Lecture 1: Introduction and Overview

January 4, 2011

Goals of the Course

- What can security decide, and what can it not decide?
- Policy models: what can systems and people do, and what can they not do?
- Information flow: how can information move around a system?

Confidentiality

- What it is
  - Concealing information, resources
  - May hide attributes (including existence) of data as well as content
  - May hide resources to keep others from using them
- How to do this
  - Cryptography
  - File access controls
  - Other access controls (*e.g.*, firewalls)

Confidentiality Example

Example: protecting a tax return on a PC

- Tax return is enciphered, so it cannot be read directly
- If owner has the cryptographic key, she can read it by deciphering the tax return
- So can anyone who has that cryptographic key
- If someone can rig the decryption program to send them the decryption key, that also compromises the tax return

Integrity

- What it is
  - Has the data been altered without authorization, or in unauthorized ways?
  - Is the data credible (trustworthy)
- Types of integrity
  - Data integrity (contents)
  - Origin integrity (source, *authentication*)
- Example: database transaction
  - If interrupted, may leave database in an inconsistent state
- Much harder to quantify than confidentiality

Integrity Example

Example: government leaking

- Newspaper prints information leaked to it from White House, attributing it to wrong source
- Data integrity: preserved, as information printed as received
- Origin integrity: corrupt, as source is mis-attributed
- Data trustworthiness: depends ...

Availability

- What it is
  - Ability to use information or resource desired
  - Key part of reliability as well as security
- Most models based on statistics, so assume a predicted pattern of use overall
  - Attackers change the pattern of use, so the model no longer applies
  - Mechanisms providing availability not designed for changed environment—and fail

Availability Example

Example: compromising a bank

- Anne controls secondary server that supplies bank balances for credit cards
- Anne blocks access to primary server, so requests sent to secondary server
- Anne supplies any balance she likes, ensuring none of her purchases is declined

Threats

A *potential* violation of security

- Actions that could cause it to occur are *attacks*
- Four classes of threats
  - Disclosure: unauthorized access to information
  - Deception: acceptance of false data
  - Disruption: interruption or prevention of correct operation
  - Usurpation: unauthorized control of some part of a system

Common Threats and Their Classes

- Snooping, passive wiretapping: disclosure
- Modification, active wiretapping: deception, disruption, usurpation
- Masquerading, spoofing: deception, usurpation
  - Delegation: a legitimate form of masquerading
- Repudiation of origin: deception
- Denial of receipt: deception
- Delay, denial of service: usurpation, may support deception

Policy and Mechanism

- Policy says what is, and is not, allowed
  - This defines “security” for the site/system/*etc.*
- Mechanisms enforce the policy
- Policy composition: if they conflict, the discrepancies may create security vulnerabilities

Expressions

- Policy expression
  - Natural language: usually imprecise, but easy to understand
  - Mathematics: usually precise but hard to understand
  - Policy languages: look like some form of programming language and try to balance precision with ease of understanding
- Mechanisms
  - Technical: controls in the computer enforce the policy
    - Require the user supply a password to authenticate herself before using the computer
  - Procedural: controls outside the system enforce the policy
    - Require the firing of someone who beings in a disk containing a game program obtained from an untrusted source

Goals of Security

- Prevention: the attack will fail
- Detection: the attack will be identified
  - Appropriate when the attack cannot be prevented
  - Appropriate to check effectiveness of preventative measures
- Recovery: return system to correct functioning during (or after) attack
  - First form: stop attack, assess and repair damage from that attack
  - Second form: continue to function correctly during the attack ("attack tolerant")

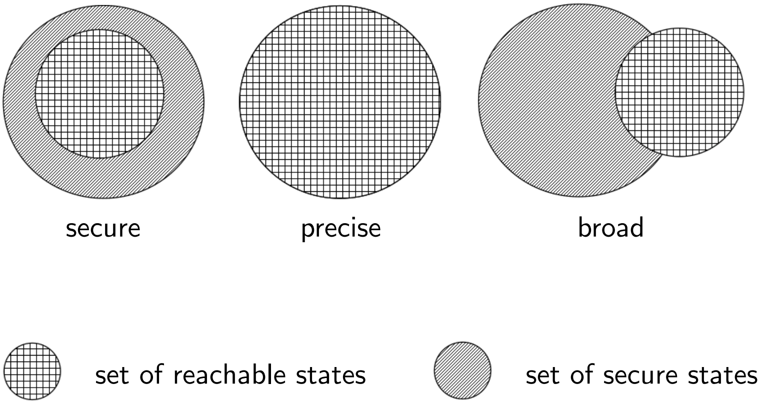
Trust and Assumptions

- Underlie *all* aspects of security
- What happens if assumptions incorrect?
  - Key needed to open a door lock $\Rightarrow$ lock cannot be picked
  - Good lock picker can pick a lock
  - Consequent false, therefore antecedent (assumption) false

