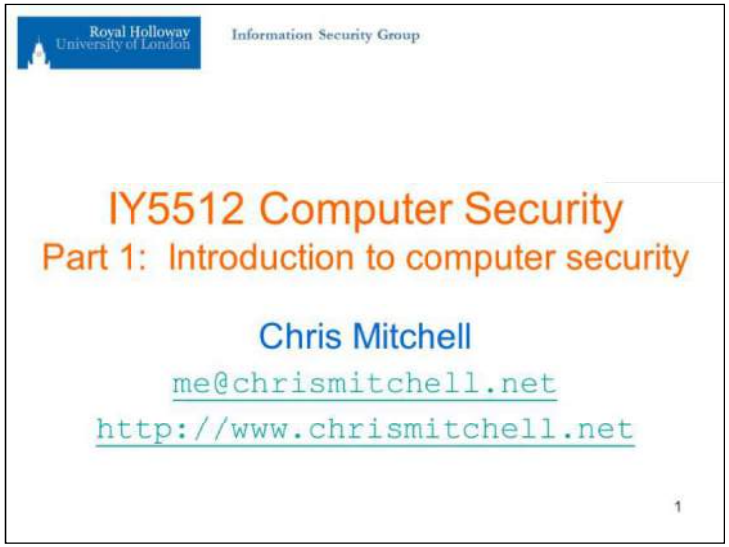
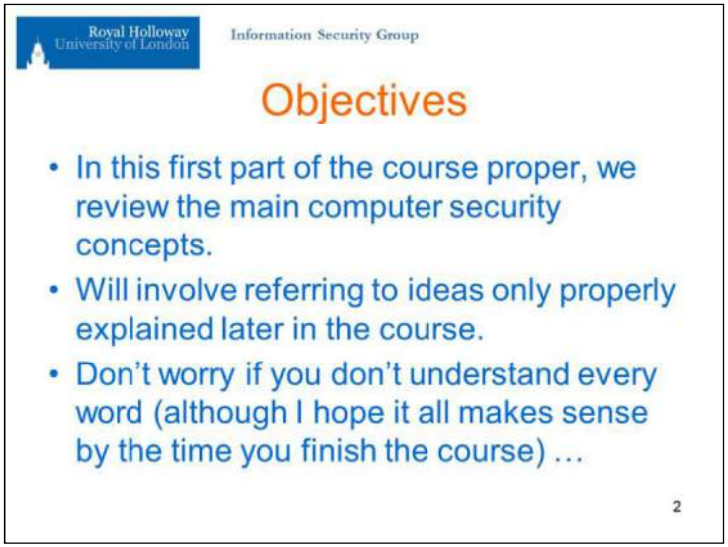


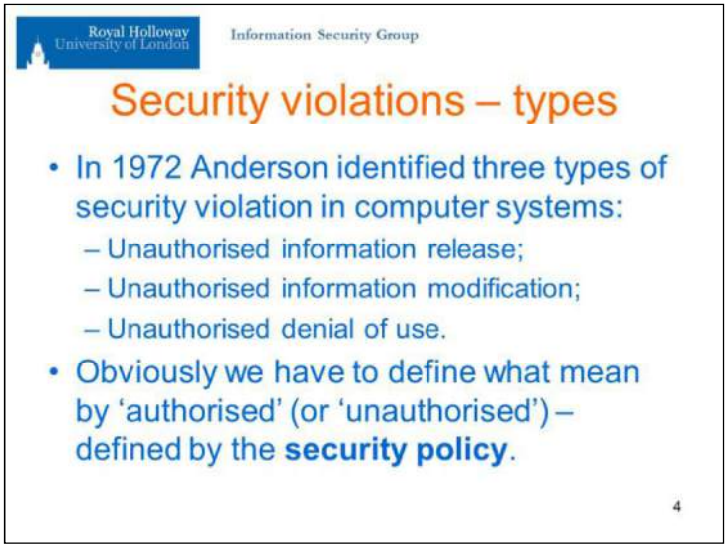
We start the course proper by providing an overview of the security problems that can arise in a computer system.



In this first part of the course proper, the main goal is to review the main computer security concepts. This will involve referring to ideas that are only properly explained later in the course.

So don't worry if you don't understand every word at this stage, although I would hope that it all makes sense by the time you finish the course …

We start by very briefly reviewing some of the threats that apply to computer systems.



In 1972 James Anderson identified three different types of security violation in computer systems:

- unauthorised information release;
- unauthorised information modification;
- unauthorised denial of use.

Obviously we have to define what we mean by 'authorised' (or 'unauthorised'). This is defined by the security policy in force. (Security policies are discussed a little more under the heading 'Implementing security').

The Anderson papers are as follows:

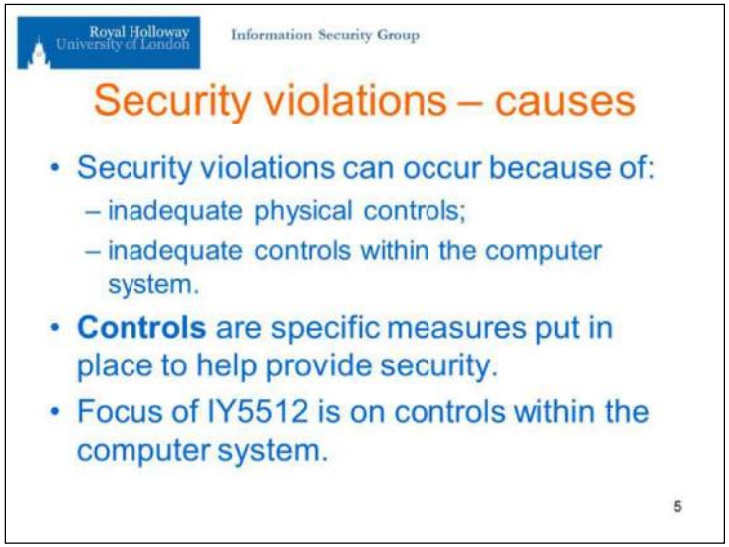
> James P. Anderson, *Computer Security Technology Planning Study*, ESD-TR-73-51, ESD/AFSC, Hanscom AFB, Bedford, MA (Oct. 1972) [NTIS AD-758 206]; Volumes I and II.

These and other seminal computer security papers are available here:

> http://seclab.cs.ucdavis.edu/projects/history/seminal.html

and here:

> http://csrc.nist.gov/publications/history/



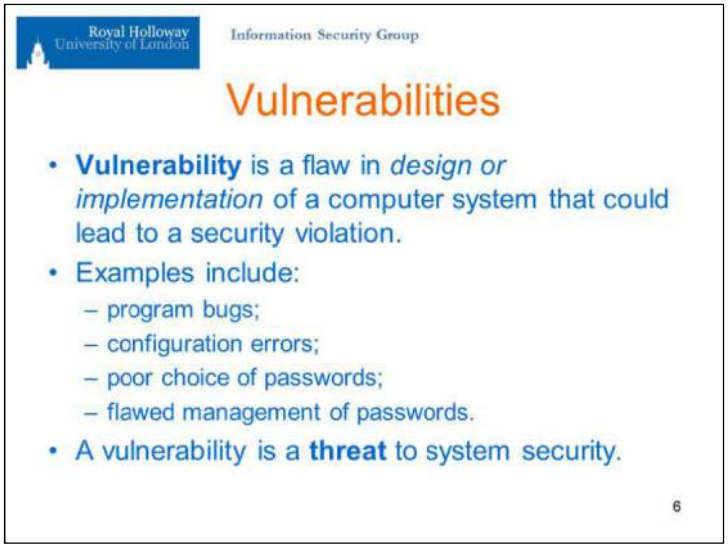
Security violations might occur because of:

• inadequate physical controls (enabling an attacker to gain physical access to the target system);

• inadequate controls within the computer system (enabling an attacker to make unauthorised access to information stored and processed within the system, as well as to the system itself).

Here, and more generally, the word **controls** is used to mean specific measures put in place to help provide security.

In this course we will focus on controls within the computer system.



A **vulnerability** is a flaw in the *design or implementation* of a computer system that could lead to a security violation.

Examples of vulnerabilities include:

• program bugs, i.e. errors in programs (including both design errors and implementation errors);

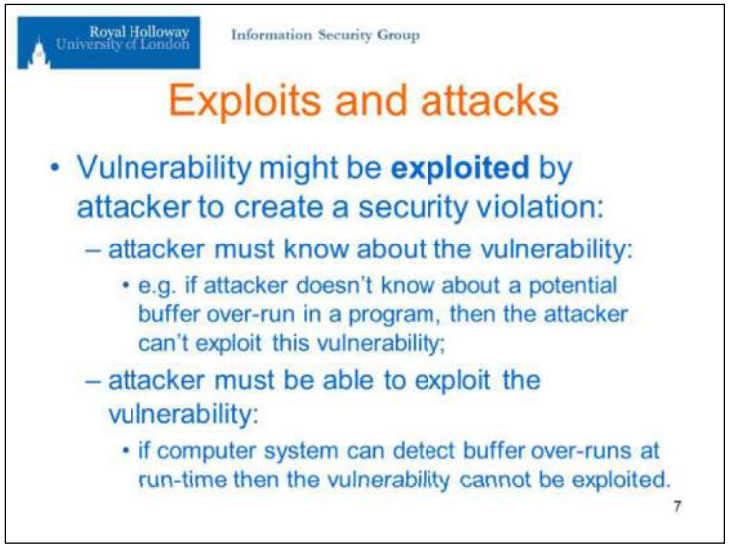
• misuse of program features, i.e. features designed for a valid purpose but which can be misused;

• configuration errors;

• poor choices for passwords;

• flawed management of passwords.

A vulnerability represents a threat to the security of a system.



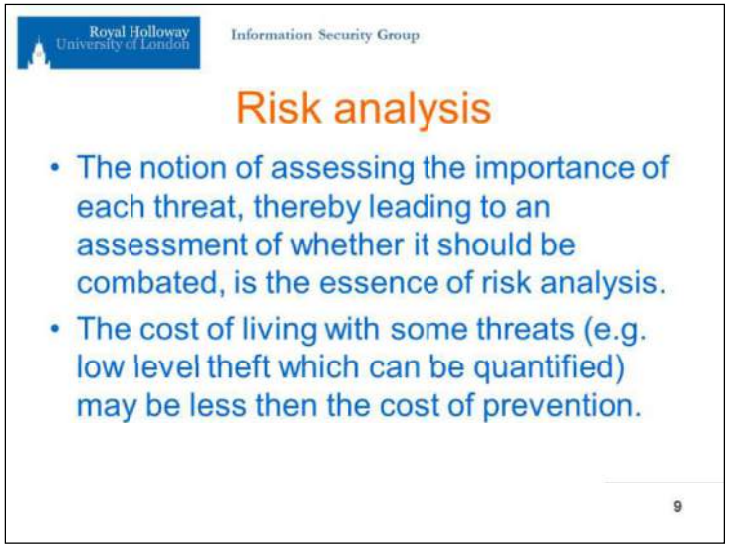
A vulnerability might be **exploited** by an attacker to create a security violation. In order to exploit the vulnerability:

- the attacker must know about the vulnerability:
  - for example, if an attacker doesn't know of the existence of a potential buffer over-run in a program, then the attacker cannot exploit this vulnerability;
- the attacker must be able to exploit the vulnerability:
  - if the computer system can detect buffer over-runs at run-time, then a buffer over-run vulnerability cannot be exploited.

When we discuss security, we will often refer to attackers, i.e. active opponents of system security.

There are, of course, many types of attacker, each of which poses different threats. It is important to be aware of possible attacks from both **insiders** (individuals with legitimate access to parts of system) and **outsiders** (other parties).

Insider attacks are a huge threat in practice, and are difficult to defend against.

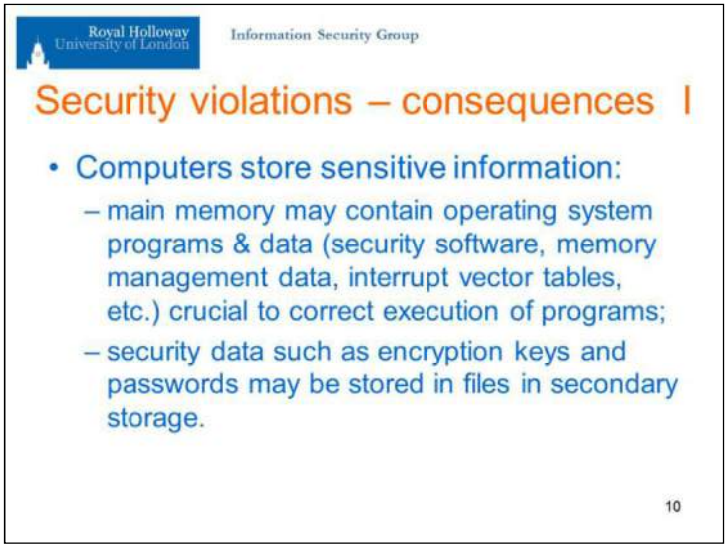


The notion of assessing the importance of each threat to a system, thereby leading to an assessment of whether it should be combated, is the essence of **Risk Analysis**. Risk Analysis involves not just listing possible threats, but also assessing the likelihood of their being realised, and also the potential cost to the system user.

The product of the probability of occurrence and the cost of the threat's realisation gives an indication of the 'cost' of the threat. If combating the threat will cost significantly more than the cost of the threat, then it will probably not be worth taking countermeasures. I.e. the cost of living with some threats (e.g. low level theft which can be quantified) may be less then the cost of combating them.

There exist risk analysis tools, e.g. CRAMM, which mechanise the risk analysis process by leading the user through a series of questions, the answers to which are processed by the tool.

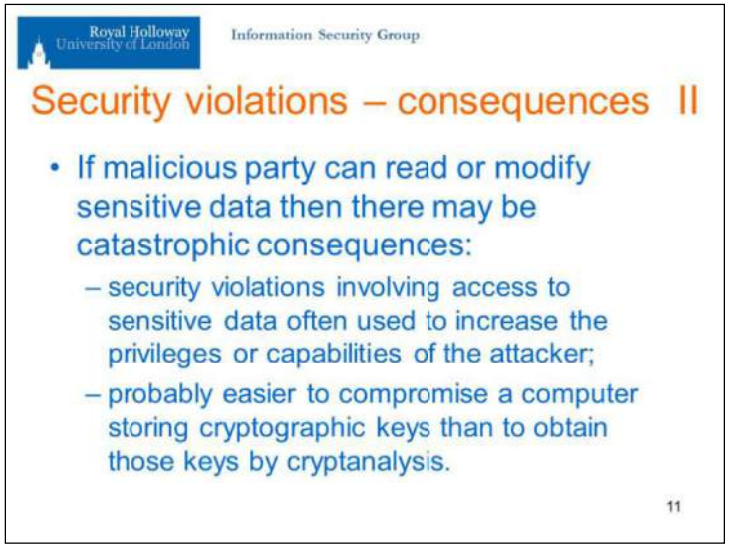
IY5512: Part 1



Computers are used to store sensitive information.

