Introduction to Information Security

Daniel Bosk

Department of Information and Communication Systems,
Mid Sweden University, SE-851 70 Sundsvall.

24th April 2017

- The main purpose of security is to protect assets.
- In our *(security) policy* we define what is to be accomplished, the goals, e.g. who may access what asset and how.
- We then have *mechanisms* to help us enforce our policy, e.g. cryptography.

- Each of our mechanisms we can say is more or less *trustworthy*: A trustworthy mechanism will not break our security policy.
- The aim of this course is to give you an idea of how to determine what is a trustworthy mechanism.

- Our protection strategies can be divided into the following:
  - Prevention, taking measures that prevent your assets from being damaged.
  - Detection, taking measures that allow detection of when, how, and by who an asset has been damaged.
  - Reaction, taking measures that allow to recover assets or recover from damage to assets.

Example (Private property)

Prevention	Locks on doors, window bars, surrounding walls, ...
Detection	Stolen items are missing, burglar alarms, video surveillance, ...
Reaction	Call the police, replace stolen items (insurance?), ...

Example (E-commerce)

Prevention	Encrypt orders, rely on merchants checking identities, ...
Detection	An unauthorised transaction appears on your bank statement, ...
Reaction	Complain to bank, ask for new card, ...

Definition (Data and information [Han73])

[Data is the] Physical phenomena chosen by convention to represent certain aspects of of our conceptual and real world. *The meanings we assign to data are called information.* [my emphasis] Data is used to transmit and store information and to derive new information by manipulating the data according to formal rules.

Example

- $x + 1 = 0 \iff x = -1$: the two equations are data, we use the formal rules of mathematics to derive the value of x.
- Timestamps of TCP packets with port set to 80 or 443, formal rules of statistical analysis.

Confidentiality Concerns unauthorised disclosure of information.

Integrity Concerns unauthorised modification.

Availability Concerns unauthorised withholding of information or resources.

Authenticity Concerns the identity of principals in systems.

Accountability (non-repudiation) Concerns proof that some principal was involved in some event.

...

Confidentiality Concerns unauthorised disclosure of information.

Integrity Concerns unauthorised modification.

Availability Concerns unauthorised withholding of information or resources.

Authenticity Concerns the identity of principals in systems.

Accountability (non-repudiation) Concerns proof that some principal was involved in some event.

...

Confidentiality

- Prevent unauthorised *reading*.
- Think about what to hide: the content of a document, or the document's existence?

Privacy

- Privacy is different, this concerns protection of personal information.
- The users should be in control of their data and of the information about their activities.
- Varying definitions: also the right to be left alone.

Integrity

- Prevent unauthorised *writing*.
- Data integrity: "The state that exists when computerized data is the same as that in the source document and has not been exposed to accidental or malicious alteration or destruction."
- Concerns detection and correction of intentional and unintentional modifications of data.

Integrity

- Clark and Wilson: "No user of the system, even if authorized, may be permitted to modify data in such a way that assets or accounting records of the company are lost or corrupted."
- I.e. make sure that everything is as it is supposed to.
- Integrity is a prerequisite to many other security services.

Availability

- This is the property of being available and usable upon demand by an authorized principal.
- Denial of Service (DoS) is an attack on availability which prevents authorized access to resources or the delaying of time-critical operations.
- A very important part of security, unfortunately not many methods for accomplishing this are available.
- Distributed Denial of Service (DDoS) gets much attention, this can also be seen as a reliability problem (unintentional).

Availability

Example (Smurf attack)

- Attacker sends ICMP echo request to a broadcast address with the victim's address as the spoofed sender address.
- The echo request is distributed to all nodes in the broadcast range.
- All nodes replies to the echo request, and the replies are sent to the victim – the victim is flooded.
- Depending on the size of the broadcast range, there is a considerable amplification.

Example (DDoS)

- Attackers infect $x \cdot 10^5$ devices (smartphones, IoT-stuff) with malware.
- Attackers commands all devices to send requests to a given address.
- No one can communicate with that address under that load.

