Cornell University
Computing and Information Science

CS 5150 Software Engineering

Security

William Y. Arms

Suggested Readings

Butler W. Lampson, Computer Security in the Real World. *IEEE Computer*, June 2004.

Trust in Cyberspace, Committee on Information Systems Trustworthiness, National Research Council (1999) http://www.nap.edu/readingroom/books/trust/ [Fred Schneider, Cornell Computer Science, was the chair of this study].

Security in the Software Development Process

The security goal

The security goal is to make sure that the agents (people or external systems) who interact with a computer system, its data, and its resources, are those that the owner of the system would wish to have such interactions.

Security considerations need to be part of the entire software development process. They may have a major impact on the architecture chosen.

Security Needs and Dangers

Needs

- Secrecy: control of who gets to read information
- Integrity: control of how information changes or resources are used
- Availability: providing prompt access to information and resources
- Accountability: knowing who has had access to resources

Dangers

• Damage to information	integrity
• Disruption of service	availability
• Theft of money	integrity
• Theft of information	secrecy
• Loss of privacy	secrecy

Butler W. Lampson, 2004

The Economics of Security

How secure should your system be?

Building secure systems adds cost and time to software development

"Practical security balances the cost of protection and the risk of loss, which is the cost of recovering from a loss times its probability... When the risk is less than the cost of recovering, it's better to accept it as a cost of doing business ... than to pay for better security."

"Many companies have learned that although people may complain about inadequate security, they won't spend much money, sacrifice many features, or put up with much inconvenience to improve it."

Butler W. Lampson, 2004

The Economics of Security

Example: a credit card system

Option A

- The card is a plastic card with all data (e.g., name, number, expiration date) readable by anybody who has access to the card. A copy of the signature is written on the card.
- This is a cheap system to implement, but does little to discourage fraud.

Banks in the USA use this system.

Option B (chip and PIN)

- The card has an embossed security chip. To use the card, the security chip must be read by a special reader and the user must type in a confidential 4-digit number.
- This provides greater protection against fraud, but is more expensive and slightly less convenient for both merchant and user.

Banks in Europe use this system.

Security and People

People are intrinsically insecure

- Careless (e.g., leave computers logged on, share passwords)
- Dishonest (e.g., stealing from financial systems)
- Malicious (e.g., denial of service attack)

Many security problems come from inside the organization

- In a large organization, there will be some disgruntled and dishonest employees
- Security relies on trusted individuals. What if they are dishonest ?

Design for Security: People

- Make it easy for **responsible** people to use the system (e.g., make security procedures simple)
- Make it hard for **dishonest** or **careless** people (e.g., password management)
- **Train people** in responsible behavior
- **Test the security** of the system thoroughly and repeatedly, particularly after changes
- **Do not hide violations**

Agents and Components

The software development challenge

- develop secure and reliable components
- protect whole system so that security problems in parts of it do not spread to the entire system

A large system will have many agents and components

- each is potentially unreliable and insecure
- components acquired from third parties may have unknown security problems

The commercial off-the-shelf (COTS) problem

- Developers of COTS software have considerable incentives to supply software that has many options and features
- In developing such software rapidly they have fewer incentives to be thorough about security.

Security Techniques: Barriers

Place barriers that separate parts of a complex system:

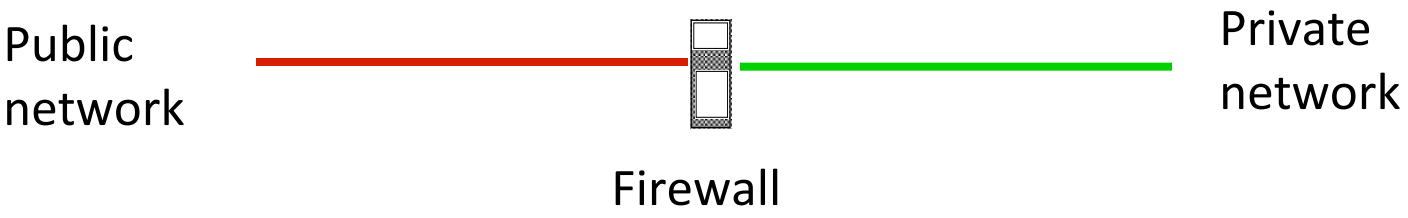
- Isolate components, e.g., do not connect a computer to a network
- Firewalls
- Require authentication to access certain systems or parts of systems