• The computer's main memory may contain operating system programs and data (security software, memory management data, interrupt vector tables, etc.) that are crucial to the correct execution of programs.

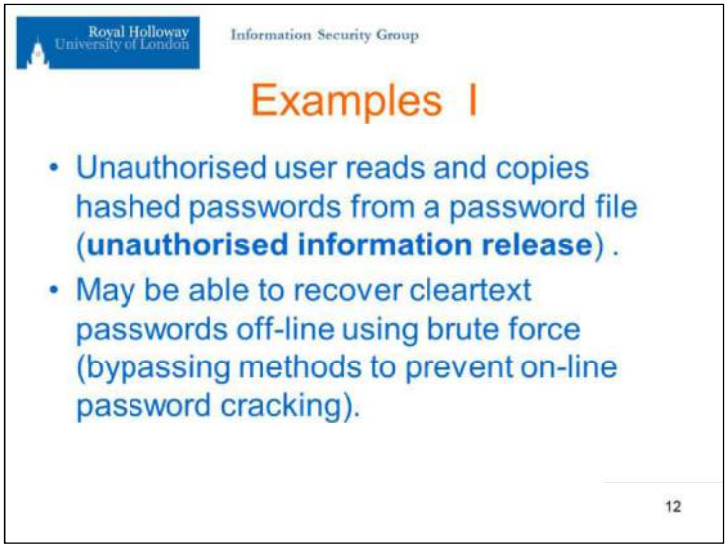
• Security-relevant data, such as encryption keys and passwords, may be stored in files in the computer's secondary storage.



The ability for an unauthorised (malicious) party to read or modify such security-related data can have catastrophic consequences.

• Security violations that involve access to sensitive data are often used to increase the privileges or capabilities of the attacker;

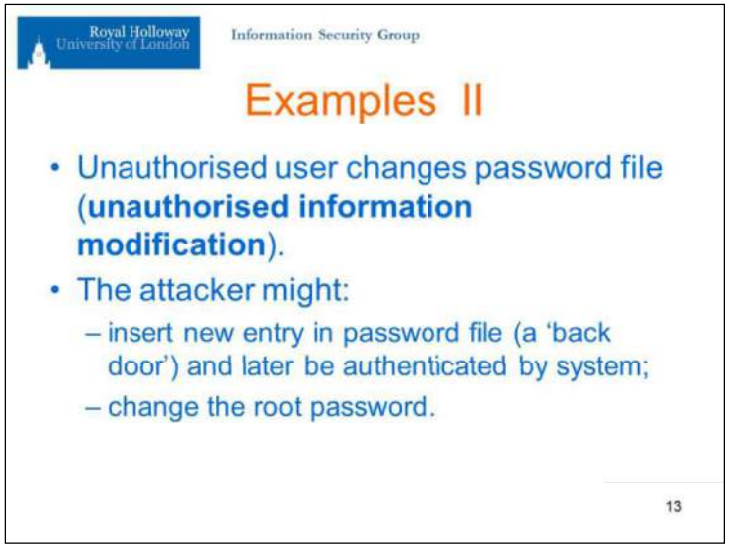
• in many cases it is likely to be easier to compromise a computer that stores cryptographic keys than it is to obtain those keys by cryptanalysis.



Suppose an unauthorised user is able to read and copy hashed (one-way encrypted) passwords from a password file. This is an example of **unauthorised information release**.

Then he/she may be able to recover the plaintext passwords off-line using a brute force (thereby bypassing methods to prevent on-line password cracking). Sadly, such events occur all too often …

Of course, life is even easier for the attacker if the passwords are stored in cleartext (unencrypted) form!

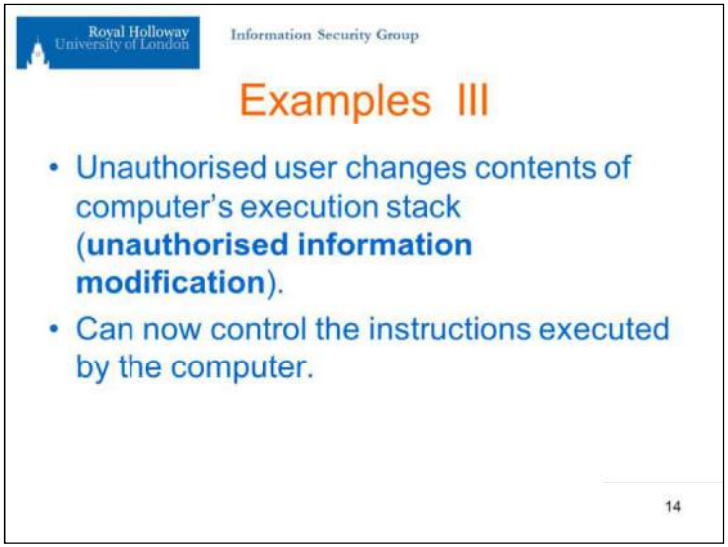


Suppose an unauthorised user is able to change the password file. This is an example of **unauthorised information modification**.

The attacker might:

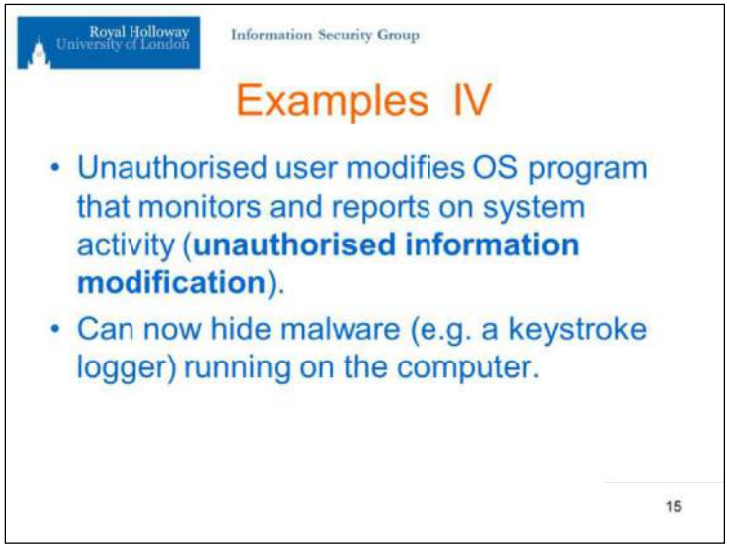
• insert a new entry in the password file (a 'back door') and subsequently be authenticated by the system;

• simply change the root password.



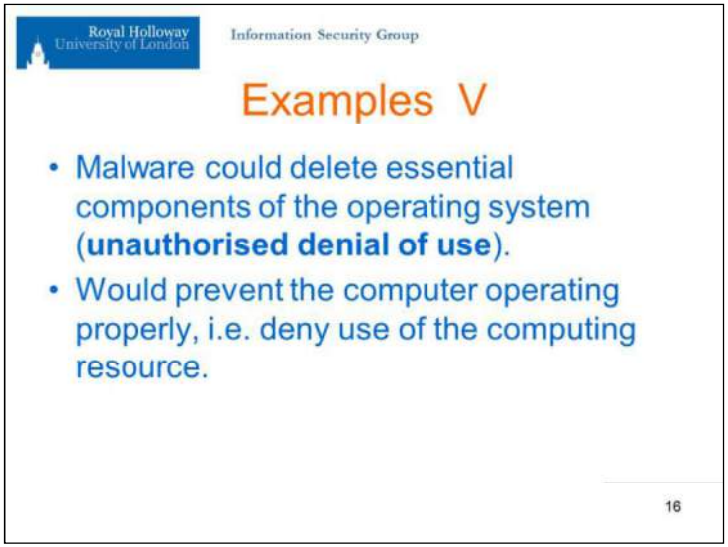
Suppose an unauthorised user changes the contents of a computer's execution stack, i.e. the list of instructions to be executed by the computer. This is an example of **unauthorised information modification**.

The attacker can then control what instructions are executed by the computer.



Suppose an unauthorised user modifies an operating system program that monitors and reports on system activity. This is an example of **unauthorised information modification**.

This may enable the attacker to hide the fact that malware (such as a keystroke logger) is running on the computer.



Suppose that malware deletes essential components of the operating system. This is an example of **unauthorised denial of use**.

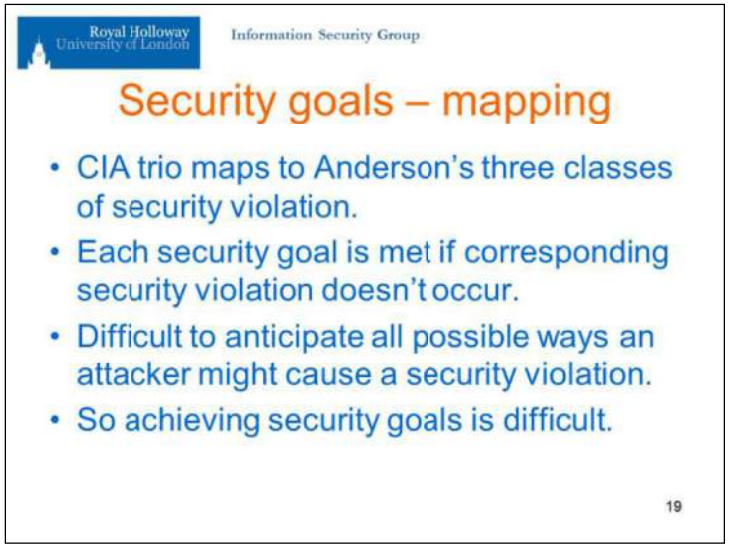
This would prevent the computer operating properly, i.e. it would deny authorised users use of the computing resource.

In order to prevent unauthorised events relating to a computer system, we can identify certain fundamental security objectives, or goals. We now introduce these goals.

There are three well-established goals for computer security:

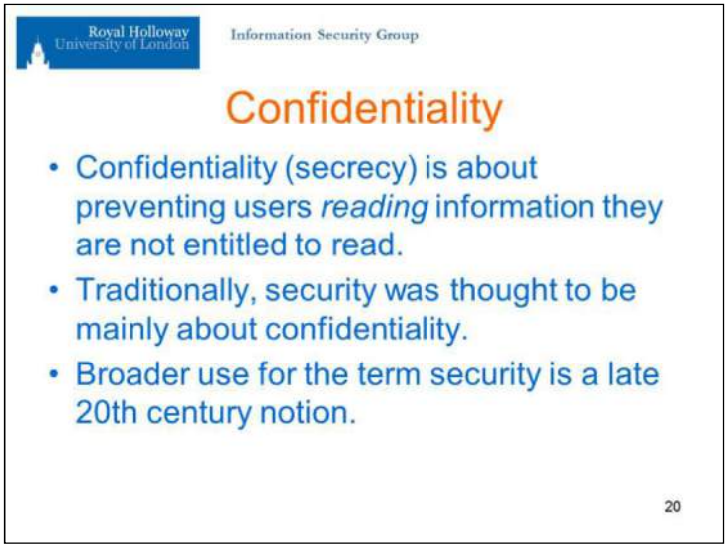
- **Confidentiality** – i.e. prevention of unauthorised information release;
- **Integrity** – i.e. prevention of unauthorised information modification;
- **Availability** – i.e. prevention of unauthorised denial of use.

These are commonly abbreviated to **CIA**.



These security goals map one-to-one to the three types of security violation identified by James Anderson. Each security goal is met if the corresponding security violation does not occur.

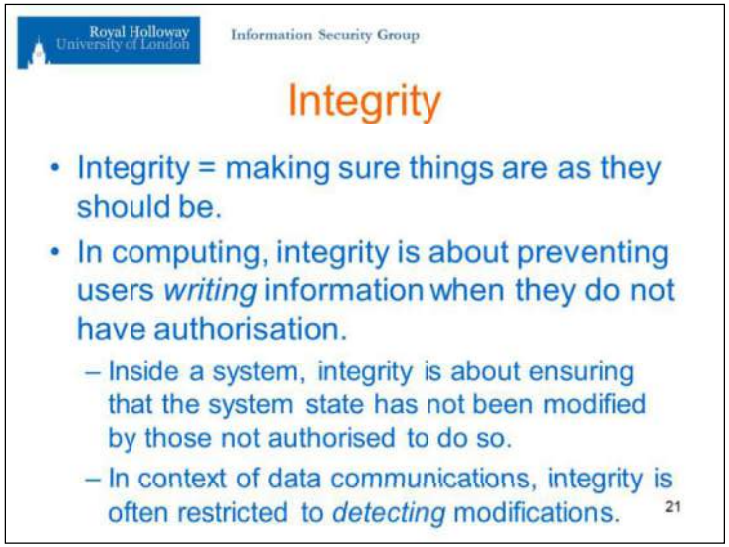
Unfortunately, it is difficult to anticipate all possible ways in which an attacker might (try to) cause a security violation. Achieving these goals is therefore difficult in practice.



Confidentiality (i.e. *secrecy*) is about preventing users *reading* information to which they are not entitled.

Traditionally, the notions of security and confidentiality were often confused. For example, in a military environment, security was traditionally associated with keeping information secret, e.g. by using ciphers to protect communicated information.

The general notion of security as being something broader than just confidentiality is something that has evolved with the broader use of computing from the 1960s onwards.



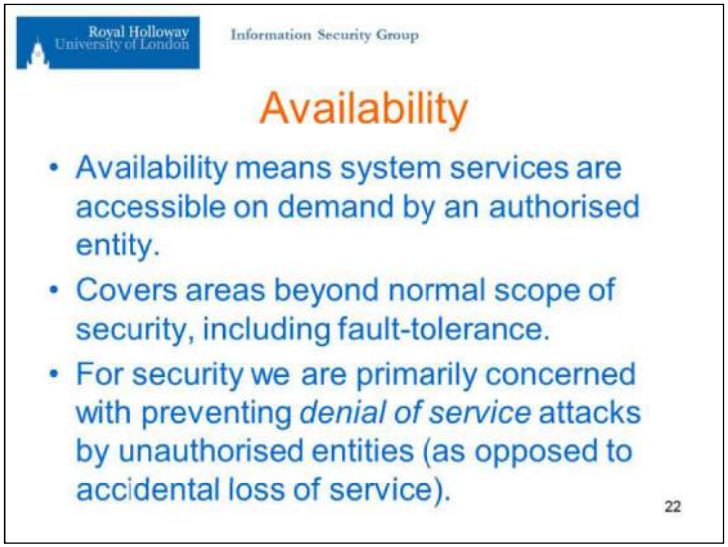
There are various definitions for Integrity. However, most of them come down to some formal version of requiring that things are as they should be.

In the context of computing, integrity is about preventing users *writing* information when they do not have the necessary authorisation. In this sense it is the formal dual of confidentiality (which is about preventing unauthorised *reading* of information).

In the context of a computer system, integrity can be defined as ensuring that the system state has not been modified by those not authorised to do so.

In the context of data communication, data integrity has acquired a subtly distinct meaning. Typically, in this context integrity refers to the *detection* (and subsequent correction) of modifications to transmitted data, since if data is sent through a public channel, preventing modifications (deliberate or accidental) is normally impossible.

