Applied Cryptography and Network Security

William Garrison
bill@cs.pitt.edu
6311 Sennott Square

Course Introduction

University of Pittsburgh

Administrivia

Applied Cryptography and Network Security

CS 1653/2053

T/H 11:00–12:15

2300 Sennott Square

Instructor:

William Garrison
bill@cs.pitt.edu
6311 Sennott Square
OH: TBA (By appt, for now)

Teaching Assistant:

TBA

Web:

https://bill-computer.science/1653

Why should I take this course?

My goal is to help you answer the following types of questions:

- How do various types of cryptosystems work?
  - Symmetric key
  - Public key
  - Threshold
  - "Modern cryptography" (elliptic curve, post-quantum)
- What sorts of cool things can we do with cryptography?
- What does it mean to be secure? Private?
- How can we design functional and secure systems?

(Some) Questions that we will not answer:

- How do I design new cryptographic algorithms?
- How do I break into the [*your favorite software*] system?
- …

Topic Outline

Introduction

- CIA properties
- Vulnerabilities, threats, and attacks
- Design principles

Cryptographic tools

- Symmetric key cryptography
- Cryptographic hash functions
- Public key cryptography
- Secret sharing and threshold cryptography

Applications of cryptography

- User authentication
- Digital certificates
- Key distribution
- Protecting digital transactions
- …

Topic Outline

Network Security

- Designing and attacking protocols
- Denial of service
- Intrusion detection

Research Issues in Privacy

- Private messaging
- Private routing
- Data privacy

Modern Cryptography

- Zero-knowledge proofs
- Homomorphic encryption
- Elliptic-curve cryptography
- Post-quantum cryptography

How will my grade be calculated?

Overall Breakdown:

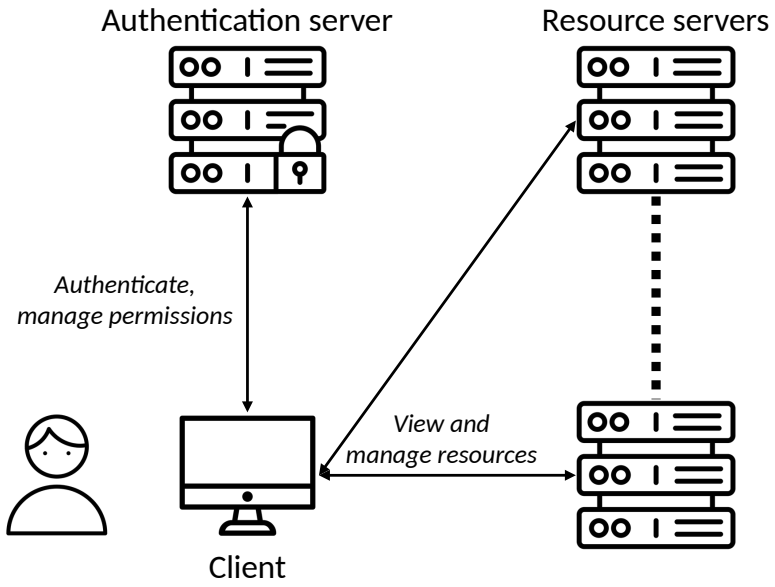
- 35% Exams
- 40% Project
- 10% Homework assignments
- 15% Lecture participation

In general, you can expect that:

- Exams will test how you think about problems, not how well you memorize facts
- Few homework assignments (2–3)
  - Short programming exercises
  - Cryptanalysis
  - Reflect on an article
  - …

Our project will take a security engineering approach to applying the material covered in class

The Scenario:



The Gist:

- Completed in groups
- Five phases (i.e., mini-projects) that build upon one another
  - Apply the concepts that we cover in class
  - Test your ability to think about problems faced during the development of “real” products
  - Cultivate your ability to work in groups

