Computer Science Tripos part 2 – Security

Lecture 1 – Multilevel Secure Systems

Ross J. Anderson

1 Revision

In the 1b 'Introduction to Security' course, we looked at a number of different secure systems and explored their requirements.

1. We considered first of all the kind of system found in banks, which implements double-entry bookkeeping and where the main control is to ensure that only authorised programs operate on files containing account balances. These programs preserve an invariant (roughly, the total amount of money) by ensuring that whenever one account is credited by a certain amount, another is debited by the same amount. It is also possible to reconstruct account balances and other state at any time in the recent past by referring to audit trails.

2. We then looked at transaction processing systems, such as automatic teller machines (ATMs). These systems must also do bookkeeping reliably, but they also have communications security requirements: customer PINs are encrypted while in transit from the ATM through a (possibly international) network to the bank that issued the card. The security policy is dual control: no single bank employee should be able to get at a customer PIN. This is implemented using encryption and key management mechanisms.

3. We looked briefly at electronic warfare systems, where issues such as covertness and jamming become important.

4. We then looked at typical military database systems, which can hold information at a number of different levels of classification (confidential, secret, top secret, ...) and which have to ensure that data can only be read by a user whose clearance level dominates its classification.

5. Finally we touched on a number of other systems in passing, including conditional access systems used in pay-TV and copyright marking schemes.

This should bring home that the phrase 'secure system' is not at all self-explanatory. The protection properties we require depend on what we are trying to protect against whom, and good practice here (as elsewhere in system engineering) involves understanding the requirements in detail as much as writing the code.

Where a top-down approach is possible, this will typically take the form of **threat model – security policy – security mechanisms**:

- first, understand the threats against the system. This should wherever possible draw on empirical evidence of actual attacks against similar systems;
- next, write down as concisely as possible what the protection mechanisms are to achieve. This will often be in the form of statements about which users may access which data. This 'security policy', of which we shall give several examples in the next few lectures, may be expressed formally; it will be the baseline for evaluating the system;
- finally, we design the system, using appropriate mechanisms (such as access control lists and cryptographic protocols) to enforce the security policy. The strength of mechanisms that we use should be proportional to the threat.

Of course, we may not understand the threats properly at the beginning, or they might change over time; so we may end up with an iterative approach (whether spiral, evolutionary or whatever).

However, there are already a number of applications which have been extensively analysed, and where there is a more or less 'off-the-peg' solution that we can adopt. This solution may be available at several levels, of which the most visible will be products on the market and the most interesting for our purposes will be the *security policy model*.

The first and best known example of a security policy model was developed by Elliot Bell and Len LaPadula in the 1970's, in response to US military concerns over the security of time-sharing mainframe systems.

As we discussed in part 1b, the security of many computer systems is somewhat ramshackle, with operating system bugs and other exploits constantly being discovered. Not only is there the constant possibility that even unskilled users will discover loopholes and make opportunistic use of them, but there was a growing awareness of the threat from malicious code. (Viruses were not discovered until a decade later; the 70's concern was about password stealing Trojans and the like.)

This led to a reappraisal by the US government and a decision that a secure system should do one or two things well, with these protection properties enforced by simple mechanisms that would be relatively easy to verify and would change only rarely. But what are the security properties that should be enforced above all others?

2 Government Infosec Policy

Government systems often have the property that information is classified at a number of levels. In Britain, these run upwards from unclassified through Restricted, Confidential, Secret and Top Secret. Their official definitions are given in the appendix.

There is also a system of codewords whereby information, especially at Secret and Top Secret, can be further restricted. For example, information which might reveal intelligence sources or methods — such as the identities of agents or decrypts of foreign government traffic — is typically classified 'Top Secret Special Compartmented Intelligence' or TS/SCI. Other classifications might include 'Top Secret Crypto', 'Top Secret Atomic', or 'Secret UK/US Eyes Only'.

The level at which information is classified determines the handling procedures which must be used. At the lowest level, 'Restricted' information may be kept in an ordinary government office under lock and key. Higher levels of information may be kept in shielded rooms in which there are no open access photocopiers and where staff wear photo-id badges at all times. The NSA security manual (see my ftp site) gives some idea of the measures taken when dealing with information classified TS/SCI.

