600.436: High-Assurance Systems

David M. Chizmadia, *Promia, Inc.*, dchizmadia@srl.cs.jhu.edu
Jonathan S. Shapiro, *Johns Hopkins University*, shap@srl.cs.jhu.edu

Course website: http://srl.cs.jhu.edu/courses/cs436.html

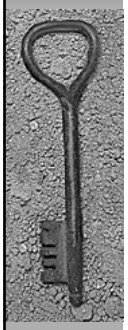
Time/Place: M 4-5, Tu 3-5, NEB 12

Required Texts:

- Anderson: *Security Engineering*
- DeMarco & Lister: *Peopleware* (2nd Ed.)
- ISO: *Common Criteria, Common Evaluation Methodology*
- Papers as distributed in class

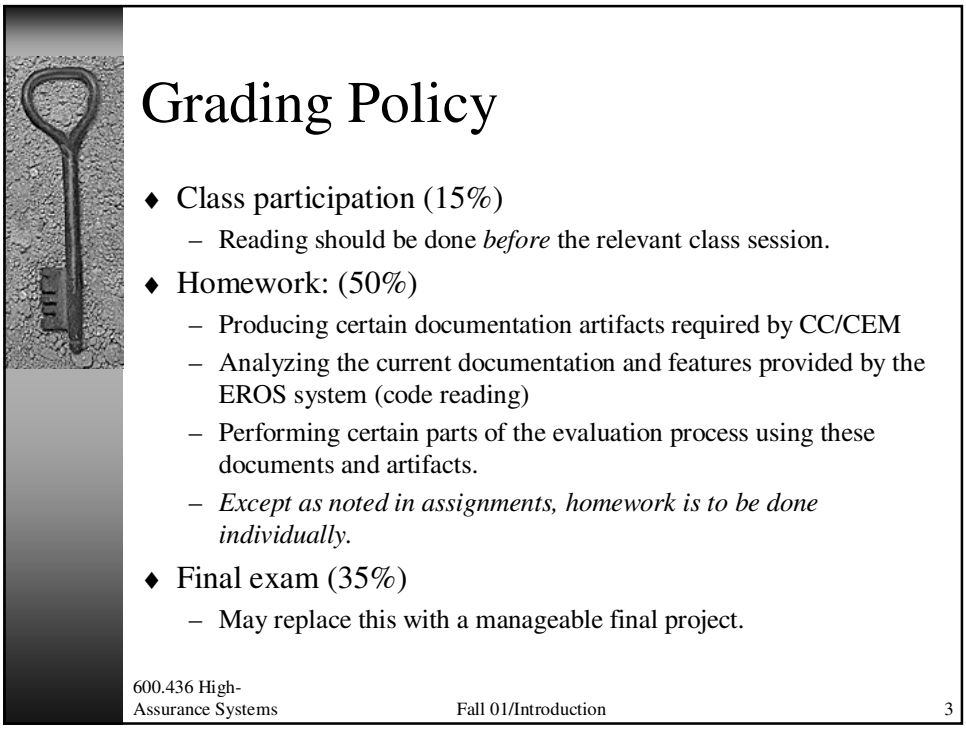
Recommended (Strongly)

- Schneier: *Secrets & Lies*



Course Plan

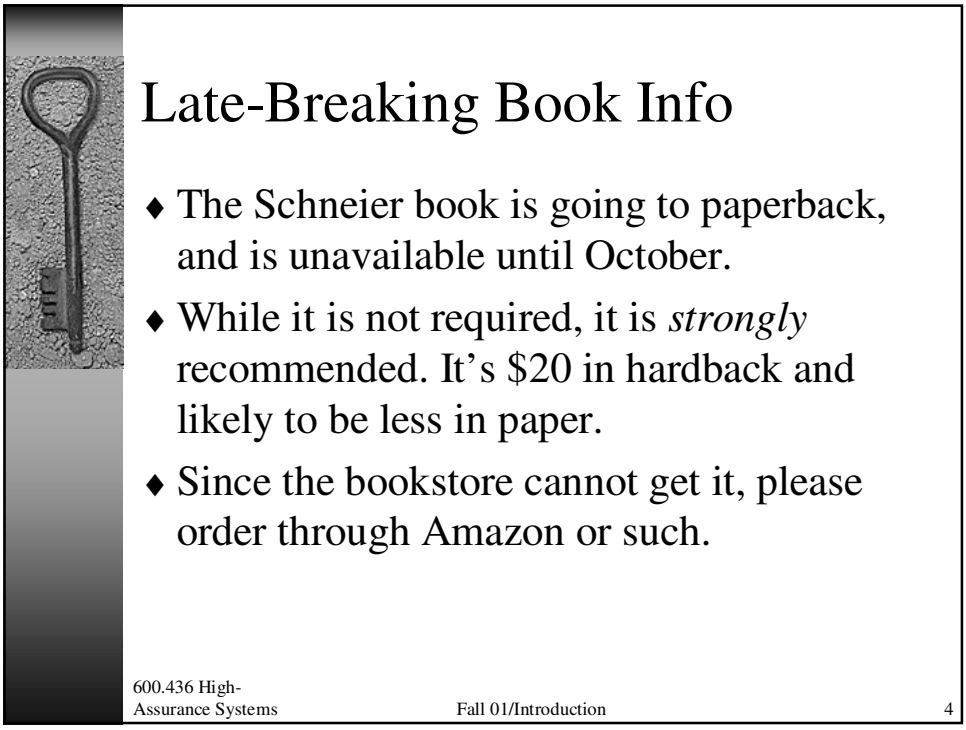
- Weeks 1-3: Groundwork for Assurance
- Weeks 4-12:
  - Assurance Track: Learning how assurance works (David)
  - Practice: Learning how EROS (supposedly) works (Jonathan)
- Weeks 13-15: Wrap-up and Integration



Grading Policy

- Class participation (15%)
  - Reading should be done *before* the relevant class session.
- Homework: (50%)
  - Producing certain documentation artifacts required by CC/CEM
  - Analyzing the current documentation and features provided by the EROS system (code reading)
  - Performing certain parts of the evaluation process using these documents and artifacts.
  - *Except as noted in assignments, homework is to be done individually.*
- Final exam (35%)
  - May replace this with a manageable final project.

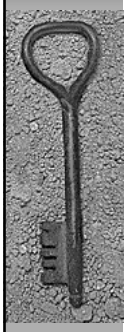
600.436 High-Assurance Systems Fall 01/Introduction 3



Late-Breaking Book Info

- The Schneier book is going to paperback, and is unavailable until October.
- While it is not required, it is *strongly* recommended. It's $20 in hardback and likely to be less in paper.
- Since the bookstore cannot get it, please order through Amazon or such.

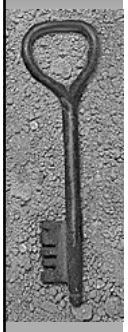
600.436 High-Assurance Systems Fall 01/Introduction 4



Caveats

- This is a new, experimental course
  - We literally *are* making it up as we go along!
- David:
  - Has done evaluations in two countries under three standards
  - Project lead and primary author: *Guide to writing the Security Features Users Guide* (part of the rainbow series)
  - Technical Editor for the Federal Criteria
  - Current member of CORBA Security standards committee
- Jonathan:
  - Has designed and built an allegedly secure system: EROS
  - Has shipped three "set new bar" products grossing > $250M (lots of production reality to check against)
  - These delivered on time
- We both:
  - Detest process *qua* process
  - Understand some of what works in practice

