

# Overview of Computer Security<sup>1</sup>

Hans-Wolfgang Loidl

<http://www.macs.hw.ac.uk/~hwloidl>

School of Mathematical and Computer Sciences  
Heriot-Watt University, Edinburgh



<sup>1</sup>Based on David Aspinall's slides on "Computer Security Landscape" and Goodrich's & Tamassia's textbook

- 1 Overview
- 2 Security properties and their protection
- 3 Reasons for security failure
- 4 Implementing a security solution
- 5 References

## Basic concepts from the Common Criteria (CC)

- Security is about protecting **assets** from **threats**.
- Threats are the potential for abuse of assets.
- **Owners** value assets and want to protect them.
- **Threat agents** also value assets, and seek to abuse them.
- Owners analyse threats to decide which apply; these are **risks** that can be costed.
- This helps select **countermeasures**, which reduce **vulnerabilities**.
- Vulnerabilities may remain leaving some residual risk; owners seek to minimise that risk, within other constraints (feasibility, expense).

## Designing secure systems

Steps in designing secure systems:

- 1 Develop sound and accurate **security models**.
- 2 Define **security properties** that need to be fulfilled.
- 3 Anticipate **attacks** on the (modelled) system.
- 4 Involve the security properties in all stages of system **design**.
- 5 The implementation needs to be **verified** or at least rigorously **tested**.
- 6 Once deployed, the system needs to be continuously **monitored**.

## Security properties

### Security Properties to Ensure

- confidentiality:** *avoidance of unauthorised disclosure of information*
- integrity:** *information has not been altered in an unauthorised way*
- availability:** *information is accessible and modifiable in a timely fashion*
- assurance:** *how is trust provided and managed in computer systems*
- accountability:** *actions traceable to those responsible*
- authentication:** *information or services available only to authorised identities*

## Confidentiality, privacy and secrecy

*Information is not disclosed to unauthorised principals*

- Confidentiality is characterised as preventing the **unauthorised reading of data**, when considering access control systems. More generally and subtly, it implies **unauthorised learning of information**.
- Confidentiality presumes a notion of authorised party, or more generally, a **security (access) policy** saying who or what can access our data. The security policy is used for access control.
- Usage: **privacy** refers to confidentiality for individuals; **secrecy** to confidentiality for organizations. Privacy is sometimes used to mean **anonymity**, keeping one's identity private.
- Example violations: your medical records are obtained by a potential employer without your permission

## Integrity

*Information has not been altered in an unauthorised, malicious way*

- Integrity has more general meanings elsewhere, but in computer security we are concerned with preventing the possibly **malicious** alteration of data, by someone who is not authorised to do so.
- Integrity in this sense can be characterised as the **unauthorised writing of data**. Again, this presumes a policy saying who or what is allowed to alter the data.
- **Techniques** to ensure integrity encompass: checksums, backups, data correcting code.
- Example violation: an on-line payment system alters an electronic cheque to read £10000 instead of £100.00

## Availability

*Information or services are accessible and modifiable in a timely fashion by those authorised to do so*

- Threats to availability cover many kinds of external environmental events (e.g. **fire**, **pulling the server plug**) as well as accidental or **malicious** attacks in software (e.g. **infection with a debilitating virus**).
- Computer security focuses on the second kind of threat, rather than considering general forms of fault-tolerance or dependability assurance.
- Ensuring availability means preventing **denial of service** (DoS) attacks, insofar as this is possible. It's possible to fix attacks on faulty protocols, but attacks exhausting available resources are harder, since it can be tricky to distinguish between an attack and a legitimate use of the service.
- Example violations: the deadly distributed DoS (DDoS) attacks against on-line services; interfering with IP routing.

## Assurance

*How is trust provided and managed in computer systems?*

- **Policies** define expectations that people or systems must fulfill. Example: define *whether to play (or copy) music from an on-line store*.
- **Permissions** describe behaviours that are allowed by the agents that interact with a person or system. Example: define *how to play (or copy) music from an on-line store*.
- **Protections** describe mechanisms put in place to enforce permissions and policies. Example: provide mechanisms *how to play (or copy) music from an on-line store*.

