CSE 127 Computer Security

Deian Stefan, Winter 2019

Lecture 1: Introduction

Course Information

- Lecturer: Deian Stefan
  - Lectures: Mon & Wed 6:30-7:50pm, Solis 104
  - Discussion: Tue 8:00-8:50pm, Center 105
  - Office Hours: Mon 1-2pm or by apt CSE 3126
- TAs (office hours TBA):
  - Nadah Feteih
  - Jonathan Luck (tutor)
  - Kaiser Pister
  - Michael Smith
- Piazza: https://piazza.com/ucsd/winter2019/cse127/home
- Course Web Page: http://cse127.programming.systems

About Me

- Research
  - I work at the intersection of computer security, programming languages and operating systems
  - Defense: language design for security, runtime systems for security, sandboxing techniques
  - Offense: static analysis and symbolic execution
- Industry
  - Co-founder of security startup called intrinsic
  - Spend summers building security products
- Working groups
  - Node.js Security Working Group
  - W3C Web Application Working Group

Course Objectives

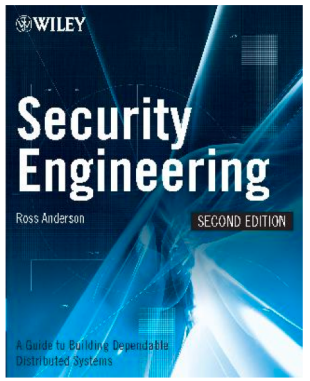
- A solid foundation of security concepts, backed by concrete examples
- Security mindset
  - How to **think like** an attacker/ security engineer
  - Looking beyond the system's intended functionality, to what it can be made to do
- Understanding how things work, how they break, and how to fix them
  - Technical details of vulnerabilities, attacks, and defenses
- Becoming a better engineer
  - Minimize number and severity of vulnerabilities you create
  - Understand the causes and impact of vulnerabilities that you are alerted to
  - Properly address vulnerabilities that are identified

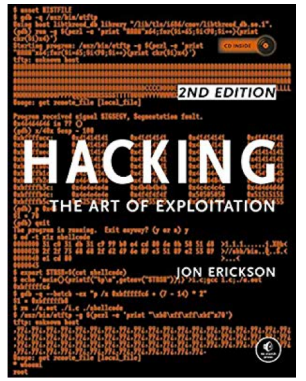
Prerequisites

- C programming and bits of assembly
- Some familiarity with:
  - OS (virtual memory, process)
  - Architecture (caches/TLBs)
  - Networking (packets, connections)
  - PL/compilers
- I'll try to touch on some of these things, but you need to be prepared to learn on your own

Course Material

- Books:
  - Security Engineering by Ross Anderson
  - Hacking by Jon Erikon (optional)
- Articles and research papers:
  - Links will be posed on course website
- Slides
  - Pretty much Stefan Savage's slides
  - Based on slides and notes from Kirill Levchenko, Alex Gantman, Alex Dent, Vitaly Shamtikov, Robert Turner, and a host of others





Grading

- Homework assignments & projects: 30%
- Midterm: 30%
  - midterm = if mideterm > 0
    then max(midterm, final)
    else 0
- Final: 40%
  - final will demand that you think beyond lectures and assignments

Rules

- Early policy
  - Can turn in assignment 3 days early to get 10% (of your grade) extra credit
  - No late days
- Regrades should be the exception not the norm
- No cheating!
  - UC San Diego policy: http://academicintegrity.ucsd.edu
  - If you are not sure if something is cheating, ask!
  - We will report ***all*** suspected cheating cases to academic integrity

Ethics

- In this class you will learn how to attack systems
- We learn attacks to understand how to defend them
- You have an obligation to use this knowledge ethically
  - You **may not** attack others
  - Many good ***legitimate*** hacking challenges (CTFs)

What is Security?

- Merriam-Webster online dictionary:

  Function: *noun*

  1 : the quality or state of being secure : as **a :** freedom from danger **: SAFETY b :** freedom from fear or anxiety **c :** freedom from the prospect of being laid off <job *security*>

  2 a : something given, deposited, or pledged to make certain the fulfillment of an obligation **b : SURETY**

  3 : an instrument of investment in the form of a document (as a stock certificate or bond) providing evidence of its ownership

  4 a : something that secures **: PROTECTION b** (1) **:** measures taken to guard against espionage or sabotage, crime, attack, or escape (2) **:** an organization or department whose task is security

What is Computer Security?