Integrity is often a prerequisite for other security properties. For example, if the integrity of an Operating System's access control system could be violated, then file confidentiality might be breached.



Availability can be defined as ensuring that the services provided by a system are accessible on demand by an authorised entity.

Availability covers areas beyond the normal scope of security. For example, much of the technology required to ensure availability comes from areas such as fault-tolerant computing.

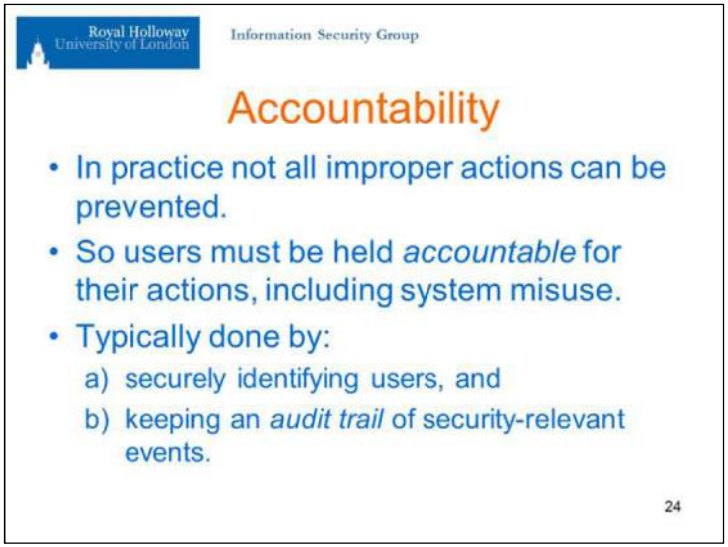
For the purposes of security we are primarily concerned with preventing attackers preventing legitimate users gaining access to their systems. Such attacks are known as *denial of service* (DoS) attacks. I.e. we are mainly concerned with defeating deliberate rather than accidental loss of availability. This gives a useful analogy with data integrity where security is normally concerned with protecting systems against deliberate loss of integrity (caused by malicious parties) rather then accidental loss of integrity, which is typically combated using a combination of non security technologies such as error-control coding and back-up procedures.

Examples of denial of service attacks include Internet 'flooding' attacks, where the attacker(s) overwhelm a server by sending it large numbers of connection requests (an example of a Distributed DoS (DDoS) attack). Availability may often be the most important security property, but there are relatively few effective mechanisms available to help provide it.

There are a number of other important goals and issues relating to security, arguably not covered by CIA. These include:

- accountability;
- reliability (including fault tolerance); and
- security event management (relating to auditability).

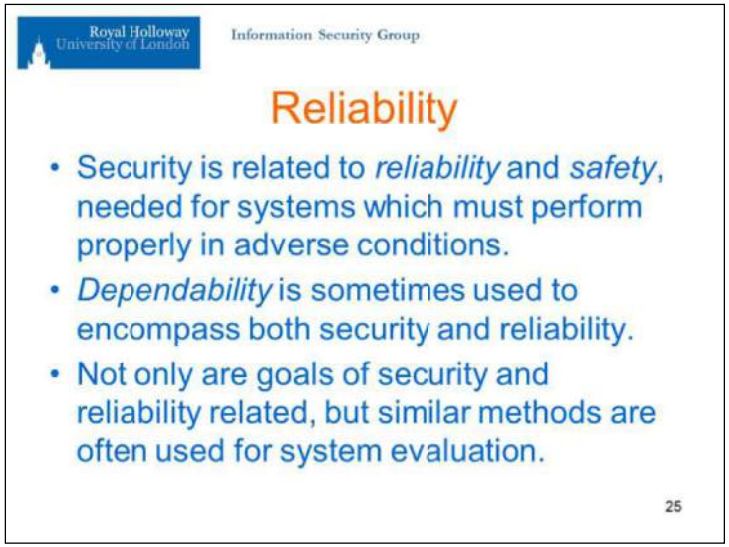
These are not the only terms that have been suggested should be added to CIA.



The traditional trio of security services (or goals), i.e. Confidentiality, Integrity and Availability, are all primarily concerned with preventing undesired events. However, in practice not all improper actions can be prevented, since authorised actions may cause security violations, and new security flaws are very often found in current systems.

Thus, in order to deal with security breaches as and when they occur, users must be held *accountable* for their actions, including system misuse.

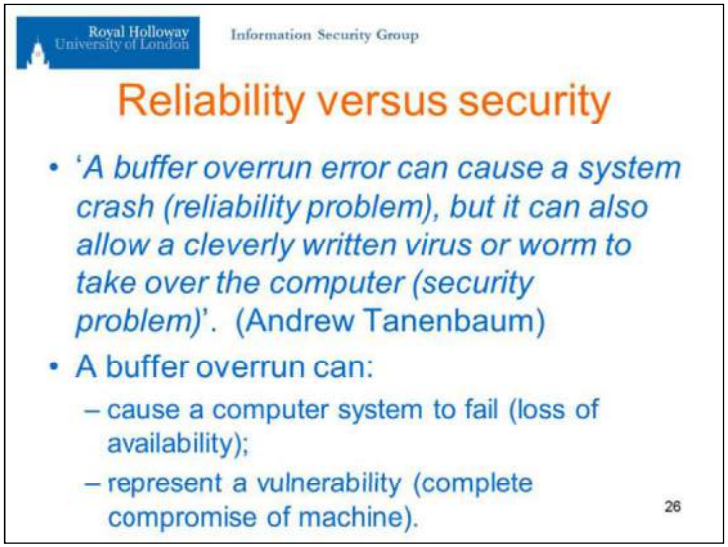
This is typically done by securely identifying users, and keeping an *audit trail* of security-relevant events, i.e. a sequence of 'records' containing event information. In the event of a security violation, the audit trail can be inspected for evidence to help identify the perpetrator of fraud. Also, in the event of disputes, the audit trail may help resolve 'who did what'.



Security is related to *reliability* and *safety*. These are notions used in connection with systems which must perform properly even in adverse conditions, e.g. nuclear power station and aircraft control systems.

Because of possible confusion in the scope of terms such as reliability and security, *dependability* is sometimes used to encompass both security and reliability.

Not only are the goals of security and reliability related, but similar methods are often used for evaluation of systems with respect to their security and reliability properties. For example, the issue of *assurance* is vital in both secure and reliable systems, and, at the highest level, in both cases this is dealt with using formal methods.



Reliability and security are intimately related …

'*A buffer overrun error can cause a system crash (reliability problem), but it can also allow a cleverly written virus or worm to take over the computer (security problem)'.* (Andrew Tanenbaum)

That is, a buffer overrun can:

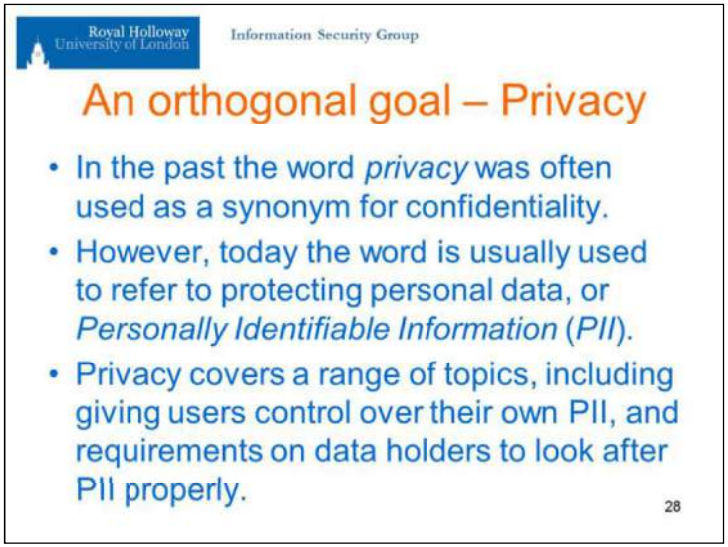
- cause a computer system to fail (loss of availability);
- represent a vulnerability (complete compromise of machine).

There are a variety of other aspects of security, not covered by the previous headings. These include:

• security event reporting (i.e. putting in place a defined procedure for reporting and managing security breaches or other suspicious events);

• security awareness training (i.e. training staff to be aware of security and privacy issues, including their legal obligations); and

• business continuity planning (BCP), which includes disaster recovery planning (i.e. putting in place measures to allow rapid resumption of IT functions after a major event, such as a natural disaster).



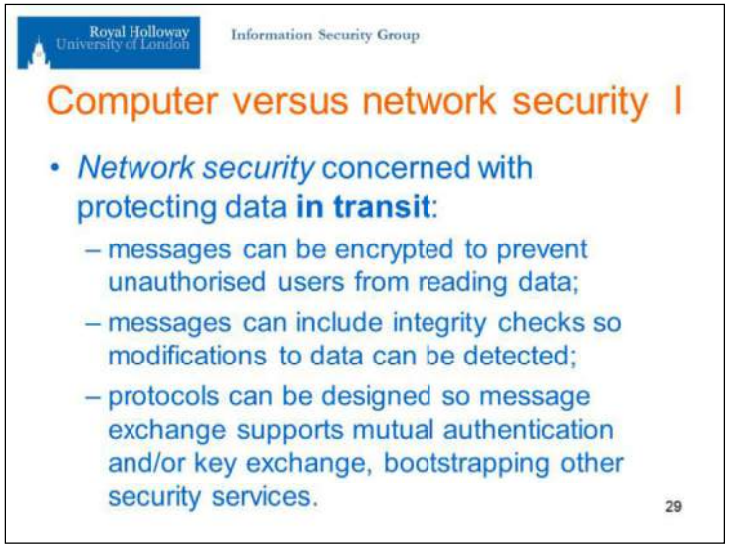
In the past, the word *privacy* was often used as a synonym for confidentiality.

However, today the word is usually used to refer to the protection of personal data, or *Personally Identifiable Information* (*PII*) as it is known.

Privacy covers a range of topics, including giving users control over their own PII, and requirements on data holders to look after PII properly.

Privacy in this sense is not really part of security, although enforcing privacy relies on the provision of security. That is, protecting user privacy requires the correct implementation of security measures, but security does not necessarily require user's privacy rights to be enforced.

We could, but we won't, spend the rest of the course looking at privacy – it is a large and important topic!

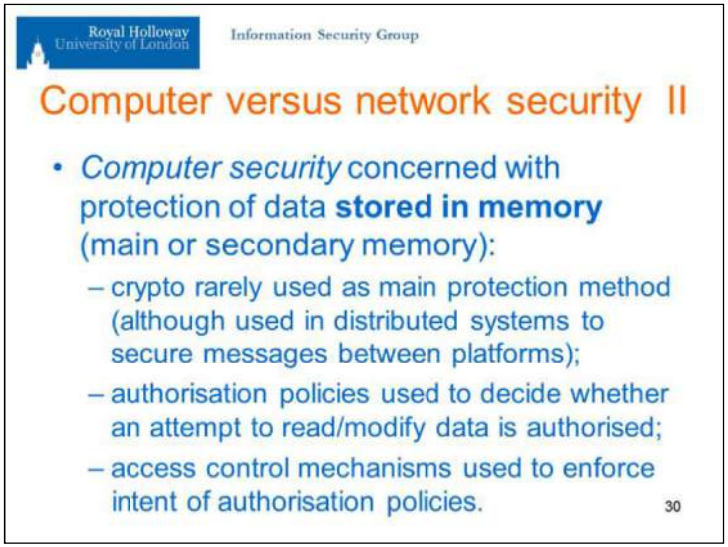


Network security is primarily concerned with the protection of data in transit. A number of measures can be used to provide such protection (typically based on the use of cryptography):

• messages can be encrypted to prevent unauthorised users from reading data;

• messages can include integrity checks so that modification to the data can be detected;

• protocols can be designed so that an exchange of messages leads to mutual authentication and/or key exchange, thereby 'bootstrapping' other security services.



Computer security is primarily concerned with the protection of data stored (and processed) in main memory or in secondary memory. A variety of measures can be used to provide such protection (typically rule-based):

• traditionally, cryptographic techniques (e.g. encryption and MACing) are rarely used as the main method of protection, although techniques from network security are used in distributed systems to secure the messages that are exchanged between system components (and encryption is sometimes used to protect entire storage media);

• authorisation policies are used to determine whether an attempt to read or modify data is authorised;

• access control mechanisms are used to enforce the intent of authorisation policies.

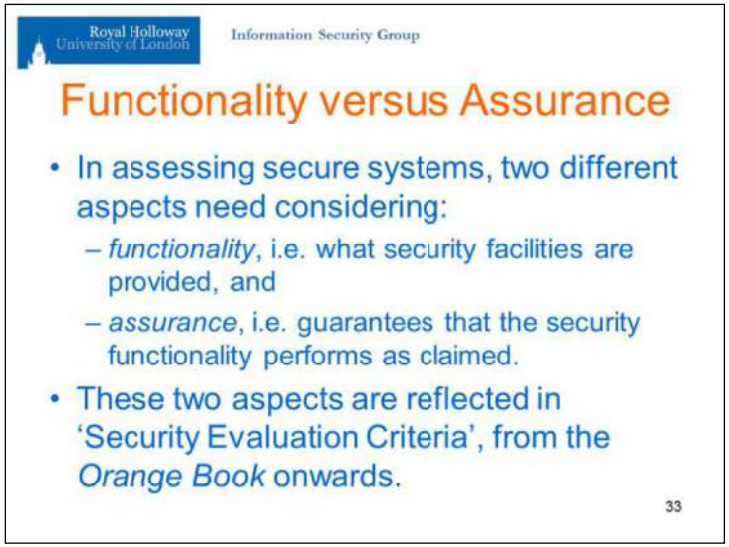
The mechanisms used to enforce security of stored data in a computer system are collectively known as the **Trusted Computing Base** ...

User expectations of, and functional requirements for, computer systems have increased dramatically over the last 30 years. This has increased the number of potential security issues. Problems introduced by relatively recent innovations include:

- multi-tasking computers require memory protection;
- complex security policies require complex access control mechanisms;
- ensuring that mobile code does not damage the host machine requires new protection mechanisms.

There are a number of ways of judging the security features of a computer system. These include looking for the following features:

- can the operating system and hardware implement memory protection?
- is it possible to identify authorised users?
- is it possible to define and enforce a discretionary security policy (i.e. where users can define controls affecting their own resources)?
- is it possible to define and enforce a mandatory security policy (i.e. where the system owner can define rules governing the handling of data, in addition to discretionary controls)?
- is it possible to store and protect audit information?
- can it be proved that the system meets the above requirements?



In assessing a secure system, two different aspects of the system need to be considered:

- *functionality*, i.e. what security facilities are provided by the system, and
- *assurance*, i.e. what guarantees are offered that the security functionality performs as claimed.

These two aspects are reflected in 'Security Evaluation Criteria', from the *Orange Book* onwards.

