Security Principles

Ruta Jawale

June 25, 2019
Announcements

- CS61C review today, 6/25 from 2-3 pm at Wozniak Lounge, Soda
- Lecture moving to 245 Li Ka Shing starting tomorrow, 6/26
- Homework 0 is due tomorrow, 6/26
- Project 1 is out, so start looking for a project partner if you want someone to work with
- https://inst.eecs.berkeley.edu/~cs161/su19/feedback
Ethical Concerns

You, the student, will be introduced to a lot of security flaws and vulnerabilities throughout the course of this class; however, you are not to exploit these, in any way, outside the scope of provided course assignments.

As a student in computer security, please focus your efforts on discovering and understanding vulnerabilities so that you can build more private, secure systems in the future.

If you find a vulnerability in an existing system, contact the company or university in question privately. Depending on their policies, they may even provide a monetary reward.
Learning objectives

- Be able to identify and reason about the differences and connections between security principles
Scenario: Purchasing a Safe

What do you need to take into consideration?

- What are you keeping in the safe?
  - Goal of your system
- Whom are you keeping it safe from?
  - Attack surface of your system
Scenario: Keep gun safe from robbers

Would you use a CA state mandated gun safe?

Yes, seems reasonable to trust standards set by the government.

What if I told you that this was legally a gun safe?

Some of these safes can be opened by simply dropping them. So, as it turns out, this “legal gun safe” could not even keep a toddler out.

Lesson: checkbox security ≠ real security
Security rating: A real safe

Let's say you decide to look into security ratings of safes...

TL-15: takes expert a minimum of 15 minutes to break using common tools
Security rating: A stronger safe

TL-30: take a minimum of 30 minutes for an expert to break using common tools
Security rating: Now we’re talking

TL-30: take 30 minutes to break using common tools and cutting torch
Security rating: maximum overkill

TXTL-60: take 60 minutes to crack, working on all 6 sides, using 8 ounces of explosives
In all seriousness, how important are you? How important is that document? If you’re not that important, then do you really need a maximum overkill security rated safe?

Part of knowing your threat model is establishing who your enemies are. Your kid sister might be deterred by a basic safe.
Scenario: Keep document safe from robbers

The other part of knowing your threat model is related to the fact that security costs serious money...

...for both you and the attacker. Let’s say that the only threat to the privacy of your document was your kid sister. Would you really buy a $52k TXTL-60 rated safe? Would she really buy explosives just to get into your safe?

For sake of the argument, let’s say no. However, I really wouldn’t know. Maybe you have that type of a sister.
Scenario: Change a traffic sign
Scenario: Change a traffic sign
Scenario: Change a traffic sign
Scenario: Change a traffic sign
Scenario: Change a traffic sign
Algorithm/System vs Keys

A specific algorithm or system does not change, but the keys used will change.

- **Cellphone**
  - System: Pixel running Android Q
  - Keys: passwords or fingerprint

- **ElGamal Encryption System**
  - Algorithm: ElGamal encryption
  - Keys: public key, private key pairing
Obscure system \(\Rightarrow\) Secure system

A system doesn’t change. As such, we strive to design systems assuming the worst, that the attacker knows the system.

- Shannon’s Maxim
  - “The enemy knows the system.”

- Kerckhoffs’s Principle
  - “A cryptosystem should be secure even if everything about the system, except the [private] key, is public knowledge.”

Well, almost the worst. Knowing the private key would be the worst case, but we trust that the key is kept securely or changed regularly.
Scenario: µTorrent

**Mac and OSX Downloads - µTorrent® (uTorrent) - a (very) tiny ...**

Download the official µTorrent® (uTorrent) torrent client for Windows, Mac, Android or Linux-- µTorrent ... For Mac (1.42 MB); English (US) - November 27, 2016.

**uTorrent (Mac)**
- µtorrent estable (1.8.7 build 43001).
- Para Mac (1.42 MB); Inglés ...

More results from utorrent.com »

**uTorrent (Mac) - Free download**

https://utorrent.en.softonic.com/mac ▼

★★★★★ Rating: 3 - 550 votes - Free - Mac OS - Utilities/Tools
uTorrent, free download. uTorrent 1.8.8: Super lightweight torrent client for Mac. uTorrent for Mac is a lightweight and efficient BitTorrent client that allows you to ...
Scenario: \(\mu\)Torrent

**License Agreement**

Please review the license terms before installing \(\mu\)Torrent

\(\mu\)Torrent (also known as uTorrent) is a peer-to-peer file sharing application distributed by BitTorrent, Inc.

