

# Introduction and Landscape

## Computer Security Lecture 1

David Aspinall and Myrto Arapinis

School of Informatics  
University of Edinburgh

13th January 2014

# Outline

Course introduction

Basic concepts

Security properties and their protection

Reasons for security failure

Implementing a security solution

References

# Outline

Course introduction

Basic concepts

Security properties and their protection

Reasons for security failure

Implementing a security solution

References

# What is Computer Security?

- ▶ **Security** is about protecting assets.
- ▶ **Computer Security** concerns assets of computer systems: the information and services they provide.
- ▶ Just as real world physical security systems vary in their security provision (e.g., a building may be secure against certain kinds of attack, but not all), so computer security systems provide different kinds and amounts of security.
- ▶ Computer security is quite vast in scope, touching on many areas besides computer science. In this course we will study the *fundamentals*, some current *internet technologies*, and a little bit about *engineering and management* aspects.

# Lecture plan

- ▶ About **16 lectures covering core topics**:
  - ▶ Risks and threats
  - ▶ Crypto
  - ▶ Protocols
  - ▶ Models
  - ▶ Network
  - ▶ Software
  - ▶ Internet
  - ▶ Usability
- ▶ Many things not included (despite their relevance):
  - ▶ War stories
  - ▶ Legalities
  - ▶ Hardcore crypto math
  - ▶ Security APIs
  - ▶ Economics
  - ▶ Criminology
  - ▶ Firewall HOWTOs
  - ▶ Personal advice
- ▶ Core lectures are in Weeks 1-5 and 6-9.
- ▶ Week 5.X is *Innovative Learning*.

## Tutorials and exercises

- ▶ There will be **4 tutorials**, in Weeks 3, 5, 8 and 10.
- ▶ We will offer **formative feedback** in tutorials, based on submitted answers to the tutorial exercises.
- ▶ The exact schedule and tutorial groups are to be determined, they will be allocated by the ITO and advertised on the course web page.
- ▶ There will be **2 assessed exercises**:
  - ▶ Assignment 1, problem solving on paper
  - ▶ Assignment 2, security testing/debugging
- ▶ These will be issued in good time, and discussed in tutorials after marking.
- ▶ Assessment for the course is based 75% on the exam and 12.5% on each exercise.

## Standard security course advisory

- ▶ *Nothing here is intended as incitement to crack!*
- ▶ Breaking into systems to “demonstrate” security problems at best causes a headache to overworked sysadmins, at worst compromises systems for many users and could lead to **prosecution**.

## Standard security course advisory

- ▶ *Nothing here is intended as incitement to crack!*
- ▶ Breaking into systems to “demonstrate” security problems at best causes a headache to overworked sysadmins, at worst compromises systems for many users and could lead to **prosecution**.
- ▶ If you spot a security hole in a running system, **don't exploit it**, instead consider contacting the relevant administrators confidentially.



## Standard security course advisory

- ▶ *Nothing here is intended as incitement to crack!*
- ▶ Breaking into systems to “demonstrate” security problems at best causes a headache to overworked sysadmins, at worst compromises systems for many users and could lead to **prosecution**.
- ▶ If you spot a security hole in a running system, **don't exploit it**, instead consider contacting the relevant administrators confidentially.
- ▶ But be aware that keeping abreast with latest security patches and methods is difficult; practical security is a matter of weighing up risks, so your advice may be not be quickly acted on.

## Standard security course advisory

- ▶ *Nothing here is intended as incitement to crack!*
- ▶ Breaking into systems to “demonstrate” security problems at best causes a headache to overworked sysadmins, at worst compromises systems for many users and could lead to **prosecution**.
- ▶ If you spot a security hole in a running system, **don't exploit it**, instead consider contacting the relevant administrators confidentially.
- ▶ But be aware that keeping abreast with latest security patches and methods is difficult; practical security is a matter of weighing up risks, so your advice may be not be quickly acted on.
- ▶ This is especially true in a relatively low security environment such as a university, where open access has traditionally been put above security, and resources for sysadmin are very tight.

