Security Principles

Haipeng Dai

haipengdai@nju.edu.cn
313 CS Building
Department of Computer Science and Technology
Nanjing University

Security Principles

- Variations of lists of security principles
- Security vulnerabilities often exploit violations of these principles
- Good security solutions or countermeasures follow these principles
- Some overlap & some tension between principles
- More generally, checklists are useful for security
- These principles can be applied at many levels, e.g., in source code of an application, between applications on a machine at OS level, at network level, within an organization, between organizations
- Here we give 16 principles
- Main Source: *software security principles* by Gary McGraw and John Viega

Principle 1: Securing the weakest

- Security is a chain; a system is only as secure as the weakest link.
- Spend your efforts on improving the security of the weakest part of a system, as this is where attackers will attack.
  - Bank vs. convenience store
  - Firewall vs. application visible through firewall
- Social engineering: a common weak link.
  - Software can be weak, surrounding infrastructure can be weaker.
  - Typical scenario: a service center gets a call from a sincere-sounding user
  - Educating users may be best investment to improve security, eg., think of phishing attacks, weak passwords

Table II. Character composition information about each password dataset

Dataset	[a-z]+	[A-Z]+	[A-Za-z]+	[0-9]+	[a-zA-Z0-9]+	[a-z]+[0-9]+	[a-z]+1	[a-zA-Z]+[0-9]+	[0-9]+[a-zA-Z]+	[0-9]+[a-z]+
Tianya	9.96%	0.18%	10.29%	**63.77%**	98.05%	14.63%	0.12%	15.64%	4.37%	4.11%
Dodonew	8.79%	0.27%	9.37%	**20.49%**	82.88%	40.81%	1.39%	42.94%	7.31%	6.95%
CSDN	11.64%	0.47%	12.35%	**45.01%**	96.31%	26.14%	0.24%	28.45%	6.46%	5.88%
Duowan	10.30%	0.09%	10.52%	**52.84%**	97.59%	23.97%	0.37%	24.84%	6.04%	5.83%
Myspace	7.18%	0.31%	7.66%	0.71%	89.95%	**65.66%**	**18.24%**	**69.77%**	6.02%	5.66%
Singles.org	60.20%	1.92%	**65.82%**	9.58%	99.78%	17.77%	2.73%	19.68%	1.92%	1.77%
Faithwriters	54.40%	1.16%	**59.04%**	6.35%	99.57%	22.82%	4.13%	25.45%	2.73%	2.37%
Hak5	18.61%	0.27%	20.39%	5.56%	92.13%	16.57%	2.01%	31.80%	1.44%	1.21%

Principle 1: Securing the weakest link

Table IV. Top 10 most popular passwords of each dataset

Rank	Tianya	Dodonew	CSDN	Duowan	Myspace	Singles.org	Faithwriters	Hak5
1	**123456**	**123456**	123456789	**123456**	password1	**123456**	**123456**	QsEfTh22
2	111111	a123456	12345678	111111	abc123	**jesus**	writer	——
3	000000	123456789	11111111	123456789	fuckyou	password	**jesus1**	timosha
Top 3 (%)	5.58%	1.49%	8.15%	5.01%	0.40%	2.10%	1.03%	4.62%
4	123456789	111111	dearbook	123123	monkey1	12345678	**christ**	ike02banaA
5	123123	**5201314**	00000000	000000	**iloveyou1**	**christ**	blessed	**123456**
6	123321	123123	123123123	**5201314**	myspace1	**love**	john316	zxczxc
7	**5201314**	a321654	1234567890	123321	fuckyou1	**princess**	**jesuschrist**	123456789
8	12345678	12345	88888888	a123456	number1	**jesus1**	password	westside
9	666666	000000	111111111	suibian	football1	sunshine	heaven	ZVjmHgC355
10	111222tianya	123456a	147258369	12345678	nicole1	1234567	faithwriters	Kj7Gt65F
Top 10 (%)	7.42%	3.28%	10.44%	6.78%	0.78%	3.40%	2.17%	7.20%

- Credit card example - threats to credit card numbers include:
  - eavesdropping on the network
  - stealing from customer database
  - me loosing my wallet – weakest link
- Solution: risk analysis.

Principle 2: Defense in depth

- If one layer fails, hopefully another layer can succeed.
- No single point of failure.
- Why is bank more secure than convenience store?
  - Redundant security measures: security guard, bulletproof glass, electronically locked doors, vault protecting the rest and requiring the presence of two individuals who are rarely at the bank at the same time, security camera,…
- Well-known principle even beyond security
  - Have a series of defenses so that if an error isn't caught by one, it will probably be caught by another. (From Bruce MacLennan's *Principles of Programming Languages*.)
- Securing the weakest link applies to nonoverlapping functions.
- Defense in depth applies to same function.

Principle 2: Defense in depth (cont.)

- Example 1: have a firewall and secure web application software, and run web application with minimal privileges
- Example 2: use OS access control to restrict access to sensitive files, and encrypt them, especially when files are stored on removable media such as USB sticks, laptops, or PCs which might be disposed.
- Counterexample: on UNIX systems, the password file, /etc/passwd, which contains hashed passwords, was world readable.
  - Solution: enforce tight access control to the file.
- Counterexample: having a firewall, and only having firewall
  - a user bringing in a laptop circumvents firewall
- Counterexample: firewall + unencrypted data within network

Principle 3: Secure failure

- When systems fail, they should not revert to insecure behavior. Otherwise, attacker only need to invoke the right failure.
- Incorrect handling of unexpected errors is major cause of security breaches
- Example: careful handling of exceptions in JAAS (Java Authentication and Authorization Service) module code!

```
isAdmin = true; // enter Admin mode
try {
        something that may throw SomeException
} catch (SomeException ex) {
        // should we log?
        log.write(ex.toString());
        // how should we proceed?
        isAdmin = false;
        // or should we exit?
}
```

Principle 3: Secure failure (cont.)