- Most of computer science is about providing ***functionality***:
  - Architecture
  - Algorithms
  - Operating Systems/Networking/Databases
  - Compilers/PL
- Computer security is ***not*** about functionality
  - It is about how the embodiment of functionality behaves ***in the presence of an adversary***

> "Software security is about integrating security practices into the way you build software, not integrating security features into your code"
> – Gary McGraw

History: two competing philosophies

- Binary model [secure vs insecure]
  - Traditional crypto and trustworthy systems
  - Assume adversary limitations X and define security policy Y

    If Y cannot be violated without needing X then system is secure, else insecure
  - You know people are invoking some version of this model if they say "proof of security", "secure by design" "trustworthy systems"
- Risk management model [more secure vs less secure]
  - Most commercial software development
  - Try to minimize biggest risks and threats
  - Improve security where most cost effective (expected value)
  - You know people are in this model if they use the words "risk", "mitigation", "defenses", "resilience", etc.

Classic example (binary model): perfect substitution cipher

$$\begin{array}{l} p_1\ p_2\ p_3\ \ldots\ p_n \\ \oplus\ b_1\ b_2\ b_3\ \ldots\ b_n \\ \hline c_1\ c_2\ \ c_3\ \ldots\ c_n \end{array}$$

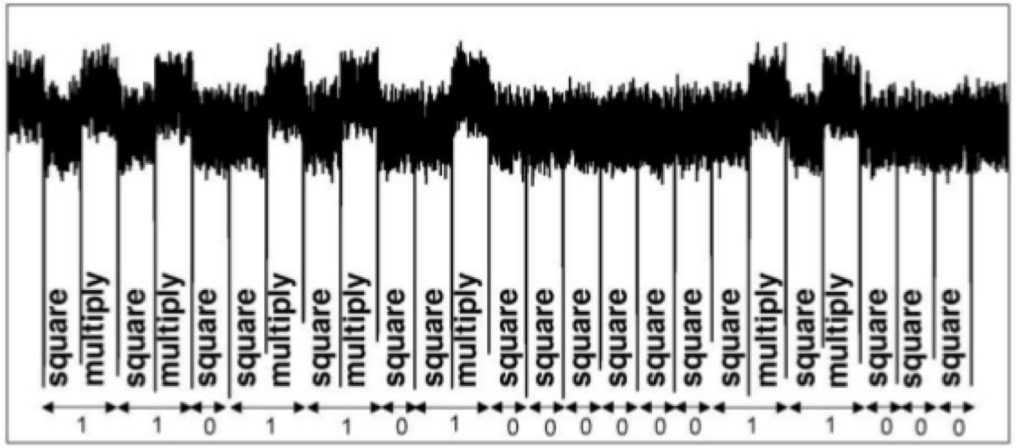
- Invited by combination of Vernam & Mauborgne (~1919)
- Choose a string of **random** bits the same length as the plaintext, XOR them to obtain the ciphertext.
- ***Perfect Secrecy*** (proved by Claude Shannon)
  - Probability that a given message is encoded in the ciphertext is **unaltered** by knowledge of the ciphertext
  - Proof: Give me any plaintext message and any ciphertext and I can construct a key that will produce the ciphertext from the plaintext. Zero information in ciphertext

Classic example (binary model): high-level languages

- Attacker limitation: can only write code in JavaScript
- Policy: cannot access arbitrary process memory
- Why is this important?
  - Whole classes of attacks are not possible if you write your code in a high-level language
  - So? Browser tab cannot read your files!

Problems with the binary model: Abstract design != Concrete artifact

- Many assumptions are **brittle** in real systems
  - Real artifacts fragile, imperfect, have bugs/limitations
    - Don't do precisely what spec says/documentation say
  - Large gap between abstraction and implementation
    - E.g., leak secret key though circuit draw:



Courtesy Oswald

Problems with the binary model: Abstract design != Concrete artifact

- JavaScript engines are notoriously complex…

Finding and Preventing Bugs in JavaScript Bindings

Fraser Brown* Shravan Narayan† Riad S. Wahby*
Dawson Engler* Ranjit Jhala† Deian Stefan†

Exploiting the Math.expm1 typing bug in V8

02 Jan 2019

- Most browsers also rely on process sandboxing for defense in depth!