## Responsible security experiments

- ▶ If you want to experiment with security holes, play with your own machine, or better, your **own private network of machines**.
- ▶ One (mostly) harmless way: use *virtualisation*: e.g., VMWare, VirtualBox, KVT/Xen/UML.
- ▶ If you discover a new security hole in a standard application, or operating system routine that may be running at many sites, then consider contacting the vendor of the software (or vendor of the operating system which contains the software) in the first case. You might also raise the issue in a security forum for discussion, perhaps without providing complete details of the hole.
- ▶ The software vendor or other security experts will be able to confirm or deny, and work can begin on fixing the problem.

# Outline

Course introduction

**Basic concepts**

Security properties and their protection

Reasons for security failure

Implementing a security solution

References

## Basic concepts from the Common Criteria (CC)

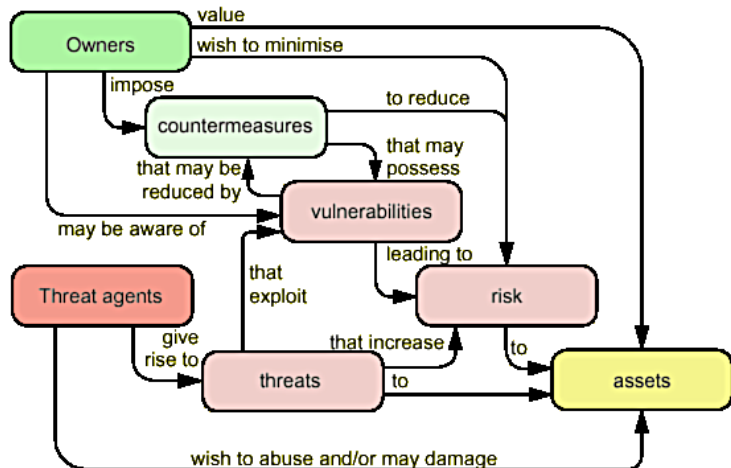
- ▶ Security is about protecting **assets** from **threats**.
- ▶ Threats are the potential for abuse of assets.
- ▶ **Owners** value assets and want to protect them.
- ▶ **Threat agents** also value assets, and seek to abuse them.
- ▶ Owners analyse threats to decide which apply; these are **risks** 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).

## 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).

Note: The CC considers threats from accidental as well as malicious activity; we focus on the latter. The CC considers abuse leading to loss of **confidentiality**, **integrity** or **availability**; we also consider the security properties of **accountability** and **authentication**.

# Concepts and relationships (CC version 2.1)



# Outline

Course introduction

Basic concepts

**Security properties and their protection**

Reasons for security failure

Implementing a security solution

References



# Security properties

## Security Properties to Ensure

<b>confidentiality</b>	<i>no improper information gathering</i>
<b>integrity</b>	<i>data has not been (maliciously) altered</i>
<b>availability</b>	<i>data/services can be accessed as desired</i>
<b>accountability</b>	<i>actions traceable to those responsible</i>
<b>authentication</b>	<i>user or data origin accurately identifiable</i>

- ▶ Remember **security is a whole system issue**.  
The whole system includes at least: software, hardware, physical environment, personnel, corporate and legal structures.
- ▶ We will be more restrictive, and some security evaluation standards like CC are deliberately so, to allow comparison between different security technologies.

## Protection countermeasures

- ▶ **Prevention.** Stop security breaches by system design and using security technologies as defences. E.g., using a firewall to prevent external access to corporate intranets. Prevention is often the most important protection measure.

## Protection countermeasures

- ▶ **Prevention.** Stop security breaches by system design and using security technologies as defences. E.g., using a firewall to prevent external access to corporate intranets. Prevention is often the most important protection measure.
- ▶ **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.

## Protection countermeasures

- ▶ **Prevention.** Stop security breaches by system design and using security technologies as defences. E.g., using a firewall to prevent external access to corporate intranets. Prevention is often the most important protection measure.
- ▶ **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.** In case a security breach occurs, have a recovery plan. Responses range from restoring from backups or claiming on insurance, through to informing stakeholders and law-enforcement agencies.