Example Assumptions

- Assumptions policies make
  - Unambiguously partition system states
  - Correctly capture security requirements
- Assumptions mechanisms make
  - Correctly implemented
  - Support tools (libraries, operating system services, *etc.*) work correctly
  - Installed, administered correctly
  - Union of mechanisms implements all aspects of security policy

Types of Mechanisms



Assurance

How much to trust a system, based on evidence obtained from specification, design, implementation, and operation

- Assurance based on assurance evidence gathered during analysis
- Assurance evidence provides a basis for assessing what one must trust in order to believe system is secure

Assurance does not guarantee correctness or security

Example: Aspirin

- Aspirin sold in safety-sealed container
  - Testing, certification of drugs by FDA
  - Manufacturing standards of company and precautions it takes to prevent contamination
- In 980s, technologies above considered sufficient to provide assurance evidence that aspirin not contaminated
  - Then someone contaminated the aspirin after manufacture but before consumer purchase
- Evidence no longer deemed sufficient sufficient
  - Safety seal on bottle added in 1980s to prevent introduction of harmful chemicals as happened above
- Assurance evidence then considered sufficient

Phases

- Specification: statement of desired functioning of system
    - Need to meet requirements (*requirements assurance*)
    - Specification may be formal or informal
    - Statement of *functionality*, not assurance
- Design: translates specification into components that will implement the specification
    - Need to prove design satisfies specification (*design assurance*)
    - Design can be given in many ways (mathematics, pseudocode, *etc.*)
    - Typically, system treated as layers of abstraction, and then components of layers, and interfaces between layers, designed

Phases

Implementation: creates a system that satisfies the design

- Problem is to prove implementation satisfies design (and, by transitivity, specification)
- Approach
  - Specify preconditions, postconditions for each line of code
  - Build function preconditions, postconditions from those of lines of code
  - Derive preconditions, postconditions for programs from these
  - Verify all preconditions hold and all postconditions satisfy design

Phases

Problems with mathematical implementation assurance

- Problem is to prove implementation satisfies design (and, by transitivity, specification)
- *Very* difficult and time-consuming to do mathematically
  - Complexity of programs *and environments* makes any preconditions subtle
  - Assumption is that implementation is correctly compiled, linked, loaded, and libraries and supporting infrastructure is correct
  - If preconditions require specific forms or values in input, programs must check that the input conforms to the preconditions

Phases

Problems with mathematical implementation assurance

- Problem is to prove implementation satisfies design (and, by transitivity, specification)
- *Very* difficult and time-consuming to do mathematically
  - Complexity of programs *and environments* makes any preconditions subtle
  - Assumption is that implementation is correctly compiled, linked, loaded, and libraries and supporting infrastructure is correct
  - If preconditions require specific forms or values in input, programs must check that the input conforms to the preconditions

Operational Issues: Cost-Benefit Analysis

- Balance benefit of security against its cost
- Analysis rarely clear-cut as benefits overlap and calculating cost, benefits involves judgement and guesswork
- Benefits may overlap, complicating the calculations

Operational Issues: Risk Analysis

- What is the probability that the threat will materialize?
- Risk is a function of environment, and changes with time
    - Computer system not connected to Internet has one set of risks, generally local
    - Add a network connection and the risks change
- "Analysis paralysis", where risk analysis made but not acted upon

Operational Issues: Laws and Customs

- Constrain availability, use of technology, procedures
  - Country X makes reading another's email illegal
  - Attackers break in by compromising mail system
  - Sysadmins gathering evidence look in mailbox—now they are criminals too!
- Systems in multiple jurisdictions complicate how they are (can be) used
  - Country A requires encryption keys to be registered with police
  - A multinational corporation has offices in Country A
  - Key and message management messy!
- That which is legal may be completely unacceptable

Human Issues: Organizational Problems

- Security a supportive service (no direct benefit, especially not financial)
- Who is responsible for security—and do they have the power to implement needed controls?
    - Often lack of people knowledgeable in security
    - Security considered something "additional" to other work rather than job in itself
    - Lack of resources for developing, implementing, acquiring security mechanisms

Human Issues: People Problems

People at the heart of every security system

- Security controls won't block unauthorized user who knows your login and password
- People trusted with access (*insiders*) who betray that trust difficult to thwart
  - Just look at the Wikileaks messages ...
  - Untrained people also a threat
- Social engineering