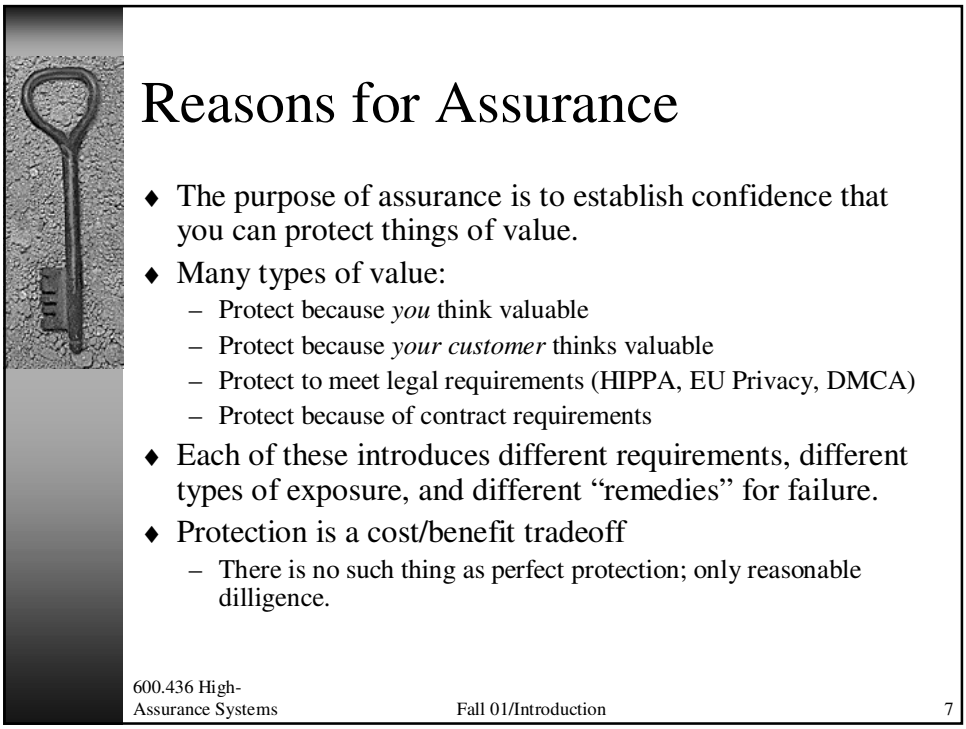
600.436 High-Assurance Systems Fall 01/Introduction 5



Informal Definition: "Assurance"

- Assurance is the process by which one obtains confidence in the security that a software system will enforce when operated correctly.
- This includes the policies enforced, the degree of confidence in the enforcement, and an assessment of the *appropriateness* of those policies for the context in which the system will be used.

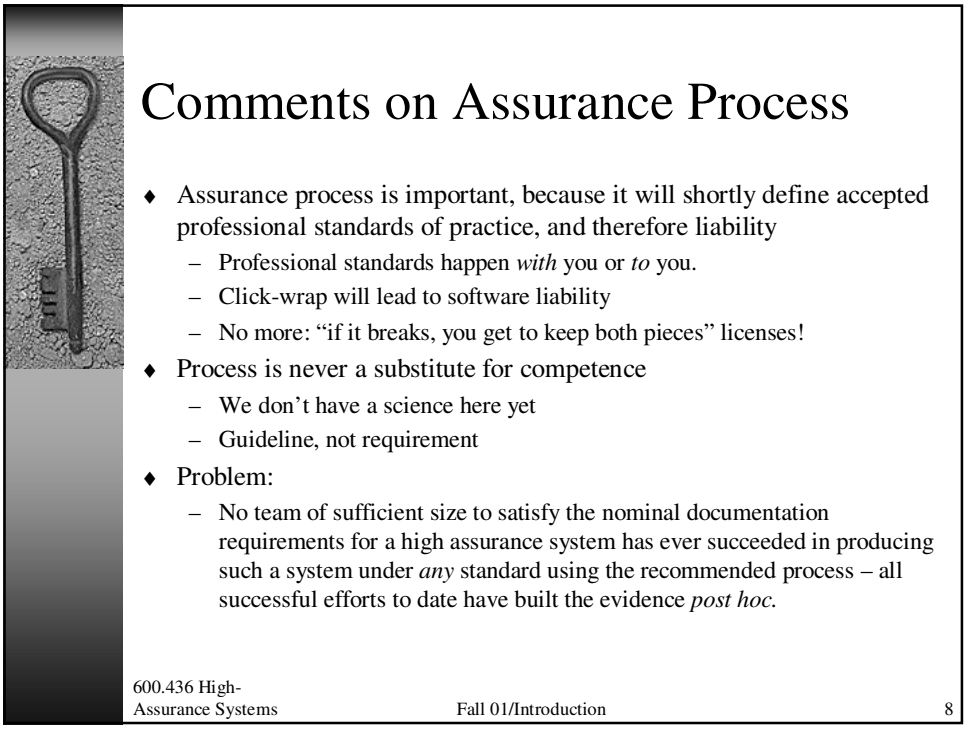
600.436 High-Assurance Systems Fall 01/Introduction 6



Reasons for Assurance

- The purpose of assurance is to establish confidence that you can protect things of value.
- Many types of value:
  - Protect because *you* think valuable
  - Protect because *your customer* thinks valuable
  - Protect to meet legal requirements (HIPPA, EU Privacy, DMCA)
  - Protect because of contract requirements
- Each of these introduces different requirements, different types of exposure, and different "remedies" for failure.
- Protection is a cost/benefit tradeoff
  - There is no such thing as perfect protection; only reasonable dilligence.

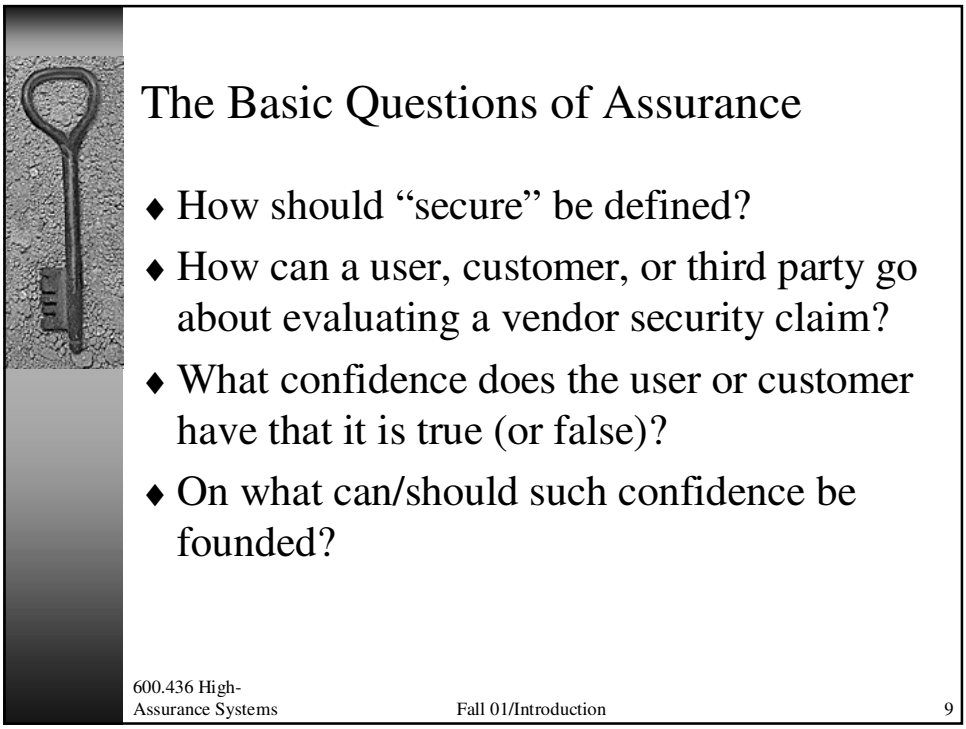
600.436 High-Assurance Systems Fall 01/Introduction 7