There are also descriptors, e.g. 'Honours'; however, these do not invoke any special handling requirements, so that we can handle a file marked 'Restricted — Honours' as if it were simply marked 'Restricted'.

Classifications together with codewords form a lattice: someone cleared to TS/SI would be entitled to read files classified 'Top Secret', 'Secret', 'Secret/SCI' and so on, but would have no access to files classified 'Top Secret Crypto' or even 'Secret Crypto'. Classification also determines how information is processed electronically, and this is our main interest in the topic.

3 Multilevel Secure Systems

It was in the context of the classification of military and intelligence data that the 'Bell-LaPadula' or BLP model of computer security was formulated in 1973. It is also known as 'multilevel security' and systems which implement it are often called multilevel secure or MLS systems. Their basic property is that processes can read down and write up, but never vice versa.

More formally, the Bell-LaPadula model enforces two properties:

- The simple security property: processes may not read data at a higher level. Also known as no read up (NRU);
- The *-property: processes may not write data to a lower level. Also known as no write down (NWD).

Using these assumptions, various results can be proved about which machine states can be reached from a given starting state.

The *-property was Bell and LaPadula's essential innovation. Without it, an MLS system cannot protect itself against malicious code; a virus or Trojan that manages to copy itself to a Top Secret directory, or have itself invoked by a user with a Top Secret clearance, could simply copy Top Secret documents down to a user without this clearance.

Multilevel security can be implemented in a number of ways. The traditional mechanism is a piece of software called a reference monitor, which supervises all operating system calls and compares the classification levels to decide whether the call can be serviced or not. Such systems are also described as having mandatory access control, as opposed to the discretionary access control in systems like Unix where users can take their own access decisions about files. Another approach that has been gaining ground as hardware costs have fallen is to replicate systems. One might, for example, have one database at 'restricted' and another at 'secret', with a 'pump' – a one-way information channel – that constantly copies information from 'restricted' up to 'secret'.

In order to simplify matters, we will assume that the system has only two levels, High and Low. The discussion here applies to general lattices with essentially no change; instead of information flows between High and Low, we consider flows between two levels one of which dominates another. (If two levels in a lattice are incompatible — neither dominates the other – then there should be no information flow between them at all.)

3.1 The standard criticisms of Bell-LaPadula

John McLean introduced System Z to show that the BLP rules were not in themselves enough. System Z has the feature that a user can ask the system administrator to temporarily declassify any file from High to Low; thus the Low user can read any High file without breaking the BLP assumptions.

This objection can be dealt with by introducing a tranquility property. The strong tranquility property says that security labels never change during system operation, while the weak tranquility property says that labels never change in such a way as to violate a defined security policy.

The motivation for the weak property is that in a real system we often want to observe the principle of least privilege and start off a process at the uncleared level, even if the owner of the process were cleared to Top Secret. Her process would then be upgraded each time it access data at a higher level – the so-called high water mark principle. Such upgrades would not normally break a sensible security policy.

There are a number of practical problems with the Bell-LaPadula model. There are always system components – such as memory management – which must be able to read and write at all levels. This objection is dealt with by abstracting it away, and assuming that memory management is part of the trusted computing base which enforces BLP. However, as we shall see later, resource management problems commonly lead to leaks in real systems.

Another objection is that it is often very inconvenient to deal with 'blind write-up' – when a low level application sends data to a higher level one, BLP prevents any acknowledgment being sent. The effect is that information vanishes into a 'black hole'. The answer to this is varied. Some organisations accept the problem as a fact of life; in the words of a former NSA chief scientist 'When you pray to God, you do not expect an individual acknowledgement of each prayer before saying the next one'.

A third is that many applications need to be rewritten or at least greatly modified to run under MLS operating systems. For example, compartmented mode workstations which display information at different levels in different windows, and prevent the user from doing cut-and-paste operations from high to low, often have problems with code which tries to manipulate the colour map. Another source of conflict with commercial software is the licence server; if a 'Top Secret' user invokes an application, which goes to a licence server for permission to execute, then an MLS operating system will promptly reclassify the server 'Top Secret', and thus deny access to lower level users.

