CS 161: Computer Security

Chufan Chen

November 4, 2021

"A good stock of examples, as large as possible, is indispensable for a thorough understanding of any concept, and when I want to learn something new, I make it my first job to build one."

– Paul Halmos.

"Constrained optimization is the art of compromise between conflicting objectives."

– William A. Dembski.

Contents

1 Wednesday, August 25th: Introduction

1.1 Course Outline

- Introduction to Security
- Memory Safety
- Cryptography
- Web Security
- Network Security
- Miscellaneous Topics

1.2 What is Security

Security enforcing a desired property in the presence of an attacker. These property includes:

- data confidentiality
- user privacy
- data and computation integrity
- authentication
- avaliablity

Security is important for our

- physical safety
- confidentiality/privacy
- functionality
- protecting our assets
- successful business
- a country's economy and safety

Everthing is hackable, especially things connected to the Internet.
What will you learn in this class?

- How to think adversarially about computer systems
- How to assess threats for their significance
- How to build programs & systems with robust security properties
- How to gauge the protections and limitations provided by today's technology
- How attacks work in practice

The rest of this lecture is largely focused on philosophical issues.

- People and Money
- Threat Model

1.3 People and Money

People attack systems for some reason. Often the most effective security is to attack the attacker's motivation.
It All Comes Down to People...
The Attackers
People attack systems for some reason
No attackers? No problem!

- They may do it for money
- They may do it for politics
- They may do it for the lulz
- They may just want to watch the world burn

1.4 Threat Model

For personal security, it best described by threat model and chill.
Threat Model is about who and why might someone attack you?

- Criminals for money
- Teenagers for laughs or to win in an online game
- Governments
- Intimate partners threat

We talked a lot about threat model because when you think about secuity you shouldn't just ask yourself this binary question is it secure or not but secuity against who, secure against what, what types of attackers, who might be trying to attack your system, what might their motivation be, what might their resources and their capabilities be, and we often don't need to defend against everyone maybe there's just some subset of people we need to defend against.
Often the most effective security is to attack the reasons for an attacker.
It All Comes Down to People...
The Users
Have you ever sacrificed your own personal security for the sake of usability?

- If a security system is unusable it will be unused
- Users will subvert systems anyway
- Programmers will make mistakes
- Social Engineering

But don't blame the Users

- Often we blame the user when an attacker takes advantage of them.
- Phishing is a classic example

Security often comes down comes down to money

- "You don't put a $10 lock on a $1 rock Unless the attacker can leverage that $1 rock to attack something more important

- "You don't risk exposing a $1M zero-day on a nobody"
- Cost/benefit analyses appear all throughout security

Prevention
The goal of prevention is to stop the "bad thing" from happening at all. On one hand, if prevention works its great. E.g. if you don't write in an unsafe language (like C) you will never worry about buffer overflow exploits. On the other hand, if you can only depend on prevention. You get Bitcoin and Bitcoin thefts.
Detection and Response
Detection: See that something is going wrong
Response: Actually do something about it
False Positive and False Negatives
False positive: You alert when there is nothing there
False negative: You fail to alert when something is there
This is the real cost of detection:
Responding to false positives is not free. And too many false positives and alarms get removed.False negatives mean a failure.
Defense in Depth
The notion of layering multiple types of protection together. Hypothesis is that attackers needs to breech all the defenses. But defense in depth isn't free. You are throwing more resources at the problem. You can have a incresead false positive rate.
Mitigation and Recovery The bad things happened, can we get back on our feet. Assumption: bad things will happen in the system, so can we design things so we can get back working? Back it up!
Password
Humans can't remember good passwords.
Something you know. Password
Something you have. RSA token
Something you are. Fingerprint
So what to do? Password Managers! E.g. 1Password
And FIDO U2F Security Keys is a very powerful second-factor for 2-factor authentication. This can not be phished.

2 Thursday, November 4th: Security Principles

2.1 The Properties We Want in a Safe

We want the inside to be inaccessible to an attacker. But what **sort** of attacker? And **how much time does** the attacker have? We want to measure how much time & capabilities needed for an attacker. For a safe, ratings communicate how much based on experts performing the attack. Such secuirty ratings are much harder in the computer security side.

Example 2.1. Security Rating: A Real Safe
TL-15($3,000): An expert with common tools will take $\geq$ 15 minutes to break in. May even have "relockers".
TLRTL-30($10,000): 30 minutes with common tools and a cutting torch.
Gun Safe: Meets the California requirements for safe storage of a handgun. But it is practically snake oil. It create an iluusion of security. It meets the legal requirement for security.

Lesson from safe: Security is economics. More security costs more. Standards often define security.
We've seen that laptop/desktop platforms grant application a lot of privilege.
Thinking About Least Privilege
When assessing the security of a system's design, identify the Trusted Computing Base(TCB). What components deos secuirty rely upon? Security requires that the TCB: correct, comlete(can't be bypassed) and secure(can't be tampered with). Best way to be assured of correctness and its security are KISS(Keep It Simple, Stupid) and Generally, Simple(Small). One powerfuldesign approach: privilege separation. Isolate privileged operations to as small a component as possible.

3 Appendix

List of Definitions and Theorems

Todo list