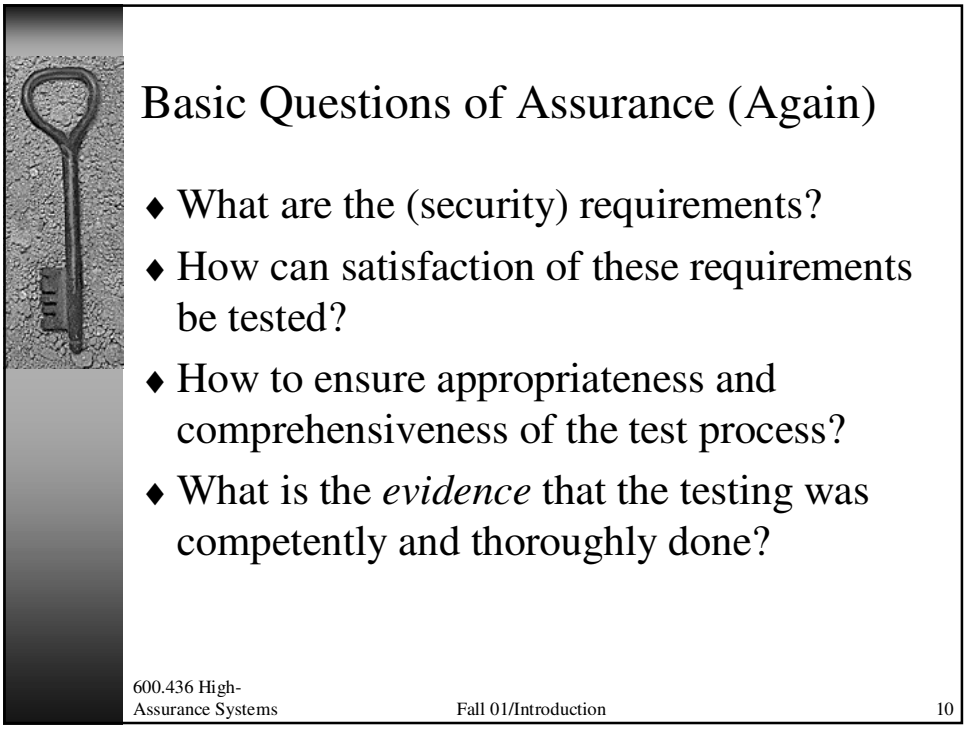


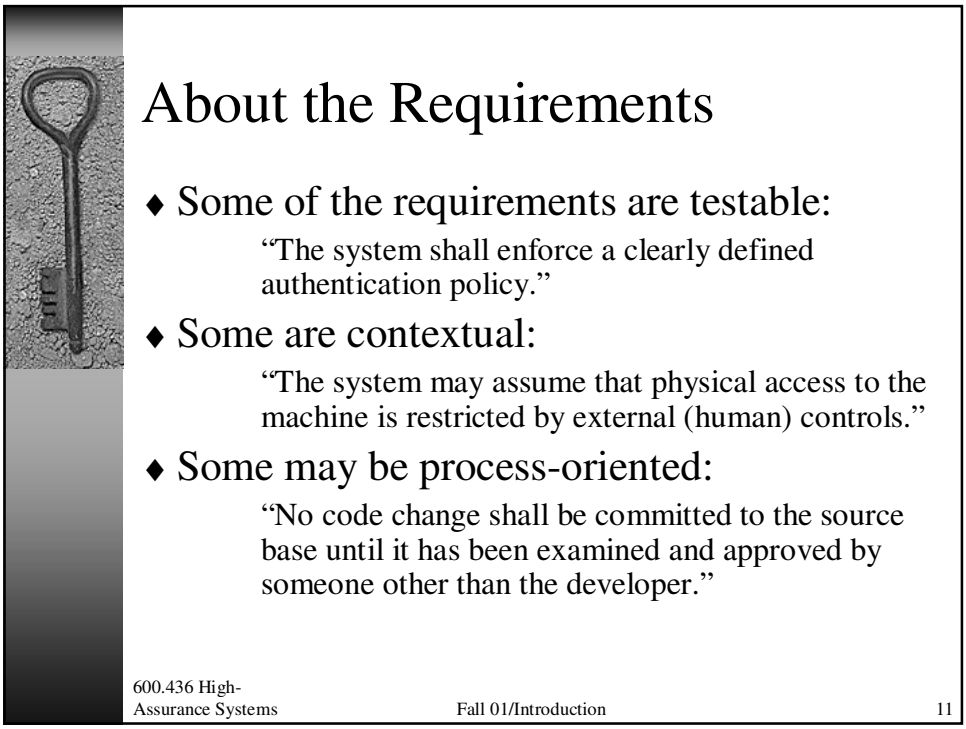
Comments on Assurance Process

- Assurance process is important, because it will shortly define accepted professional standards of practice, and therefore liability
  - Professional standards happen *with* you or *to* you.
  - Click-wrap will lead to software liability
  - No more: "if it breaks, you get to keep both pieces" licenses!
- Process is never a substitute for competence
  - We don't have a science here yet
  - Guideline, not requirement
- Problem:
  - No team of sufficient size to satisfy the nominal documentation requirements for a high assurance system has ever succeeded in producing such a system under *any* standard using the recommended process – all successful efforts to date have built the evidence *post hoc.*

600.436 High-Assurance Systems Fall 01/Introduction 8


The Basic Questions of Assurance
♦ How should "secure" be defined?
♦ How can a user, customer, or third party go about evaluating a vendor security claim?
♦ What confidence does the user or customer have that it is true (or false)?
♦ On what can/should such confidence be founded?
600.436 High-Assurance Systems
Fall 01/Introduction
9


Basic Questions of Assurance (Again)
♦ What are the (security) requirements?
♦ How can satisfaction of these requirements be tested?
♦ How to ensure appropriateness and comprehensiveness of the test process?
♦ What is the *evidence* that the testing was competently and thoroughly done?
600.436 High-Assurance Systems
Fall 01/Introduction
10



About the Requirements

- Some of the requirements are testable:

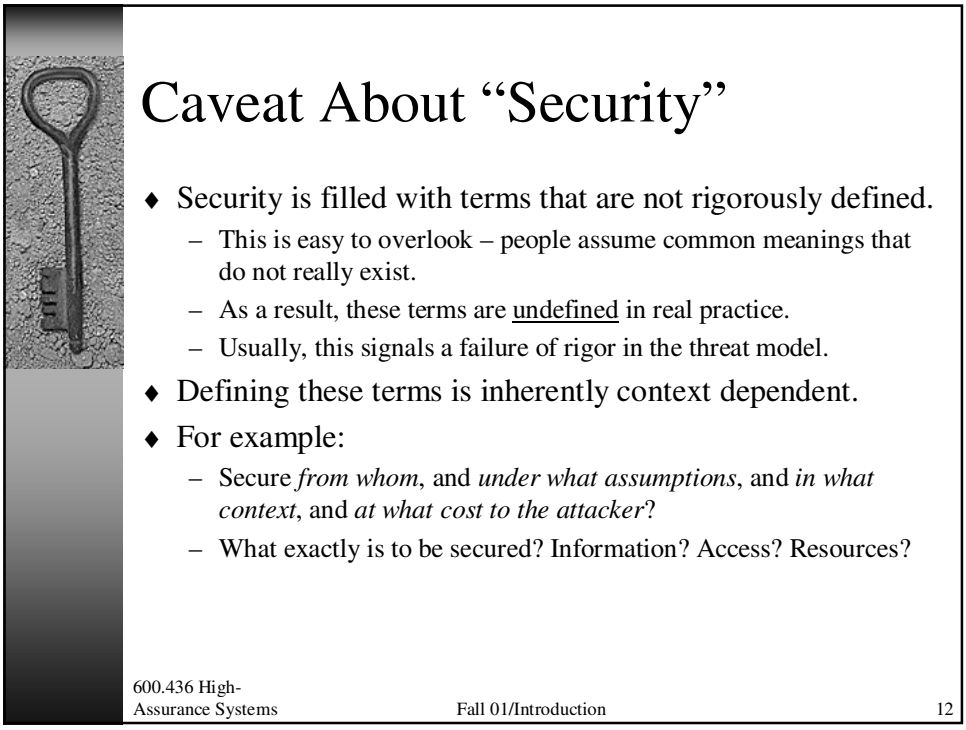
  "The system shall enforce a clearly defined authentication policy."

- Some are contextual:

  "The system may assume that physical access to the machine is restricted by external (human) controls."