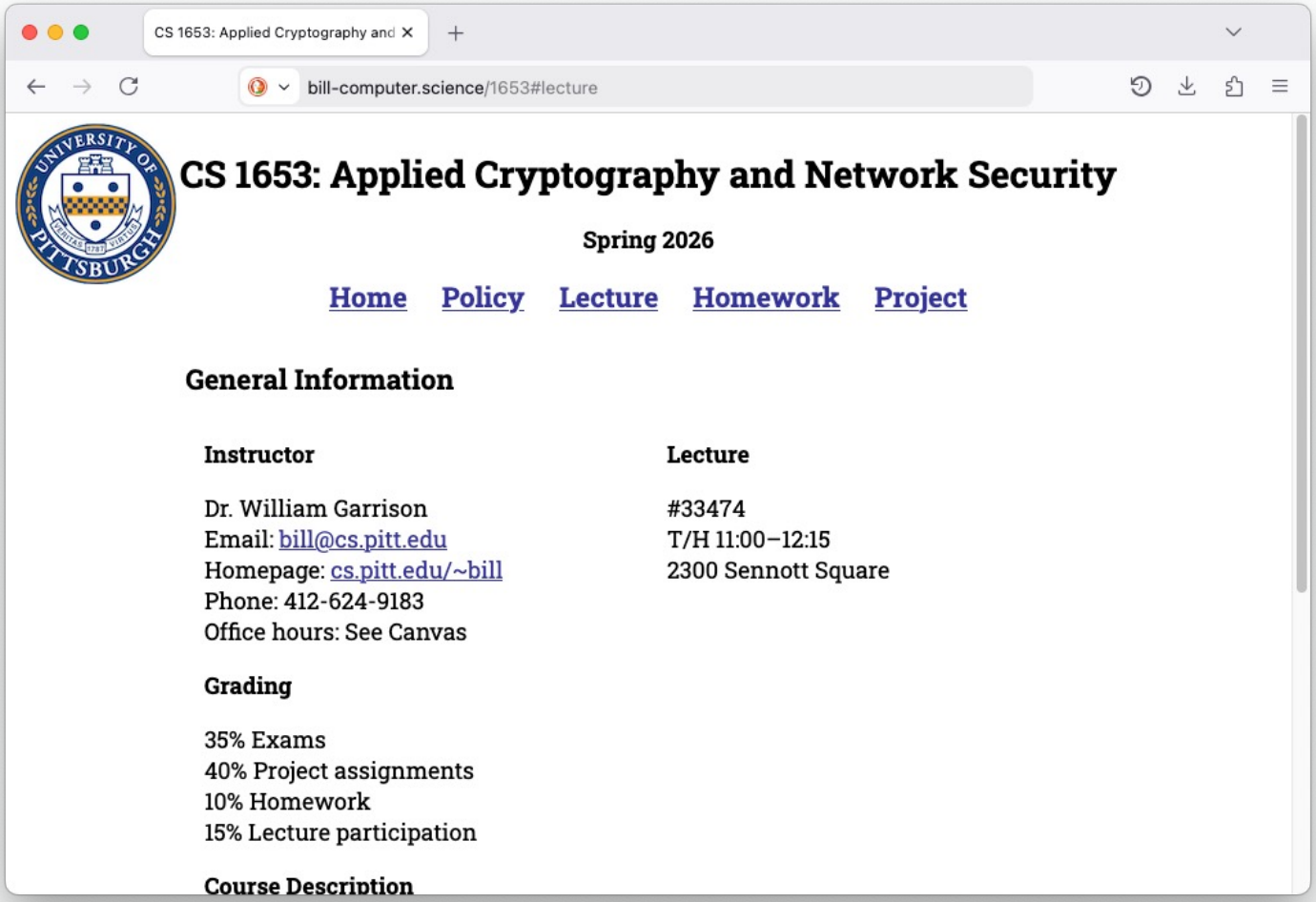
How will my final project grade be calculated?

Description	*Rough Due Date*	*Percent of Grade*
Proposal and requirements	Late Jan	15%
Prototype implementation	Mid Feb	20%
Security features I	Early Mar	25%
Security features II	Early Apr	25%
Attack and defend	Finals week	15%

Important notes:

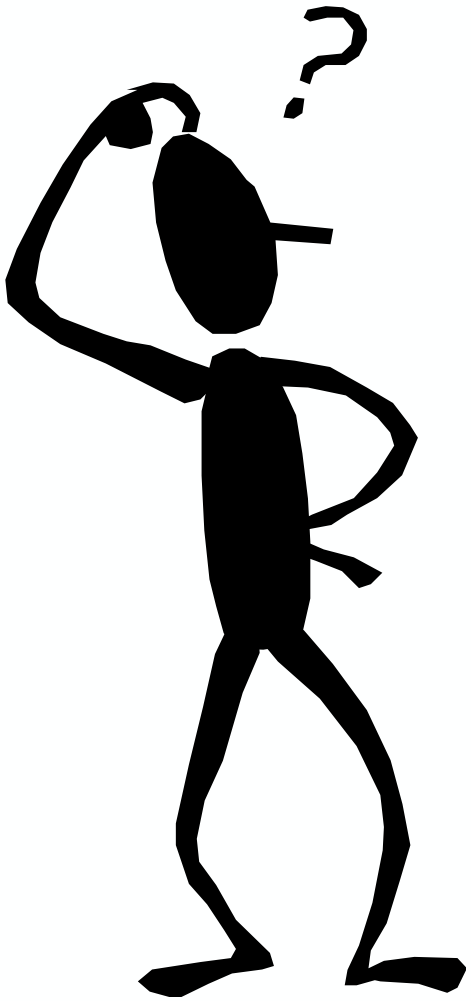
- Choose your group carefully
  - You cannot "quit" your group part way through the semester
  - But you can (and must!) report on your division of labor
- This project is progressive in nature
  - That is, later phases build on earlier phases
  - Working hard early will pay dividends later
  - What if your early phases don't work out so well?

Check the website!



Class Introductions

Why study computer security?



The good ol' days

Security was much less complicated years ago…

Early desktops had

- A single user
- A single address space
- No permissions
- No network
- Limited data-processing abilities

Leading threat: Computer viruses

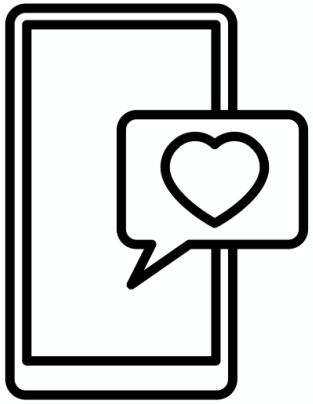
Boot-sector viruses

Executable viruses

Both threats could be controlled reasonably well by using anti-virus software, and basic “software hygiene”

The times, they are a-changin'

The Internet



Mobile/Smartphones

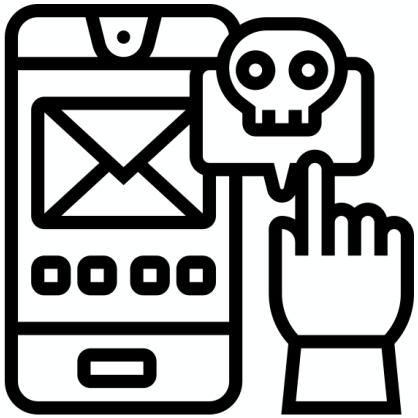
Changes introduced

- Constant data exchange
  - Email, chat, web, social media…
- Machines no longer isolated
- Looooooong uptimes
- More and more valuable data

Results

- Faster malware propagation
  - Email: Days to weeks; Active worms: Minutes
- Active attacks against end-user hosts
- Spyware, phishing, in-browser malware

The times, they are a-changin'



Software/Data Complexity

Changes introduced

- Data formats more complex
- Software functionality bloating
- Boundary between data and executable content is ~~blurred~~ gone

Results

- "Data hygiene" not as easy
- Plethora of new vulnerabilities
- Vulnerabilities more easily exploited

The times, they are a-changin'

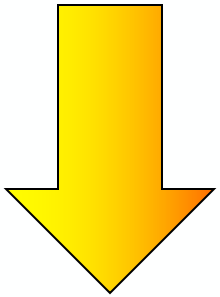
Attacker Motivation

Old days

- Attacks on end hosts not really valuable
- Get "fame" for viruses, etc.

Today

- Everything is online!
- Today's computers have tons of sensors, are always on us
- Attackers can benefit financially from viruses, worms, and compromised hosts