A fourth is that the classification of data is not entirely straightforward:

- in the run-up to a military operation, the location of 'innocuous' stores such as food could reveal tactical options and intentions, and may be suddenly upgraded from 'restricted to 'secret';
- classifications are not necessarily monotone, in that an aircraft's inertial navigation set classified at 'restricted' in the peacetime inventory might contain a new laser gyro platform classified 'secret';
- information may be downgraded. One of the more important (and difficult) tasks is that of the intelligence analyst who takes agent and sigint reports classified at 'top secret' or above, and produces assessments for field commanders at 'secret'. Since these might fall into enemy hands, they must not reveal intelligence sources or methods;
- aggregation may be important, in that for example one might be able to declassify any single satellite photo, but declassifying the whole collection would reveal surveillance capability and the history of intelligence priorities.

For these reasons, the tranquility property cannot be assumed. However, the resulting information leakage will be too low to be a problem in most applications.

But the two problems that has most exercised the research and development communities are covert channels and polyinstantiation.

3.2 Polyinstantiation

Suppose that our High user has created a file named `agents`, and our Low user now tries to do the same. If the MLS operating system prohibits him, it will have leaked information – namely that there is a file called `agents` at High. But if it lets him, it will now have two files with the same name.

Often we can solve the problem by a naming convention, which could be as simple as giving that Low and High users different directories.

However, the problem remains a hard one in the world of databases. Suppose for example that a High user allocates a classified cargo to a ship. The system will not divulge this information to a Low user, who might think the ship is empty, and try to allocate it another cargo or even to change its destination.

The solution favoured in the USA for such systems is that the High user allocates a Low cover story at the same time as the real High cargo. Thus the underlying data will look something like

Level	Cargo	Destination
Secret	Missiles	Iran
Unclassified	Engine spares	Cyprus

In the UK, with no Freedom of Information Act, the practice is simpler – the system will automatically reply 'classified' to a Low user who tries to see or alter a High record. The two available views would be:

Level	Cargo	Destination
Secret	Missiles	Iran
Restricted	Classified	Classified

This makes the system engineering simpler; it also prevents the mistakes which can still arise with cover stories (e.g., a Low user tries to allocate a container of ammunition for Cyprus).

An example of a multilevel secure system is the RAF's Logistics Information Technology System (LITS). This was a 10 year (1989–99), £500m project to provide a single stores management system for the RAF's 80 bases, which are located in 12 countries. It was designed to operate on two levels: 'Restricted' for ordinary stores such as jet fuel and boot polish, and 'Secret' for special stores (this originally meant nuclear weapons, spare parts for classified aircraft, and stores whose location might divulge tactical intentions, e.g. in Bosnia).

It was initially implemented as two separate database systems, with a pump which lets information flow upwards from 'Restricted' to 'Secret'. This is a typical multilevel system design, and illustrates the philosophy of keeping the TCB as small as possible (in this case, essentially the pump and the 'Secret' level encryption devices). Note that there may be discretionary security features as well — for example, military stores systems may alert users who try to breach ordnance rules by putting explosives and detonators in the same magazine — but these reside in the application code rather than in the TCB.

With the end of the Cold War, the classification rules were eased. As a result, it was found that almost all the 'secret' information was now static (e.g., operating manuals for air-drop nuclear bombs which are now kept in strategic stockpiles rather than at airbases). In order to save money, the 'secret' information is now kept on a CD which is just locked up in a safe.

3.3 Covert Channels

For various reasons the protection which can be given by an MLS system is never absolute; all practical systems leak in one way or another, and the most famous (and extensively studied) problem with multilevel secure systems is the covert channel.

Covert channels were first pointed out by Lampson in 1973. A typical covert channel arises when a high process can signal to a low process by affecting some shared resource. For example, it could position the disk head at the outside of the drive at time t_i to signal that the i-th bit in a Top Secret file was a 1, and position it at the inside to signal that the bit was a 0.

All systems with shared resources suffer from them to some extent; the high process can signal by filling up the disk drive, or by using a lot of cpu or bus cycles (we would call the former case a storage channel and the latter a timing channel). Various strategies have been adopted to minimise their bandwidth; for example, we can arrange that the scheduler assigns a fixed disk quota to each level, and reads the boot sector each time control is passed downwards; we might also allocate a fixed proportion of the available time slices to processes at each level, and change these proportions infrequently. Each change might allow one or more bits to be signalled, but such strategies can enormously reduce the available bandwidth.