- Counterexample: old version software did not use encryption, new version does; for backward compatibility, when new version talks to old version, new version disables encryption.
  - Attack: when new version talks to new version, attackers can tamper message on the wire to make both think the other is over version.
  - Solution: new version notifies old version to download patches from a secure place that the old version knows.
  - Question: what about old version downloads patches from new version?
  - Security problem: then new version is not authenticated.
- Counterexample - Remote Method invocation (RMI)
  - When a client and server want to communicate over RMI, if the server wants to use an encryption protocol that client does not support, the client downloads the protocol implementation from the server at runtime.
  - Security problem: if the client fails to establish a secure connection (a failure), it will establish a connection using whatever protocol an untrusted entity gives it.

Principle 4: Least privilege

- Only the **minimum access** necessary to perform an operation should be granted, and that access should be granted only for the **minimum** amount of **time** necessary.
- Example: you go vacation, ask a friend to pick up mail
- Example: U.S. government -- the policy of “need to know.”
- Counterexample: famous violations of least privilege exist in UNIX systems (-- needs root privilege for running a service on a port number less than 1024)
  - Some e-mail servers is that they don't give up their root permissions once they have grabbed the mail port (Sendmail is a classic example).
- Counterexample: device drivers having to run in kernel mode
- Counterexample: Several calls in the Windows API for accessing objects that grant all access if you pass "0" as an argument.
  - Programmers are lazy.

Principle 4: Least privilege (cont.)

- In organization
  - don’t give everyone access to root passwords
  - don’t give everyone administrator rights
- On computer
  - Run process with minimal set of privileges
  - For example, don’t run web application as root or administrator
- for Java application: not the default policy

grant codeBase "file:${{java.ext.dirs}}/" {*

permission java.security.AllPermission;

};

but minimum required

grant codeBase "file:./forum/" {*

permission java.security.FilePermission;

"/home/forumcontent/","read/write";};*

Principle 4: Least privilege (cont.)

- Expose minimal functionality in interfaces of objects, classes, packages, applications.

- in code:
  - not **public int x;**
  - but **private int x;**
  - not **public void m()**
  - but **package void m()**

- Least privilege example:
  - Standard coding standard

    not to use **import java.lang.*;**

    but always **import java.lang.String;**

Principle 4: Least privilege (cont.)

- Use Secure Defaults
- By default,
  - security should be switched on
  - permissions turned off
- This will ensure that we apply principle of least privilege
- Counterexample: bluetooth connection on mobile phone is by default on, but can be abused

Principle 5: Compartmentalization

- Break the system up into as many isolated units as possible
  - Simplicity
  - Containing attacker in case of failure
- Example: submarines are built with many chambers, each separately sealed
- Example: prison.
- Counterexample: Famous violations of this principle exist standard UNIX privilege model
  - A program with root privilege can do everything (including erase logs)
- A few operating systems, such as Trusted Solaris, do compartmentalize.
- Tradeoff with manageability.
- Counterexample: OS that crashes if an application crashes.

Principle 5: Compartmentalization (cont.)

- Use different machines for different tasks
- Example: run web application on a different machine from employee salary database
- Example: use different user accounts on one machine for different tasks
- Compartementalization provided by typical OS is poor!
- Partition hard disk and install OS twice

Principle 6: Simplicity

- KISS mantra -- "Keep it simple, stupid!"
- Designs and implementations should be as simple as possible
  - Complexity increases the risk of problems; unavoidable in any system.
  - Complex code tends to be harder to analyze and maintain. It also tends to be far more buggy.
- Should try to reuse components whenever possible.
- Be careful in applying this principle
  - Keep system simple on the condition of keeping system secure.
- Use choke points to improve simplicity
  - Force all security-critical operations through a few *choke points*.

Principle 7: Promote privacy

- Privacy of users, but also of systems
- Counterexample: services tend to give information about themselves that can help the attacker figure out how to break in.
  - Telnet service tends to give the operating system name and version.

    – > telnet somemachine

    Trying 1.2.3.4

    Connected to somemachine (1.2.3.4)

    Red Hat Linux release 7.0 (Hedwig)

    Kernel 1.2.3.4 on an i686

    login:
  - Solution 1: use firewalls to block unnecessary services
  - Solution 2: remove such info from software (e.g., changing telenet login)
  - Solution 3: give the WRONG info! No hurt to lie to attackers!
- Counterexample: SQL error messages on webpage
- Counterexample: HTTP (http://www.cse.msu.edu/~alexliu/a.html)

Principle 8: Hard to hide secrets

- Don’t rely on security by obscurity [Kerckhoff principle]
- Don’t assume attackers don’t know the application source code, and can’t reverse-engineer binaries
  - Don’t hardcode secrets in code.
  - Don’t rely on code obfuscation
- Counterexample
  - DVD encryption
  - webpages with hidden URLs
  - passwords in javascript code – this happens!

Principle 9: Be Reluctant to Trust

- Minimize Trusted Computing Base (TCB), i.e., the part of the system that has to be trusted
- “Trusted” is not the same as “trustworthy”
- Counterexample: trust is extended far too easily in customer support.
- All user input is evil !
  - Unchecked user input leads to
    - buffer overflows, SQL injection, XSS on websites
  - User input includes cookies, environment variables, ...
- User input should not be trusted, and subjected to strong input validation checks before being used
- Don’t trust your employees.
- Trust is transitive – make sure that trusted programs never invoke untrusted programs.

Principle 10: Use Community Resources

- Repeated use without failure promotes trust. Public scrutiny does as well.
- Never design your own cryptography
- Never implement your own cryptography
- Researchers help you to examine your algorithms for free!
- Don’t repeat known mistakes
- If you’re making an application of kind X using programming language Y on platform Z and operating system W, look for
  - known threats for application of kind X
  - known vulnerabilities in programming language Y and platform Z, ...
  - existing countermeasures
- Counterexample: homebrew cookie schemes in Kevin Fu’s “Dos and Don'ts of Client Authentication on the Web”

Principle 11: Minimize Attack Surface

- Minimize
  - number of open sockets
  - number of services
  - number of services running by default
  - number of services running with high priviliges
  - number of dynamic content webpages
  - number of accounts with administrator rights
  - number of files & directories with weak access control
- Counterexample
  - Linux box kind.cs.ru.nl was hacked, by someone guessing the root password
  - Why was root login over the network allowed anyway?

Principle 11: Minimize Attack Surface (cont.)