In general, **trust management** deals with the design of effective, enforceable policies, methods for granting permissions to trusted users, and the components that can enforce those policies and permissions for protecting and managing the resources of a system.

## Accountability

*Actions are traceable to the party responsible*

- If prevention methods and access controls fail, we may fall back to detection: keeping a **secure audit trail** is important so that actions affecting security can be traced back to the responsible party.
- A stronger form of accountability is **non-repudiation**, when a party cannot later deny some action.
- Creating an audit trail with machine logs is tricky: if a system is compromised, logs may also be tampered with. Ways around this: send log messages to an append-only file, a separate server, or even a physically isolated printer.
- Example violation: an audit trail is tampered with, lost, or cannot establish where a security breach occurred.

## Authentication

*Data or services available only to authorised identities*

- Authentication is **verification of identity** of a person or system.
- Authentication is a prerequisite for allowing access to some people but denying access to others, using an **access control** system. An alternative is to use an previously obtained **credential** or **permission**.
- Authentication methods are often characterised as:
  - something you have** e.g. an entrycard
  - something you know** e.g. a password or secret key
  - something you are** e.g. a signature, biometric.

Also, **where you are** may be implicitly or explicitly checked. Several methods may be combined.

- Example violations: using cryptanalysis to break crypto and learn a secret key; purporting to be somebody else (**identity theft**) using stolen information.

## Security Principles

Some classic security principles (as identified by Saltzer and Schröder, 1975):

- 1 **Economy of mechanism:** Strive for simplicity in your design.
- 2 **Fail-safe defaults:** Default configurations of a system should have a conservative protection scheme. Example: *minimal group membership for new users*. This may be in contention with usability issues: eg. allow to execute any downloaded code in a web browser.
- 3 **Complete mediation:** Every access to a resource must be checked for compliance with a protection scheme. This may be in contention with performance considerations.
- 4 **Open design:** The security and architecture design of a system should be made publicly available. Security should be based only on the secrecy of (private) keys, not on ignorance of the cryptosystem. This is the opposite view from **security-by-obscurity**, which is tempting from a management perspective, but has failed for various reasons in practice.

## Security Principles (cont'd)

- 1 **Separation of privilege:** Multiple conditions should be required to achieve access to restricted resources. For large software systems, this can be extended to separation of components with independent security mechanisms.
- 2 **Least privilege:** Each user or program in a system should operate with the bare minimum of privileges necessary to function properly. Reflected in the military **need-to-know** concept.  
Example: minimal set of permissions for a user program.
- 3 **Least common mechanism:** In multi-user systems, mechanisms allowing resources to be shared should be minimised. This may be in contention with usability issues.

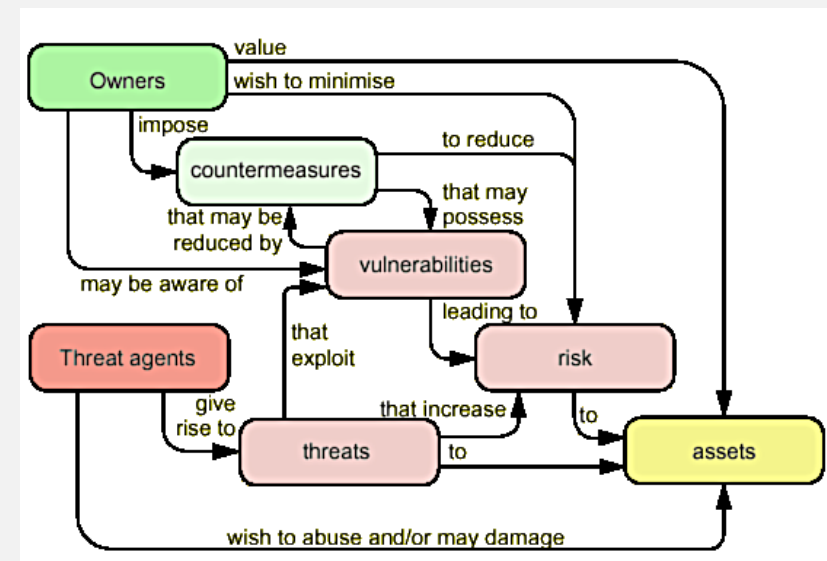
## Security Principles (cont'd)