Non-Repudiation

- *Non-repudiation* concerns unforgeable evidence that a specific action has occurred.
- *Non-repudiation of origin*: protects against a sender of data denying data was sent.
- *Non-repudiation of delivery*: protects against a receiver denying data was received.
- Note: what is meant my received? E.g. a mail delivered to your mailbox.
- Also note that a simple audit log doesn't necessarily give non-repudiation, it might have been forged by the system administrator.
- It's usually accomplished by using crypto.

Non-Repudiation

- A commonly found definition: "Non-repudiation provides irrefutable evidence about some event."
- Is anything ever irrefutable?
- Non-repudiation generates mathematical evidence.
- This does not necessarily mean it is accepted, e.g. in court of law.
- In Sweden, this is regulated in law SFS 2000:832 (among others).

Reliability

- Reliability addresses the consequences of unintentional errors.
- On a PC (offline), you are in control of the software components sending input to each other.
- The aim is to avoid mistakes.

Security

- Once online, hostile adversaries can provide input.
- Protection against mistakes is not enough — they will ensure to make them for you.
- And they will do things that you can never accomplish by mistake.

Reliability

- Reliability addresses the consequences of unintentional errors.
- On a PC (offline), you are in control of the software components sending input to each other.
- The aim is to avoid mistakes.

Security

- Once online, hostile adversaries can provide input.
- Protection against mistakes is not enough — they will ensure to make them for you.
- And they will do things that you can never accomplish by mistake.

- To make software more reliable, it is tested against typical usage patterns.
- To make software more secure, it has to be tested against non-typical usage patterns.
- In fact, you can never be sure the software is secure by just testing – you need to prove it secure.

Situation

Security-unaware users have specific security requirements but no security expertise.

Dilemma

- If you provide them with a standard ("best-practice") solution it might not meet their requirements.
- If you want to tailor your solution to the users' needs, they may be unable to tell you what they require.

- The other dilemma is the conflict between security and usability.
- Security mechanisms may need additional computational resources.
- Security interferes with the ordinary working pattern which users are accustomed to.
- Effort has to be put into managing security — or we do security correct and aligned with usability!

1. Where to focus security controls?
2. Where to place security controls?
3. Complexity or assurance?
4. Centralised or decentralised control?
5. Blocking access to the layer below?

- Focus of control may be on data, operations, or users.
- If we look at the control of integrity, its requirements may refer to rules on:
  - Format and content of data items, e.g. account balance must be integer.
  - Operations that may be performed on a data item, e.g. credit, debit and transfer.
  - Users who are allowed access to a data item, e.g. account holder and bank clerk.

- Often the location of a security mechanism in the man–machine scale correlates with its complexity.
- Generic mechanisms are simple, applications are usually feature rich.
- Back to the fundamental dilemma:
  - Simple generic mechanisms may not match specific security requirements.
  - To choose the right features from a rich selection, you need to be a security expert.
  - Security-unaware users are at a loss.

- Within the domain of a security policy, the same controls should be enforced everywhere.
- Having a centralized entity to do this makes it easy to achieve uniformity, however, this entity may become a bottleneck.
- A distributed solution might be more efficient, however, then you must ensure they all enforce consistently with each other.

- Every security mechanism defines a *security perimeter*.
- The parts of a system which can malfunction without breaking the mechanism are said to be outside the perimeter.
- The parts of the system that can disable the mechanism are within the perimeter.

- Attackers will try to bypass security mechanisms.
- How do you ensure an attacker cannot get access to the layer below the security mechanism?

- Recovery tools, read the sectors directly from the disk; logical access control is implemented in the operating system.
- Buffer overruns, a value assigned to a variable is too large for the memory buffer allocated; memory allocated for other variables may be overwritten.
- Side-channel analysis, look at the time different operations take to perform, look at power consumption.
- JavaScript to perform security checks? The client can use a Web page without JavaScript enabled.

[Han73] Per Brinch Hansen. *Operating system principles*. Prentice-Hall, Inc., 1973.