The Orange Book, or, to give it its proper title, the *Trusted Computer System Evaluation Criteria*, was published in 1985 by the US Department of Defense. The main purpose of the Orange Book (so called because of the colour of its cover) was to specify six 'security levels' for computer systems, as an aid to US government purchasers of such systems. The levels combine increasing security functionality with increasing levels of assurance. The Orange Book has been enormously influential, and many of the notions used in the Orange Book have become 'standard'. Since its publication it has been superseded, in Europe by the Information Technology Security Evaluation Criteria (ITSEC) and in the US by the Federal Evaluation Criteria. Both the ITSEC and the Federal Criteria have themselves been superseded by the globally agreed 'Common Criteria'.

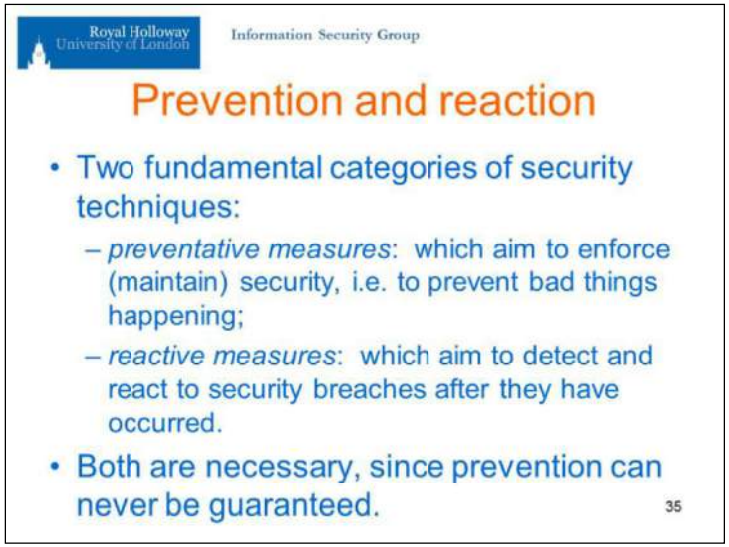
The Orange Book is available here:

http://seclab.cs.ucdavis.edu/projects/history/seminal.html

and here:

http://csrc.nist.gov/publications/history/

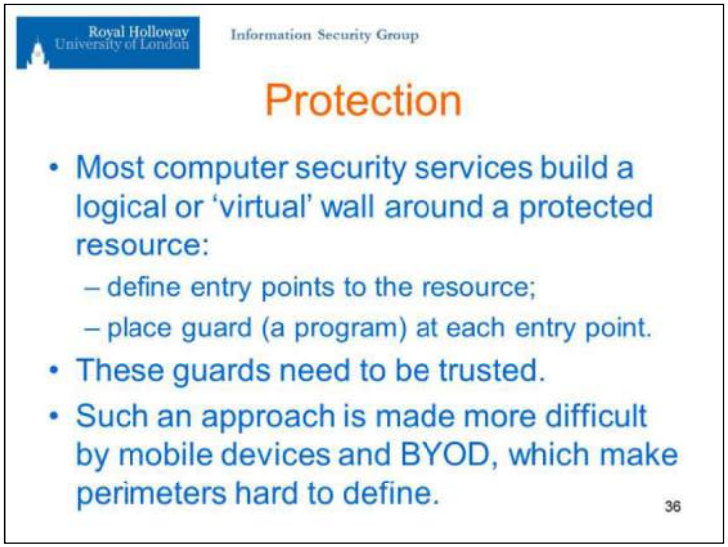
We next review at a high level the security measures (controls) that can be put in place to try to meet the security goals identified in the previous section.



There are two fundamental categories of security techniques:

- *preventative measures*: which aim to enforce (maintain) security, i.e. to prevent bad things from happening;
- *reactive measures*: which aim to detect and react to security breaches after they have occurred.

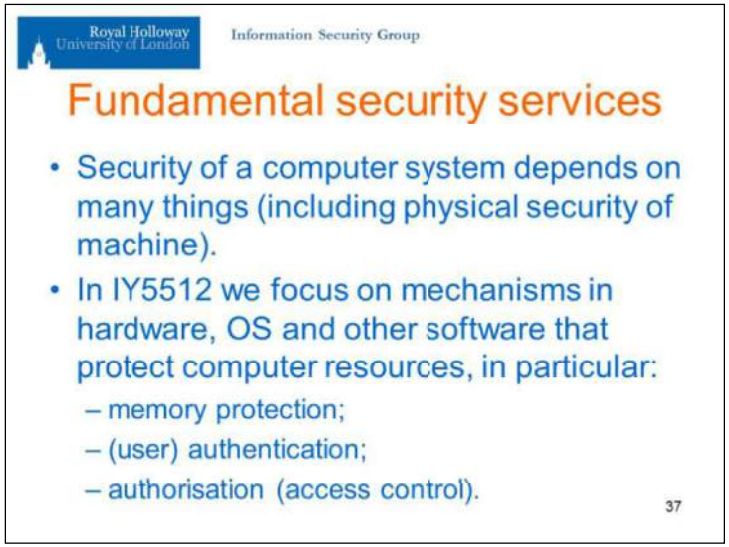
In practice, it is necessary to deploy both types of technique, since, while prevention of security breaches is clearly the goal, it can never be guaranteed. Detection is perhaps the most challenging aspect since only once issues have been detected can appropriate countermeasures be taken.



Most computer security services build a logical or 'virtual' wall around a protected resource. The system designers:

- define particular entry points to the resource;
- place a guard (typically a program) at each entry point – these guards (which are examples of security controls) need to be trusted.

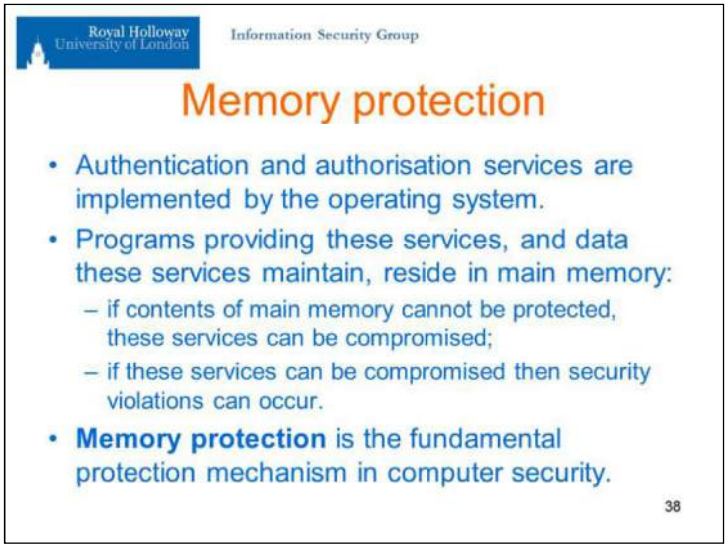
Such an approach is made more difficult by the use of mobile devices and the notion of **Bring Your Own Device (BYOD)**, i.e. using your own computing device, such as a smart phone, for work purposes. Such developments make perimeters hard to define.



The security of a computer system depends on many things (including the physical security of the machine).

In this course we focus on mechanisms in hardware, operating system and other software that protect computer resources. These include:

- memory protection mechanisms;
- authentication (of users); and
- authorisation (or access control).



Authentication and authorisation services are implemented by the operating system. The programs that provide these services, and the data these services maintain, reside in main memory. Hence:

- if the contents of main memory cannot be protected, then these services can be compromised;
- if these services can be compromised then security violations can occur.

As a result, **memory protection** is arguably the fundamental protection mechanism in computer security.

The correct identification of users is a prerequisite for authorisation and audit:

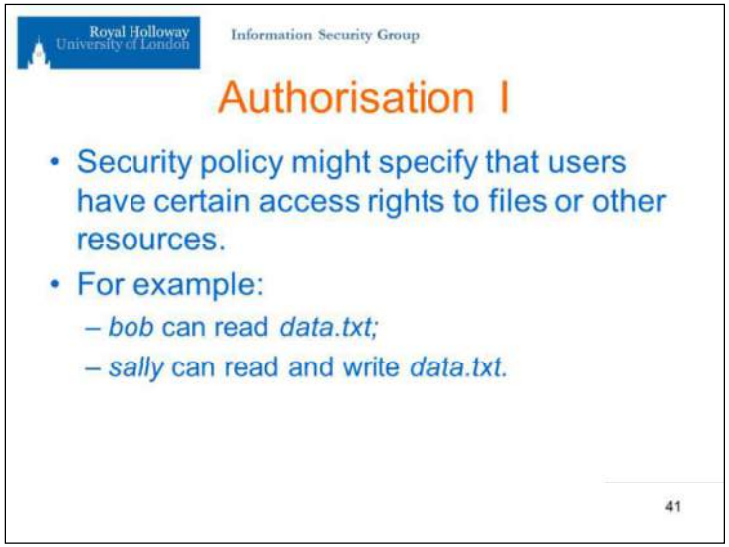
• if the authentication service can be bypassed or tricked, then unauthorised users can masquerade as authorised users;

• if you don't know the identity of a user you cannot tell whether or not the user should be allowed to access a protected resource, and you cannot determine with confidence who accessed a resource in the past (i.e. auditing is not reliable);

• without audit facilities it is impossible to provide accountability.

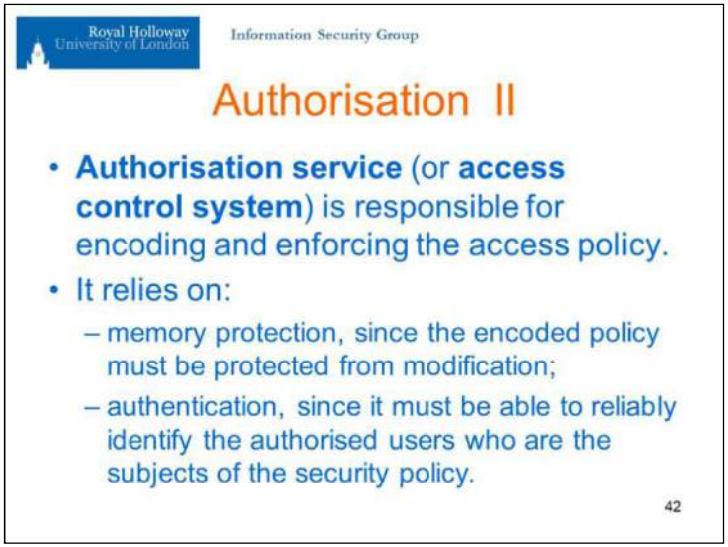
(User) authentication provides a vital link between a user's identity and the programs executed by the user. This link is crucial for access control, audit and other operating system functions. However, in practice, authentication mechanisms are often quite weak.

For example, passwords remain the dominant technique for user authentication, despite the fact that the technology is often easily broken.



The security policy in force (i.e. the current set of rules governing how security should be managed in a system) may specify that users have certain access rights to particular files or other resources. For example, the policy might specify that:

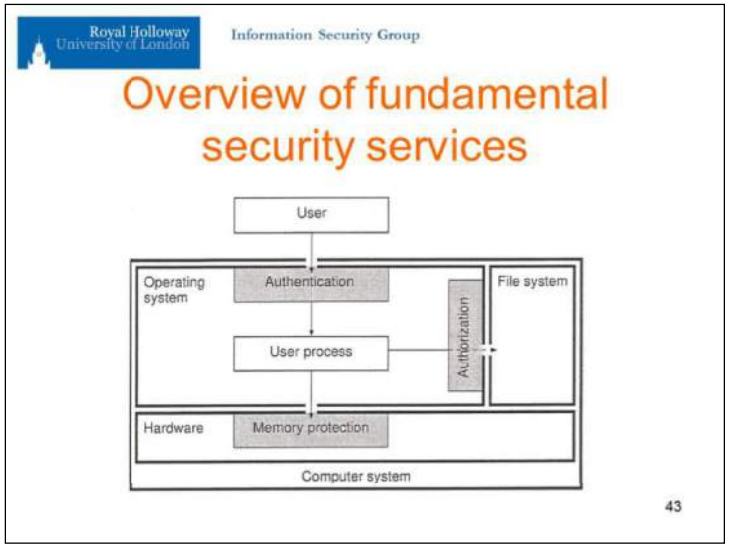
- *bob* can read *data.txt;*
- *sally* can read and write *data.txt.*



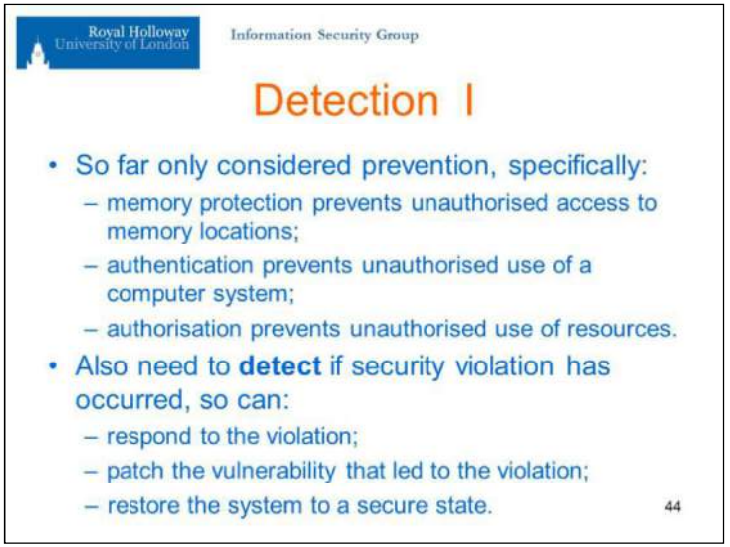
The **authorisation service** is responsible for encoding and enforcing the access policy (whatever it is). The correct operation of the service relies on:

- memory protection, since the encoded policy must be protected from modification;
- authentication, since it must be able to reliably identify the authorised users who are the subjects of the security policy.

In practice, this service is often referred to as the access control system.



The picture shows the relationship between the user, the operating system (and the authentication and authorisation services), the file system, and the underlying hardware (including the memory protection service).

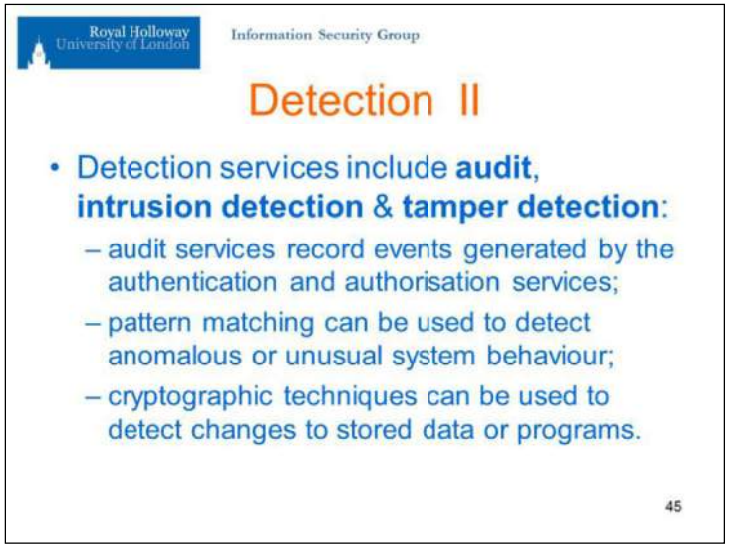


So far we have only considered preventative measures. In particular:

- memory protection prevents unauthorised access to memory locations;
- authentication prevents unauthorised use of a computer system;
- authorisation prevents unauthorised use of computer resources.