- 1 **Psychological acceptability:** User interfaces should be well-designed and intuitive, and all security-related settings should adhere to what an ordinary user might expect. Example: software for encrypting and signing email is readily available but rarely used
- 2 **Work factor:** The cost of circumventing a security mechanism should be compared with the resources of an attacker when designing a security scheme. Adjust the security mechanism used, to the security level desired. Rapid changes in technology can radically change the view, eg. security of passwords
- 3 **Compromise recording:** Sometimes it is more desirable to record the detail of an intrusion, rather than to develop sophisticated mechanisms preventing it. Example: CCTV cameras.

## Security in-the-large

- **Security is a whole system issue.**  
The whole system includes at least: software, hardware, physical environment, personnel, corporate and legal structures.
- In this course we focus on the **technology** aspects of security, covering mainly software issues.
- Security evaluation standards like CC have a similar focus, to allow comparison between different security technologies.

## Concepts and relationships (CC version 2.1)



## Threats and attacks

- **Eavesdropping:** interception of information intended for someone else during transmission over a communication channel. Example: packet-sniffer. **Attack on confidentiality.**
- **Alteration:** unauthorised modification of information. Examples: man-in-the-middle-attacks, certain viri. **Attack on integrity.**
- **Denial-of-service:** the interruption or degradation of a data service or of information access. Example: Spam. **Attack on availability.**
- **Masquerading:** the fabrication of information that is purported to be from someone who is not actually the author. Example: phishing, spoofing. **Attack on authenticity, and possibly on confidentiality.**
- **Repudiation:** the denial of commitment or data receipt. Example: back out of an on-line deal. **Attack on assurance**
- **Correlation and traceback:** the integration of multiple data sources and information flows to determine the source of a particular data stream or piece of information. Example: applying data mining on large sets of anonymous transactions or activities. **Attack on anonymity.**

## Protection countermeasures

- **Prevention.** Stop security breaches by system design and using security technologies as defences. Prevention is the most important protection measure. Example: using a firewall to prevent external access to corporate intranets.
- **Detection.** If an attempted breach occurs, make sure it is detected. Particularly pertinent in computer security, where “theft” of data does not imply denial of access for the owner. Logging and MACs (file hashes to detect alteration) are typical detection methods; **intrusion detection systems** actively monitor systems for suspicious behaviour.
- **Response.** If a security breach occurs, have a recovery plan. Responses range from restoring backups through to informing concerned parties or law-enforcement agencies.

## Protecting authentication

Protecting authentication amounts to getting it right.

**Spoofing** authentication opens the door to breaking other security properties, so it is a focal point.

- Example issues:
  - ▶ Time of check to time of use (TOCTOU)
  - ▶ Failures: logging and lock-out
  - ▶ Unilateral vs mutual
- There are **serious drawbacks with passwords:**
  - ▶ Verify knowledge of a secret, not identity.
  - ▶ May be cracked or stolen.
  - ▶ A trade-off: memorability/size versus strength
  - ▶ Single sign-on appealing but delegates trust: “all eggs in one basket”.

## Why does security fail?

- Opposing forces act against system security:
  - ▶ the **complexity** of the system, and inevitable faults;
  - ▶ wide-ranging and unpredictable **human factors**;
  - ▶ the cunning and **expertise and of attackers**.
- Engineering and management aspects impact:
  - ▶ **poor security design** from the outset;
  - ▶ standard **off-the-shelf but insecure** systems;
  - ▶ **changes in environment**; bad feature interaction;
  - ▶ a **failure to invest** in providing security;
  - ▶ other economic **perverse incentives**.

