BSc in Computer Engineering
CMP4103
Computer Systems and Network Security

Lecture 2

Security Architecture and Design

Eng Diarmuid O'Briain, CEng, CISSP

Department of Electrical and Computer Engineering,
College of Engineering, Design, Art and Technology,
Makerere University

Table of Contents

This page is intentionally blank

1. Operating Systems and Security

Many Operating Systems (OS) include some level of security. Security is based on the two ideas that:

- The OS provides access to a number of resources, directly or indirectly, such as files on a local disk, privileged system calls, personal information about users, and the services offered by the programs running on the system.
- The OS is capable of distinguishing between some requesters of these resources who are authorised to access the resource, and others who are not authorised.

1.1 Graphical user interfaces

Most modern OSs contain a Graphical User Interface (GUI). A few older OSs like the original implementations of Mac OS and Microsoft Windows tightly integrated the GUI to the kernel. More modern OSs are modular, separating the graphics subsystem from the kernel as is the case with GNU/Linux and Mac OS X.

Many OSs allow the user to install or create any user interface they desire. The X Window System in conjunction with a desktop environment like Gnome, KDE, Cinnamon, MATE, Xfce, LXDE or Enlightenment to name but a few are a commonly found setup on most UNIX and UNIX derivative (BSD, GNU/Linux) systems.

1.2 Device drivers

A device driver is a specific type of computer software developed to allow interaction with hardware devices. Typically this constitutes an interface for communicating with the device, through the specific computer bus or communications subsystem that the hardware is connected to, providing commands to and/or receiving data from the device, and on the other end, the requisite interfaces to the OS and software applications.

It is a specialised hardware-dependent computer program which is also OS specific that enables another program, typically an OS or applications software package or computer program running under the OS kernel, to interact transparently with a hardware device, and usually provides the requisite interrupt handling necessary for any necessary asynchronous time-dependent hardware interfacing needs.

1.3 UNIX®

UNIX is a computer OS originally developed in the 1960s and 1970s by a group of AT&T employees at Bell Labs including Ken Thompson, Dennis Ritchie and Douglas McIlroy. Today's UNIX systems are split into various branches, developed over time by AT&T as well as various commercial vendors and non-profit organisations.

The present owner of the trademark UNIX® is The Open Group, an industry standards consortium. Only systems fully compliant with and certified to the Single UNIX Specification qualify as "UNIX®" (others are called "UNIX system like" or "UNIX like").

During the late 1970s and early 1980s, UNIX's influence in academic circles led to large-scale adoption of UNIX by commercial start-ups, the most notable of which is Sun Microsystems. Today, in addition to certified UNIX systems, UNIX-like OSs such as GNU/Linux, Mac OS X and BSD derivatives are commonly encountered.

Examples include: Sun Solaris, HP UX, SCO UNIX, BSD UNIX.

http://www.unix.org

1.4 BSD UNIX

Berkeley Software Distribution (BSD, sometimes called Berkeley UNIX) is the UNIX derivative distributed by the University of California, Berkeley, starting in the 1970s. BSD should not be used to refer to the different BSD like OS around today. Instead they should be called BSDlike or BSD descendants.

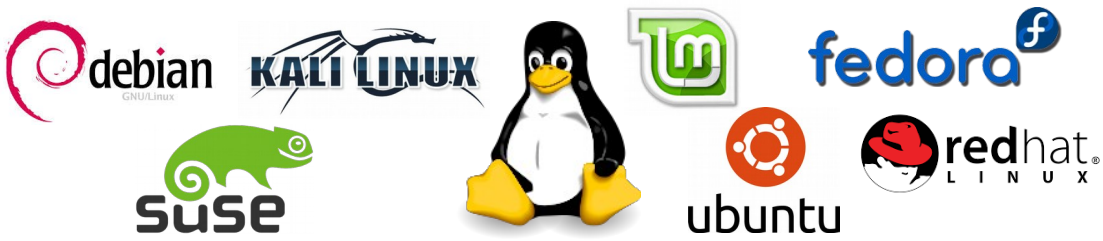
BSD is one of several branches of UNIX OSs. Another one is evolved from UNIX System V developed by AT&T's UNIX System Development Labs. A third consists of the GNU/Linux OSs which draw from UNIX System V and BSD, as well as Plan9, and non UNIX OSs.

http://www.freebsd.org http://www.bsd.org

1.5 GNU/Linux

GNU/Linux is a UNIX-like computer OS family, as well as one of the most prominent examples of Free and Open Source Software (FOSS) development; its underlying source code can be modified, used, and redistributed by anyone, freely.

After the Linux kernel was released to the public on 17 September 1991, the first GNU/Linux systems were completed by combining the kernel with system utilities and libraries from the GNU project, which led to the coining of the term GNU/Linux. From the late 1990s onward GNU/Linux gained the support of corporations such as IBM, Sun Microsystems, Hewlett-Packard, and Novell.



