PRINCIPLES OF COMPUTER SECURITY

MISC

▸ Admin 1 due tomorrow
▸ Homework 1 released later today, due 23:59 Thursday, Jan 17
▸ Lab 1 due 23:59 Thursday, January 25
▸ Problems completing Admin 1? Post to Piazza or email me.

PRINCIPLES OF COMPUTER SECURITY

TODAY

▸ Principles of Computer Security
▸ Common Security Assumptions

PRINCIPLES OF COMPUTER SECURITY

LAST TIME

▸ To protect systems, you must know your enemy
▸ Absolute security is usually impossible, infeasible, or undesirable.
  ▸ The best approach is to fully assess what threats you are likely to encounter and create defenses that can stop those threats.

PRINCIPLES OF COMPUTER SECURITY

SECURITY RATING: "MEH"

▸ TL-15
▶ Expert with common hand tools can crack it in >= 15 minutes

PRINCIPLES OF COMPUTER SECURITY

SECURITY RATING: "EH, NOT BAD"

▸ TL-30
▶ Expert with common hand tools can crack it in >= 30 minutes
Security Rating: “Now You’re Talking”

- TRTL-30
  - Expert with cutting torch can crack it in >= 30 minutes

Security Rating: “Wow”

- TXTL-60
  - Expert with power tools, cutting torch, and up to 4oz of explosives (!!!!) can crack it in >= 60 minutes
  - But if someone knows the combination...

Security Rating: “Why Bother?”

- Meets all Californian requirements for “safe” storage of a hand gun
- Child with a BIC pen can open the cylindrical locks in < 10 seconds
- Some models will spring open if you drop them just 30cm
- So why bother?
  - They create the illusion of security
  - They meet the legal requirement for security

PRINCIPLE: SECURITY IS ECONOMICS

- More security (usually) costs more
- If monetary costs are the same or less and don’t impose other costs, reasonable people would always choose “more”
- Standards often define security
  - The safe standards from Underwriters Laboratories
  - The “gun safe” standards from the California Department of Justice
- Security imposes costs on attackers, too.
  - More tools, time, people, expertise, and risk of getting caught.
PRINCIPLES OF COMPUTER SECURITY

What is this software able to do?

- Can it read all your files?
- Can it delete all your files?
- Can it send spam e-mails?
- Can it modify system settings?

What does this software need to do?

Perhaps:
- Access the screen
- Manage a directory of downloads
- Modify its own settings
- Open connections for certain protocols
- Receive connections as a server

PRINCIPLES OF COMPUTER SECURITY

PRINCIPLE: LEAST PRIVILEGE

▸ Give software the set of access privileges it legitimately needs to do its job — but nothing more
▸ Try to minimize the privilege you give each program and system component
▸ LP doesn’t reduce the probability of failure, but it can reduce the cost of failures
▸ The less privilege a program has, the less harm it can do if it goes wrong or becomes subverted

PRINCIPLES OF COMPUTER SECURITY

TRUSTED COMPUTING BASE

▸ The Trusted Computing Base (TCB) is the set of mechanisms required to function correctly for the system to behave as expected.
▸ Examples:
  - CPU logic (e.g., 2 + 2 = 4)
  - Language runtimes (JVM, C libraries, etc.)
  - Memory safety (e.g., no buffer overflows)
▸ Security requires that the TCB:
  - Is correct
  - Is complete (can’t be bypassed)
  - Is self-secure (can’t be tampered with)

PRINCIPLES OF COMPUTER SECURITY

TRUSTED COMPUTING BASE

▸ The Trusted Computing Base (TCB) is the set of mechanisms required to function correctly for the system to behave as expected.
▸ What assurances are there of correctness, completeness, and security?
  - Simple, small design: easy to understand, test, audit, and/or verify.
  - Design approach: privilege separation
  - Split TCB into components, only give privileges to necessary components
**Least Privilege in Practice: Web Browsers**

**Proposition:** 70% of browser vulnerabilities are in the rendering engine.

**Goal:** Prevent "drive-by malware," where a malicious web page exploits a browser bug to access local files.

**Proposition:** Time of Check To Time of Use (TOCTTOU) vulnerability.

**Example Code:**

```c
void withdraw(int cash) {
    int bal = getBalance();
    if (bal >= cash) {
        setBalance(bal - cash);
        dispense(cash);
    }
}
```