Downgrading information brings a whole raft of problems. Information could have been covertly hidden in a 'Top Secret' file by a rogue program, in order that it can be retrieved later once the file is declassified. For this reason, downgrading procedures may involve special word processors which remove nonprinting characters and reformat text, in the hope that the only information remaining is that which lies overtly in the text itself. I append some images showing how easy it is to hide one picture in another.

It is also possible to limit the covert channel capacity by introducing noise (recall Shannon's theorem that capacity = bandwidth$\{1+$ signal/noise$\}$). Some machines have had randomised system clocks for thus purpose; and one might try to prevent information leakage in declassified pictures by inserting as much noise as was acceptable into the low order bits.

However, some covert capacity always remains, and the best which developers have been able to do consistently with time sharing multilevel operating systems is to limit it to 1 bit per second or so. This may be adequate in an environment where we wish to prevent large TS/SI files – such as satellite photographs – leaking down from TS/SI users to Secret users, but it is considered insufficient to protect Top Secret data from uncleared users.

It is also totally inadequate if we want to prevent the leakage of a cryptographic key. There are some applications in which the best multilevel secure systems are just not good enough. This is one of the reasons for the military doctrine of always doing crypto in special purpose hardware rather than in software (nonproliferation is another).

3.4 The Effect of Viruses

The vast majority of viruses are of course to be found on PCs and Macs. However the defence computer security community was shocked when Fred Cohen, the inventor of the computer virus, used viruses to repeatedly penetrate multilevel secure systems in 1983 and 1984. In his first experiment, a file virus which took only eight hours to write managed to take over an MLS Unix system in between 5 and 30 minutes.

There are a number of ways in which viruses, and other malicious code, can be used to perform such attacks. If the reference monitor (or other TCB components) can be corrupted, then a virus could deliver the entire system to the attacker, for example by issuing him with an unauthorised clearance. If the TCB remains intact, then the virus could still use any available covert channel to signal information down.

A well designed TCB will provide some protection against viral attacks, as well as against careless disclosure by users or application software – which is often more important than the risk of malicious disclosure. However, the main effect of viruses on multilevel systems is that their existence strengthens the case for multilevel security. Designers assume that 'Walker is a component of the system' (after the Walker family who sold US Naval key material to the Soviets for a generation); even if personnel can be trusted, one cannot rely on technical measures short of total isolation to prevent viruses moving up the system, so one must do whatever reasonably possible to stop them signalling back down.

3.5 Alternative formulations

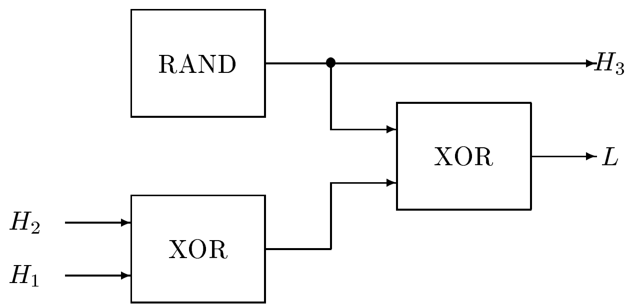
You may also see multilevel security properties expressed in other ways. The two most common are noninterference and nondeducibility.

Noninterference was introduced by Goguen and Meseguer in 1982, and its basic idea is to try and prove that what High does has no effect on what Low can see. Nondeducibility is similar; here the idea is to try and prove that Low cannot deduce anything about the state of High, and was introduced by Sutherland in 1986.

These formulations may be more useful in some specific situations; for example, nondeducibility may be helpful when we are trying to prove results about databases, and have to take into account any information which can be inferred from data structures (such as from partial views of data with redundancy) as well as considering the traces of executing programs, as we might under the noninterference or BLP models.

3.6 Composability

These formulations are also important when trying to prove results about the composition of two or more secure systems. Consider for example a simple device that accepts two 'high' inputs H_1 and H_2, multiplexes them, encrypts them by xor'ing them with a one-time pad, and then outputs the other copy of the pad on H_3 and the ciphertext, which being encrypted with a cipher system giving perfect secrecy, is considered to be low (output L).