By accepting this agreement or by installing \(\mu\)Torrent, you agree to the following \(\mu\)Torrent-specific terms, notwithstanding anything to the contrary in this agreement.

License.

Subject to your compliance with these terms and conditions, BitTorrent, Inc. grants you a royalty-free, non-exclusive, non-transferable license to use \(\mu\)Torrent, solely for your personal, non-commercial purposes. BitTorrent, Inc. reserves all rights in \(\mu\)Torrent not expressly granted to you here.

Restrictions.

The source code, design, and structure of \(\mu\)Torrent are trade secrets. You will not disassemble, decompile, or reverse engineer it in whole or in part.
Scenario: μTorrent
Scenario: μTorrent
Scenario: $\mu$Torrent

Let’s consider what this program is able to do:

- Can it leak your files? Yes
- Can it delete all your files? Yes
- Can it send spam? Yes
- Can it add a new executable to your search path? Yes
Scenario: µTorrent

What does this program need to be able to do?

- Access screen
- Manage a directory of downloaded files
- Access config and documentation files
- Open connections for a given set of protocols
- Receive connections as a server
Scenario: Phone apps

Allow “Adult Cat Finder” to access your location while you use the app?
We use your location to find nearby adorable cats.

Don’t Allow  Allow
Scenario: You don’t trust a web application

If you still want to run it, you can use a sandbox. A sandbox isolates a process from the rest of the system, preventing access to the file system and system calls.

An example sandbox implementation in the Linux kernel is seccomp (secure computing). A user first opens the files it wants the untrusted process to work on, then calls seccomp to isolate and run that process.
**Sandbox** *(No, not the playground kind)*

Browser kernel provides restricted API to RE:

- RE can send a bitmap image with webpage to display to the user, but not active code.
- RE cannot access what files it wishes. BK might ask the user to upload a file the user chooses when RE requests so, but BK will not fetch if RE asks for it.
The trusted computing base (TCB) is the subset of the system that has to be correct, for some security goal to be achieved.

What is the TCB that ensures that if the RE fails, the browser does not fail?

The sandbox

What is the TCB that ensures the RE cannot fetch whatever files it wants from the file system?

The sandbox and the browser kernel
Related concept of privilege separation

Privilege separation is a separate design principle where we isolate privileged operations to as small a component as possible.
“Don’t reinvent the wheel”

- Use tools that have been tested extensively
- Creating a new tool will probably lead to lots of bugs and vulnerabilities
- For example, in end-to-end encrypted communication
  - Open SSL has +185 vulnerabilities reported on the CVE website
  - You probably don’t want to deal with all of these yourself
Let’s take a break!

Stretch! Introduce yourself to a person sitting next to you that you don’t know.
Scenario: You’re running a toll gate

What’s wrong with this picture? There are tire tracks on either side. Some people aren’t paying the toll gate!
Ensure complete mediation

Consider a toll gate structure more similar to those on bridges.

There’s water on either side. You don’t want to pay the toll? Oh? Well, where are you going to go?
Ensuring complete mediation

Construct a reference monitor to secure there’s a single point through which all access must occur to some capability/resource.

What are properties that we want out of this reference monitor or TCB?

- un-bypassable ("complete mediation")
- tamper-proof (single point must be secure)
- verifiable (can check its security holds)

However, reference monitors are still vulnerable to race conditions.
Time of Check To Time of Use (TOCTTOU)

**procedure** WITHDRAWAL(w)

// contact central server to get balance
1. let \( b := \text{balance} \)
2. if \( b < w \), abort

// contact server to set balance
3. set balance := \( b - w \)
4. dispense \( w \) to user
Time of Check To Time of Use (TOCTTOU)

**procedure** WITHDRAWAL\([w]\)

// contact central server to get balance
1. let \( b := balance \)
2. if \( b < w \), abort
   // balance could have decreased at this point
   // contact server to set balance
3. set balance := \( b - w \)
dispense \( w \) to user

If multiple threads of execution are running, it’s important to have a lock or semaphore for the resource.
Scenario: Nuclear Bunker

Announcements
Review
Objectives
Know your threat model
Security is economics
Don't rely on security through obscurity
Least privilege
Don't reinvent the wheel
Ensure complete mediation
Division of trust
Consider human factors
Only as secure as weakest link
Make system easy to use securely
Defense in depth
Design security in from the start
Summary
Two “Man” Policy: Require Two to Launch Nukes
Scenario: Movie Theater
Scenario: Company mandates password length

Attention all employees! All of your passwords must be at least 10 characters long, contain at least 2 digits, 1 uppercase character, 1 lowercase character, and 1 special character!
What ends up happening?