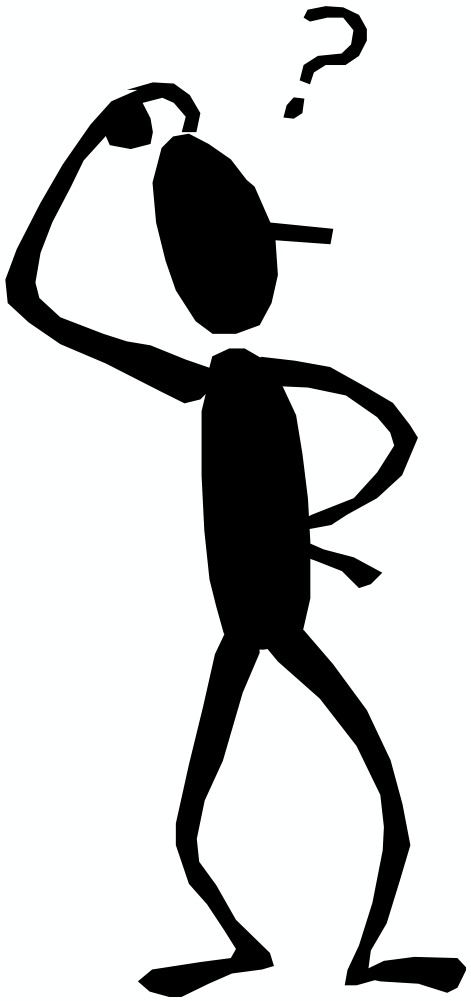
Results

- Spam and botnets
- Organized online crime

Studying security is more important now than ever before

Security is a challenging problem

What is computer security?



Computer security is typically defined with respect to three types of properties

How do I ensure that my secrets remain secret?

Confidentiality

Can I trust the services that I use?

Integrity

Availability

Am I able to do what I need to do?

Confidentiality refers to the need to conceal information or resources

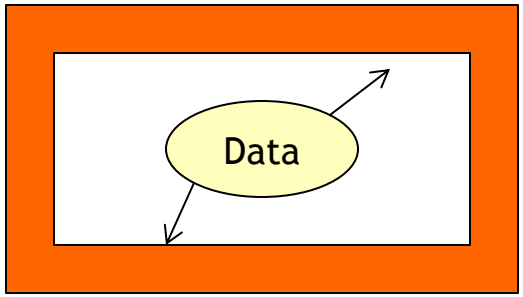
The need for confidentiality arises in both military and civilian information systems

- Military: Classified information processing and intelligence
- Civilian: Proprietary data, sensitive records, etc.

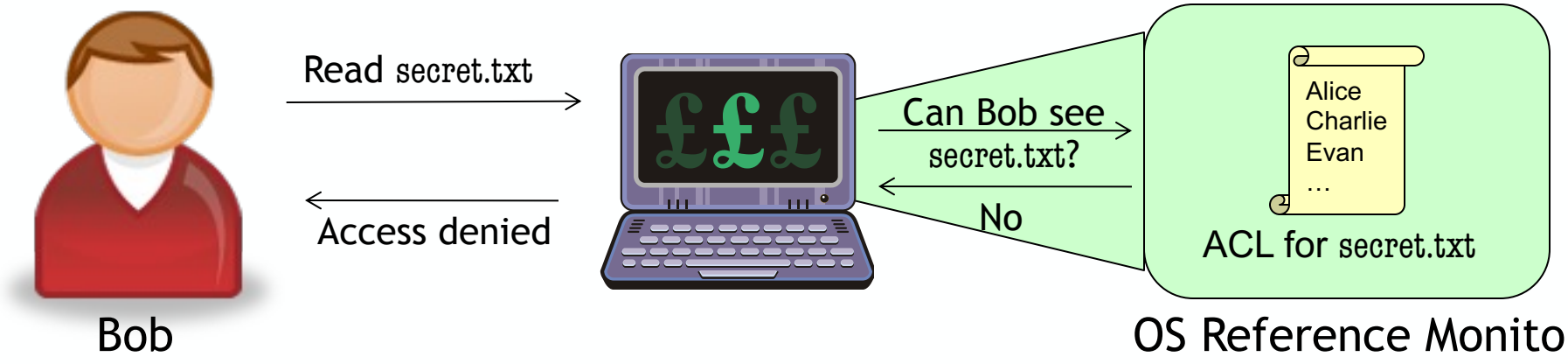
There are many flavors of confidentiality. For example:

- Data: "No one should know my bank account balance"
- Existence: "No one should know that I shop at XYZ.com"
- Configuration: "No one should know what software I run"

The precise notion of confidentiality used in a system depends on the environment in which the system will be used



Access control systems are designed to preserve data confidentiality



Note: We must trust the OS kernel in order for the reference monitor approach to work.

