# Secure Computer Systems

## Design Principles for Secure Systems

**Mustaque Ahamad, Ph.D**

Professor & Associate Director: Educational Outreach, Institute for Information Security & Privacy

*Georgia Tech - College of Computing*

## Principles to Follow When Designing for Security

Georgia Tech

# Before We Begin

- Our challenge is to make a complex system trustworthy
- Ideally, we should not have any vulnerabilities or errors
    - One vulnerability is all an attacker needs
- How can we systematically minimize the likelihood of security weaknesses?
    - By following design principles
    - Design principles help avoid actions that have negative impact on security

# Design Principles for Secure Systems

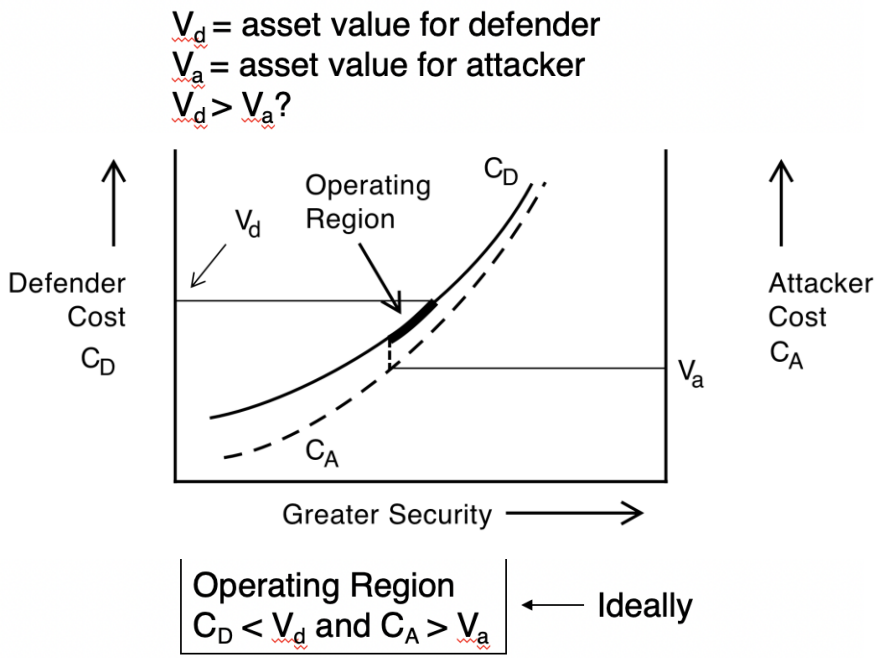
Georgia Tech

# Real-World Security

- **Assets**
  - How much value do I place on what needs to be secured?
- **Threats actors**
  - Who does it need to be protected from?
  - Why are we a target?
- **Defenses**
  - Lock and key, alarms
  - Guards, unarmed vs. armed, armies
- **Design principles should be informed by cost and effectiveness of defenses**

# Economics of Security

- How do we decide how much to spend on security?
- Defender security cost and benefit
  - Cyber risk = Attack likelihood * Attack Impact
  - Reduce risk to an acceptable level
  - Defense vs. response cost
- Attacker cost and benefit
  - Work factor (cost of attack) vs. Gain




Design Principle 1: Security cost must be commensurate with threat level and asset value

Georgia Tech

# Principle of User Acceptability

Georgia Tech

# Design Principle 2: User Acceptability

**Another Kind of Cost of Security**

- Security is annoying, gets in the way of usability

**Examples**

- Password requirements
- Multi-factor authentication

**Related Observation**

- A system is only as secure as its weakest link in the security chain
- People are often the weak link