- Minimize Attack Surface in code
  - not public int x; but private int x;
  - not public void m(); but package void m()
  - This is applying principle of least privilege, and also reduces attack surface, from buggy or hostile code

- Minimize attack surface in time
  - Automatically log off users after n minutes
  - Automatically lock screen after n minutes
  - Unplug network connection if you don't use it
  - Switch off computer if you don't use it
  - On smartcards, it's good practice to zero-out arrays that contains sensitive information (usually, decrypted information) as soon as it's no longer needed)

Principle 12: Don’t mix data & code

- This is the cause of many problems
- Counterexample: traditional buffer overflow attacks, which rely on mixing data and code on the stack
- Counterexample: VB scripts in Office documents
  - leads to attacks by hostile .doc or .xls
- Counterexample: javascript in webpages
  - leads to XSS (cross site scripting attacks)
- Counterexample: SQL injection relies on user data (user input!) as part of SQL query

Principle 13: Clearly Assign Responsibilities

- At organisational level:
  - eg. make one person responsible for something rather than two persons or a whole group.
- At coding level:
  - make one module/class responsible for input validation, access control, ...
  - for a method

    public void process(String str)

    is the caller or callee responsible for checking if for instance

    str!=null & !(str.equals(“”)) ?
- But still practice defence in depth...

Principle 14: Identify Your Assumptions

- Including obvious, implicit assumptions
  - these may be sources of vulnerability, and may change in long run
- Examples
  - laptops invalidate implicit assumption that computers don't move past the company firewall
  - assumption that user is a human may cause you to ignore possibility of brute-force password guessing
  - TCP SYN flood attack exploits implicit assumptions in the spec "If we receive a TCP packet with the SYN flag set, it means the sender wants to start a dialog with us"
  - assumption that *new LoginContext()* won't throw an *OutOfMemory* Exception
  - assumption that a Java applet won't use all CPU time

Principle 15: Audit Your System

- To keep a system secure, you need a record of changes made to the system, either by its own utilities or by intrusion detection systems such as Snort and Tripwire.
- While you can keep some records of changes by hand, on Unix-like systems, logs of changes or errors are traditionally saved in /var/log by system applications.
  - This system is not ideal, since altering logs to hide an intrusion is one of the first steps that an expert cracker makes.
- However, since many attacks are by script-kiddies with little understanding of the system, a change in logs is often the first sign that a system has been compromised.
  - Some intrusion detection programs, such as Tripwire, can automate the checking of logs and other key files.

Principle 16: Have Good Usability

- Understand your users
  - They will switch off security if too cumbersome.
  - They do not read documentation.
  - They ignore warnings – they just want to get what they want!

- Counterexample: Norton Internet Security keeps popping up windows to ask questions

- User education may improve the situation, but only up to a point

Overview

Haipeng Dai

haipengdai@nju.edu.cn
313 CS Building
Department of Computer Science and Technology
Nanjing University

Cryptographic algorithms and protocols can be grouped into four main areas:

Symmetric encryption

- Used to conceal the contents of blocks or streams of data of any size, including messages, files, encryption keys, and passwords

Asymmetric encryption

- Used to conceal small blocks of data, such as encryption keys and hash function values, which are used in digital signatures

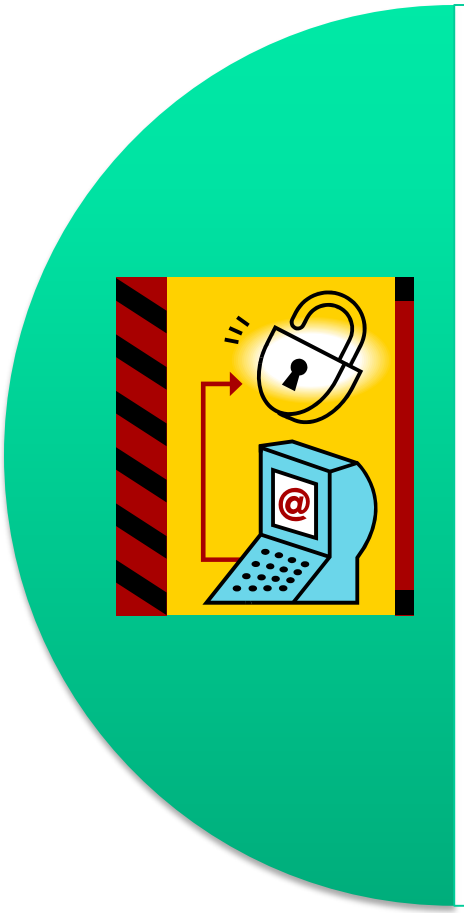
Data integrity algorithms

- Used to protect blocks of data, such as messages, from alteration

Authentication protocols

- Schemes based on the use of cryptographic algorithms designed to authenticate the identity of entities

The field of network and Internet security consists of:



measures to deter, prevent, detect, and correct security violations that involve the transmission of information

Computer Security

- The NIST *Computer Security Handbook* defines the term computer security as:

  "the protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability and confidentiality of information system resources" (includes hardware, software, firmware, information/data, and telecommunications)

Computer Security Objectives

Confidentiality

- Data confidentiality
  - Assures that private or confidential information is not made available or disclosed to unauthorized individuals
- Privacy
  - Assures that individuals control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed

