Lecture 2:
Security Principles
Don't Blame The Users...

- Often we blame the user when an attacker takes advantage of them...
- Yet we've consistently constructed systems that encourage users to do the wrong thing!
- Phishing is a classic example:
- Which is a phishing email and which is an actual email from Chase?
Security often comes down to money...

• You don't put a $10 lock on a $1 item...
  • Unless the attacker can leverage that $1 item to attack something more important

• You don't risk exposing a $1M zero-day on a nobody
  • So I'm quite content to use my iPhone in a hostile network: free market cost of a zero-day (unknown/unpatchable) exploit chain for iOS is somewhere between $500k to $1.5M

• 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 it's great
  • E.g. if you write in a memory-safe language (like Python) you are immune from buffer overflow exploits

• On the other hand, if prevention fails, it can fail hard
  • Example: $68M stolen from a Bitcoin exchange, can't be reversed
  • Or Ethereum's July 2018: four separate multi-million-dollar theft incidents
  • Or Coinbase accounts: Averaging a theft a day!
Detection & Response

• Detection: See that something is going wrong
• Response: Do something about it
  • Example: Reverse the harmful actions (restore from backup), prevent future harm (block attacker)
  • Need both — no point in detection without a way to respond and remediate
### False Positive and False Negatives

- **False positive:**
  - You alert when there is nothing there

- **False negative:**
  - You fail to alert when something is there

- **Cost of detection:**
  - Responding to false positives is not free, and if there are too many false positives, detector gets removed or ignored
  - False negatives mean a failure
Defense in Depth

• The notion of layering multiple types of protection together
  • EG, the Theodesian Walls of Constantinople: Moat -> wall -> depression -> even bigger wall

• Idea: attacker needs to breach all the defenses to gain access

• But defense in depth isn't free:
  • You are throwing more resources at the problem
Composing Detectors for Defense In Depth

• The best case: the two detectors are independent
  • With FP1 and FP2 false positive rates and FN1 and FN2 false negative rates
    • Rate is 0-1:
      0 is it never has a false positive/negative,
      1 is it is always a false positive/negative...

• Parallel composition: either detector may alert to trigger a response
  • Reduces false negatives: new rate is FN1*FN2
  • Increases false positive rate: new rate is FP1 + (1-FP1)*FP2

• Serial composition: both detectors must alert
  • Reduces false positives: new rate is FP1*FP2
  • Increases false negatives: new rate is FN1 + (1-FN1) * FN2
Password authentication

• People have a hard time remembering multiple strong passwords, so they reuse them on multiple sites
  • Consequence: security breach of one site causes account compromise on other sites

• Solution: password manager
  • Remember one strong password, which unlocks access to site passwords

• Solution: two-factor authentication
  • Need both correct password and separate device to access account

• Free advice: to protect yourself, use a password manager and two-factor authentication
The Properties We Want in a Safe

• We want the contents to be inaccessible to an attacker
  • But what sort of attacker?
  • But 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 security ratings are much harder in the computer security side
Security Rating: A Real Safe

- TL-15:
  - An expert with common tools will take $\geq 15$ minutes to break in
Security Rating: A Stronger Safe

- **TL-30:**
  - The same expert and tools now takes 30 minutes
Security Rating: A Real Safe

• TL-15:
  • An expert with common tools will take $\geq 15$ minutes to break in

• Quiz: Suppose we sign up for a security alarm service. What guarantees do we need from it, for TL-15 to be adequate?
Security Rating: Now We Are Talking

- TRTL-30
  - 30 minute to break with tools and/or a cutting torch
Security Rating: Maximum Overkill...

- **TXTL-60:**
  - 60 minutes with tools, torches, and up to 4 oz of *explosives!*
  - Far easier to use "Rubber Hose Cryptanalysis" on someone who knows the combination

![Diagram of a crypto nerd's imagination vs. what would actually happen]
Lesson:
Security is economics

- More security often costs more
  - Need to balance expected loss from undefended system, vs cost of defense
- More purchasers often makes security cheaper...
Mac and OSX Downloads - µTorrent® (uTorrent) - a (very) tiny ...
www.utorrent.com/downloads/mac
Download the official µTorrent® (uTorrent) torrent client for Windows, Mac, Android or Linux-- uTorrent ...
... 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 ...

Download
µTorrent Stable(1.8.7 build 43001).
Für Mac (1.42 MB); Englisch ...

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 1.8.6: Super lightweight torrent client for Mac. uTorrent for Mac is a lightweight and efficient BitTorrent client that allows you to...
IMPORTANT - Read this License Agreement carefully before clicking on the "Agree" button. By clicking on the "Agree" button, you agree to be bound by the terms of the License Agreement.