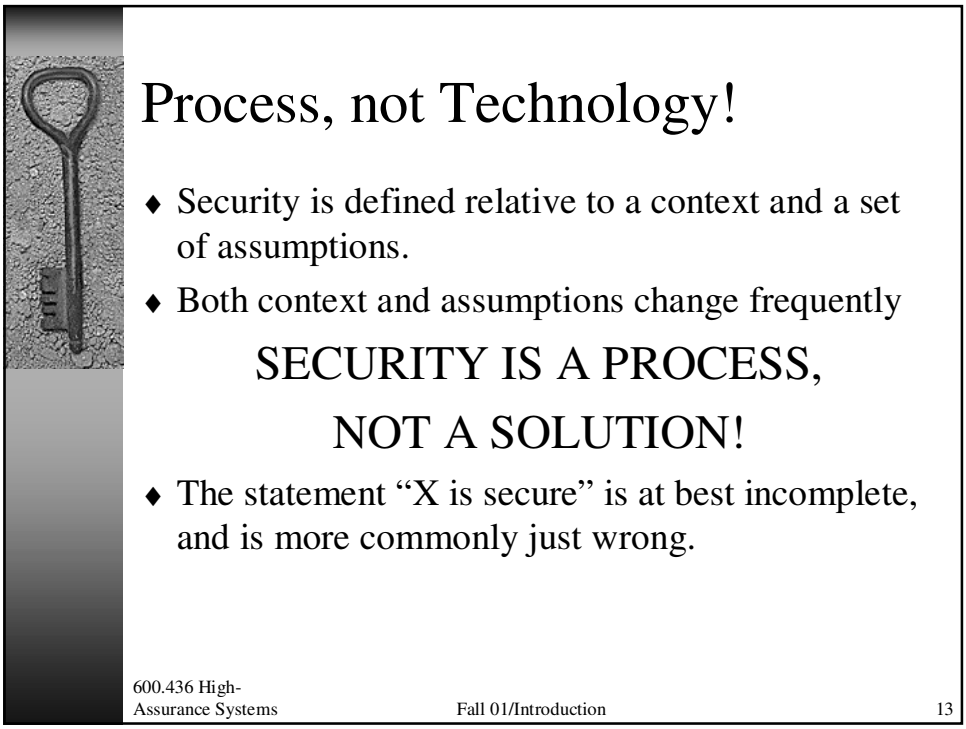
- Some may be process-oriented:

  "No code change shall be committed to the source base until it has been examined and approved by someone other than the developer."



Caveat About "Security"

- Security is filled with terms that are not rigorously defined.
  - This is easy to overlook – people assume common meanings that do not really exist.
  - As a result, these terms are undefined in real practice.
  - Usually, this signals a failure of rigor in the threat model.
- Defining these terms is inherently context dependent.
- For example:
  - Secure *from whom*, and *under what assumptions*, and *in what context*, and *at what cost to the attacker*?
  - What exactly is to be secured? Information? Access? Resources?



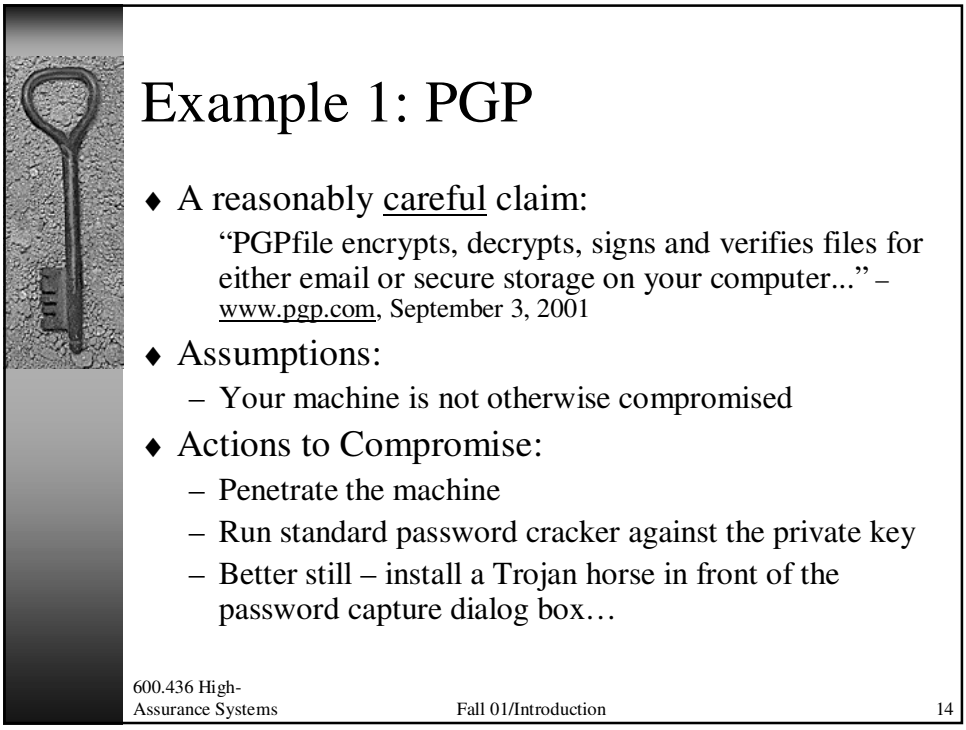
Process, not Technology!

- Security is defined relative to a context and a set of assumptions.
- Both context and assumptions change frequently

SECURITY IS A PROCESS,

NOT A SOLUTION!

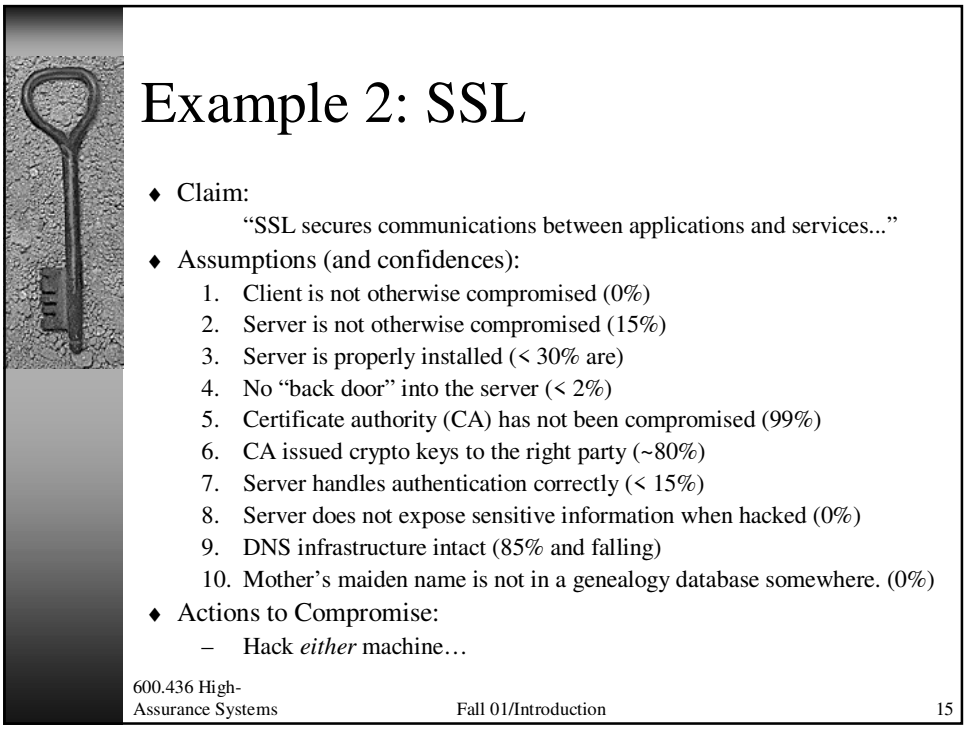
- The statement "X is secure" is at best incomplete, and is more commonly just wrong.



Example 1: PGP

- A reasonably careful claim:

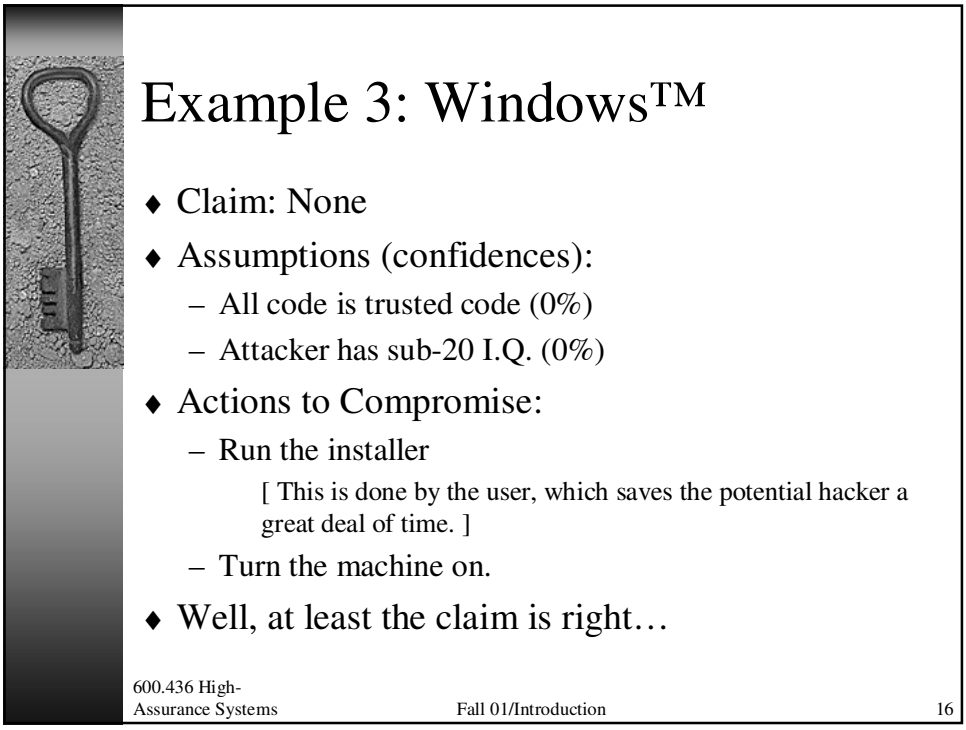
  "PGPfile encrypts, decrypts, signs and verifies files for either email or secure storage on your computer..." – www.pgp.com, September 3, 2001
- Assumptions:
  - Your machine is not otherwise compromised
- Actions to Compromise:
  - Penetrate the machine
  - Run standard password cracker against the private key
  - Better still – install a Trojan horse in front of the password capture dialog box…