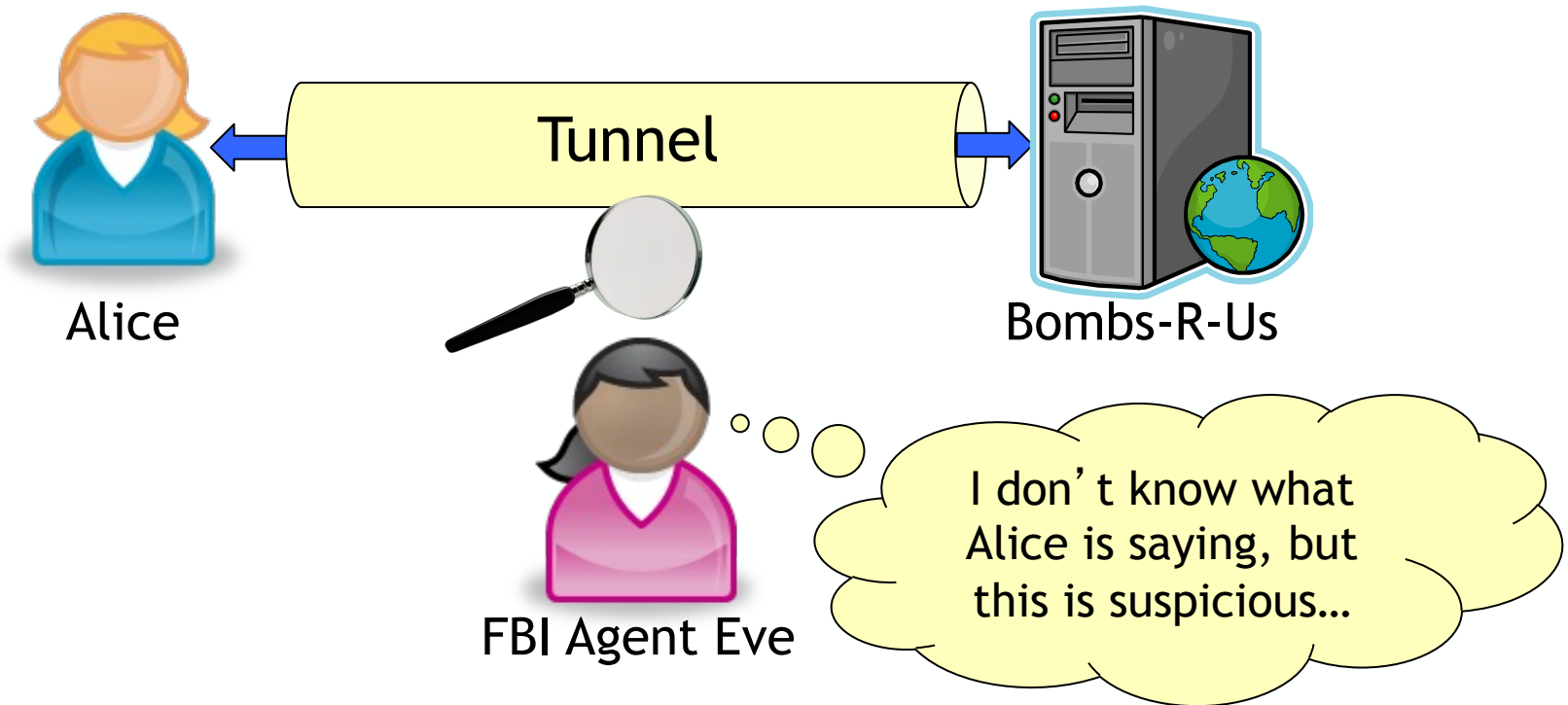
Can you think of a solution that avoids this trust?

- Cryptographic protection!

Does cryptography guarantee confidentiality?

While cryptography does provide data confidentiality, it does not always provide existence confidentiality.

Example: SSL



Integrity refers to the trustworthiness of information or resources

Systems are typically concerned with two types of integrity

- Data integrity: "Is my bank account balance correct"
- Origin integrity: "Was this application really written by Microsoft?"

Integrity mechanisms can either prevent or detect violations

Pro: Keep system consistent

Con: Difficult to design!

- "Easy" to keep bad guys out…
- But how do we identify malicious insiders?

Pro: Easier to design in some cases

- File checksums
- Transaction consistency checks
- etc.

Con: Causes of violations often remain unknown…

Availability refers to the ability to use information or resources

Systems are usually designed with the expectation of certain patterns of usage

- Electricity demand is higher during the workday
- Roads are more utilized during commute hours
- Web sites experience peak traffic during certain hours
- …

By violating these assumptions, the security properties of the system can be altered!