**TOCTTOU = Time of Check To Time of Use**
PRINCIPLE: SEPARATION OF DUTY

- Split up privilege so no one user or program has complete power
- Require more than one party to approve before access is granted

Avoiding Accidental Armageddon:
- In a nuclear missile silo, two launch officers must agree before the missile can be launched, both switches can not be reached by one officer

Fraud Prevention:
- In a movie theater, you pay the teller and get a ticket; then when you enter the movie theater, a separate employee tears your ticket and retains half
- In many organizations, purchases over a certain $ amount must be approved both by the requesting employee and by a separate purchasing office

PRINCIPLE: FAIL-SAFE DEFAULTS

- Use default-deny policies
  - Start by denying all access, then allow accesses only to support specific functionality.
  - For example:
    - Default UMASK for new users is usually 0022 Readable by everyone!
    - On grunhilda.soe.ucsc.edu, the UMASK for new users is 0077 Readable only by you

Ensure that if the security mechanisms fall back to secure behavior, not to insecure behavior
- Firewalls must explicitly decide to forward a given packet or else the packet is dropped
- If the firewall fails, no packets are forwarded

As governor of Maryland, Spiro Agnew had full control of how some state contracts were awarded, and used this authority to set up a kickback scheme.

As Nixon’s vice president, he tried to make himself the authority over some federal contracts, but ultimately he was prevented.
PRINCIPLE: DEFENSE-IN-DEPTH

▸ If you use multiple protections, they should be independent: all of them would need to be breached before the system's security will be endangered.

▸ When one protection is down for maintenance or upgrade, the system as a whole remains secure.

PRINCIPLE: PSYCHOLOGICAL ACCEPTABILITY

▸ Ensure your users buy into the security model.

▸ Will your users abide by the mechanism or try to subvert it?

▸ What would happen if:

▸ UCSC network admins decided to configure the firewalls to block all traffic to https://stackexchange.com/?

▸ Faculty & students would instantly decide the firewall was a hindrance to their work and route around it, setup SSH tunnels, VPNs etc.

▸ UCSC admins decide that all CruzID passwords (Blue&Gold) must be reset to automatically generated strings that are at least 17 characters long, and must be changed once a week?

PRINCIPLE: HUMAN FACTORS MATTER

▸ Security mechanisms should not assume something about human behavior when interacting with the system that might not hold, even in the absence of conscious decisions by the users to subvert that system.

"Humans are incapable of securely storing high-quality cryptographic keys, and they have unacceptable speed and accuracy when performing cryptographic operations. They are also large, expensive to maintain, difficult to manage, and they pollute the environment. It is astonishing that these devices continue to be manufactured and deployed. But they are sufficiently pervasive that we must design our protocols around their limitations."

**PRINCIPLES OF COMPUTER SECURITY**

**PRINCIPLE: KNOW YOUR THREAT MODEL**

- Intellectual Curiosity
- Bragging Rights
- Financial Gain
- Political Motivation
- Things we haven’t thought of yet...

*Be careful with old code, the assumptions originally made may no longer be valid; the threat model may have changed.*

**PRINCIPLES OF COMPUTER SECURITY**

**PRINCIPLE: DETECT IF UNABLE TO PREVENT**

- If you can’t prevent break-ins, **at least detect them** (and, where possible, provide a way to recover or to identify the perpetrator)
- **Save logs** so that you have a way to analyze break-ins after the fact
- Regularly back-up these logs off-site
- Example tamper-resistant hardware: (NIST FIPS 140-1)
- Type III devices (the highest level) are intended to be tamper-resistant, but are very expensive
- Type II devices are only required to be tamper-evident (less expensive)

**PRINCIPLES OF COMPUTER SECURITY**

**PRINCIPLE: DON’T RELY ON SECURITY THROUGH OBSCURITY**

- Don’t rely on the confidentiality of your design, algorithms, source code, etc.
- Hard to keep internal details of any system secret from a sufficiently motivated adversary
- Hard to design and deploy a new system: once they are leaked, game over.
- Systems that rely upon their code or design remaining secret to maintain security tend to fail miserably and publicly
- Kerckhoffs’ Doctrine (1880’s): “A crypto system should be secure even if everything about the system, except the key, is public knowledge”
- Shannon’s Maxim (1950’s): “The enemy knows the system”