Example 2: SSL

- Claim:
  "SSL secures communications between applications and services..."
- Assumptions (and confidences):
  1. Client is not otherwise compromised (0%)
  2. Server is not otherwise compromised (15%)
  3. Server is properly installed (< 30% are)
  4. No "back door" into the server (< 2%)
  5. Certificate authority (CA) has not been compromised (99%)
  6. CA issued crypto keys to the right party (~80%)
  7. Server handles authentication correctly (< 15%)
  8. Server does not expose sensitive information when hacked (0%)
  9. DNS infrastructure intact (85% and falling)
  10. Mother's maiden name is not in a genealogy database somewhere. (0%)
- Actions to Compromise:
  - Hack *either* machine…

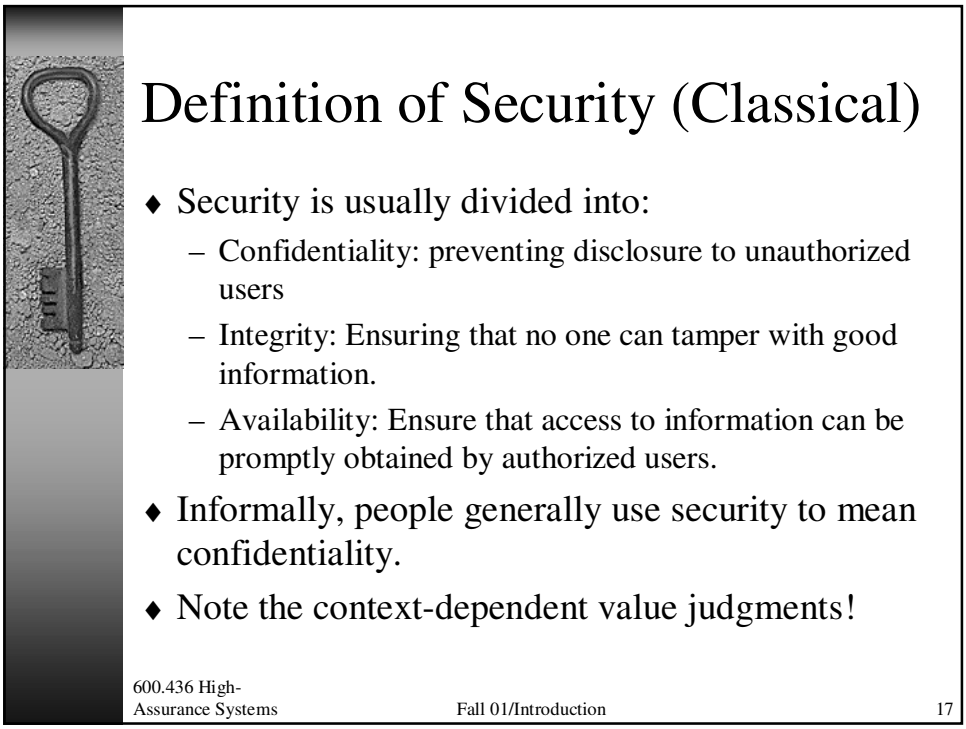
600.436 High-Assurance Systems Fall 01/Introduction 15



Example 3: Windows™

- Claim: None
- Assumptions (confidences):
  - All code is trusted code (0%)
  - Attacker has sub-20 I.Q. (0%)
- Actions to Compromise:
  - Run the installer
    [This is done by the user, which saves the potential hacker a great deal of time.]
  - Turn the machine on.
- Well, at least the claim is right…

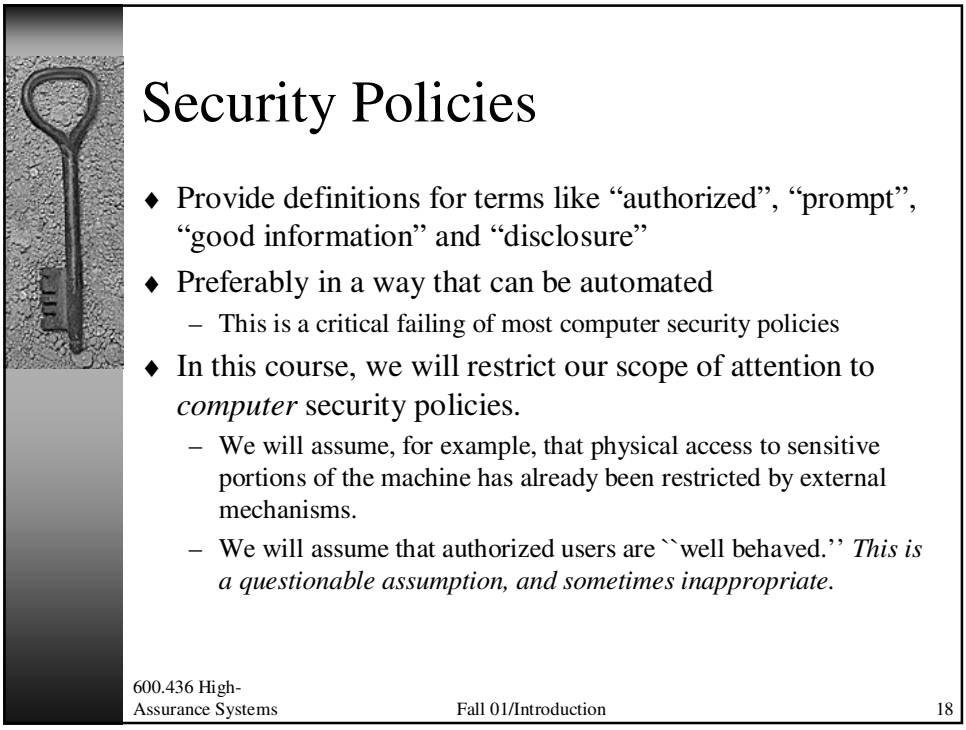
600.436 High-Assurance Systems Fall 01/Introduction 16



Definition of Security (Classical)

- Security is usually divided into:
  - Confidentiality: preventing disclosure to unauthorized users
  - Integrity: Ensuring that no one can tamper with good information.
  - Availability: Ensure that access to information can be promptly obtained by authorized users.
- Informally, people generally use security to mean confidentiality.
- Note the context-dependent value judgments!