## Opposing forces: complexity

- Complexity is one of the worst enemies of security
  - ▶ all computer systems have bugs and design errors
  - ▶ a proportion of bugs will be security related, leading to accidental breaches or easy exploits
  - ▶ more complex systems have more bugs  $\rightsquigarrow$  more security bugs
- Two approaches to deal with this:
  1. **Limit complexity**, e.g.
    - ▶ use *special purpose* solutions: firewalls on stand-alone machines, or dedicated devices.
    - ▶ use *proven designs* and open protocols.
  2. **Design around complexity**, e.g.
    - ▶ multiple levels of protection: *defence-in-depth*;
    - ▶ invest more in *detection* than *prevention*;
    - ▶ use *end-to-end* and pervasive authentication

## Opposing forces: attackers

- Attackers have many reasons to act maliciously: *play, publicity, theft, fraud, vandalism, surveillance, terrorism.*
- Profile by expertise, resourcing, dedication:
  - ▶ **Script kiddies**. Use downloaded scripts to exploit well-known holes, without much understanding.
  - ▶ **Hobbyist hackers**. More knowledgeable; write own tools, find new flaws; Internet “underground”.
  - ▶ **Determined hackers**. Have a cause, e.g. disgruntled IT professionals, cyber criminals. More directed attacks, perhaps insider knowledge.
  - ▶ **Professional consultants**. E.g. industrial espionage. Highly knowledgeable, well-funded.
  - ▶ **Security services expert**. E.g., national security, law-enforcement. Highly-specialised, maybe unique access, tacit governmental and legal support.

A **risk assessment** can consider profiles of attackers, and how much to protect against them.

## Opposing forces: human factors

Human factors are wide-ranging. People are liable to:

- **sloppy procedure**: e.g. choosing weak passwords, turning off or skipping security checks, ignoring warnings;
- **social engineering attacks**: giving information inadvertently, accidentally, or being corruptible;
- **fail to understand security implications** of actions (e.g. opening unexpected attachments);
- **choose system features over security** (e.g. judge security products by their GUIs);
- **handle exceptional circumstances improperly** (e.g. preferring to believe a security exploit has not happened, so believing the perpetrator’s lie; following on-screen instructions without question).

## Engineering challenges in security

- Software engineering is a difficult activity: *making systems behave in a clearly specified way.*
- Security engineering is, in a sense, even more difficult: *preventing systems misbehaving in many unspecified ways.*
- It’s hard to design tests to evaluate *emergent properties* like the security behaviour of a system in an unknown environment, before deployment.
- Because emergent properties are fragile, a *change in environment* is a common reason for failure.
- Current trends are extending security evaluation into the engineering process, along with techniques for meeting other non-functional requirements.








## Managing security: implementing a solution



A **security analysis** surveys the *threats* which pose *risks* to assets, and then proposes *policy* and *solutions* at an appropriate cost.

- 1 A **threat model** documents the possible threats to a system, imagining all the vulnerabilities which might be exploited.
- 2 A **risk assessment** studies the likelihood of each threat in the system environment and assigns a cost value, to find the risks.
- 3 A **security policy** addresses the threats, and describes a coherent set of countermeasures.
- 4 The costs of countermeasures is compared against the risks, and juggled to make a sensible **trade-off**.
- 5 This allows a **security solution** to be designed, deploying appropriate technologies at an appropriate cost. Partly this is budgeting; but it's also important to spend effort in the right place.

## References

-  David Aspinall et al, University of Edinburgh. “*Computer Security*”, On-line: <http://www.inf.ed.ac.uk/teaching/courses/cs/>
-  Michael T. Goodrich and Roberto Tamassia “*Introduction to Computer Security*”, Addison Wesley, 2011. ISBN-10: 0321512944
-  Common Criteria (CC) for Information Technology Security Evaluation, September 2007. <http://www.niap-ccevs.org/cc-scheme/>.
-  NIST Risk Management Guide for Information Technology Systems, <http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf>
-  Dieter Gollmann. *Computer Security*. John Wiley & Sons, second edition, 2006.

## Recommended Reading

-  Michael T. Goodrich and Roberto Tamassia *Introduction to Computer Security*, Addison Wesley, 2011. ISBN-10: 0321512944  
Chapter 1.
-  Dieter Gollmann. *Computer Security*. John Wiley & Sons, second edition, 2006.  
Chapter 1.