In isolation this device is provably secure. However, if feedback is permitted, then the output from H_3 can be fed back into H_2, with the result that the high input H_1 now appears at the low output L.

As the attached classic paper by McCullough shows, timing inconsistences can also lead to the composition of two secure systems being insecure.

4 Verification and Evaluation

A mandatory access control mechanism is a trusted computing base, and so we might reasonably hope to find some way of evaluating the level of trust which we can place in it.

Here, there are essentially two traditions. IVV, or independent verification and validation, arose from the rocketry community and is mandatory both in nuclear weapons command and control and in manned space flight. In this tradition, a team from one contractor develops a piece of hardware or software, and a team from another contractor is then paid to look for bugs in it.

The much better known tradition is the evaluated products list, which is maintained by the US Department of Defense. Vendors submit their products to be evaluated by the NCSC, an arm of the NSA, who evaluate products according to the 'Trusted Computer System Evaluation Criteria' or TCSEC, also known as the Orange Book. This book is interpreted using the 'Rainbow Series' of books; for example, covert channel measurements are specified in the 'Pink Book', whose full title is 'A Guide to Understanding Covert Channel Analysis of Trusted Systems'.

The main evaluation classes are as follows:

C1 : discretionary access control by groups of users

C2 : discretionary access control by single users; object reuse; audit

B1 : mandatory access control — all objects carry security labels and the security policy (e.g., Bell-LaPadula) is enforced independently of user actions. Labelling is enforced for all input information.

B2 : structured protection — as B1 but there must also be a formal model of the security policy which has been proved consistent with security axioms. Tools must be provided for system administration and configuration management. The TCB must be properly structured and its interface clearly defined. Covert channel analysis must be performed. A trusted path must be provided from the user to the TCB. Severe testing, including penetration testing, must be carried out.

B3 : security domains — as B2 but the TCB must be minimal, it must mediate all access requests, it must be tamper-resistant, and it must withstand formal analysis and testing. There must be real-time monitoring and alerting mechanisms, and structured techniques must be used in implementation.

A1 : verification design. As B3, but formal techniques must be used to prove the equivalence between the TCB specification and the security policy model.

The evaluation class of a system determines what 'spread' of information may be processed simultaneously on it. For example, an A1 system can process 'Top Secret' through 'Confidential', or 'Secret' through unclassified. (Note in passing that if we connect two systems, one processing TS, S and C, and one S, C and U, then the combined system is processing TS through U – which should not happen. This is another hard problem with composing trusted systems.)

In Europe, the system is slightly different; there is a system of Commercial Licenced Evaluation Facilities (or CLEFs) which are accredited by GCHQ (and its foreign counterparts) to perform evaluations. These are done using criteria which have been harmonised across all the EC countries – the 'Information Technology Security Evaluation Criteria', or ITSEC. The US, Canada and Europe are in the process of moving to common criteria, in order to provide for a larger homogeneous defence equipment market. The common criteria are now available for public comment (see `www.itsec.gov.uk`).

So in both Europe and the USA, equipment sold to governments may have already been evaluated; even then, there is a further process, known as accreditation, during which a further assessment is carried out of whether they have been integrated properly and whether the system owner understands and has accepted responsibility for the residual risk.

Some commercial systems have been evaluated to class C2 (e.g., mainframe access control packages such as RACF, and Windows NT when run with a certain configuration on diskless workstations). However, many software firms consider the evaluation process to be too slow and expensive; a product evaluation may take a year or two and cost of the order of $1m, while in the commercial market, an inspection carried out for insurance purposes might cost a few thousand pounds and take a week or two.

A common abuse is for vendors to claim that their products are 'C2 compliant' or something similar. In fact, many products are never actually evaluated because of cost, and even where they are, the evaluation may be conditional on using the product with an obsolete version of the operating system. Claims above C2 are also doubtful, as such products are typically export controlled and are usually only supplied to cleared purchasers. It is thus prudent to clarify vendors' security claims and to ask for copies of their certificates.