We also need to **detect** if a security violation has occurred, so that we can:

- respond to the violation;
- patch the vulnerability that led to the violation;
- restore the system to a secure state.

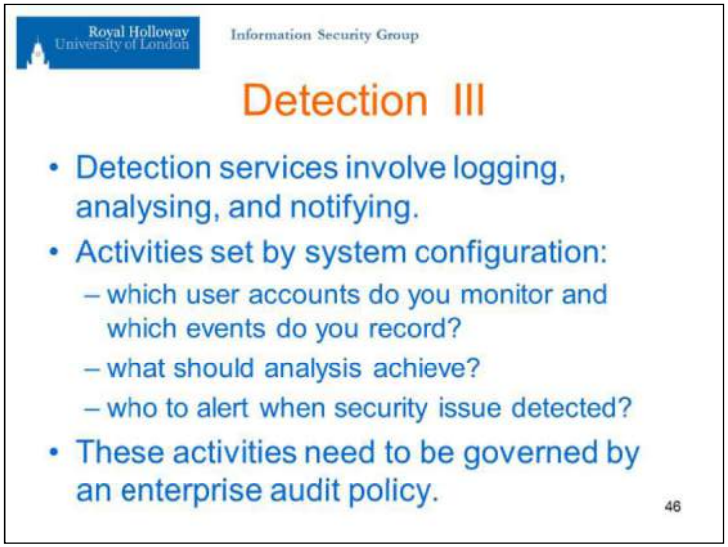


Detection services include **audit**, **intrusion detection** and **tamper detection**:

• audit services typically record events generated by the authentication and authorisation services;

• pattern matching can be used to detect anomalous or unusual system behaviour (this is what **intrusion detection systems** do);

• cryptographic techniques can be used to detect changes to stored data or programs (in much the same way as changes to data during transmission can be detected).



Event detection services involve three main activities:

• logging events;

• analysing events; and

• notifying administrators when possible suspicious behaviour is detected.

These activities are determined by the system configuration process, that determines:

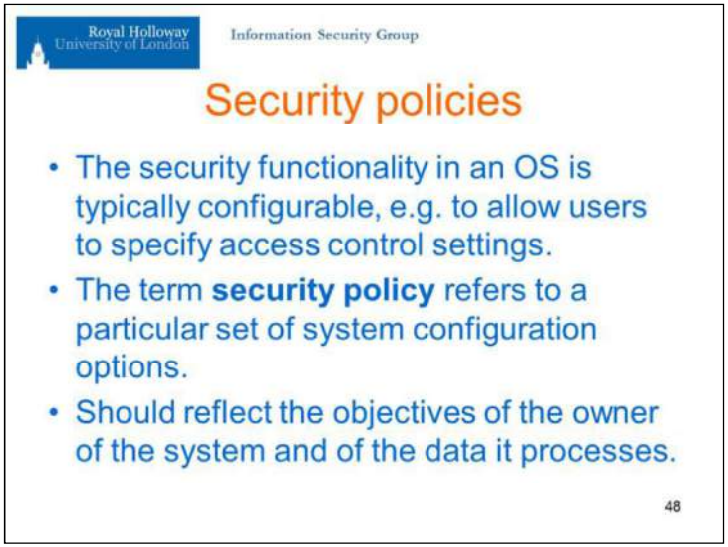
• which user accounts do you monitor and which events do you record?

• what should your analysis achieve?

• who do you alert when the analysis reveals a security problem?

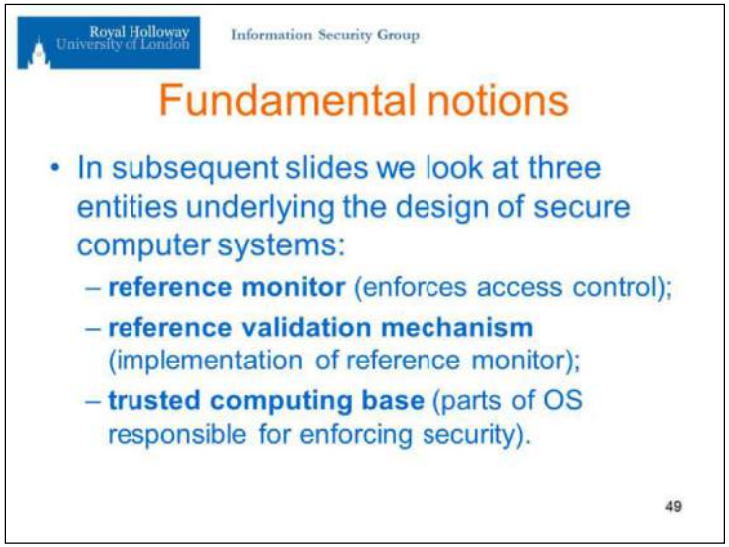
These activities need to be governed by an enterprise audit policy.

We next consider some of the fundamental ideas behind the provision of the security services we have identified.



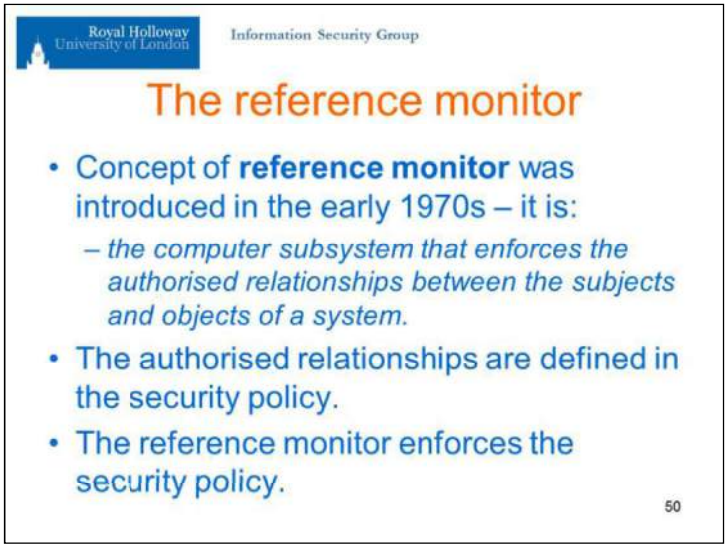
The security functionality in an OS is typically configurable, e.g. to allow users to specify access control settings to files and other resources managed by the system. The term **security policy** refers to a particular set of configuration options for a computer system.

The security policy should reflect the security objectives of the owner of the system and of the data it processes. Indeed, the term security policy is sometimes also used to refer to the underlying rules (objectives) that the system owner wishes to enforce. Of course, if the system does not have the configuration options to enforce the objectives of the system owner then there is a problem! This underlines the need for the purchaser of a system to have in mind any particular security objectives when making the purchasing decision.



In subsequent slides we look at three fundamental notional entities that underlie the design of secure computer systems:

- the **reference monitor** (which enforces access control);
- the **reference validation mechanism** (the implementation of reference monitor);
- the **trusted computing base** (i.e. the parts of the OS responsible for enforcing security).

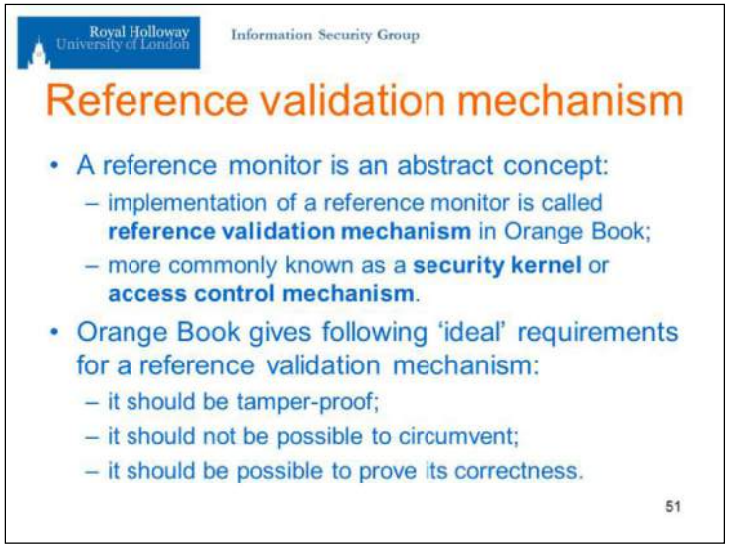


The concept of a **reference monitor** was introduced in the early 1970s. It is defined to be:

• *the computer subsystem that enforces the authorised relationships between the subjects and objects of a system*;

The authorised relationships are defined in the security policy, i.e. the implicit or explicit set of rules defining who is allowed to do what. The reference monitor enforces the security policy.

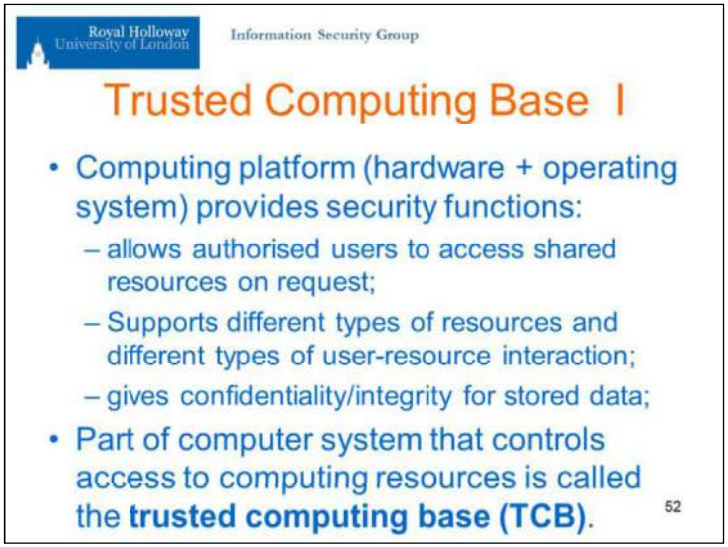
In this context the **subjects** are parties which want to access a resource (e.g. to read a file or execute a program), and the **objects** are the resources (e.g. files).



A reference monitor is an abstract concept. An implementation of a reference monitor is called a **reference validation mechanism** in the Orange Book. The reference validation mechanism is also more commonly known as the **security kernel** or **access control mechanism**.

The Orange Book identifies the following 'ideal' list of design requirements for a reference validation mechanism:

- it should be tamper-proof;
- it should not be possible to circumvent;
- it should be possible to prove its correctness.

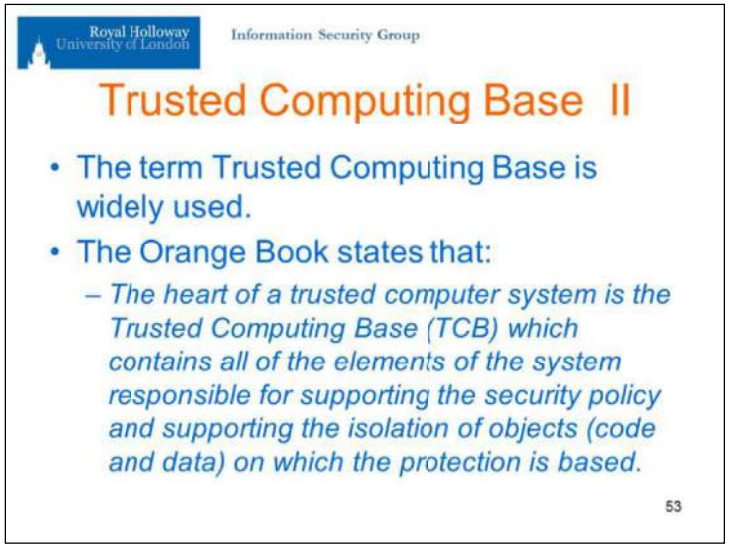


A computing platform (i.e. the computer hardware and the operating system) provides security functions on behalf of its owners/authorised users.

- It allows authorised users to access shared resources on request;
- It can support many different types of resources and many different types of user-resource interaction;
- It can provide confidentiality and integrity services for stored data;

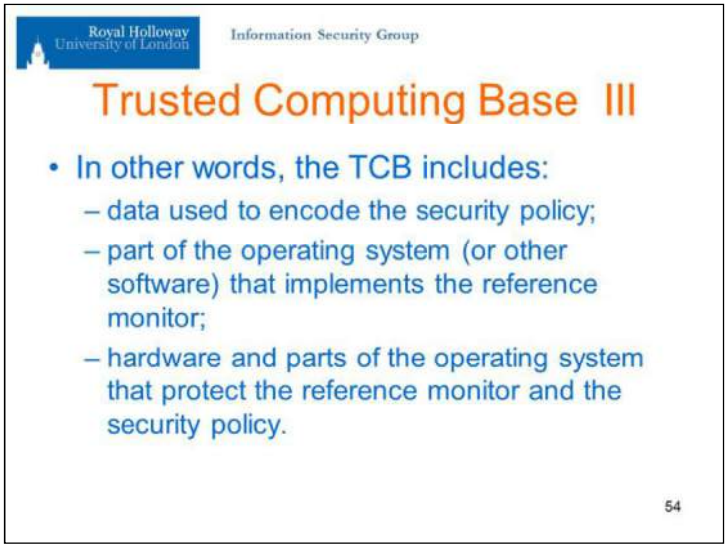
The part of a computer system that is responsible for controlling access to computing resources is called the **trusted computing base** (**TCB**).

The TCB clearly includes the reference validation mechanism.



The term TCB is widely used in the context of building secure computer systems. We take our definition from the Orange Book. The Orange Book states that:

• *The heart of a trusted computer system is the Trusted Computing Base (TCB) which contains all of the elements of the system responsible for supporting the security policy and supporting the isolation of objects (code and data) on which the protection is based.*

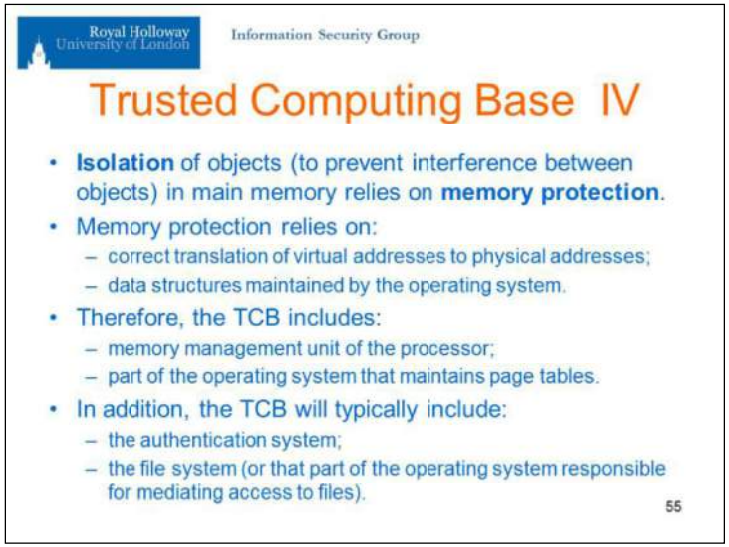


That is, the TCB includes:

• the data that is used to encode the security policy;

• that part of the operating system (or other software) that implements the reference monitor;

• the hardware and those parts of the operating system that protect the reference monitor and the security policy.



