Cornell University
Computing and Information Science

CS 5150 Software Engineering
14. Security

William Y. Arms

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 system 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, Computer Security in the Real World
IEEE Computer, June 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

Credit cards: 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 have traditionally used this system.

Credit cards: 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.

For many years, banks in Europe have used this system.

The new system in the USA is less secure than option B, but cheaper to install.

Security within an Organization: People

Many security problems come from people inside the organization

- In a large organization, there will be some dishonest and disgruntled employees.
  - -> Dishonest (e.g., stealing from financial systems)
  - -> Malicious
- Security relies on trusted individuals. What if they are dishonest?

People are intrinsically insecure

- Careless (e.g., leave computers logged on, share passwords)

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.**

External Intruders

All network systems are vulnerable to security breaches by external intruders:

- financial
- malicious
- secrets
- and worse

Modern software is so complex that it is impossible to eliminate all vulnerabilities.

Many skilled individuals and organizations are continually seeking to discover and exploit new vulnerabilities.

External Intruders

Examples of external security vulnerabilities:

- unauthorized access — modify software, install listening devices
- backdoors — bypass authentication
- denial of service — overload and other forms of blocking
- eavesdropping
- spoofing
- phishing
- etc., etc.

This list is derived from Wikipedia.

External Intruders: Minimizing Risk

There is no way to guarantee security from external intruders, but careful software development can make a major difference.

How to minimize the risks:

- System design — secure protocols, authentication, barriers to access
- Programming — defensive programming and rigorous testing
- Operating procedures — backup, auditing, vulnerability testing
- Training and monitoring personnel

Minimizing Risks: System Architecture

The system architecture can minimize risks in various ways:

- Secure protocols, e.g., HTTPS encryption.
- Authentication, e.g., encryption of passwords in transmission and when stored, two factor authentication.
- Barriers, e.g., firewalls, private networks, and virtual private networks.
- Data security, e.g., encryption of stored data, backup.

Security Techniques: Barriers

Place barriers that separate parts of a complex system:

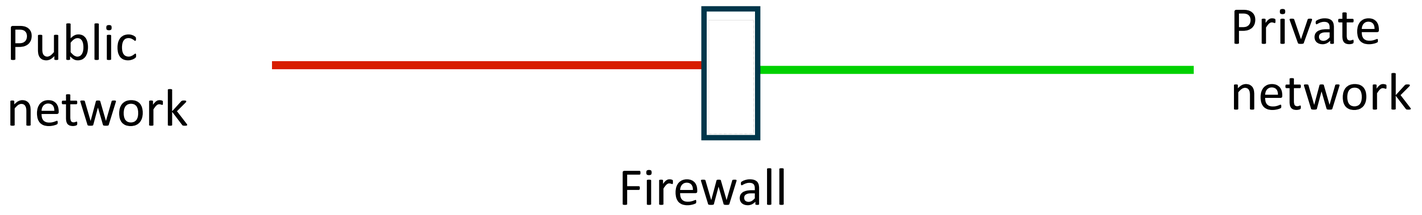
- Isolate components, e.g., do not connect a computer to a network
- Firewalls
- Require authentication to access certain systems or parts of systems

Every barrier imposes restrictions on permitted uses of the system.

Barriers are most effective when the system can be divided into subsystems.