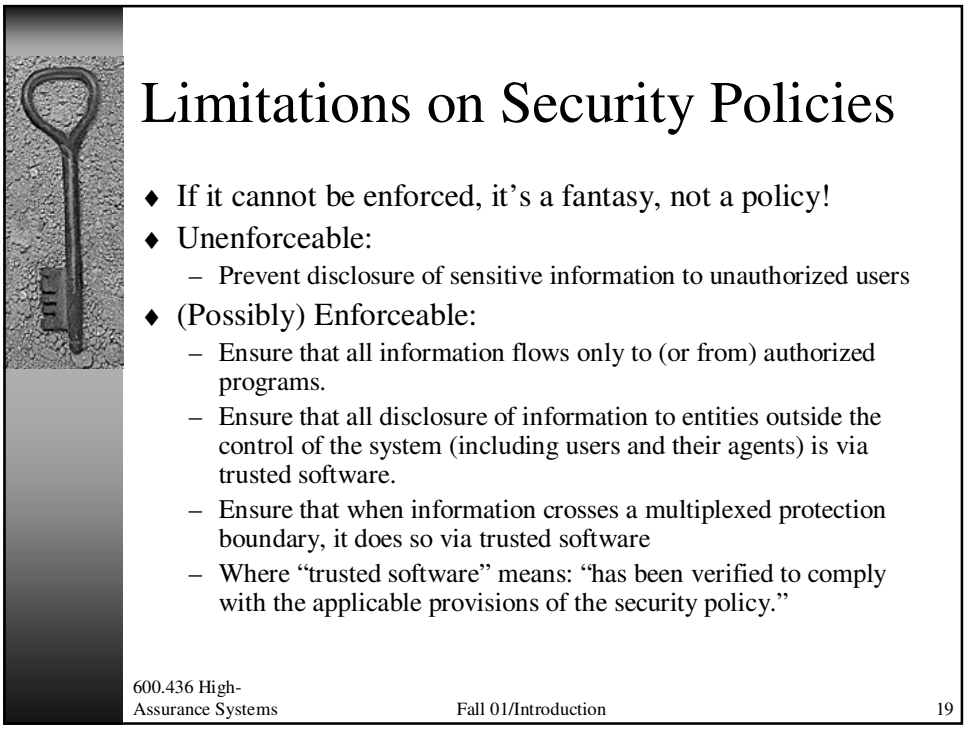
600.436 High-Assurance Systems Fall 01/Introduction 17



Security Policies

- Provide definitions for terms like “authorized”, “prompt”, “good information” and “disclosure”
- Preferably in a way that can be automated
  - This is a critical failing of most computer security policies
- In this course, we will restrict our scope of attention to *computer* security policies.
  - We will assume, for example, that physical access to sensitive portions of the machine has already been restricted by external mechanisms.
  - We will assume that authorized users are ``well behaved.'' *This is a questionable assumption, and sometimes inappropriate.*

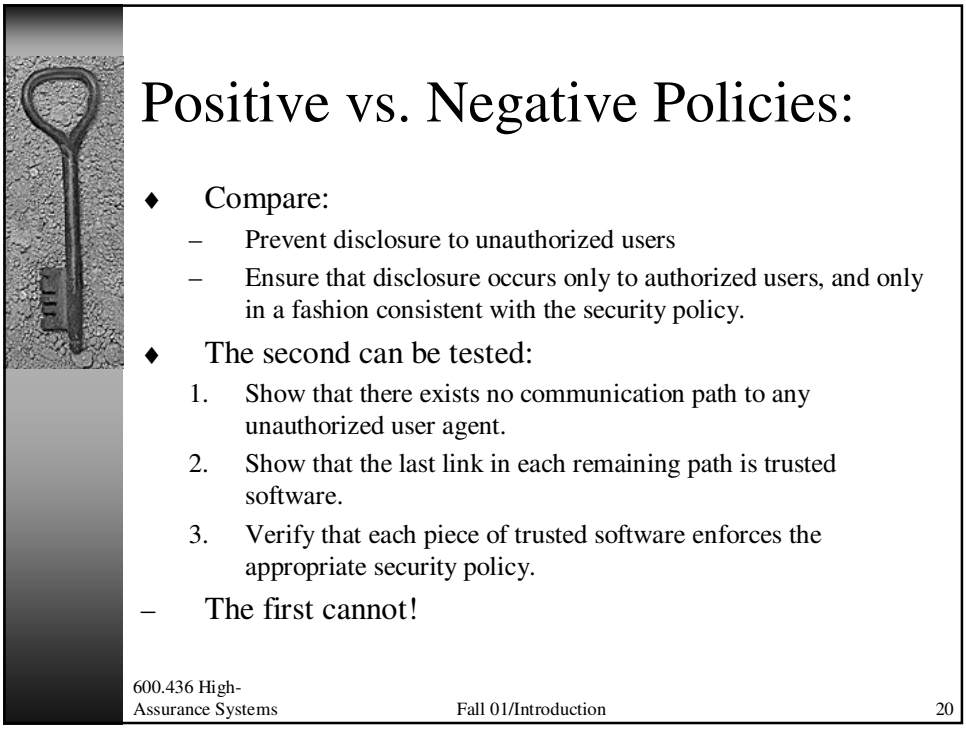
600.436 High-Assurance Systems Fall 01/Introduction 18



Limitations on Security Policies

- If it cannot be enforced, it's a fantasy, not a policy!
- Unenforceable:
  - Prevent disclosure of sensitive information to unauthorized users
- (Possibly) Enforceable:
  - Ensure that all information flows only to (or from) authorized programs.
  - Ensure that all disclosure of information to entities outside the control of the system (including users and their agents) is via trusted software.
  - Ensure that when information crosses a multiplexed protection boundary, it does so via trusted software
  - Where "trusted software" means: "has been verified to comply with the applicable provisions of the security policy."

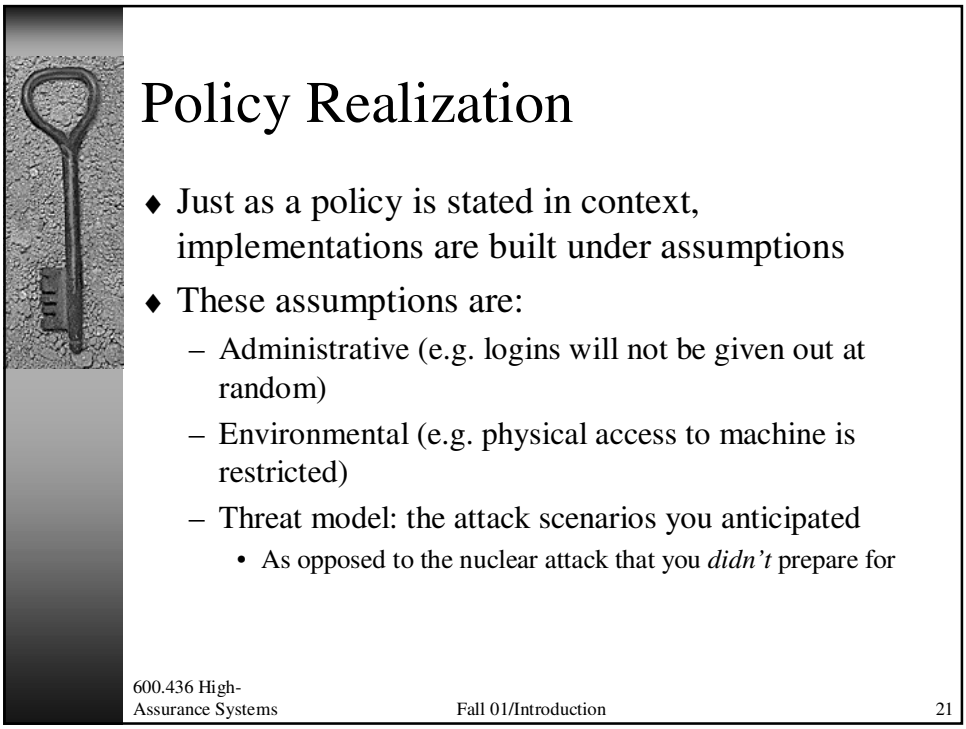
600.436 High-Assurance Systems Fall 01/Introduction 19



Positive vs. Negative Policies:

- Compare:
  - Prevent disclosure to unauthorized users
  - Ensure that disclosure occurs only to authorized users, and only in a fashion consistent with the security policy.
- The second can be tested:
  1. Show that there exists no communication path to any unauthorized user agent.
  2. Show that the last link in each remaining path is trusted software.
  3. Verify that each piece of trusted software enforces the appropriate security policy.
- The first cannot!