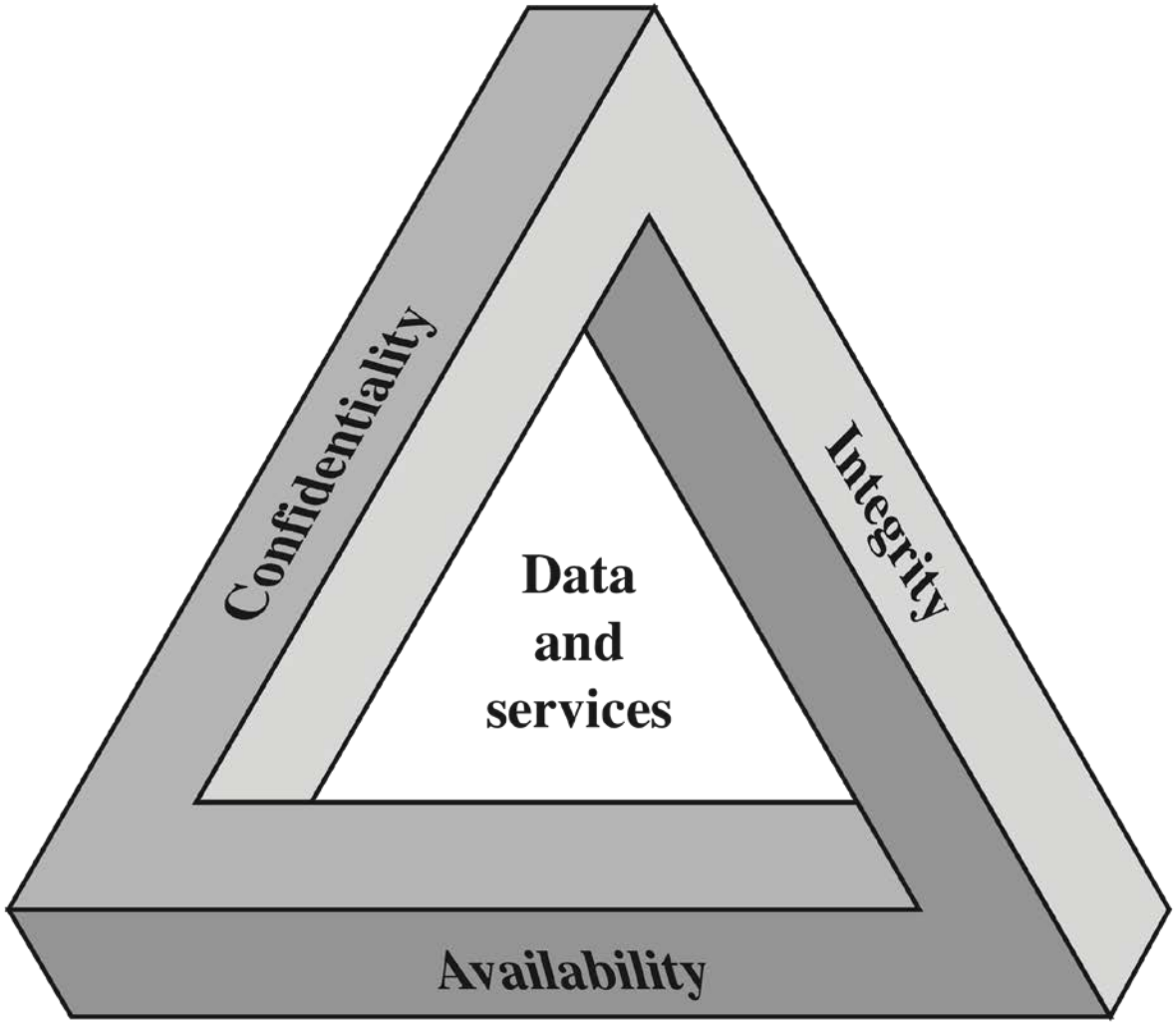
Integrity

- Data integrity
  - Assures that information and programs are changed only in a specified and authorized manner
- System integrity
  - Assures that a system performs its intended function in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation of the system

Availability

- Assures that systems work promptly and service is not denied to authorized users

CIA Triad


Confidentiality
Integrity
Data
and
services
Availability

Possible additional concepts:

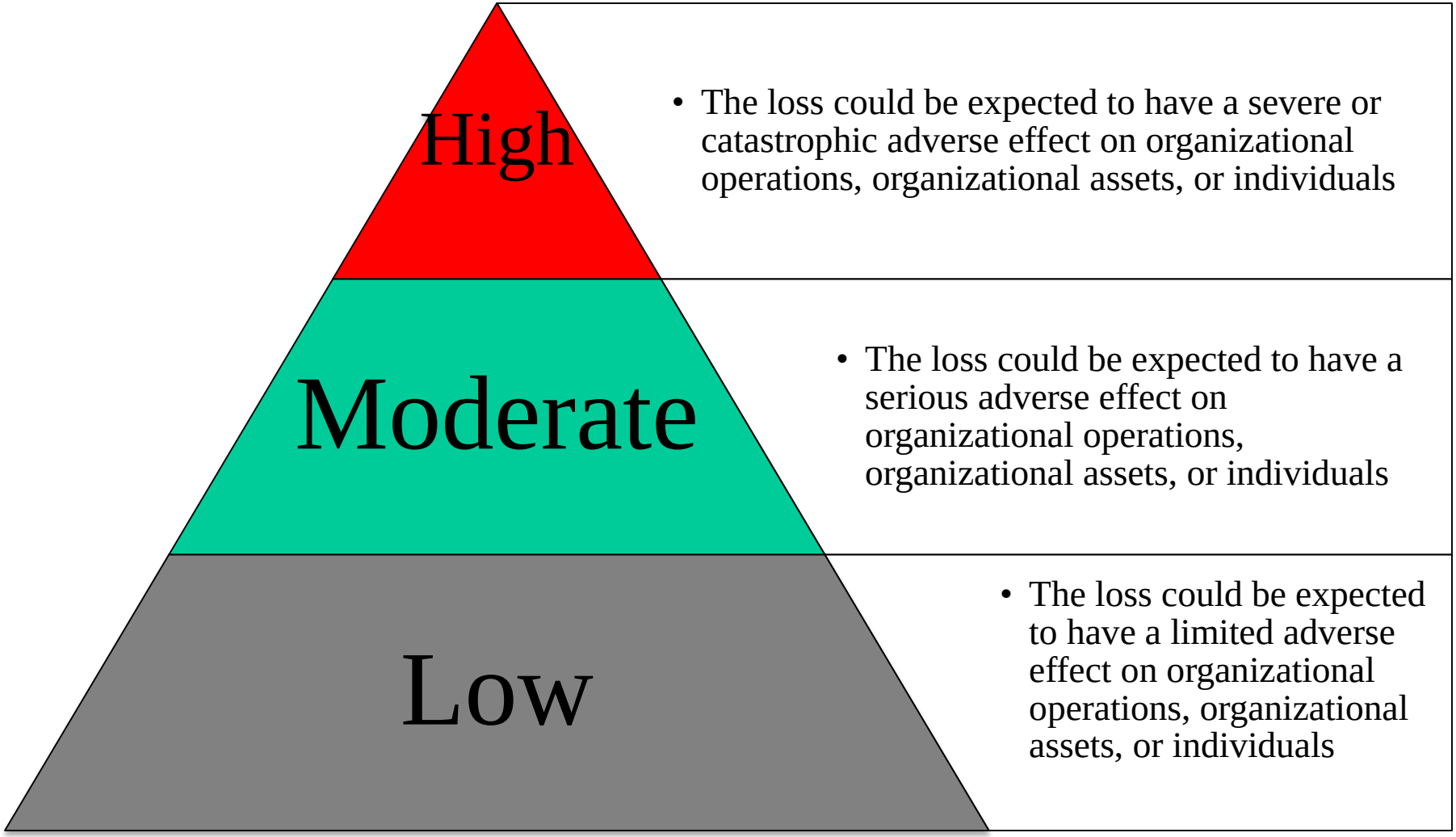
Authenticity

- Verifying that users are who they say they are and that each input arriving at the system came from a trusted source