Isolation of objects is necessary to protect one object from another, and to prevent a subject with access to one object gaining access to another object. The isolation of objects which reside in main memory relies on **memory protection**. Memory protection relies on:

- correct translation of virtual addresses to physical addresses;
- data structures maintained by the operating system.

Therefore, the TCB includes:

- the memory management unit of the processor;
- that part of the operating system responsible for maintaining page tables.

In addition, the TCB will typically include:

- the authentication system;
- the file system (or that part of the operating system responsible for mediating access to files).

A trusted entity (in the context of computer security) is defined to be an entity that can break the security policy but is trusted not to do so;

- the operating system is trusted;
- some 'superuser' computer accounts, such as root in Unix systems, are trusted;
- all trusted programs and trusted users must be thoroughly checked to establish trustworthiness.

Roughly speaking, all trusted programs are contained in the TCB:

- to reduce the threat posed by trusted programs (and to minimise the size of the TCB), the number of trusted programs should be as small as possible.

If an operating system contains bugs, then trust in that system may be misplaced:

- it is essential that the operating system behaves as expected;
- if an attacker can compromise the operating system (perhaps because of a bug) that trust will certainly be misplaced!

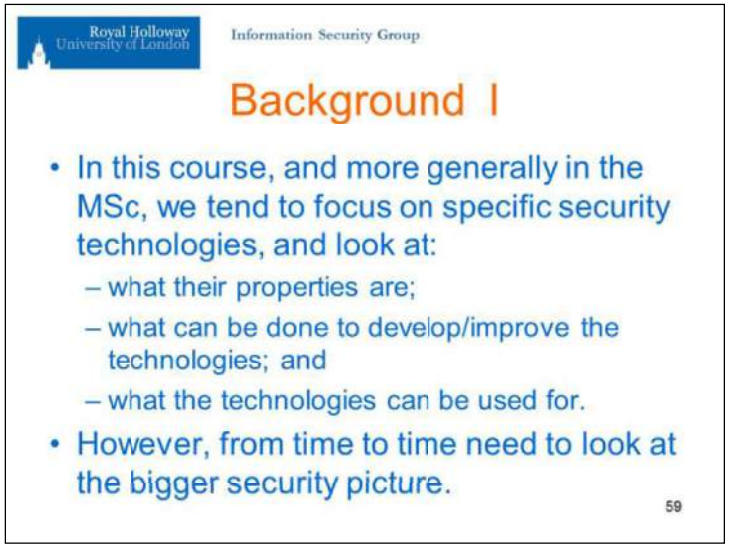
Proving that an operating system is trustworthy is very difficult:

- the highest level of certification for computer systems requires a formal (mathematical) proof of correctness.

In general, establishing program correctness (and hence trustworthiness) is very difficult:

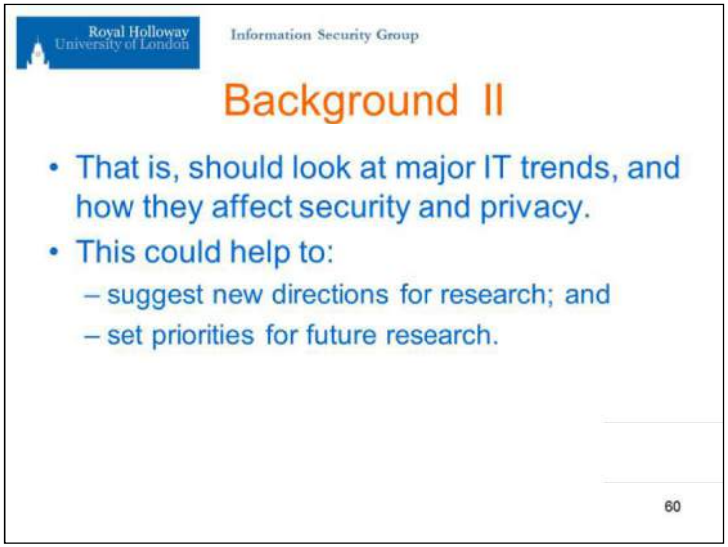
- it is generally only possible for simple and small programs;
- operating systems are neither simple nor small!.

In the final main part of this introduction, we look at how certain long-term trends in computing can increase existing threats and/or create totally new threats.



When faced with new security technologies, we tend to think about their security properties, how they can be developed and improved, and possible applications. Indeed, these issues are the main focus of the MSc in Information Security. However, this is not enough to understand the entirety of information security and how it will develop.

As security experts and professionals we also need to think about the larger trends in technology, and how security and privacy will be affected by them.



That is, from time to time we should try to look at the major IT trends, and see how they affect security and privacy. This could help to suggest new directions for research, and set priorities for future research.

Unfortunately, right now the signs for the future of computer security don't look very good. It is particularly worrying that security and privacy requirements seem to be pushing in very different directions to very powerful economic, technological and social pressures. Many ongoing trends are helping to make security and privacy problems worse, and the security community faces an uphill battle in trying to manage these new risks. We examine key emerging technology trends which have a major effect on information security and privacy.

We look at two key issues for future information security:

• **Technology trends** – what do they mean for future information security?

• **Conflicting requirements** – how security/privacy requirements are often pushing in very different directions to economic and technological pressure (and observing that economic/business pressures are very powerful).

We consider six key emerging technology trends with serious security and privacy implications:

• Ubiquitous/ambient computing, one aspect of which has become known as the Internet of Things (IoT);

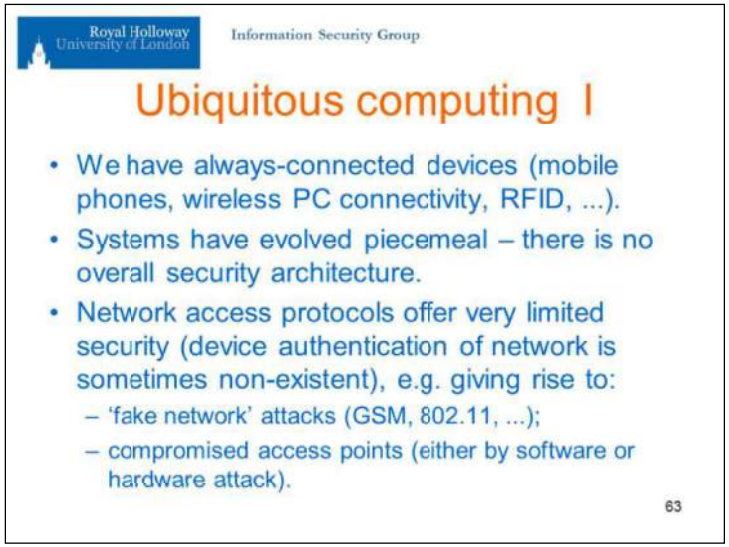
• Clouds/proxies/Grids;

• Growing system and component complexity;

• Integrated peripherals;

• System intelligence/autonomy;

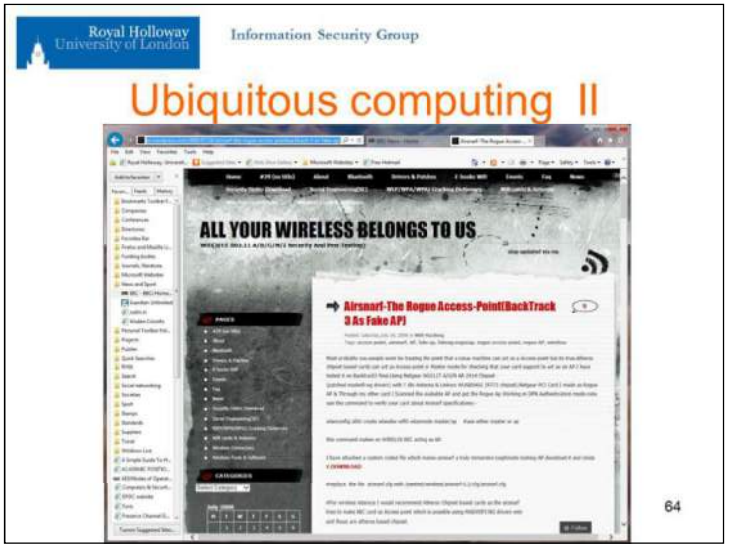
• Orchestrated attacks.



Always-connected devices have gone from the research laboratory to reality, including smart phones and PCs connecting via wireless networks, as well as still emerging technologies such as RFID tags, sensor nodes, and computing facilities in everyday devices ranging from cars to toasters (the IoT). These systems are evolving in a piecemeal way and there is no overall security architecture. This poses a huge risk, since these devices routinely communicate with each other, often without human intervention.

The common element is the use of TCP/IP across a wide range of underlying networks. However, TCP/IP has also evolved over a long period, and IPsec is far from universally deployed and in any case does not address security issues arising at lower layers of the protocol hierarchy.

Technology-specific protocols often offer a very limited set of security features. For example, authentication of the 'access network' to the device is sometimes non-existent, e.g. as is the case for GSM and IEEE 802.11 Wi-Fi (although more recent versions of 802.11 include measures for two-way authentication, they do not seem to be widely implemented). Existing security measures aim at controlling access to the network to protect the investment of the network owners, rather than the serious threat to end nodes posed by unauthenticated access points.



The effects of such a lack of network authentication has been widely documented in print and on the Internet. This has given rise to a series of public domain implementations of 'fake network' attacks on GSM and IEEE 802.11, as well as attacks arising from compromised access points (where the compromise might arise from software or hardware attack).

There are a host of examples of such software, including AirJack:

http://sourceforge.net/projects/airjack/

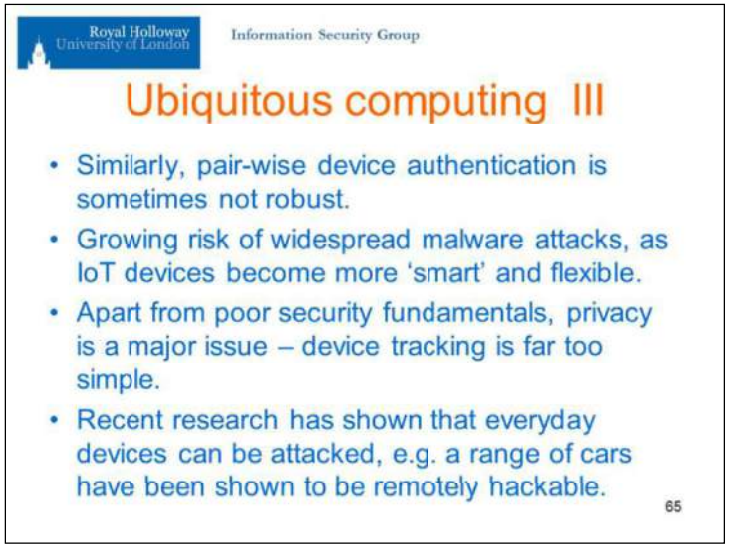
and airsnarf:

http://airsnarf.shmoo.com/

Airsnarf is a rogue wireless access point utility designed to demonstrate how a rogue access point can steal usernames and passwords from public wireless hotspots. A graphic description of how airsnarf could be used to compromise user security is provided in many places, e.g.:

http://wifi0wn.wordpress.com/2008/07/19/airsnarf-the-rogue-access-pointbacktrack-3-as-fake-ap/

http://www.newswireless.net



Pair-wise device authentication can also be vulnerable; for example the original Bluetooth pairing scheme was rather weak. In general, as a result of the lack of comprehensive and integrated security solutions for mobile connected devices, there is an ever-growing risk of widespread malware attacks, as IoT devices become more 'smart' and flexible. This is all happening in an environment where malware attacks on mobile devices continue to become more numerous and serious.

Apart from poor security fundamentals, privacy is a major issue. Device tracking is a particular problem. In any network protocol, addresses of some sort are exchanged between devices, and, at least at some level of the protocol hierarchy, these addresses need to be exchanged in cleartext. If the address of the mobile device is fixed, then this offers a simple way of tracking the location of that device, and by implication, its owner. Of course, work is ongoing to address this problem for a wide variety of protocols, including for mobile networks.

Recent research has shown that everyday devices can be attacked, e.g. a range of cars have been shown to be remotely hackable – see, for example:

IY5512: Part 1

http://www.theregister.co.uk/2015/07/21/jeep_patch/

65

Mobile malware is a big and growing problem. Kaspersky Labs' 2015 Q2 cyber threats report (from July 2015) states that:

- 291,800 new mobile malware programs emerged in Q2, which is 2.8 times greater than in Q1 of 2015.
- 1 million mobile malware installation packages were identified in Q2, seven times greater than in Q1 of 2015.
- Mobile banking has remained a main target for mobile threats, especially Trojans.

The above information is taken from:

http://usa.kaspersky.com/about-us/press-center/press-releases/kaspersky-lab-releases-q2-it-threat-evolution-report-2015

There is growing trend to move data and processing to the cloud. The security and privacy concerns arising from such a move are very well-documented. It is a particular worry since many cloud providers offer few guarantees about the security, privacy and availability of the data they store. Reports of real-life security breaches are not hard to find.

Of course, the move to the cloud is just one part of a long-term trend to outsource IT provision. All users of outsourced services need to ask deep questions about security and availability.

Cloud security and privacy has become a huge and pressing concern. There are risks to the user of cloud services from both incompetent and malign cloud providers.

There is a growing set of standards governing security, which cloud providers are under pressure to seek conformance to (in addition to the usual ISO/IEC 27001 and 27002 security management standards). One example is provided by recently published IDSO/IEC 27018 (Code of practice for protection of personally identifiable information (PII) in public clouds acting as PII processors).

The Wikipedia article on cloud computing security (http://en.wikipedia.org/wiki/Cloud_computing_security) flags up some of the major areas of concern.

A quick search of the web reveals a plethora of articles revealing concerns about the impact of security breaches at cloud providers, and what this might mean for the client of cloud services.

The article shown (at http://www.govtech.com/security/Data-Breaches-in-the-Cloud-Whos-Responsible.html) discusses who might take responsibility for losses resulting from such a breach.

IY5512: Part 1

ngs

Social networking provides a particularly pervasive example of third party computing. Huge numbers of individuals trust social networking sites with very sensitive personal data, but the measures taken to protect this data are far less clear. Other privacy issues abound. For example, Facebook user privacy settings continue to attract much negative commentary.