- Bank password: goMets12
- Credit card: browser8
- Brokerage: Initial23
- E-mail: letmein
- Company Portal Password: !secret
Scenario: Warning notifications

What does the average user see? How do they respond?
This ends up happening...

Unable to verify the identity of svn.xiph.org as a trusted site.
Blah blah geekspeak geekspeak geekspeak.

Before accepting this certificate, your browser can display a second dialog full of incomprehensible information. Do you want to view this dialog?

- View Incomprehensible Information
- Make this message go away permanently
- Make this message go away temporarily for this session
- Stop doing what you were trying to do

OK    Cancel
By the way, use fail-safe defaults

Make sure to have fail-safe defaults in your system, so this doesn’t happen...

...this happened to one of our TAs while boarding a flight to come teach you all.
Security as a chain

Trust can be thought of as a chain. Recall Trusted Computing Base (TCB). Trusting that the TCB is secure, we can establish trust in the entirety of our system.

Of course, if our system can be seen as a chain, it will only be as secure as the weakest link. Attackers follow the path of least resistance, and they will attack the system at its weakest point.
Scenario: Break Into a Phone

Let’s say you wisen up and set a high entropy password on your phone. In total, say you have the following features enabled to unlock your phone:

- password
- fingerprint
- facial recognition

What’s the easiest way an attacker can gain access to your phone?
**Scenario: Break Into a Phone**

- **Guess your password?**
  - attacker would have to spend lots of time to brute force a high entropy password
  - phones sometimes have auto-lock features after too many incorrect attempts

- **Use your fingerprint?**
  - sure! this is easy if they have physical access to you
  - again, this is why the intimate partner threat is so dangerous

- **Get around facial recognition?**
  - easiest! facial recognition software is known to be buggy

In which case, who or what is the weakest link? Up to interpretation. Let’s say the programmers who built the facial recognition feature.
Scenario: Break Into Encrypted Device

User is the weakest link!
Often times...

...humans can be the “weakest link”. Even if you design an complex, secure system, a non-malicious, average user might not use your system correctly, rendering its security useless.

However, don’t blame the user! Make your system easy to use securely.
So how should you design a useable secure system?

That’s a tough security question! It’s different in every case. Let’s consider a small example.

What’s good about this design?
Tells users they need to protect it!
Can we think of other examples of useable secure systems?
Scenario: Medieval Castle Design

Basic Layers of Defense
- Layer 1: Moat
- Layer 2: Outer wall
- Layer 3: Inner wall

Photo Credit: Sucuri Blog
Scenario: Independent Layers of Defense

Often times layers of security are not shaped like an onion. They are independent of each other.

How should you combine each security layer alerts to generate an alarm in defense in depth?

If you combine using OR, then an attacker will only need to trigger one of the security layers to set off an alarm. If you combine using AND, then an attacker will need to trigger both.

As food for thought, there are use cases for both AND and OR.
Independent layers of defense

Often times layers of security are not shaped like an onion. They are independent of each other.

How should you combine each security layer alerts to generate an alarm in defense in depth?

If you combine using OR, then an attacker will only need to trigger one of the security layers to set off an alarm. If you combine using AND, then an attacker will need to trigger both.

As food for thought, there are use cases for both AND and OR.
Two-factor authentication (2FA)

Types of user authentication:

- **Something you know**
  - account password
  - answers to security questions

- **Something you own**
  - physical keys
  - mobile phone

- **Something you are**
  - fingerprints
  - biometric data

2FA involves picking two authentication methods from two different categories.

For example, account password and mobile phone.
Scenario: UC Berkeley campus login
Recently UC Berkeley added this important 2FA...

...from a different perspective, 2FA appears to be an afterthought or security bolted on after the fact since passwords were revealed (quite long ago) to be insecure.
Bolt on security

2FA is important, so it wasn’t a clearly bad example. For a clearer example...

If they’re enabling remote upgrades so they can cover up an insecure system, that’s bolt on security.
In your future...

Consider security aspects during the design phase of your system. For that matter, also consider user privacy.

As a programmer interested in computer security, you have power to make technology more private and secure.
Summary

- Know your threat model
- Security is economics
- Don’t rely on “security through obscurity”
- Least privilege
- Don’t reinvent the wheel
- Ensure complete mediation
- Division of trust
- Consider human factors
- Only as secure as weakest link
- Make system easy to use securely
- Defense in depth
- Design security in from the start