- ***Example:*** Poorly-protected backup servers

Interesting: Detecting malicious availability violations is non-trivial

- e.g., Botnet-based DDoS attack or "Slashdotted" article?

Discussion

Which of the CIA properties do you feel is most important? Why? Are there other properties that are as important or more important than these?

If security is characterized by the CIA properties, how do we characterize insecurity?

A threat is a potential violation of security; i.e., a threat is something that you want to prevent from happening.

Note: A violation need not occur in order for a threat to exist!

Threats are sometimes broadly classified into four categories:

- Disclosure: Information leakage
- Deception: Acceptance of false information
- Disruption: Interruption or prevention of correct operation
- Usurpation: Unauthorized control of some part of a system

These four classes encompass many, many threats…

A few examples…

Snooping: Unauthorized interception of information

- Category: Disclosure
- Defenses: Confidentiality mechanisms

Modification or Alteration: Unauthorized change of information

- Category: Deception, Disruption, and/or Usurpation
- Defenses: Integrity mechanisms

Spoofing: Impersonation of one entity by another

- Category: Deception and/or Usurpation
- Defenses: Integrity mechanisms (e.g., crypto or authentication services)

Threats simply define the types of insecurity that we are concerned with…

Vulnerabilities are situations or conditions that allow a threat to be realized.

For example:

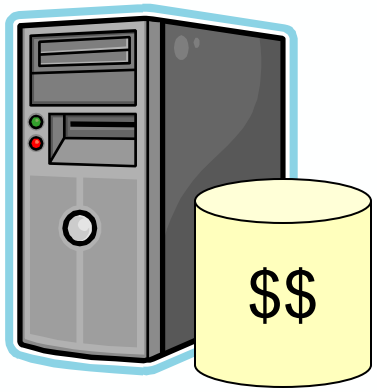
- Incorrect assumptions about operating environment
- Implementation bugs
- Backdoors
- Incompletely specified policies (e.g., firewalls)
- Corrupted hardware
- …

There are many online databases, such as CERT/CC, that list vulnerabilities in widely-used software packages

An attack is the exploitation of a vulnerability to realize a threat.

An Example: Bob's Bank

Bob's Bank runs an online banking site that allows its users to check their account balances and pay bills online. Bob's Bank wants to make sure that clients can only see the data in their own account. Due to financial woes, the banking portal is run using an out-of-date web server.



Threat: Seeing another user's bank records

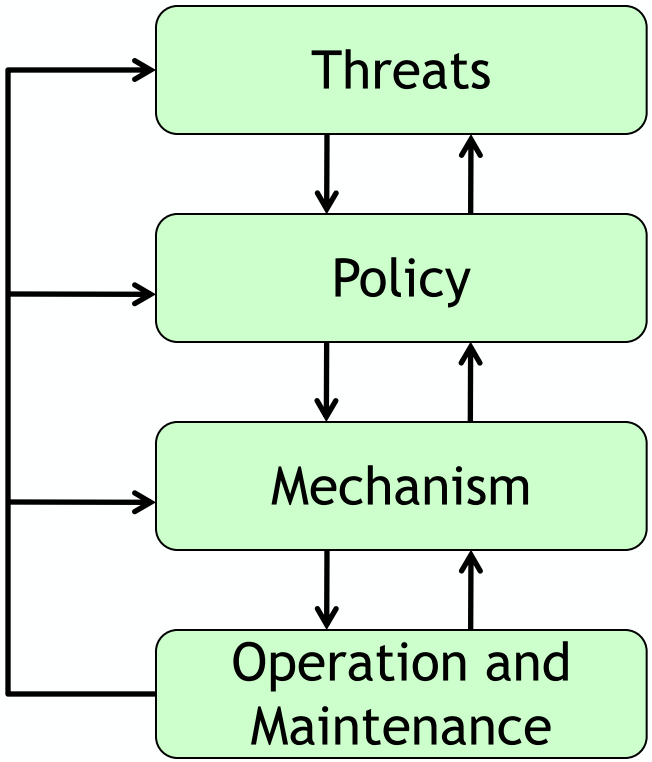
Vulnerability: Unsafe string operations in C code of server

Attack: A buffer overflow that gives the adversary root-level access to the system

So what?

Security is <u>not</u> an absolute property!

Security is a process!



Steps:

1. Identify threats for the domain of interest

2. Define policies to protect against these threats
   - High-level organizational policies
   - Low-level logical policies
   - Everything in between!