LICENSE AGREEMENT
Please review the license terms before installing µTorrent

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

By accepting this agreement or by installing µTorrent, you agree to the following µTorrent-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 µTorrent, solely for your personal, non-commercial purposes. BitTorrent, Inc. reserves all rights in µTorrent not expressly granted to you here.

Restrictions.

The source code, design, and structure of µTorrent are trade secrets. You will not disassemble, decompile, or reverse engineer it in whole or in part.
LIGHT. LIMITLESS. ENGINEERED FOR POWERFUL DOWNLOADING.
What is this program able to do?

Can it leak your files elsewhere?
What is this program able to do?

Can it leak your files elsewhere?
Can it delete all of your files?
Can it send spam?
Can it add a new executable to your search path?

YES. Why?
What does this program need to be able to do?

Maybe:

- access screen
- manage a directory of downloaded files
- access config & documentation files
- open connections for a given set of protocols
- receive connections as a server
Check for Understanding

- We’ve seen that laptop/desktop platforms grant applications a lot of privileges

- Quiz: Name a platform that does a better job of least privilege
Does this follow the principle of least privilege?

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
Thinking About Least Privilege

• When assessing the security of a system’s design, identify the Trusted Computing Base (TCB).
  • What components does security rely upon?

• Security requires that the TCB:
  • Is correct
  • Is complete (can’t be bypassed)
  • Is itself secure (can’t be tampered with)

• Best way to be assured of correctness and its security?
  • KISS = Keep It Simple, Stupid!
  • Generally, Simple = Small

• One powerful design approach: privilege separation
  • Isolate privileged operations to as small a component as possible
The Base for Isolation: The Operating System...

• The operating system provides the following "guarantees"
  • Isolation: A process can not access (read OR write) the memory of any other process
  • Permissions: A process can only change files etc if it has permission to
    • This *usually* means "Anything that the user can do" in something like Windows or MacOS
    • It can be considerably less in Android or iOS
    • But even in Windows, MacOS, & Linux one can say "I don't want any permissions"
Web browser

“Drive-by malware”: malicious web page exploits browser bug to infect local files

Trusted Computing Base
The Chrome browser

Goal: prevent "drive-by malware", where a malicious web page exploits a browser bug to infect local files
The Chrome browser

70% of vulnerabilities are in the rendering engine.

1M+ lines of code

Now it sandboxes each web context so you can't even read out other web page content (E.g. spectre)
All your money in one place

We bring together all of your accounts, bills and more, so you can conveniently manage your finances from one dashboard.

- See all of your bills and money at a glance
- Create budgets easily with tips tailored to you
- Enjoy access to unlimited free credit scores, without harming your credit

Sign Up Free

Effortlessly stay on top of bills

Bills are now easier than ever to track. Simply add them to your dashboard to see and monitor them all at once.
Discuss with a partner

- How would you architect mint.com to reduce the likelihood of a breach that reveals everyone’s bank passwords?

- How would you architect mint.com to reduce the likelihood of a breach that empties everyone’s bank account?
Ensuring Complete Mediation

- To secure access to some capability/resource, construct a reference monitor
- Single point through which all access must occur
  - E.g.: a network firewall
- Desired properties:
  - Un-bypassable (“complete mediation”)
  - Tamper-proof (is itself secure)
  - Verifiable (correct)
  - (Note, just restatements of what we want for TCBs)
- One subtle form of reference monitor flaw concerns race conditions …
A Failure of Complete Mediation

Every security-relevant action must be checked for authenticity, integrity and authorization.
More security principles

• Use fail-safe defaults
• Consider human factors
• Only as secure as the weakest link
• Don’t rely on security through obscurity
• Trusted path
Time of Check to Time of Use
Vulnerability: Race Condition

procedure withdraw(w)
// contact central server to get balance
1. let b := balance
2. if b < w, abort

// contact server to set balance
3. set balance := b - w
4. dispense $w to user

TOCTTOU = Time of Check To Time of Use

Suppose that here an attacker arranges to suspend first call, and calls withdraw again concurrently.
A Hundred Million Dollar
TOCTTOU Bug...

- Ethereum is a cryptocurrency which offers "smart" contracts
  - Program you money in a language that makes JavaScript and PHP look beautiful and sane
- The DAO (Distributed Autonomous Organization) was an attempt to make a distributed mutual fund in Ethereum
  - Participants could vote on "investments" that should be made
- The DAO supported withdrawals as well
A "Feature" In The Smart Contract

• To withdraw, the code was:
  • Check the balance, then send the money, then decrement the balance

• But sending money in Ethereum can send to another program written by the recipient

• So someone "invested", then did a withdraw to his program
  • Which would initiate another withdraw...