Problems with the binary model: Abstract design != Concrete artifact

- Not limited to software…

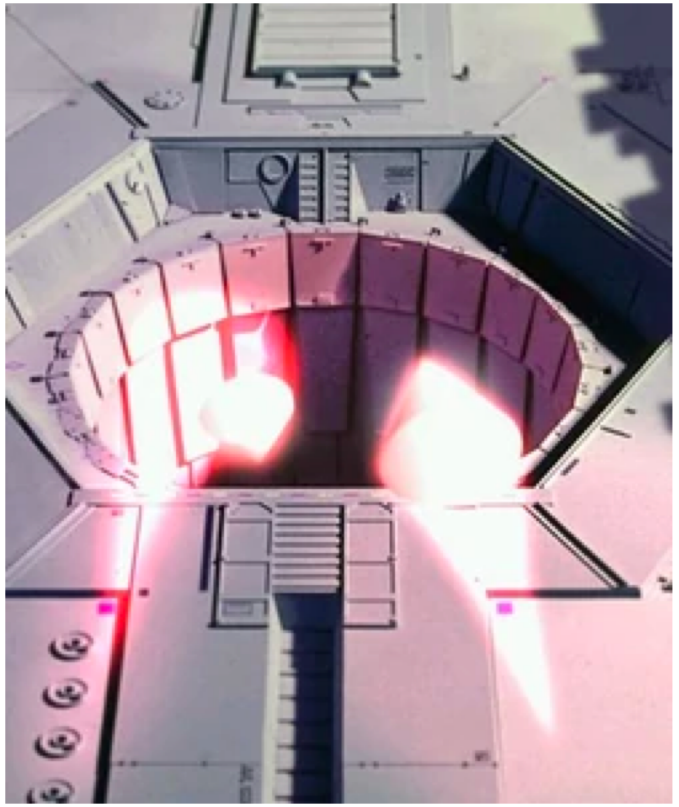
Problems with the binary model: security evolution

- As engineers, we often delude ourselves into thinking that we understand our own creations
  - or that we can create complex systems to do only what we meant them to do
- But … nobody knows how these systems really work
  - Complexity of computer systems is approaching complexity of biological organisms
    - 3 billion base pairs in human genome
    - 10+ billion transistors in modern CPUs
- Complex systems co-evolve with attacks against them
  - How we use systems, how we depend on them and how they might be attacked – all change over time
  - Systems deemed secure today may not be resilient to new threats

Classic example (risk management): Concrete barricades

- Prevent incursion by car bombers

Problems with the risk management model: One vulnerability can matter…



Problems with the risk management model: You never win

- Creates arms race – forced co-evolution

- The best you can hope for is **stalemate**

Problems with the risk management model: How to measure

- Its fine to say security is a spectrum, but how to **evaluate** risk or reward?
  - How many units of security does your anti-virus product give you?

- Big question: how do we measure security?
  - How is this different from airplane safety?
  - Car safety?
  - Drug safety?

Key meta issues in Security

- Policy
- Assets, Risks & Threats
- Value
- Protection
- Deterrence

Policy

- What **is** a bad thing?

- Remarkably tricky to define for known threats
  - The software on your computer likely has 100s of security options… How should you set them?
  - What might be a good security policy for who gets to access faculty salary data?
- Even harder for unknown threats
  - What is a reasonable policy for spam?

- Should a highly privileged user have more rights on a system or less?

Assets, Risks & threats

- Assets
  - What you want to protect
- Threats
  - Actions likely to cause damage, harm or loss
  - Includes both kinds of attacks (e.g., virus, social engineering) and kinds of attackers (e.g., script kiddie vs state sponsored actor)
  - Need to reason about requirements of each threat (what capabilities does the attacker need) and what it enables (what harm might come? What motivations might drive such a threat)
- Risk
  - What is the potential likelihood of a something bad happening (i.e., what threats are likely)

- These tend to be well formalized in some communities (e.g. finance sector) and less in others (e.g. energy sector)
- We'll talk more about threat models next class…

Value

- What is the cost if the bad thing happens?
- What is the cost of preventing the bad thing?

- Example: credit card fraud
  - Who pays if someone steals your credit card # and buys a TV with it?

- Example: Permissive Action Links for nuclear weapons
  - http://www.cs.columbia.edu/~smb/nsam-160/pal.html

Protection

- The mechanisms used to protect resources against threats
  - This is most of academic and industrial computer security

- Many classes of protections
  - Cryptographic protection of data
  - Software guards
  - Communication guards
  - User interface design (protect user against own limitations)
- Can be either proactive or reactive

Deterrence

- There is some non-zero expectation that there is a future cost to doing a bad thing
  - i.e. going to jail, having a missile hit your house, having your assets seized, etc
  - Criminal cost-benefit: $M_b + P_b > O_{cp} + O_{cm}P_aP_c$ [Clark&Davis 95]
    - M_b : Monetary benefit
    - P_b : Psychological benefit
    - O_{cp} : Cost of committing crime
    - O_{cm} : Monetary cost of conviction
    - P_a : Probability of getting caught
    - P_c : Probability of conviction

- Need meaningful forensic capabilities
  - Audit actions, assign identity to evidence, etc
  - Must be cost effective relative to positive incentives

Next Lecture…

Security Foundations: Threat Models and Risk Analysis