Accountability

- The security goal that generates the requirement for actions of an entity to be traced uniquely to that entity

Breach of Security Levels of Impact



OSI Security Architecture

- Security attack
  - Any action that compromises the security of information owned by an organization
- Security mechanism
  - A process (or a device incorporating such a process) that is designed to detect, prevent, or recover from a security attack
- Security service
  - A processing or communication service that enhances the security of the data processing systems and the information transfers of an organization
  - Intended to counter security attacks, and they make use of one or more security mechanisms to provide the service

Threats and Attacks

Threat

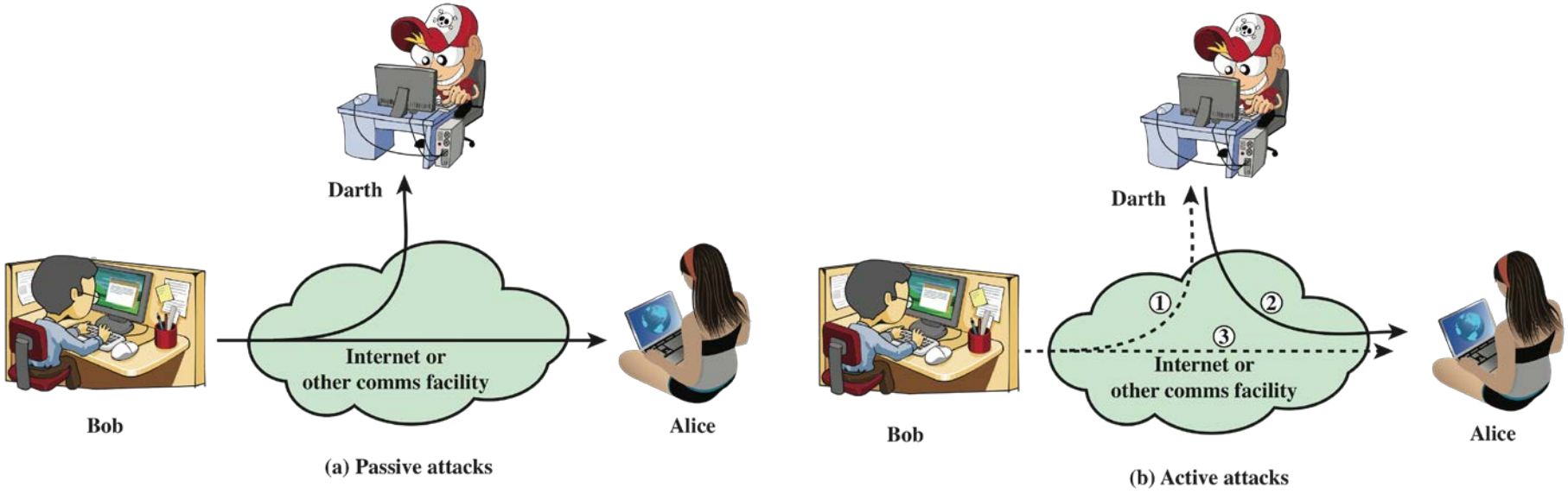
A potential for violation of security, which exists when there is a circumstance, capability, action, or event that could breach security and cause harm. That is, a threat is a possible danger that might exploit a vulnerability.

Attack

An assault on system security that derives from an intelligent threat; that is, an intelligent act that is a deliberate attempt (especially in the sense of a method or technique) to evade security services and violate the security policy of a system.

Security Attacks

- A means of classifying security attacks, used both in X.800 and RFC 4949, is in terms of *passive attacks* and *active attacks*
- A *passive attack* attempts to learn or make use of information from the system but does not affect system resources
- An *active attack* attempts to alter system resources or affect their operation



Passive Attacks

- Are in the nature of eavesdropping on, or monitoring of, transmissions
- Goal of the opponent is to obtain information that is being transmitted

- Two types of passive attacks are:
  - The release of message contents
  - Traffic analysis

Active Attacks

- Involve some modification of the data stream or the creation of a false stream
- Difficult to prevent because of the wide variety of potential physical, software, and network vulnerabilities
- Goal is to detect attacks and to recover from any disruption or delays caused by them

Masquerade

- Takes place when one entity pretends to be a different entity
- Usually includes one of the other forms of active attack

Replay

- Involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect

Modification of messages

- Some portion of a legitimate message is altered, or messages are delayed or reordered to produce an unauthorized effect

Denial of service

- Prevents or inhibits the normal use or management of communications facilities

Security Services

- Defined by X.800 as:
  - A service provided by a protocol layer of communicating open systems and that ensures adequate security of the systems or of data transfers
- Defined by RFC 4949 as:
  - A processing or communication service provided by a system to give a specific kind of protection to system resources

X.800 Service Categories

- Authentication
- Access control
- Data confidentiality
- Data integrity
- Nonrepudiation

Authentication

- Concerned with assuring that a communication is authentic
  - In the case of a single message, assures the recipient that the message is from the source that it claims to be from
  - In the case of ongoing interaction, assures the two entities are authentic and that the connection is not interfered with in such a way that a third party can masquerade as one of the two legitimate parties

Two specific authentication services are defined in X.800:

- Peer entity authentication
- Data origin authentication

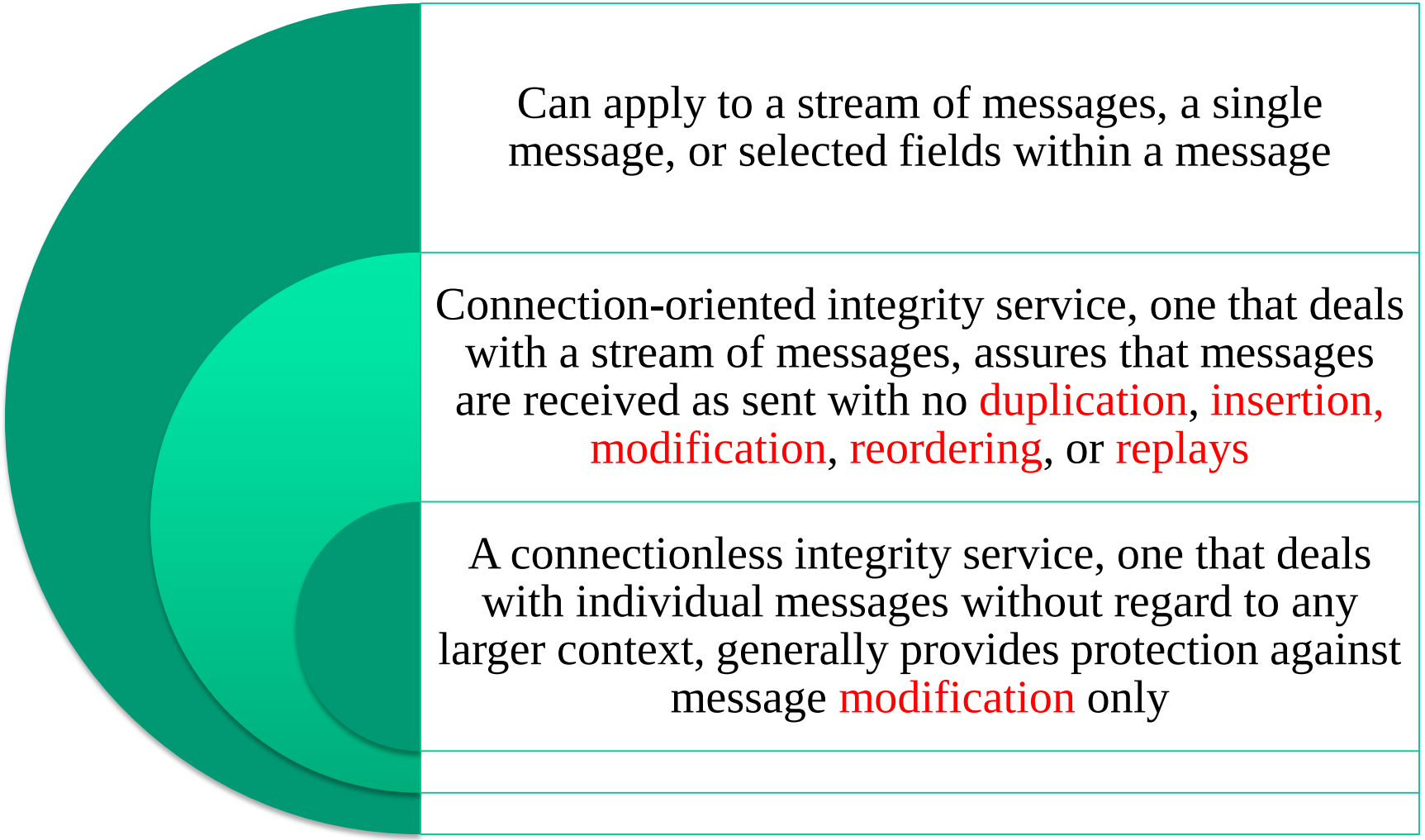
Access Control

- The ability to limit and control the access to host systems and applications via communications links
- To achieve this, each entity trying to gain access must first be indentified, or authenticated, so that access rights can be tailored to the individual

Data Confidentiality

- The protection of transmitted data from passive attacks
  - Broadest service protects all user data transmitted between two users over a period of time
  - Narrower forms of service includes the protection of a single message or even specific fields within a message
- The protection of traffic flow from analysis
  - This requires that an attacker not be able to observe the source and destination, frequency, length, or other characteristics of the traffic on a communications facility

Data Integrity



Nonrepudiation

- Prevents either sender or receiver from denying a transmitted message
- When a message is sent, the receiver can prove that the alleged sender in fact sent the message
- When a message is received, the sender can prove that the alleged receiver in fact received the message

Security Services (X.800)

AUTHENTICATION

The assurance that the communicating entity is the one that it claims to be.

Peer Entity Authentication
Used in association with a logical connection to provide confidence in the identity of the entities connected.

Data-Origin Authentication
In a connectionless transfer, provides assurance that the source of received data is as claimed.

ACCESS CONTROL

The prevention of unauthorized use of a resource (i.e., this service controls who can have access to a resource, under what conditions access can occur, and what those accessing the resource are allowed to do).

DATA CONFIDENTIALITY

The protection of data from unauthorized disclosure.

Connection Confidentiality
The protection of all user data on a connection.

Connectionless Confidentiality
The protection of all user data in a single data block

Selective-Field Confidentiality
The confidentiality of selected fields within the user data on a connection or in a single data block.

Traffic-Flow Confidentiality
The protection of the information that might be derived from observation of traffic flows.

DATA INTEGRITY

The assurance that data received are exactly as sent by an authorized entity (i.e., contain no modification, insertion, deletion, or replay).

Connection Integrity with Recovery
Provides for the integrity of all user data on a connection and detects any modification, insertion, deletion, or replay of any data within an entire data sequence, with recovery attempted.

Connection Integrity without Recovery
As above, but provides only detection without recovery.

Selective-Field Connection Integrity
Provides for the integrity of selected fields within the user data of a data block transferred over a connection and takes the form of determination of whether the selected fields have been modified, inserted, deleted, or replayed.

Connectionless Integrity
Provides for the integrity of a single connectionless data block and may take the form of detection of data modification. Additionally, a limited form of replay detection may be provided.

Selective-Field Connectionless Integrity
Provides for the integrity of selected fields within a single connectionless data block; takes the form of determination of whether the selected fields have been modified.