The Daily Mirror recently (22/9/15) had the headline: 'Facebook as bad as NSA spies for snooping'. This arises from the fact that Facebook has received a lawsuit from the Belgian data protection authorities accusing it of acting like the NSA. 'When it became known that the NSA was spying on people all around the world, everybody was upset. [Facebook] is doing the very same thing, albeit in a different way' said Frederic Debussere (representing the Belgian privacy commission (BPC).

See:

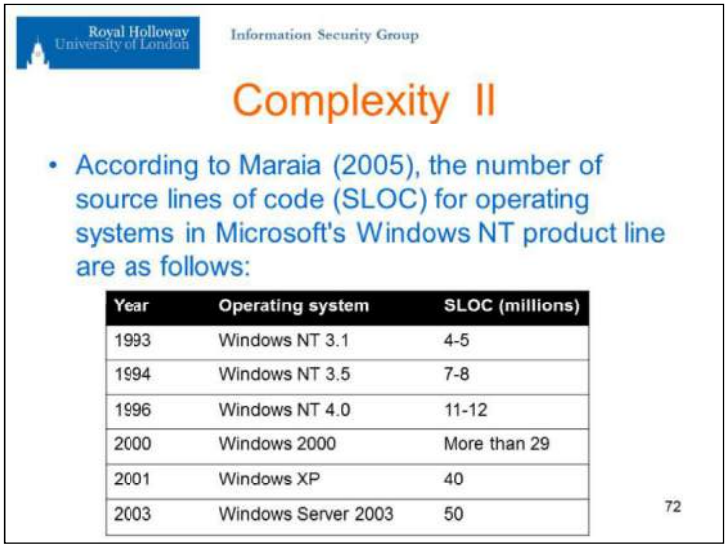
http://www.mirror.co.uk/news/technology-science/technology/facebook-bad-nsa-spies-snooping-6492069

for the news article, and for the report see:

https://securehomes.esat.kuleuven.be/~gacar/fb_tracking/index.html#findi

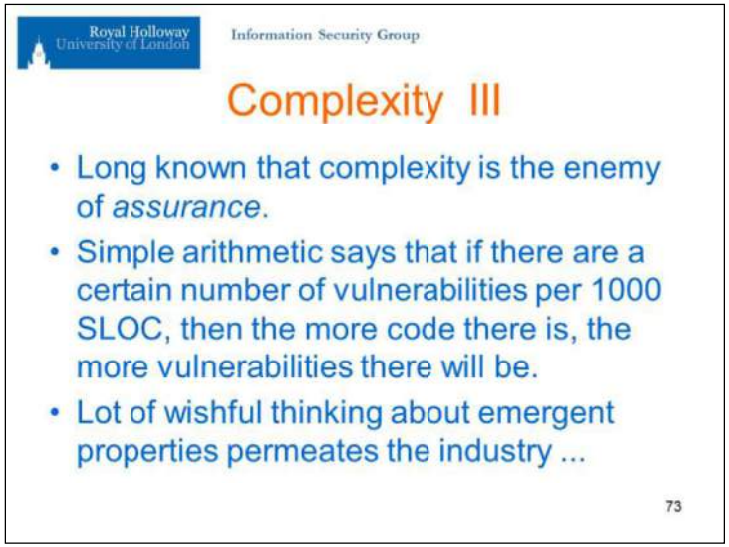
70

Another long-term trend with serious security effects is the ever-increasing complexity of both hardware and software. The complexity of both devices and the software running on them has continued to increase for many years.



As an example of growing software complexity, we consider the development of Microsoft Windows NT. Following Maraia (2005), the numbers of source lines of code (SLOC) for operating systems in Microsoft's Windows NT product line are as given in the table.

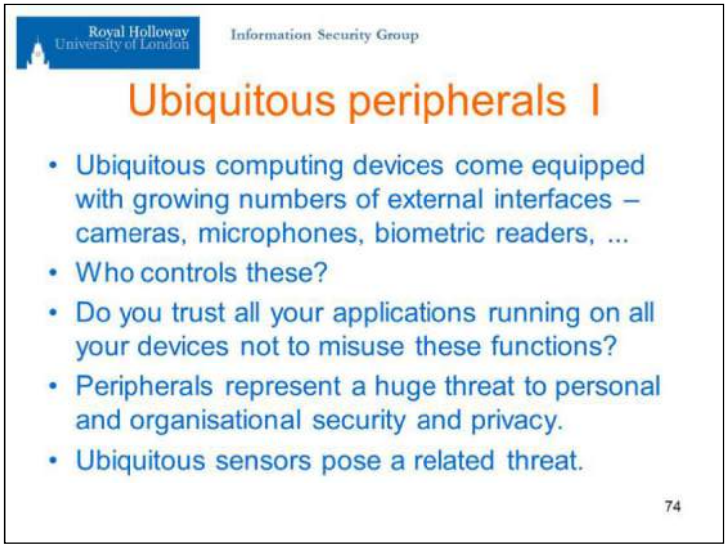
A similar table for various versions of Unix can be found on the Wikipedia page for SLOC: http://en.wikipedia.org/wiki/Source_lines_of_code



This complexity is often not just to increase functionality. For example, the rapidly reducing cost of sophisticated computing makes it simpler and cheaper to put a powerful processor and a complete operating system into a small embedded device, rather than write special-purpose code. Systems built out of individual components are also becoming more complex – growing interconnectivity potentially adds huge complexity.

A long-standing fundamental principle is that complexity is the enemy of assurance. Simple arithmetic says that if there are a certain number of vulnerabilities per 1000 SLOC, then the more code there is, the more vulnerabilities there will be. These issues have been explored by many authors.

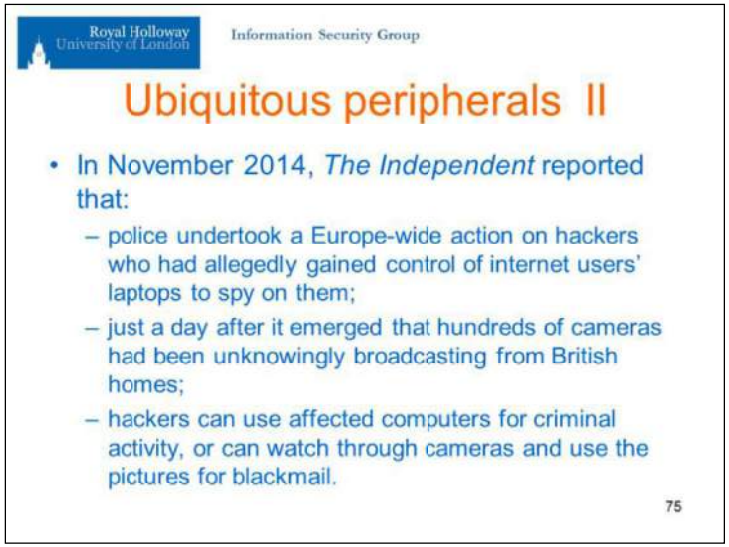
A lot of wishful thinking about emergent properties appears to permeate the industry; in particular that, somehow, a secure system can be built out of a collection of insecure components. The notion of an emergent property is, of course, well-established, but it is far from clear whether security and reliability can 'emerge' in this way.



Computing devices (laptops, phones, PDAs, etc.) now come with a growing number of external interfaces, including cameras, microphones and biometric readers. Users need to consider who or what controls these devices. For example, does a user trust the applications not to misuse these functions? In fact these peripherals represent a huge threat to personal and organisational security and privacy.

This is not just a theoretical threat. For example, in October 2008 it was reported (Keizer, 2008) that Adobe Systems had warned users that hackers could use 'clickjacking' attack tactics to secretly turn on a computer's microphone and web camera. By duping users into visiting a malicious website, hackers could hijack seemingly-innocent clicks that, in reality, would be used to grant the site access to the computer's webcam and microphone without the user's knowledge.

These threats look likely to grow. Additional, highly privacy-sensitive peripherals, including GPS receivers, are becoming commonplace in smart phones and kindred devices. The desire for autoconfiguration of systems and devices also increases the risks.

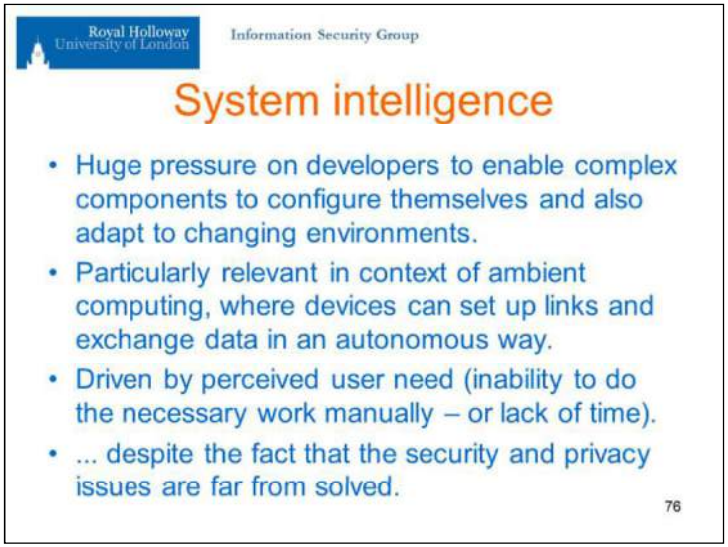


In November 2014, *The Independent* reported on the following story.

Police undertook a Europe-wide action on hackers who had allegedly gained control of internet users' laptops to spy on them — just a day after it emerged that hundreds of cameras had been unknowingly broadcasting from British homes. Police say that the arrested people, four of whom were in the UK, had been using software to remotely control computers, known as 'Ratting'. Once into the computers, hackers can use them for criminal activity, or watch through cameras and use the pictures for blackmail.

See:

http://www.independent.co.uk/news/uk/another-webcam-security-threat-uncovered-as-police-launch-computer-hijack-crackdown-9875594.html

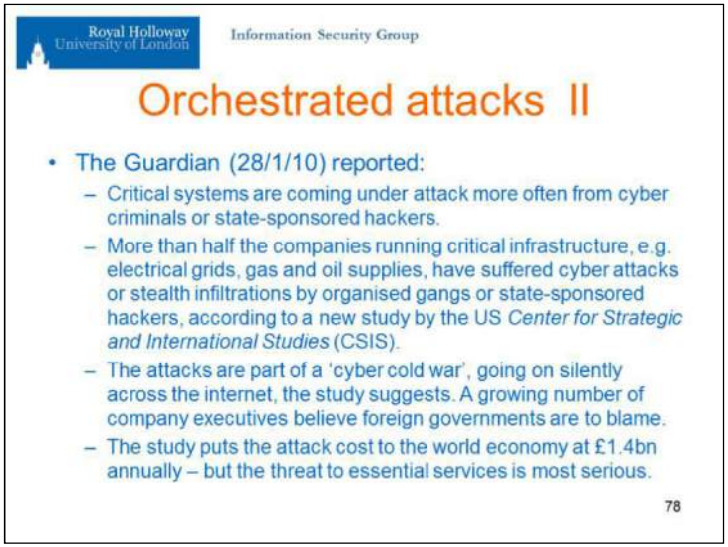


With the growing use of small, network-enabled, components, there is a corresponding pressure on developers to enable these components to configure themselves and automatically adapt to changing environments. For example, users expect newly purchased components, such as smart phones, wireless headsets, PDAs, and notebook computers, to seamlessly intercommunicate and interoperate. This clearly poses a non-trivial task for the developer.

In the future world of ambient computing, the problem will only grow, and will probably involve many small devices without a complex user interface (or, in the case of small sensor nodes or tags, without any user interface at all). There is thus potentially huge demand for technology to enable devices to set up communications links and exchange data in completely autonomously. However, the security and privacy issues are far from solved. Reputation systems of many types have been proposed as a possible solution, but whilst this may be appropriate for informal relationships of low sensitivity, it is hard to see such schemes as a solution to medium or high-level security requirements, since such systems are notoriously easy to 'game'.

It is perhaps too soon to see much practical impact in real life attacks of the threat from autonomous configuration. However, there is already concern in the research community.

Over the last decade or two, a key trend in the evolution of malware has been the shift from 'proof of concept' by amateurs to attacks by criminals or others with sinister intent. Continued growth in such orchestrated attacks is almost inevitable, involving governments, terrorist groups, criminal gangs, protesters, etc.

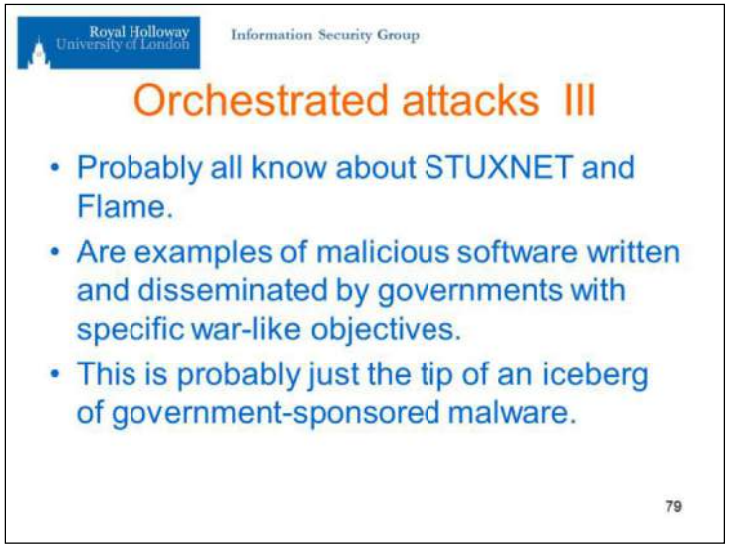


The Guardian newspaper reported in early 2010 that 'Critical systems are coming under attack more often from cyber criminals or state-sponsored hackers. More than half the companies running critical infrastructure, e.g. electrical grids, gas and oil supplies, have suffered cyber attacks or stealth infiltrations by organised gangs or state-sponsored hackers, according to a study by the US Center for Strategic and International Studies (CSIS). The attacks are part of a 'cyber cold war', going on silently across the internet, the study suggests. A growing number of company executives believe foreign governments are to blame. The study puts the attack cost to the world economy at £1.4bn annually – but the threat to essential services is most serious.'

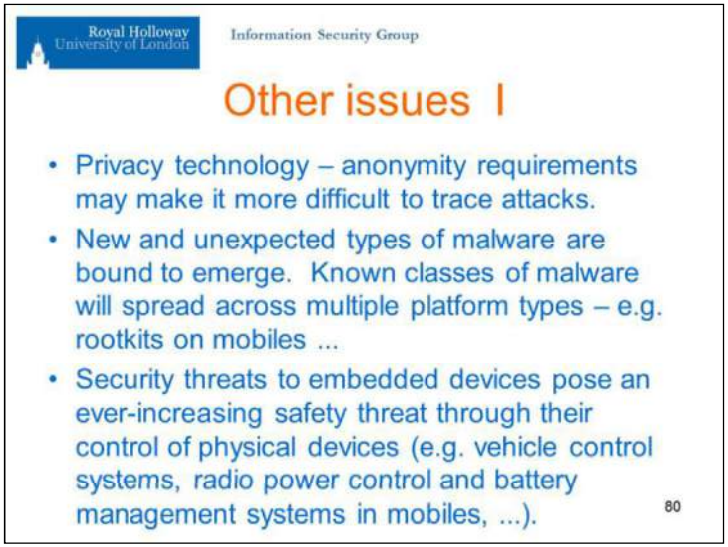
Some idea of the scale of the organised attacks can be obtained from Microsoft's Security Intelligence Report series available at:

http://www.microsoft.com/security/portal/Threat/SIR.aspx

These extensive reports provide a wealth of statistical data on malware and other attacks gained from Microsoft's own experience.