**PRINCIPLES OF COMPUTER SECURITY**

**PRINCIPLE: DESIGN SECURITY IN FROM DAY ONE**

- Retrofitting security (or anything) to an existing system is always difficult
- Backwards compatibility is particularly painful
- Might get stuck with supporting/defending/mitigating the insecurities of all versions of the software.
- A system is only as secure as its weakest component.
- Components that are secure in isolation may be insecure when combined
- Using mechanisms that provide compositional security guarantees make it easier to safely construct complex systems
**PRINCIPLES OF COMPUTER SECURITY**

**PRINCIPLE: DESIGN CONSERVATIVELY**
- Systems should be evaluated according to the worst plausible security failure under assumptions favorable to the attacker
- If there is any plausible circumstance under which the system can be rendered insecure, it is prudent to seek a more secure system
- But... this must be balanced against Security is Economics
  - We must decide the degree to which our threat model indicates resources should be spent mitigating the given scenario

**PRINCIPLE: PROACTIVELY STUDY ATTACKS**
- We should devote considerable effort to trying to break our own systems; in doing so we gain confidence in their security
- Security is a game where the attacker gets the last move
- It can be very costly if a security hole is discovered after a system is widely deployed, it pays to try to **identify attacks before individuals tempted to exploit them do**
- Automation is crucial here!

---

**COMPUTER SECURITY PRINCIPLES**
- Security is Economics
- Least Privilege
- Fail-Safe Defaults
- Separation of Duty
- Defense in Depth
- Psychological Acceptability
- Human Factors Matter
- Complete Mediation
- Know Your Threat Model
- Detect if Unable to Prevent
- Don’t Rely on Security Through Obscurity
- Design Security in From Day One
- Design Conservatively
- Proactively Study Attacks

---

**REASONABLE ASSUMPTIONS**
- Attackers can interact with your systems without particular notice
- Probing (poking at systems from inside, port scanning etc.) may go unnoticed
- ...even if highly repetitive, lead to crashes, and are easy to detect
- Attackers know general information about your systems
  - OS, software versions, usernames, server ports, IP address, activity patterns, admin procedures, etc
- Attackers can obtain access to an exact copy of your system and configuration to measure and determine how it works.
- Attackers will make enthusiastic use of **automation**
- Attackers can initiate sophisticated **coordinated activity** across geographically and architecturally disparate systems
- Attackers can bring **large resources** to bear if required
  - Computational, network capacity.
  - Botnets are like EC2 for attackers
- If it helps the attacker in some way, assume they can obtain elevated privileges
- But if the privilege gives everything away (attack becomes trivial) then the best we can do is worry about unprivileged attacks

---

**REASONABLE ASSUMPTIONS**
- Attackers can interact with your systems without particular notice
- Probing (poking at systems from inside, port scanning etc.) may go unnoticed
- ...even if highly repetitive, lead to crashes, and are easy to detect
- Attackers know general information about your systems
  - OS, software versions, usernames, server ports, IP address, activity patterns, admin procedures, etc
- Attackers can obtain access to an exact copy of your system and configuration to measure and determine how it works.
- Attackers will make enthusiastic use of **automation**
- Attackers can initiate sophisticated **coordinated activity** across geographically and architecturally disparate systems
- Attackers can bring **large resources** to bear if required
  - Computational, network capacity.
  - Botnets are like EC2 for attackers
- If it helps the attacker in some way, assume they can obtain elevated privileges
- But if the privilege gives everything away (attack becomes trivial) then the best we can do is worry about unprivileged attacks

---

**FINAL THOUGHTS**
- The ability to robustly detect that an attack has occurred does not mean you should stop trying to prevent it
- Infrastructure systems (ISPs, high level domain servers, etc.) tend to be well protected, i.e. hard to directly take over
- Hence, a vulnerability that requires compromised infrastructure is consequently less worrisome than one that does not
- Even attacks that only work one-in-a-million tries are dangerous: attacker just needs to try a million times!
- Attackers are rarely daunted by the fear of getting caught
NEXT LECTURE

- Software Security