Every barrier imposes restrictions on permitted uses of the system

Barriers are most effective when the system can be divided into subsystems with simple boundaries

Example. Integration of Internet Explorer into Windows

Barriers: Firewall



A **firewall** is a computer at the junction of two network segments that:

- Inspects every packet that attempts to cross the boundary
- Rejects any packet that does not satisfy certain criteria, e.g.,

  an incoming request to open a TCP connection

  an unknown packet type

Firewalls provide increased security at a loss of flexibility, inconvenience for users, and extra system administration.

Security Techniques: Authentication & Authorization

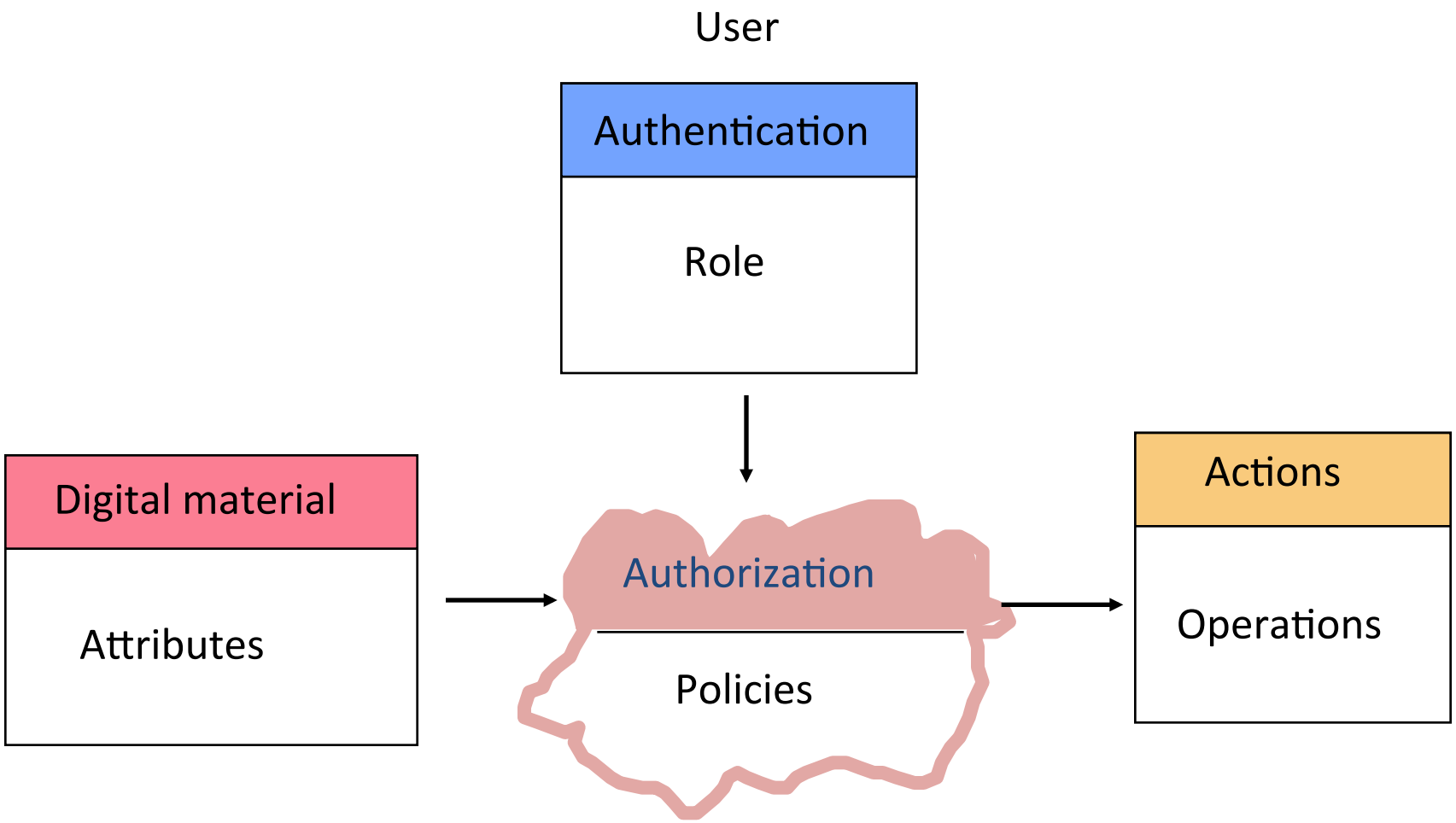
Authentication establishes the identity of an agent:

- What does the agent know (e.g., password)?
- What does the agent possess (e.g., smart card)?
- What does the agent have physical access to (e.g., crt-alt-del)?
- What are the physical properties of the agent (e.g., fingerprint)?

Authorization establishes what an authenticated agent may do:

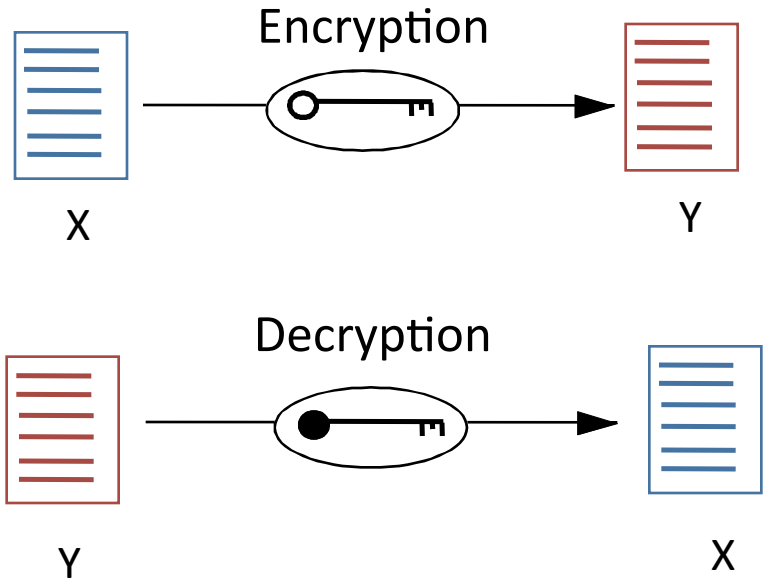
- Access control lists
- Group membership

Example: An Access Architecture for Digital Content



Security Techniques: Encryption

Allows data to be stored and transmitted securely, even when the bits are viewed by unauthorized agents and the algorithms are known.



- Private key and public key
- Digital signatures

Programming Secure Software

Programs **that interface with the outside world** (e.g., web sites, mail servers) need to be written in a manner that resists intrusion.

For the top 25 programming errors, see: *Common Weakness Evaluation: A Community-Developed Dictionary of Software Weakness Types*. http://cwe.mitre.org/top25/

- Insecure interaction between components
- Risky resource management
- Porous defenses

Project management and test procedures must ensure that programs avoid these errors.

Programming Secure Software

The following list is from the SANS Security Institute, *Essential Skills for Secure Programmers Using Java/JavaEE*, http://www.sans.org/

- Input handling
- Authentication & session management
- Access control (authorization)
- Java types & JVM management
- Application faults & logging
- Encryption services
- Concurrency and threading
- Connection patterns