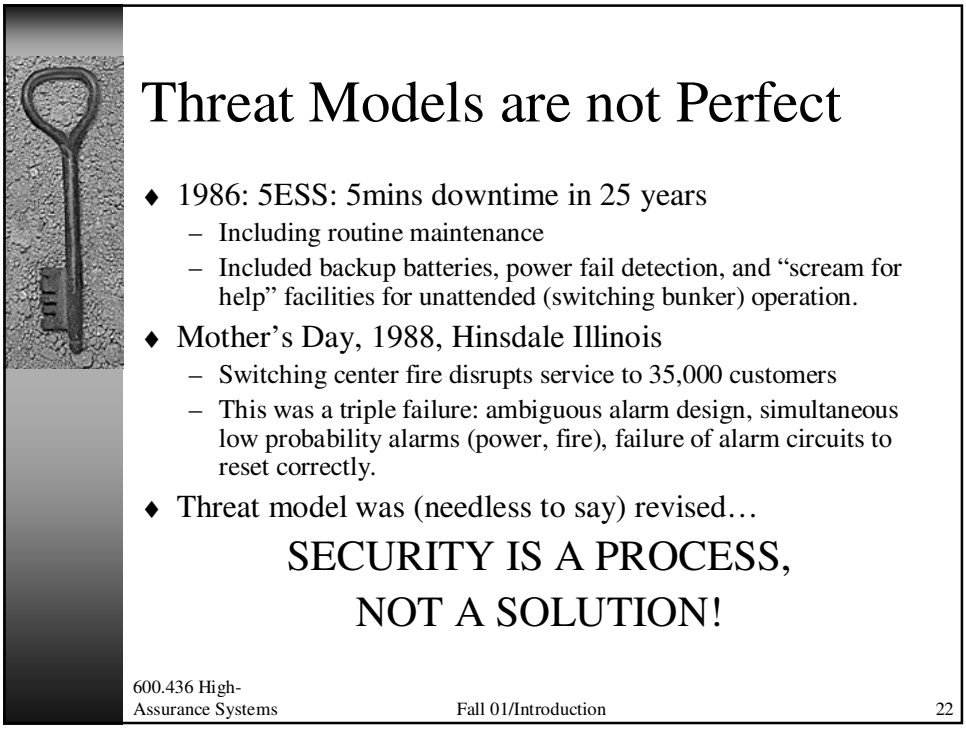
600.436 High-Assurance Systems Fall 01/Introduction 20



Policy Realization

- Just as a policy is stated in context, implementations are built under assumptions
- These assumptions are:
  - Administrative (e.g. logins will not be given out at random)
  - Environmental (e.g. physical access to machine is restricted)
  - Threat model: the attack scenarios you anticipated
    - As opposed to the nuclear attack that you *didn't* prepare for

600.436 High-Assurance Systems Fall 01/Introduction 21

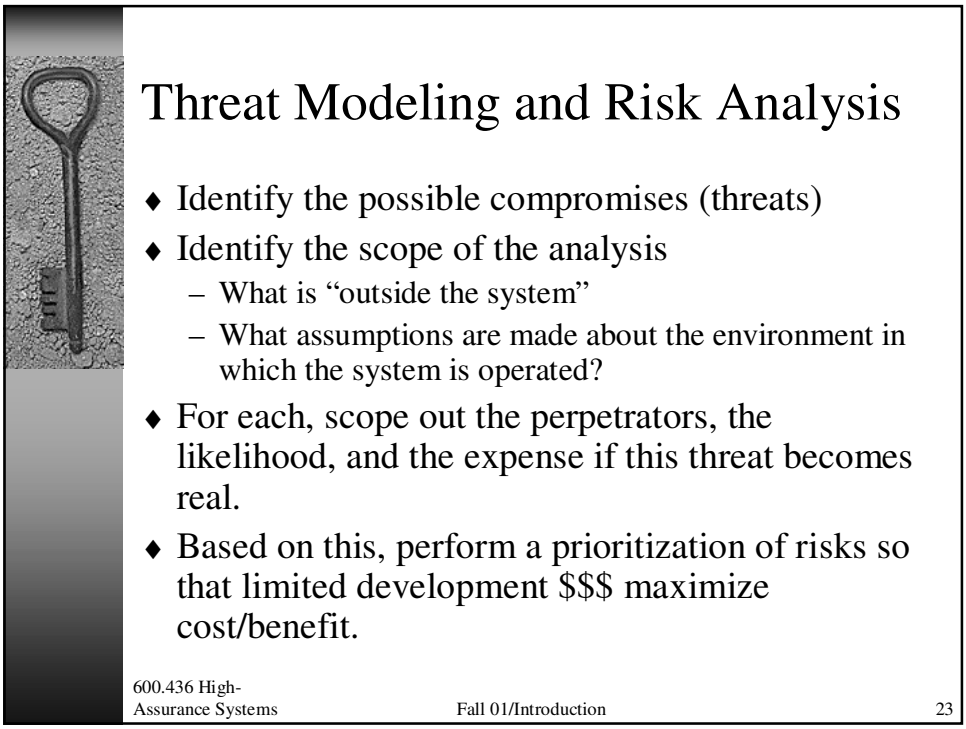


Threat Models are not Perfect

- 1986: 5ESS: 5mins downtime in 25 years
  - Including routine maintenance
  - Included backup batteries, power fail detection, and "scream for help" facilities for unattended (switching bunker) operation.
- Mother's Day, 1988, Hinsdale Illinois
  - Switching center fire disrupts service to 35,000 customers
  - This was a triple failure: ambiguous alarm design, simultaneous low probability alarms (power, fire), failure of alarm circuits to reset correctly.
- Threat model was (needless to say) revised…

SECURITY IS A PROCESS,
NOT A SOLUTION!

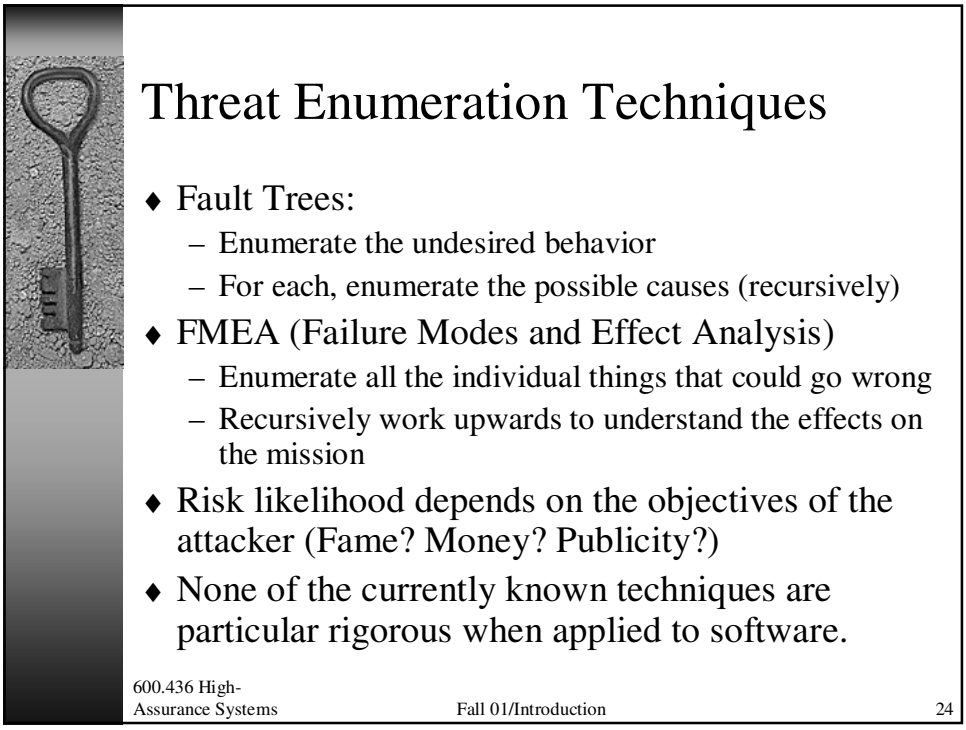
600.436 High-Assurance Systems Fall 01/Introduction 22



Threat Modeling and Risk Analysis

- Identify the possible compromises (threats)
- Identify the scope of the analysis
  - What is "outside the system"
  - What assumptions are made about the environment in which the system is operated?
- For each, scope out the perpetrators, the likelihood, and the expense if this threat becomes real.
- Based on this, perform a prioritization of risks so that limited development $$$ maximize cost/benefit.