# Confidentiality, privacy and secrecy

*Information is not learned by unauthorized principals*

- ▶ Confidentiality is characterised as preventing the *unauthorized reading of data*, when considering access control systems. More generally and subtly, it implies *unauthorized learning of information*.
- ▶ Confidentiality presumes a notion of authorized 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.

# Confidentiality, privacy and secrecy

*Information is not learned by unauthorized principals*

- ▶ Confidentiality is characterised as preventing the *unauthorized reading of data*, when considering access control systems. More generally and subtly, it implies *unauthorized learning of information*.
- ▶ Confidentiality presumes a notion of authorized 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; a competitor steals Cadwaladers' secret ice-cream formula.

# Integrity

*Data has not been maliciously altered*

- ▶ 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 authorized to do so.
- ▶ Integrity in this sense can be characterised as the *unauthorized writing of data*. Again, this presumes a policy saying who or what is allowed to alter the data.

# Integrity

*Data has not been maliciously altered*

- ▶ 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 authorized to do so.
- ▶ Integrity in this sense can be characterised as the *unauthorized writing of data*. Again, this presumes a policy saying who or what is allowed to alter the data.
- ▶ **Example violation: an on-line payment system alters an electronic cheque to read £10000 instead of £100.00**



# Availability

*Data or services are accessible as expected*

- ▶ 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.

# Availability

*Data or services are accessible as expected*

- ▶ 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.

# Accountability

*Actions are recorded and can be traced 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.

# Accountability

*Actions are recorded and can be traced 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 authorized 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.

# Authentication

*Data or services available only to authorized 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.

# 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

For user authentication, **passwords** are a cornerstone.

# 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

For user authentication, **passwords** are a cornerstone.

- ▶ But passwords have serious drawbacks:
  - ▶ Verify knowledge of a secret, not identity
  - ▶ May be cracked or stolen
  - ▶ A trade-off: memorability/size versus strength
  - ▶ Single sign-on is appealing but delegates trust: “all eggs in one basket”.



# Outline

Course introduction

Basic concepts

Security properties and their protection

**Reasons for security failure**

Implementing a security solution

References

# 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 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.

# Investing in security

- ▶ Security may be a low priority, either deliberately or unintentionally:
  - ▶ projects have limited budgets and users (unaware or without interest in security) may prefer features over additional security;
  - ▶ understanding and providing security requires expertise which may be unavailable/expensive;
  - ▶ security risks are judged to be small.
- ▶ More generally, economic incentives act in many ways to affect security, e.g.:
  - ▶ First-mover advantage: release software too early
  - ▶ Inhibit switching ↔ proprietary, obscure solutions.
  - ▶ DRM: inconvenience encourages piracy, file sharing.
  - ▶ Asymmetric knowledge: economic advantage in keeping flaws private may outweigh altruism.

# Outline

Course introduction

Basic concepts

Security properties and their protection

Reasons for security failure

**Implementing a security solution**

References



## 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.
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.

# Review

- ▶ Course introduction and format
- ▶ Basic security **concepts** you should understand:
  - ▶ assets, threats, vulnerabilities, countermeasures
- ▶ Security **properties**, you should explain:
  - ▶ confidentiality, integrity, availability, accountability, authentication
- ▶ Basic **process** of security implementation
- ▶ Reasons for failure of security
- ▶ **Next**: lecture block on cryptography.

# Outline

Course introduction

Basic concepts

Security properties and their protection





Reasons for security failure

Implementing a security solution

References

## References

Basic definitions from ISO Common Criteria and early chapters of any text, e.g. Gollmann.

-  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, third edition, 2010.
-  Ross Anderson. *Why Information Security is Hard – An Economic Perspective*. Available from <http://www.cl.cam.ac.uk/~rja14/econsec.html>.

### Recommended Reading

NIST Guide (concepts, not detail), Anderson's paper.