5 The BIBA model

The second security policy model we will mention is the Biba model. We saw in the crypto lectures that confidentiality and integrity are in some sense dual concepts – confidentiality is a constraint on who can read a message, while integrity is a constraint on who may have written or altered it.

This duality also manifests itself in the multilevel secure world; where we build a multilevel integrity system, the rule is that we must read up (from higher integrity levels) and write down (to lower levels). This 'Biba integrity model' is exact opposite of BLP; it is often referred to as BLP upside-down. It is often formulated in terms of the low water mark principle, which is the dual of the high water mark principle discussed above.

6 Further Reading

One of the best introductions to MLS systems, and especially the problems of databases, is Dieter Gollmann's *Computer Security*. Ed Amoroso's *'Fundamentals of Computer Security technology'* is also very good, as is Castano's *Database Security*. Fred Cohen's experiments on breaking MLS systems using viruses are described in his book, *'A Short Course on Computer Viruses'*.

7 Problems

1. Create a noncomputing scenario to illustrate the duality between confidentiality and integrity in record keeping.

2. The CIA has called for proposals for an information pump through which its analysts can read usenet newsgroups. They do not want anyone to observe which groups are of interest at any particular time, so they want to subscribe to all groups everywhere and read them through an optical coupler. How would you go about designing such a system, and what problems would you anticipate?

3. Discuss the problems likely to be encountered during the annual audit of an airbase which uses a multilevel secure stores system.

4. What are the effects of priority of covert channel bandwidth (i.e., if High level tasks are assigned either a higher or lower priority than Low level ones)? Is this likely to provide us with anything useful?

5. Explain the Biba model in terms of low water marks, and discuss the strong and weak tranquility conditions in this context. Can you think of applications in which upgrading, nonmonotonicity or aggregation have interesting effects?

6. Unlike Unix, VAX/VMS allows you to give a user execute access to a file without giving him read access. However, he will still get write access to it; this is to support dynamic linking. Show how the write access can be used to read at least parts of an arbitrary executable file.

7. If a multilevel secure system has to operate in real time – with guaranteed maximum waiting times for processes at all levels – how does this affect security?

8. How would you implement a mail transfer agent on a multilevel secure system?

8 Appendix

The official definitions of the four main UK security levels are as follows. The US levels are similar, except in that there is no 'Restricted' level, and the interlevel boundaries are typically higher; thus a document marked 'Secret' in Britain might only be marked 'Confidential' in America.

TOP SECRET: The compromise of this information or material would be likely to threaten directly the internal stability of the UK or friendly countries; to lead directly to widespread loss of life; to cause exceptionally grave damage to the effectiveness or security of UK or allied forces or to the continuing effectiveness of extremely valuable security or intelligence operations; to cause exceptionally grave damage to relations with friendly governments; or to cause severe long-term damage to the UK economy.

SECRET: The compromise of this information or material would be likely to raise international tension; to damage seriously relations with friendly governments; to threaten life directly or seriously prejudice individual security or liberty; to cause serious damage to the operational effectiveness or security of UK or allied forces or the continuing effectiveness of highly valuable security or intelligence operations; to cause significant damage to UK economic and commercial interests; or to lead to additional government expenditure on a scale likely to affect the UK economy as a whole.

CONFIDENTIAL: The compromise of this information or material would be likely materially to damage diplomatic relations (i.e. to cause formal protest or other sanction); to prejudice individual security or liberty; to cause damage to the operational effectiveness or security of UK or allied forces or the effectiveness of valuable security or intelligence operations; to work significantly against UK economic and commercial interests; to cause significant financial loss to government or to other organisations or individuals; to lead to significant additional government expenditure; to impede the investigation, or facilitate the commission, of serious crime; to impede seriously the development or operation of major government policies; or to shut down or otherwise substantially disrupt significant national operations.

RESTRICTED: The compromise of this information or material would be likely to affect diplomatic relations adversely; to cause substantial distress to individuals; to make it more difficult to maintain the operational effectiveness of UK or allied forces; to prejudice the investigation of a crime; to breach proper undertakings to maintain the confidence of information provided by third parties; to impede the effective development of government policies; to breach statutory restrictions on disclosure of information; or to disadvantage government in negotiations with others.