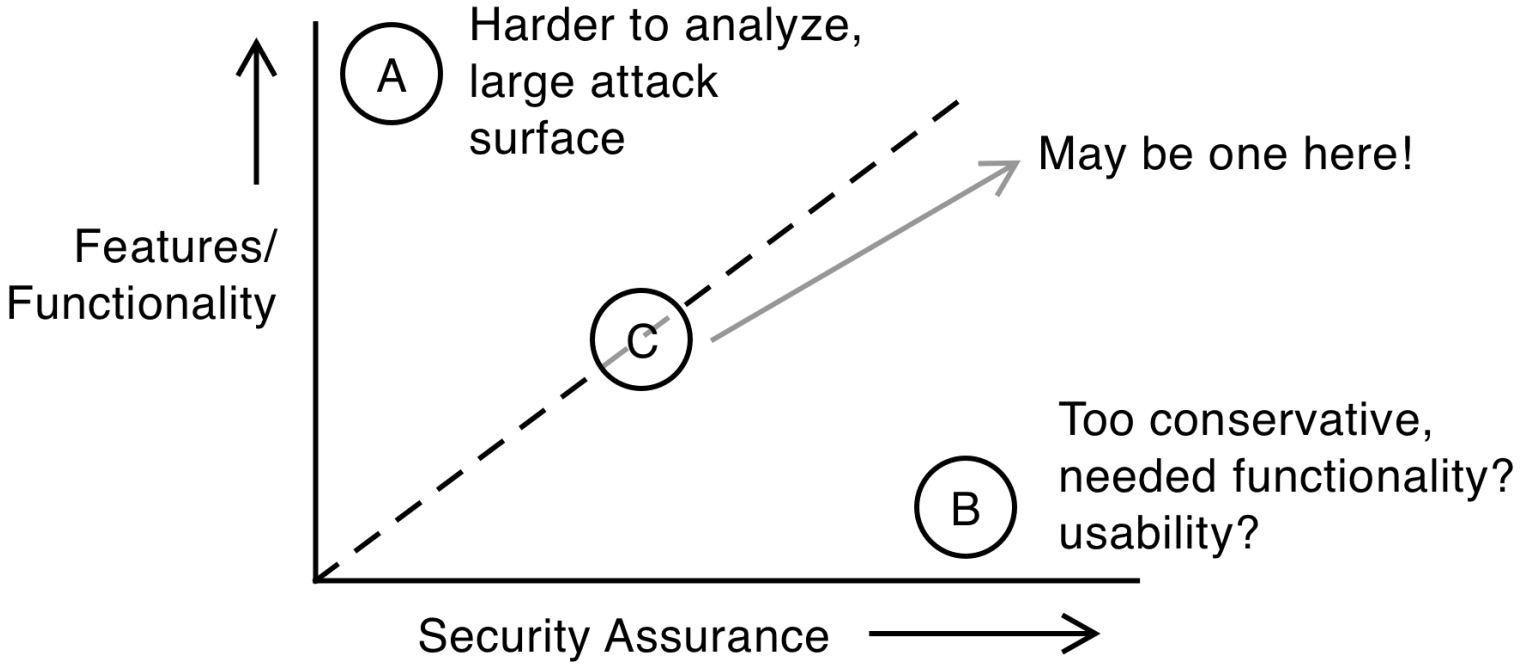
**Make Right Assumptions about People so they are not the Weak Link**

**User or psychological acceptability design principle**

# Economy of Mechanisms

Georgia Tech

# Another Tradeoff: Complexity vs. Security




Georgia Tech

# Design Principle 3: Keep it Simple

**Economy of Mechanism**

- User fewer and simpler mechanisms when possible
- Related one
  - Open design – do not count on security by obscurity or attacker's ignorance of how system works (only need to find one way to compromise it)
  - Does open source help?
    - Heartbleed vulnerability
- Do widely used systems follow this principle?
  - Windows 10 – about 50 MLOC
  - Android, in the 10~15 MLOC
- OS vs. hypervisor as the TCB
  - More compact (about 300 KLOC)

Georgia Tech

# Least Privilege & Separation of Privileges

Georgia Tech

# Design Principle 4: Least Privilege

- Now that we have our system and users, let us focus on what happens when they run their applications
- Users are authenticated
- Users are given access rights or privileges
- Least Privilege Principle – At any time, your program runs with the fewest privileges that allow it to successfully complete its execution
- Do common systems do this?
    - Unix UID, all your files are accessible while you browse the web?
    - Android – Different UIDs for different apps

# Least privilege (cont.)

**Separation of Privileges**

- Example: Separate keys for different secure areas (each office has a different key)
- Fine-grain access control
  - Different resources are accessed with different privileges