NONREPUDIATION

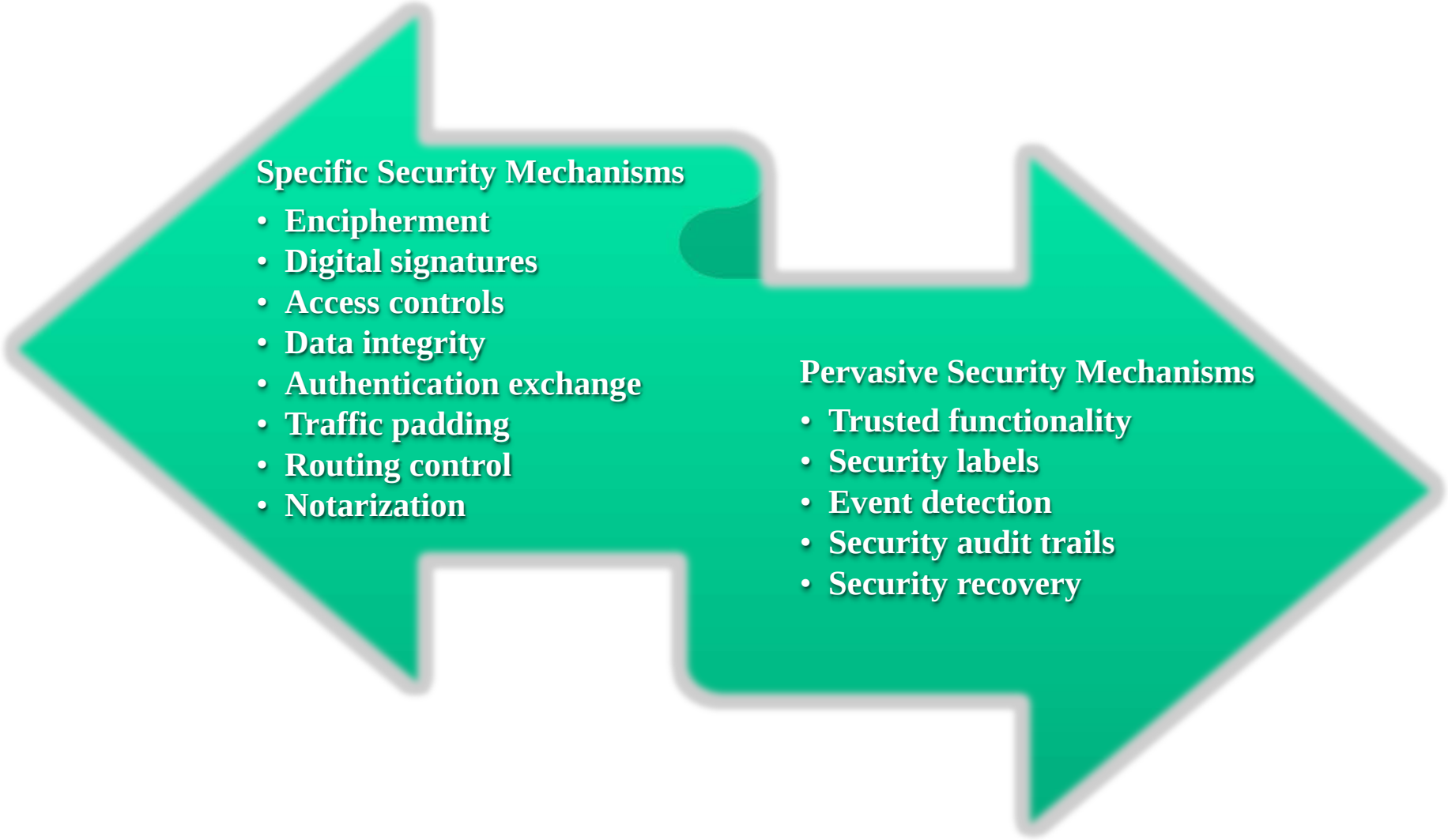
Provides protection against denial by one of the entities involved in a communication of having participated in all or part of the communication.

Nonrepudiation, Origin
Proof that the message was sent by the specified party.

Nonrepudiation, Destination
Proof that the message was received by the specified party.

(This table is found on page 18 in textbook)

Security Mechanisms (X.800)