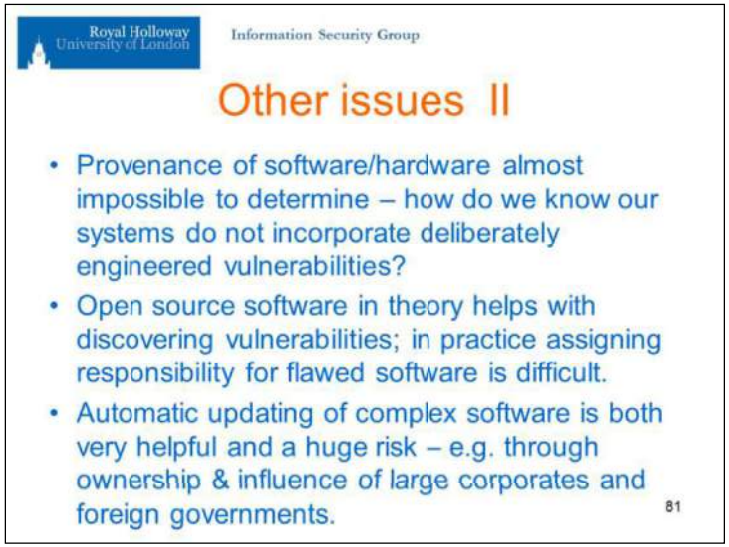
I suspect everyone reading this slide has heard about STUXNET and Flame. These are just two widely publicised examples of malicious software written and disseminated by governments with specific war-like objectives. They probably represent just the tip of an iceberg of government-sponsored malware.



Security and privacy sometimes push in opposite directions, particularly with respect to accountability and anonymity. Much research has been conducted in recent years on ways of supporting technologically-guaranteed anonymity – indeed, the new subject area of *Privacy Enhancing Technology* has emerged. Whilst this technology helps protect individual freedom, it may also make it more difficult to trace cyber attacks on individuals and organisations. That is, anonymous criminals may be much harder to detect and convict.

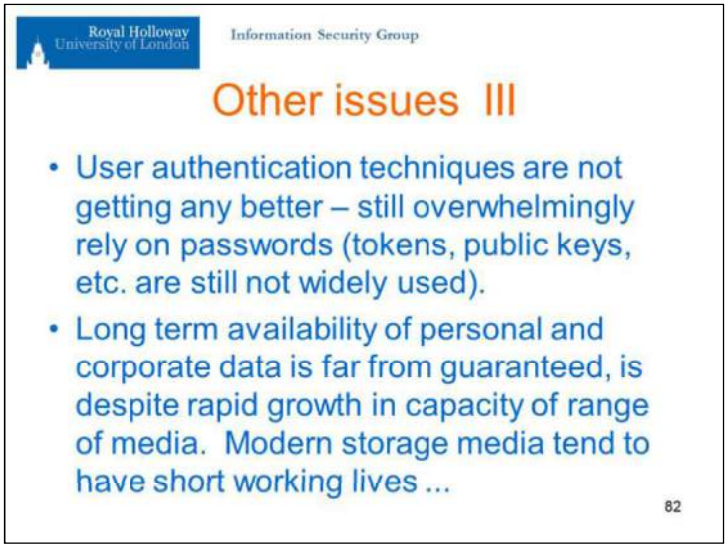
New and unexpected types of malware are bound to emerge. Even in the absence of new techniques, known types of malware will spread across multiple platform types; for example, if they are not already out there, we should soon expect to see rootkits on mobiles.

Security threats to embedded devices pose an increasing safety threat through their control of safety-critical physical devices. For example, highly complex embedded computing systems are very widely used, including in vehicle control and engine management systems, radio emission power control and battery management systems for mobiles, and control of domestic appliances. Whilst this is not so new, the increasing use of highly complex operating systems as the basis for such embedded systems greatly increases their vulnerability.



The provenance of both software and hardware has become almost impossible to determine. As a result, how do we know our systems do not incorporate deliberately engineered vulnerabilities? Indeed, it is almost inevitable that this is already occurring, possibly at the instigation of governments. In principle, open source software helps with discovering accidental or deliberately incorporated vulnerabilities but, in practice, use of such software makes assigning responsibility for identifying and fixing flawed software difficult or impossible.

The automatic updating of complex software is both very helpful and a huge risk. Its widespread use makes just about every computer system vulnerable to large corporations and (by implication) governments.



In practice, user authentication techniques are not getting much better, despite the increasing range of available technology. That is, tokens, public keys, etc., are still not widely used. We still overwhelmingly rely on passwords, with all their well-documented shortcomings.

The long-term availability of personal and corporate data is far from guaranteed, despite (or perhaps even because of) rapid growth in the capacity of storage media. Modern electronic media tend to have short working lives, which contrast poorly with the lifetime of paper. Without appropriate action, huge volumes of data could be lost.

There is huge business pressure to market products first and worry about security second. This causes enormous security problems. However, technology gets used in ways unanticipated by its designers (with obvious examples being GSM SMS and the universal use of IP); this means that, even if conducted rigorously, initial threat analyses are no longer valid.

This combines with the principle that retrofitting security is very difficult, perhaps even impossible, in practice.

Indeed, where it is available, `retrofit' security technology is often not used (e.g. trusted computing, identity management systems, SET, etc.).

Above all else, improving security and privacy rarely has a big visible financial pay-off to the user (individual or corporate). Of course, serious fraud may be averted, but that only seems like a gain in retrospect, i.e. after a disastrous event you may wish you had spent more resource on security, but the loss is not known in advance.

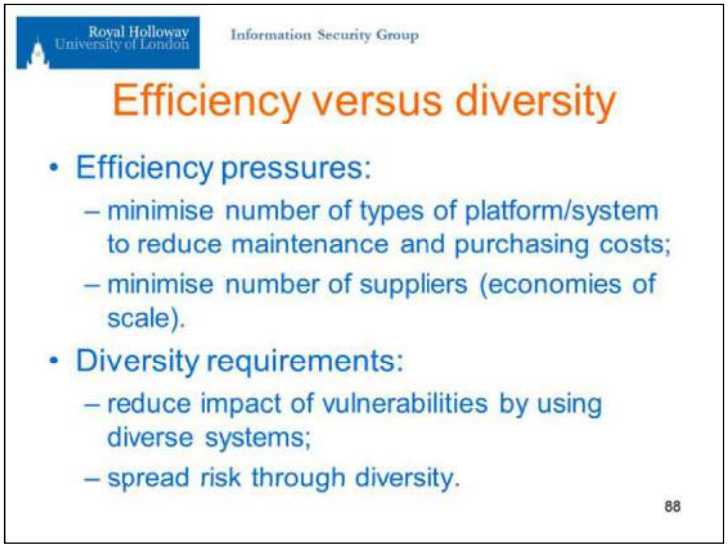
Another way of looking at the problems we have identified is in terms of conflicts. Not only are there technological trends which threaten security, but many commercial, technological and social pressures conflict directly with improving security and privacy.

There are two major security and privacy requirements of relevance here, namely the need for *high robustness*, because of the criticality of IT, and the need for *privacy protection*, not least because of emerging legal frameworks and user demands. The major economic, technological and social factors of relevance here include increasing *complexity*, arising from inevitable technological drift and which directly threatens robustness, the increased use of third parties (*outsourcing*) which makes privacy and security assurance very hard to achieve, and the use of *intelligence* (sophisticated IT) everywhere, not least to improve flexibility which also directly threatens robustness.

These security, privacy and reliability requirements often conflict directly with business, technological and social forces, which are inevitably a lot more powerful than security and privacy requirements.

We next look at a few examples of such conflicts.

Business pressures force organisations to improve their operational efficiency. This often involves the use of third party providers, integration across sectors, just in time operation (and minimisation of IT investment), and taking account of green issues. However, our robustness requirement suggests that we should avoid reliance on systems outside of our direct control and on single points of failure, avoid the possibility of cascading failures, and build in redundancy (employ multiple parallel systems, etc.). The conflicts here are clear.



Efficiency pressures also suggest that we should minimise the number of types of platform/system to reduce maintenance and purchasing costs, and minimise the number of suppliers to achieve optimal economies of scale. However, in contrast, trying to achieve reliability argues in favour of maximising diversity to reduce the impact of vulnerabilities, and spread risk across multiple technologies.

Continuous technological development results in increasing complexity. In particular, hardware and software development is more and more removed from direct human understanding through a growing number of intermediary layers (libraries, CAD tools, etc.). However our desire for reliability suggests we should take note of the oft observed maxim that the simpler a system is, the easier it is to make it reliable.

The desire for maximising flexibility in business investments, for obvious economic reasons, suggests maximising re-use of a standard platform (e.g. a PC), even in embedded applications. This reduces cost and helps hasten the development of new products. However, our requirement for stability (as part of reliability) suggest that keeping things simple will increase assurance, and it is again well-established that maximising flexibility also increases the attack surface.

We finally observe the business and social pressure for novelty, almost for its own sake. Manufacturers want to get their latest idea in the marketplace as quickly as possible in order to grab market share; also, end users want the latest gadget for social and fashion reasons. However, experience suggests that new almost certainly means less stable. We have all been told never to buy version 1 of anything, as it will be full of unanticipated flaws; in general systems become more stable over time.

We all see news items about security breaches on almost a daily basis. As security experts we are inclined to shrug our shoulders and say 'I told you so'. However, no-one seems to pay attention to us, and despite our best efforts things are getting worse. Perhaps this is inevitable?

The key question would appear to be 'What should we all do about this?' Well, we all have a role to play.

Governments can help by tailoring regulation to help enforce reliability. They can also invest in law enforcement to help detect and prevent criminal activity.

Major technology providers, such as Microsoft, Google, Apple, etc., need to change their business practices, as far as is commercially practical. In particular they need to move away from the model of deploying products first and fixing them later. To be fair, the signs are encouraging, at least from some of the big players.

End users need to be made aware of their responsibilities for their own security and for protecting the privacy of their and others personal data. However, can we reasonably expect users to be sensible?

The *academic community* needs to try to take account of commercial realities when developing new and improved security technologies. It is far from clear whether the solution is yet more new cryptographic schemes and security protocols. Perhaps ways of simplifying the deployment of the technologies we already have would be a better goal.

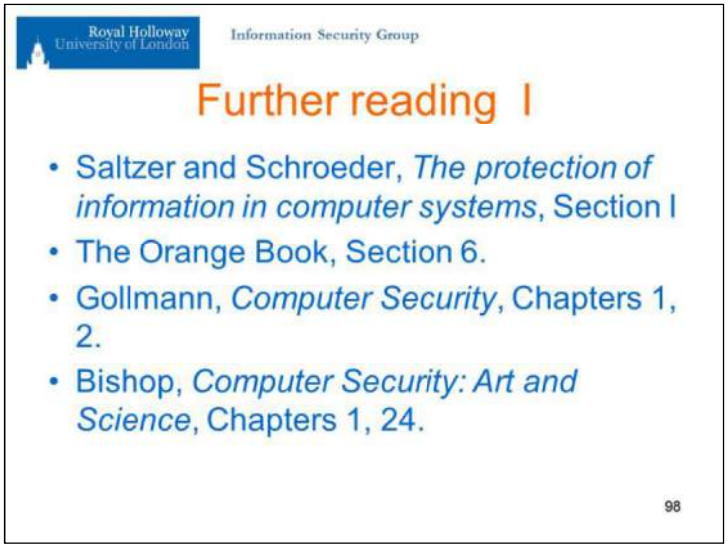
More generally, can any of us resist business and social pressure? Can we turn these pressures to our advantage?

Overall, it does not seem to be a problem of the availability of good security and privacy technology. Instead we need to find ways of getting this technology properly deployed. Typically this means finding evolutionary paths with low costs to all parties (as opposed to revolutions, which almost never happen, not least because of the chicken and egg problem).

With the growth in types of interconnected device (as part of the developing **Internet of Things**) we need somehow to educate manufacturers about the importance of security *before* they connect their devices to the Internet and not afterwards. Historically this is a major challenge, because manufacturers of such devices have never had to confront the security challenges before, and they lack the expertise to understand the risks.

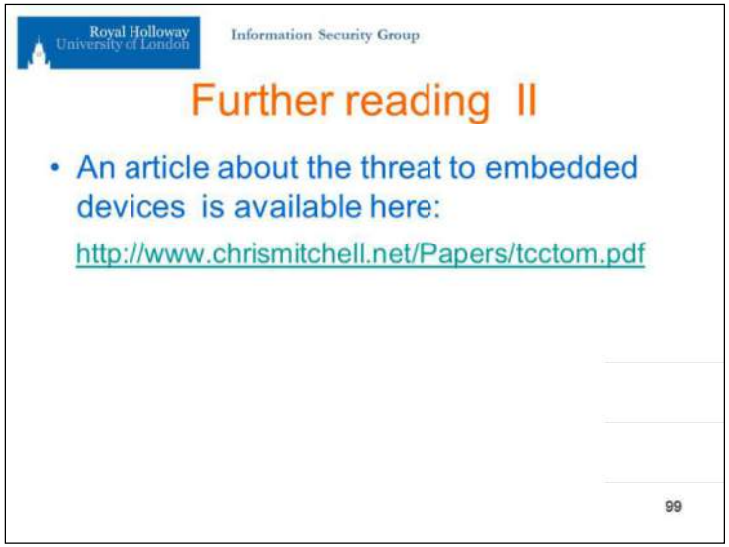
Are we all doomed? Well, perhaps not. There are some areas in which we might discern security-positive events. There is a growing diversity of platform types, covering everything from smart phones to games platforms, all of which seem to have possibilities as platforms for IT. Software engineering practices appear to be improving, at least as far as the big players are concerned. There is, it would seem, a growing awareness by the community at large of the seriousness of security threats. Finally, it is also becoming evident that not everyone wants the most sophisticated computing technology, as the rapid growth in netbooks and tablets demonstrates. Perhaps there is a future for the simpler, less flexible, and more secure/reliable product after all?

IY5512: Part 1



The following reading is recommended:

- Saltzer and Schroeder, *The protection of information in computer systems*, Section I
- The Orange Book, Section 6.
- Gollmann, *Computer Security*, Chapters 1, 2.
- Bishop, *Computer Security: Art and Science*, Chapters 1, 24.



An article about the threat to embedded devices is available here:

http://www.chrismitchell.net/Papers/tcctom.pdf