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

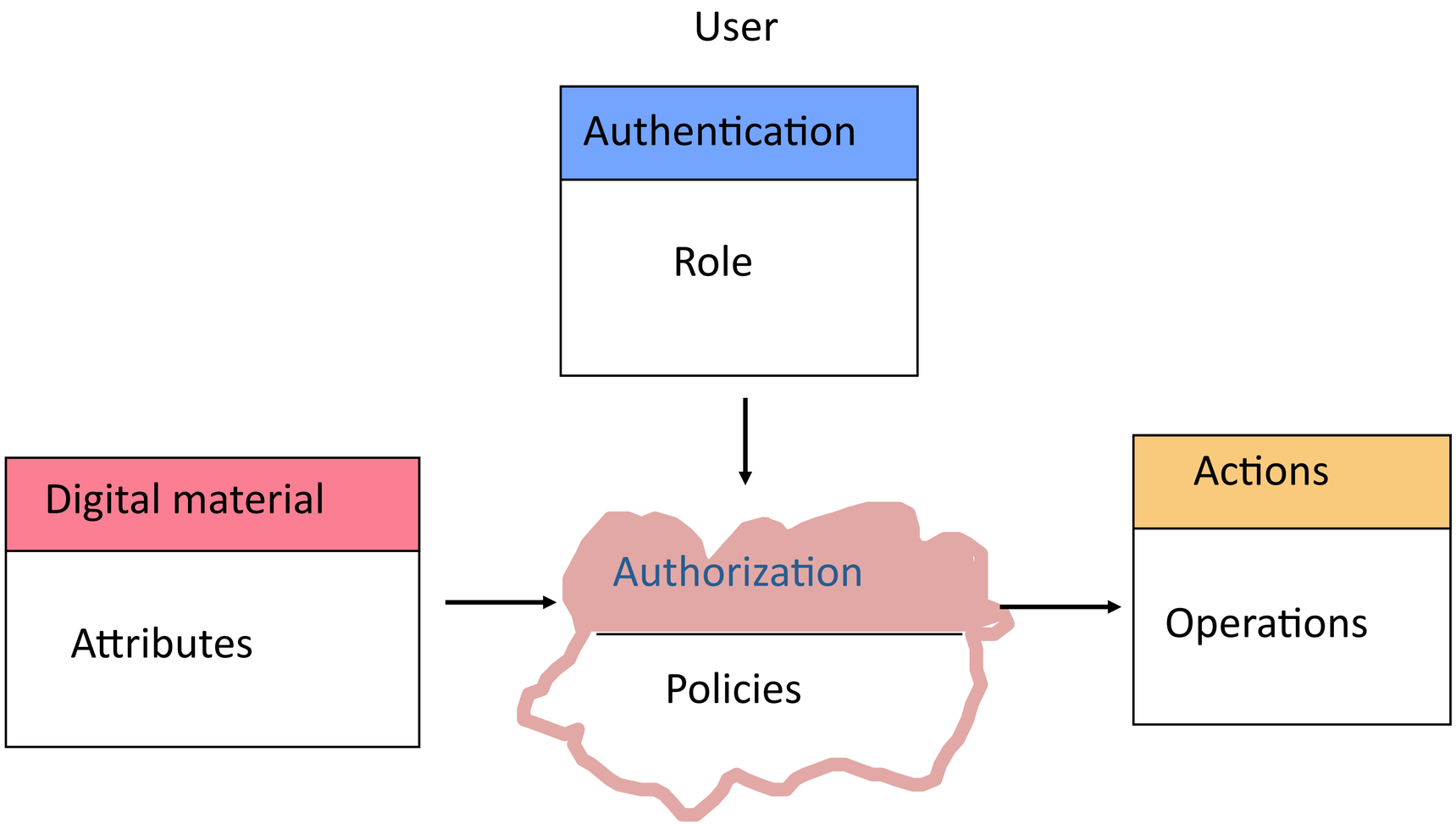
Authentication establishes the identity of an agent:

- 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:

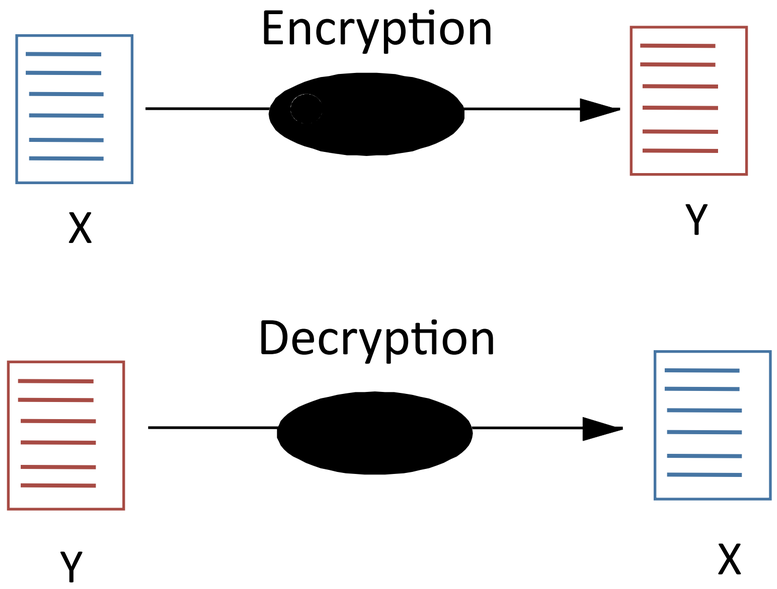
- Access control lists
- Group membership

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

Minimizing Risks: Programming

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 problem

- Developers of off-the-shelf 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.

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.

Minimizing Risks: Procedures

The operating procedures must anticipate security problems.

A senior member of staff must have responsibility for security.

Equipment

- All system software should be kept up to date with latest security patches.
- All systems should run up to date virus checking software.
- Rules about passwords, personal equipment, and non-standard software should be explicit.

Routine checks

- Run network security tests regularly.
- Run password checkers.

Training

Keep staff informed about security. Ask for their advice.

Operations: Recovery

Sooner or later every system fails because of hardware, software, operational, or security problems.

The operating procedures must anticipate loss of data and damage to systems, which can happen at any moment.

Backup techniques

- At regular intervals check point the system
- At regular intervals backup all data.
- Keep full audit trails of all important transactions

Recovery software

Recovery software is complex. It needs to be tested regularly in realistic situations.

A good practice is to rebuild the entire system, but this may not be possible with large collections of data.

Security in the Software Development Process

Conclusion

You can never guarantee that a system is completely secure, but you can do a great deal to minimize the risks and to be able to recover from problems.