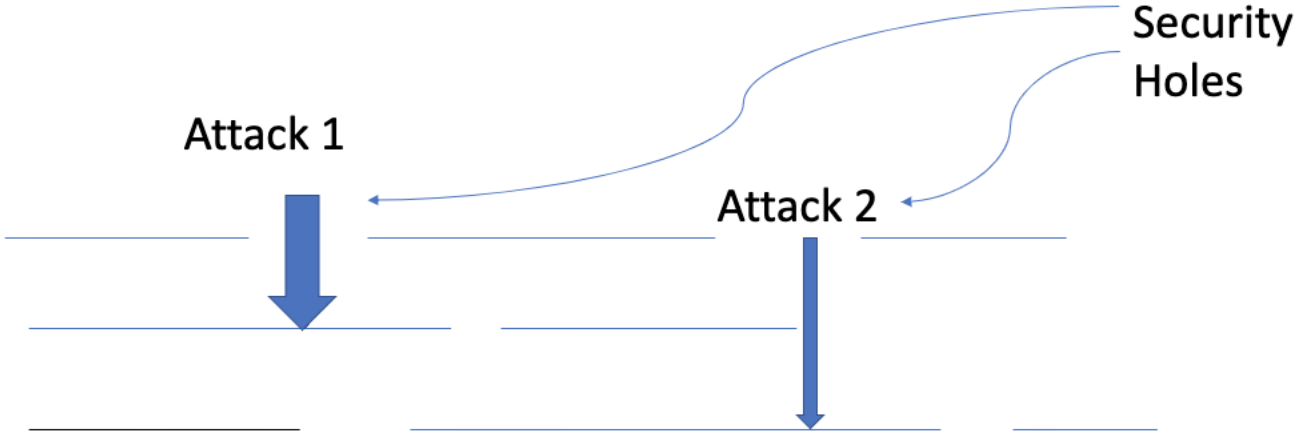
**Fail-safe Default**

- Default is deny (fail rather than allow potentially insecure access)
- Default password of 12345 ??.
  - A popular router had this vulnerability.

# Defense in Depth

Georgia Tech

# Design Principle 5: Layered Defense or Defense in Depth




Corollary: Diverse mechanisms are less likely to share the same vulnerability.

Georgia Tech

# Detecting Login Trojan with Defense in Depth

**Revisiting Reflections on Trusting Trust (Bruce Schneier blog, attributed to Wheeler**

- Two independently developed compilers A and B. Assume no more than one source is likely to introduce a Trojan
- Assume trojan is present in A (we do not know it)
- $S_A$ and $E_A$ are source and executable of A. Similarly $S_B$ and $E_B$ for compiler B
- Step 1: Compile $S_A$ with $E_A$. Let X be new executable obtained
- Step 2: Compile $S_A$ with $E_B$. Let Y be new executable obtained (X not being equal to Y is fine)
- Step 3: Compile $S_A$ with X, yielding X'. Similarly, compile $S_A$ with Y, yielding Y'
- If X' is not equal to Y', there is an issue with one of the compilers

Georgia Tech

# Should X’ = Y’?

- The two compilers $E_A$ and $E_B$, compile the same source program $S_A$. If both of them are correct (e.g., trojan free), X and Y should be functionally equivalent.
- Two functionally equivalent programs, given the same input $S_A$, should produce the same output
  - Hence X’ should be same as Y’.
- If this is not the case, one of the compilers is buggy.
- Defense in depth principle requires more than one defense but provides greater security.

# Some Thoughts on Design Principles for Secure Systems

Georgia Tech

# Are Design Principles Commonly Followed?

**US Postal Service exposed data of 60 million users**

- Once logged in, anyone could read anyone else's information
- No access control done!
- Least privilege not enforced

**Mirai botnet**

- IoT botnet, routers, cameras etc.
- Used bots to mount DDoS attacks
- Used small set of possible username/passwords, default in many cases

**Remote debug feature exploited by Morris Work**

- Fail-safe defaults

**An enterprise network uses both a firewall and an IDS/IPS**

- Defense in depth

**Many more examples, we will discuss them throughout the course**

Georgia Tech

# Cybersecurity is More Than Prevention

**Security = Prevention + Detection + Response/Remediation**

**Prevention => Keep the Attacker Out**

- This is what we focused on

**Prevention Only Goes This Far**

- Popular cybersecurity axiom: Everyone is compromised, some know they are and other will know in the future.
- Some major breaches lasted years

**What Can be done to Facilitate Detection, Response and Remediation?**

- False alerts and security analyst overload (related to first principle)
- Modularity for patching (remediation)
- Secure in design, by default and in deployment (Microsoft SD3)

Georgia Tech

# Summary

- **Design principles for secure systems help us understand**
  - Cost of security, including impact on usability
  - Importance of safe defaults and only granting privileges that are necessary
  - Open design and keeping the system simple
- **TCB that follows secure design principles is more likely to meet its requirements**
  - Next module will explore how hardware can help meet a necessary requirement

Georgia Tech