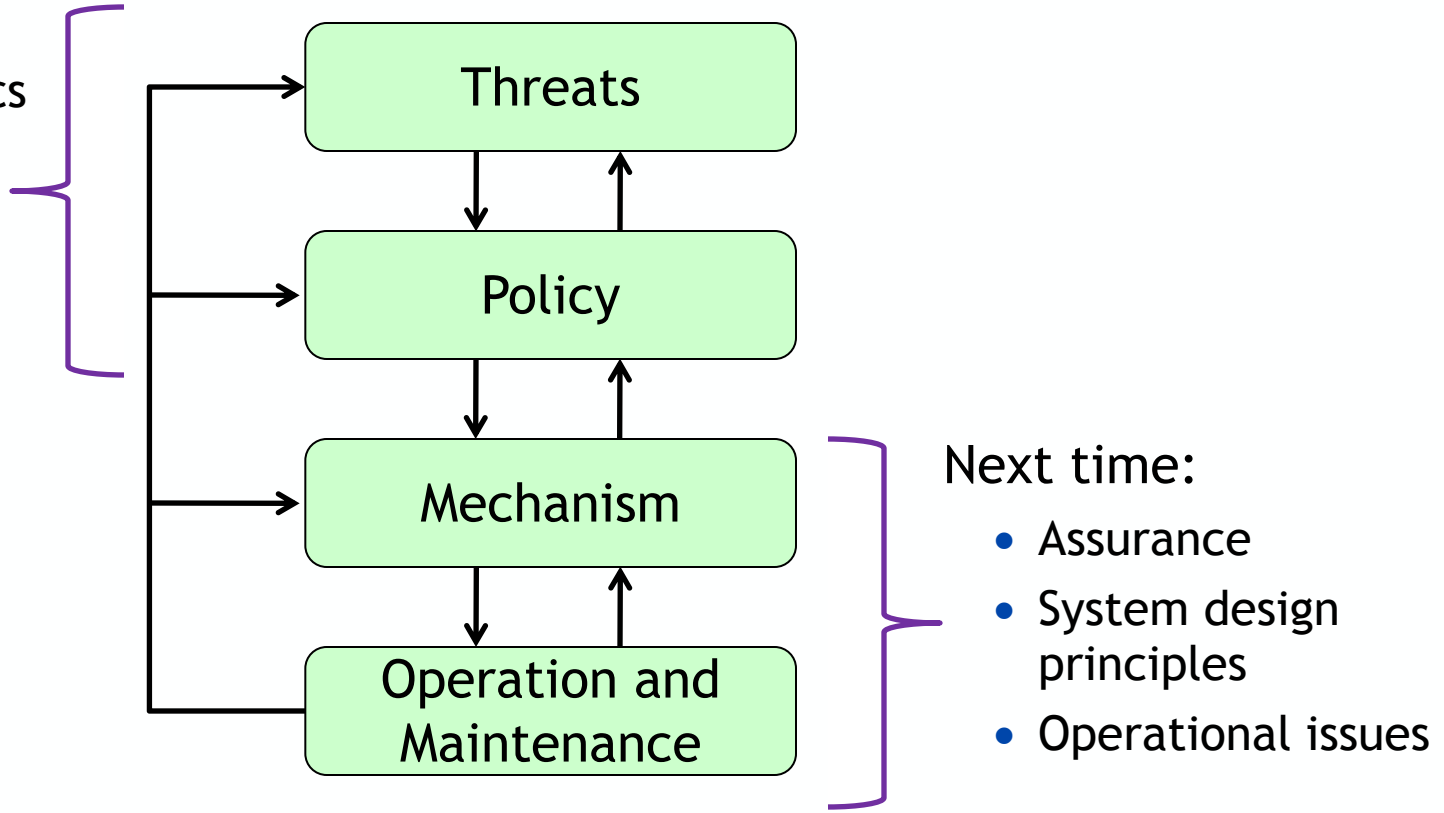
3. Develop mechanisms to enforce these policies

4. Wash, rinse, repeat

Recap and Preview

Today:

- Security basics
- Terminology
- What is security?



Next time:

- Assurance
- System design principles
- Operational issues

Onward: Details, details, details…

Discussion

What are your opinions regarding online vulnerability databases? Do they represent a helpful tool to the community, or simply provide attackers with an easy means of exploiting systems? What do you feel should be the “protocol” used with regard to newly discovered vulnerabilities?