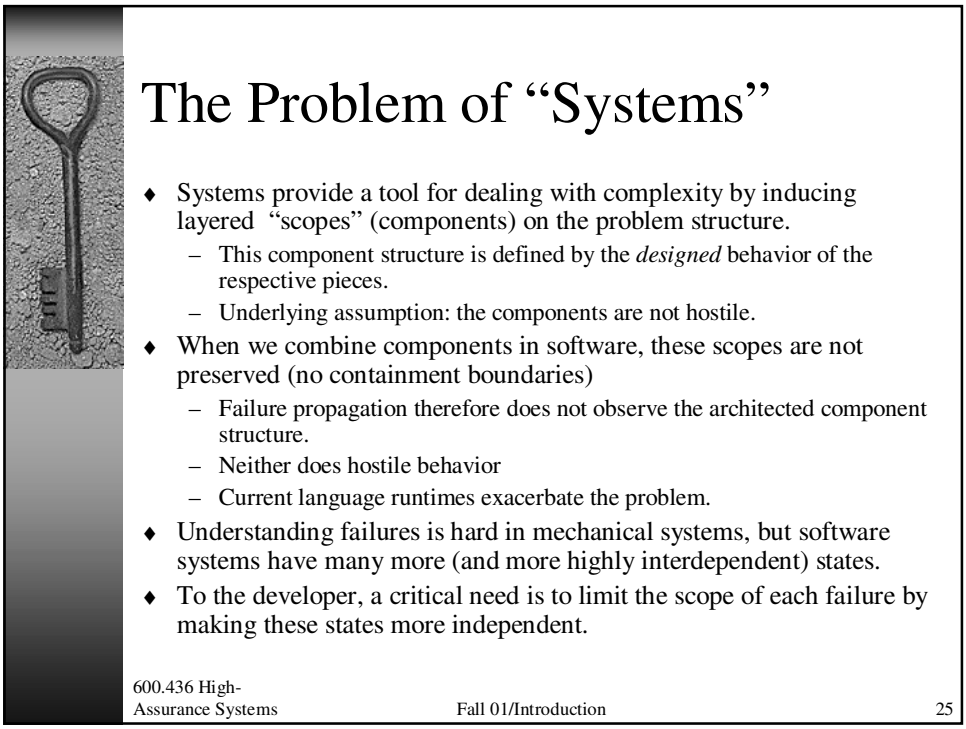
600.436 High-Assurance Systems Fall 01/Introduction 23



Threat Enumeration Techniques

- Fault Trees:
  - Enumerate the undesired behavior
  - For each, enumerate the possible causes (recursively)
- FMEA (Failure Modes and Effect Analysis)
  - Enumerate all the individual things that could go wrong
  - Recursively work upwards to understand the effects on the mission
- Risk likelihood depends on the objectives of the attacker (Fame? Money? Publicity?)
- None of the currently known techniques are particular rigorous when applied to software.

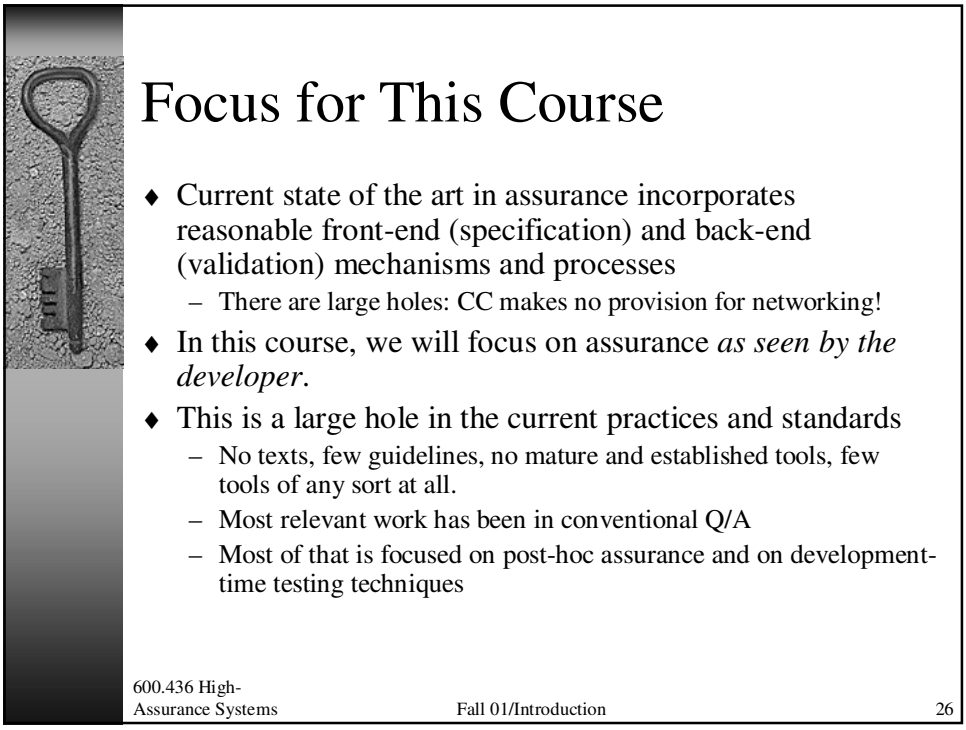
600.436 High-Assurance Systems Fall 01/Introduction 24



The Problem of "Systems"

- Systems provide a tool for dealing with complexity by inducing layered "scopes" (components) on the problem structure.
  - This component structure is defined by the *designed* behavior of the respective pieces.
  - Underlying assumption: the components are not hostile.
- When we combine components in software, these scopes are not preserved (no containment boundaries)
  - Failure propagation therefore does not observe the architected component structure.
  - Neither does hostile behavior
  - Current language runtimes exacerbate the problem.
- Understanding failures is hard in mechanical systems, but software systems have many more (and more highly interdependent) states.
- To the developer, a critical need is to limit the scope of each failure by making these states more independent.

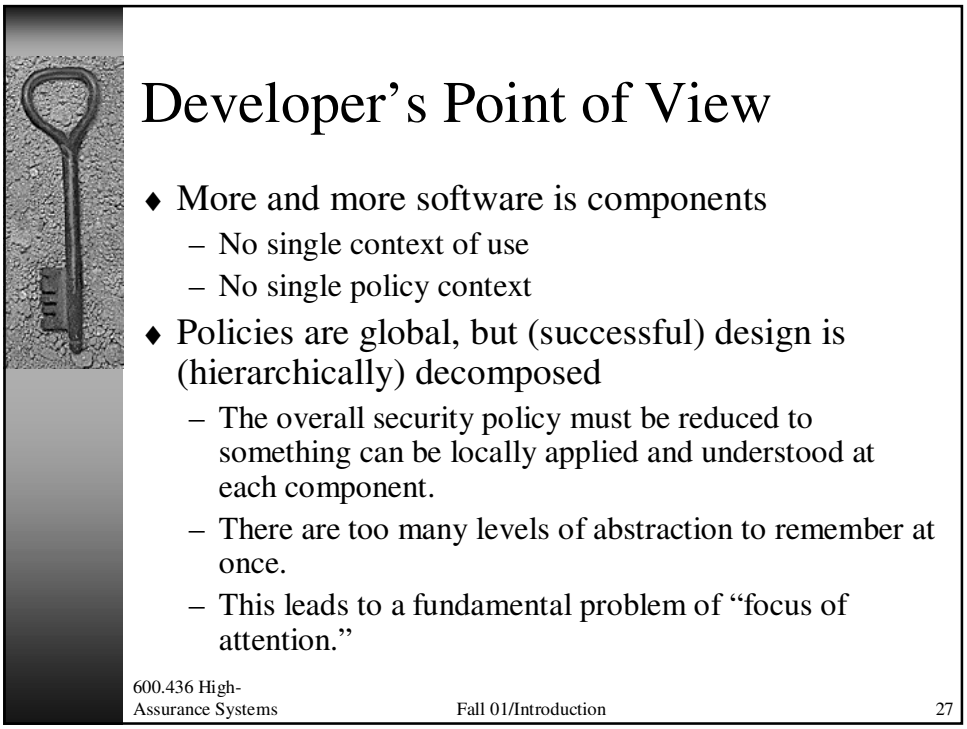
600.436 High-Assurance Systems Fall 01/Introduction 25



Focus for This Course

- Current state of the art in assurance incorporates reasonable front-end (specification) and back-end (validation) mechanisms and processes
  - There are large holes: CC makes no provision for networking!
- In this course, we will focus on assurance *as seen by the developer*.
- This is a large hole in the current practices and standards
  - No texts, few guidelines, no mature and established tools, few tools of any sort at all.
  - Most relevant work has been in conventional Q/A
  - Most of that is focused on post-hoc assurance and on development-time testing techniques

600.436 High-Assurance Systems Fall 01/Introduction 26


Developer's Point of View
More and more software is components
No single context of use
No single policy context
Policies are global, but (successful) design is (hierarchically) decomposed
The overall security policy must be reduced to something can be locally applied and understood at each component.
There are too many levels of abstraction to remember at once.
This leads to a fundamental problem of "focus of attention."
600.436 High-Assurance Systems
Fall 01/Introduction
27