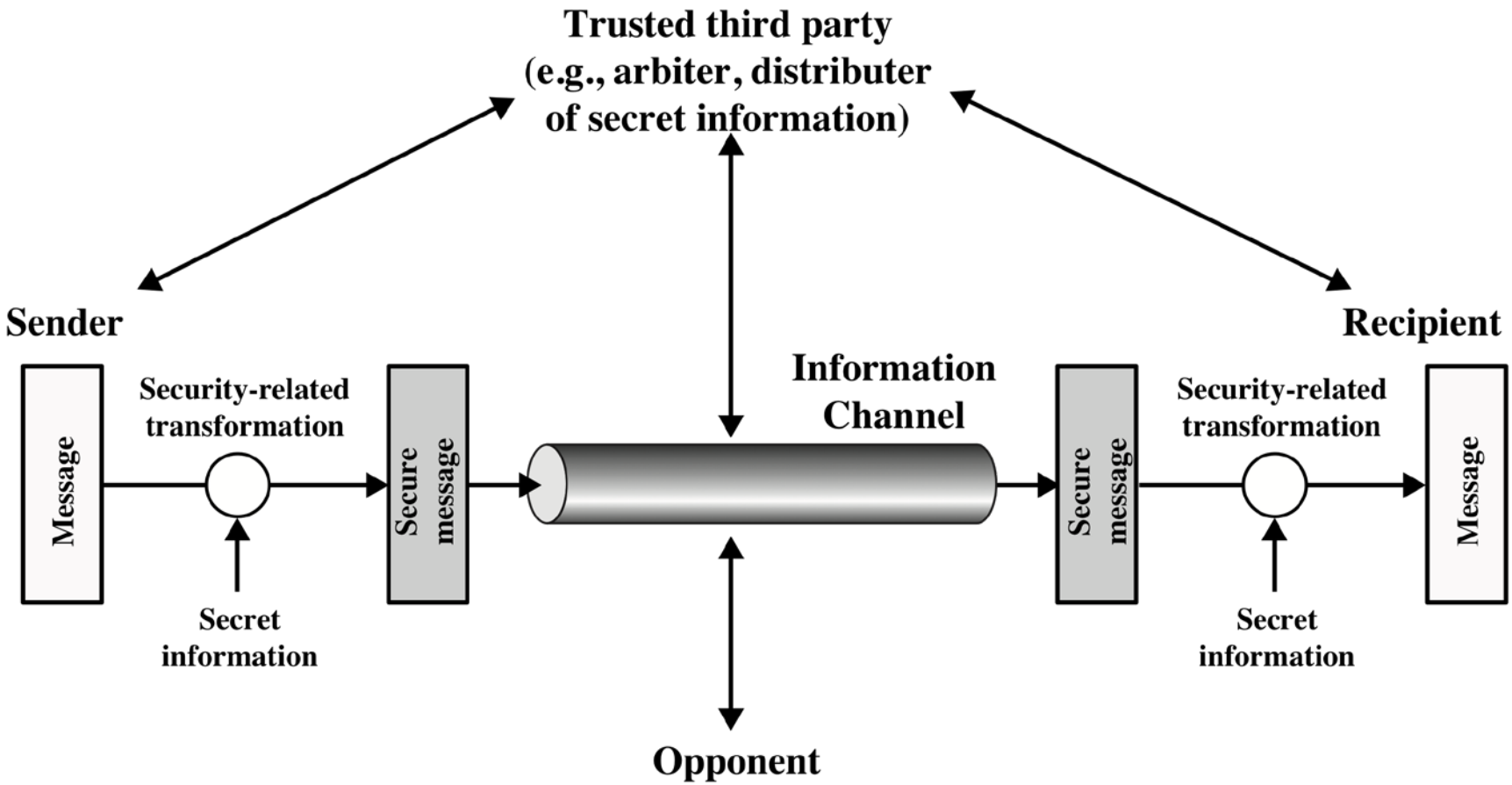


Security Mechanisms (X.800)

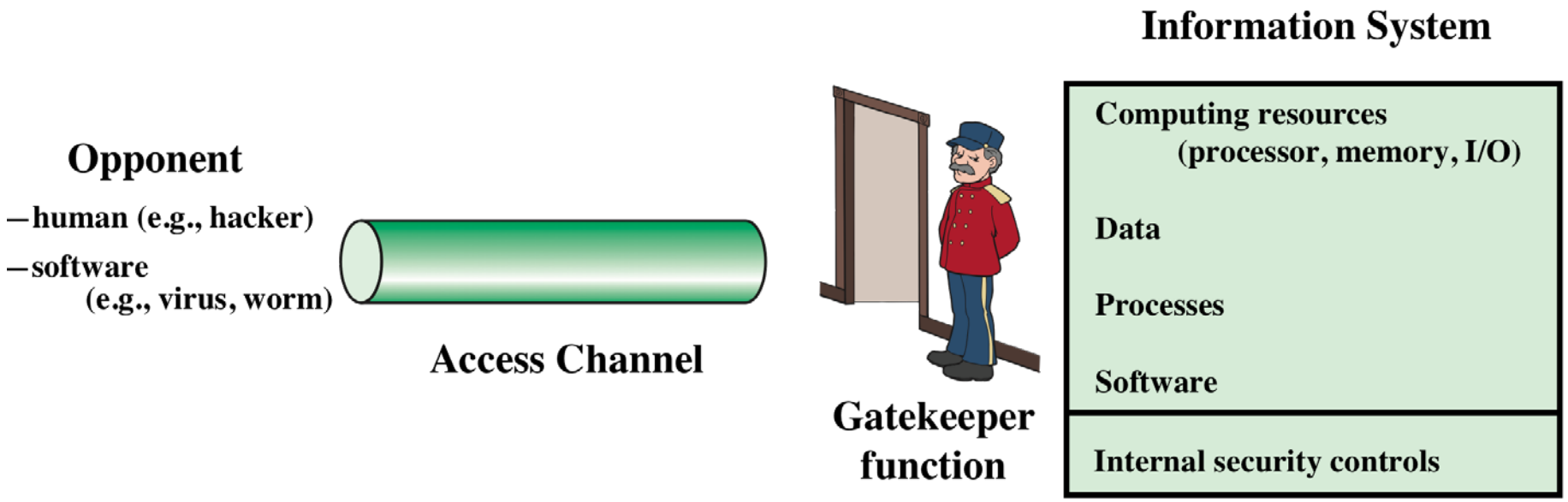
SPECIFIC SECURITY MECHANISMS	PERVASIVE SECURITY MECHANISMS
May be incorporated into the appropriate protocol layer in order to provide some of the OSI security services.	Mechanisms that are not specific to any particular OSI security service or protocol layer.
Encipherment The use of mathematical algorithms to transform data into a form that is not readily intelligible. The transformation and subsequent recovery of the data depend on an algorithm and zero or more encryption keys.	**Trusted Functionality** That which is perceived to be correct with respect to some criteria (e.g., as established by a security policy).
Digital Signature Data appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery (e.g., by the recipient).	**Security Label** The marking bound to a resource (which may be a data unit) that names or designates the security attributes of that resource.
Access Control A variety of mechanisms that enforce access rights to resources.	**Event Detection** Detection of security-relevant events.
Data Integrity A variety of mechanisms used to assure the integrity of a data unit or stream of data units.	**Security Audit Trail** Data collected and potentially used to facilitate a security audit, which is an independent review and examination of system records and activities.
Authentication Exchange A mechanism intended to ensure the identity of an entity by means of information exchange.	**Security Recovery** Deals with requests from mechanisms, such as event handling and management functions, and takes recovery actions.
Traffic Padding The insertion of bits into gaps in a data stream to frustrate traffic analysis attempts.	
Routing Control Enables selection of particular physically secure routes for certain data and allows routing changes, especially when a breach of security is suspected.	
Notarization The use of a trusted third party to assure certain properties of a data exchange.	

(This table is found on pages 20-21 in textbook)

Model for Network Security



Network Access Security Model



Unwanted Access

- Placement in a computer system of logic that exploits vulnerabilities in the system and that can affect application programs as well as utility programs such as editors and compilers
- Programs can present two kinds of threats:
  - Information access threats
    - Intercept or modify data on behalf of users who should not have access to that data
  - Service threats
    - Exploit service flaws in computers to inhibit use by legitimate users