Encourage thoughtful reading habits rather than rote memorization.
Rising Cybersecurity Demand
Source: ISACA 2017 Cyber Security Jobs Report

2017 Cybersecurity Skills Gap

Too Many Threats

- $1 billion: Personally identifiable information (PII) records stolen in 2014
- 97% believe APTs represent a credible threat to national security and economic stability
- More than 1 in 4 organizations have experienced an APT attack
- $150 million: Average cost of a data breach by 2020
- 1 in 2 believe the IT department is unaware of all of organization's Internet of Things (IoT) devices
- 74% believe likelihood of organization being hacked through IoT devices is high or medium

Too Few Professionals

- 2 million: Global shortage of cybersecurity professionals by 2019
- 3X rate of cybersecurity job growth vs. IT jobs overall, 2010-14
- 84% organizations believe half or fewer of applicants for open security jobs are qualified
- 53% of organizations experience delays as long as 6 months to find qualified security candidates
- 77% of women said that no high school teacher or guidance counselor mentioned cybersecurity as a career
- 89% of U.S. consumers believe it is important for organizations to have cybersecurity-certified employees
Next Steps

• Formal Methods for Secure Software
  • CS 6371: Advanced Programming Languages
  • CS 6389: Formal Methods and Programming Methodology
  • CS 6367: Software Testing, Validation and Verification

• Other Security Courses
  • CS 6377: Intro to Cryptography
  • CS 6348: Data and Applications Security
  • CS 6324: Information Security
  • CS 6349: Network Security

• U.S. Citizens: See Dr. Sarac about UTD’s SFS cyber-security program.

• Other Security Researchers at UTD
  • Dr. Latifur Khan – data mining for intrusion detection
  • Dr. Murat Kantarcioglu – privacy (e.g., health care data management)
  • Dr. Alvaro Cardinas – critical infrastructure security
  • Dr. Yvo Desmedt – cryptography
  • Dr. Kamil Sarac – network security
  • Dr. Bhavani Thuraisingham – database security, inference control
  • Dr. Yiorgos Makris – hardware security
Opportunities at UTD

• Scholarship for Service (SFS) Cyber Program
  • Open to U.S. Citizens (usually upper-level undergrads)
  • See Dr. Kamil Sarac

• Computer Security Group (CSG)
  • learn hacking and cybersecurity hands-on
  • participate in “capture-the-flag” (CTF) competitions

• My Lab: Software Languages Security Lab (SL)²
  • Student research projects
    • I will pay (good!) students to work on Picinae
    • Opportunities for Masters thesis / independent study (non-paid)
  • Or just come learn more about what we’re doing
    • Weekly research meeting: weekly afternoons (email me next semester for an invite)
    • Email me next semester: Dr. Kevin Hamlen (hamlen@utdallas.edu)
Reminder: Project Presentations

- Next Monday (~20 min. + 5 min. Q&A per team)
- Select one spokesman to present, or divide presentation time among multiple speakers.
  - No penalty for team members who don’t speak. Full team gets the same project grade, regardless.
- Use slides
- General presentation flow:
  - **Problem description & motivation:** What are you trying to verify? Why is that property/problem important? How is it used or relied upon in practice?
  - **Background:** Summarize how the framework you’re using works. How is it defined? What does a student need to know to get started with it?
  - **Progress:** What have you completed so far?
  - **Future Work:** What obstacles are you currently facing? Are there unsolved problems? What do you hope to accomplish by the end?
Questions/Feedback?

• Questions about any topics we covered?
• Questions about cybersecurity in general? Big picture questions?
• Course suggestions
  • Did you find the course useful? intellectually stimulating? challenging?
  • Topics you wish had been covered but weren’t?
  • Alternative diagnostic approaches?
    • exams? different assignments? alternative quiz format?
• More/fewer readings?
  • two papers per class?
  • optional extra papers?
• Content per lecture
  • too much / too little?
  • higher / lower level of detail
  • slides vs. whiteboard presentation style