Predominantly known for its use in servers where it has displaced UNIX, GNU/Linux receives use as an OS for a wider variety of computer hardware than any other OS, including desktop computers, supercomputers, mainframes, and embedded devices such as mobile phones and tablets. GNU/Linux is packaged for different uses in GNU/Linux distributions, which contain the kernel along with a variety of other software packages tailored to requirements.

Examples include: Debian GNU/Linux, Red Hat Enterprise Linux (RHEL) and SuSE Linux.

Additionally there are many specialist GNU/Linux distributions that are based on the main distributions, for example Ubuntu and Kali Linux are based in Debian GNU/Linux while Linux Mint is based on Ubuntu. CentOS is a derivative of Red Hat Enterprise Linux and Fedora Core is a Red Hat derived OS that is used for advanced feature trialling before incorporation back into RHEL.

http://www.linux.org http://www.linux.com http://www.linuxfoundation.org

1.5.1 Mac OS X

Mac OS X is a line of proprietary, graphical OSs developed, marketed, and sold by Apple Inc., the latest of which is pre-loaded on all currently shipping Macintosh computers. Mac OS X is the successor to the original Mac OS, which had been Apple's primary OS since 1984. Unlike its predecessor, Mac OS X is a UNIX-like OS built on technology that had been developed at NeXT through the second half of the 1980s and up until Apple purchased the company in early 1997.

The OS was first released in 1999 as Mac OS X Server 1.0, with a desktop-oriented version (Mac OS X v10.0) following in March 2001. Since then more distinct 'end user' and 'server' editions of Mac OS X have been released like the being Mac OS X v10.4, which was first made available in April 2005. Older releases of Mac OS X were named after big cats; for example Mac OS X v10.4 is usually referred to by Apple and users as 'Tiger'. This has changed lately with v10.10 called OS X Yosemite released in the summer of 2014 and OS X El Capitan due for release in the autumn of 2015.

http://www.apple.com/osx/

1.6 Microsoft Windows

Microsoft Windows is the name of several families of proprietary software OSs by Microsoft. Microsoft first introduced an operating environment named Windows in November 1985 as an add-on to MS-DOS in response to the growing interest in graphical user interfaces (GUI). Microsoft Windows eventually came to dominate the world's personal computer market, overtaking OS/2 and Mac OSx which had been introduced earlier. In more recent times with the personal computer being overtaken by tablets and smart phones Microsoft has lost significant market share and its attempt to win back with Windows 8 with a tiled desktop frontend was not

successful. Windows 10, the latest release of the Windows OS has attempted to fix the perceived failings of Windows 8 and has been described by Microsoft as an 'OS as a Service'.

http://www.microsoft.com/en-us/windows/

1.7 Execution Types

1.7.1 Multitasking

This is a method by which multiple processes, share common processing resources such as a CPU. In the case of a computer with a single CPU, only one task is said to be running at any point in time, meaning that the CPU is actively executing instructions for that task. Multitasking solves the problem by scheduling which task may be the one running at any given time, and when another waiting task gets a turn. The act of reassigning a CPU from one task to another one is called a context switch. When context switches occur frequently enough the illusion of parallelism is achieved. Even on computers with more than one CPU (called multiprocessor machines), multitasking allows many more tasks to be run than there are CPUs.

1.7.2 Multiprocessing

Multiprocessing is the use of two or more CPUs within a single computer system. The term also refers to the ability of a system to support more than one processor and/or the ability to allocate tasks between them. There are many variations on this basic theme, and the definition of multiprocessing can vary with context, mostly as a function of how CPUs are defined (multiple cores on one die, multiple chips in one package, multiple packages in one system unit, etc.).

1.7.3 Multiprogramming

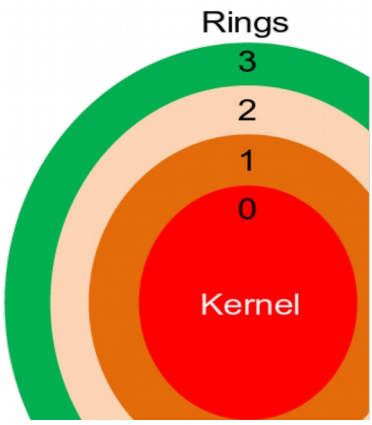
This is similar to Multitasking and comes from an era where CPU time was expensive, and peripherals were very slow. When the computer ran a program that needed access to a peripheral, the CPU would have to stop executing program instructions while the peripheral processed the data. This was deemed very inefficient.

The first efforts to create multiprogramming systems took place in the 1960s. Several different programs in batch were loaded in the computer memory, and the first one began to run. When the first program reached an instruction waiting for a peripheral, the context of this program was stored away, and the second program in memory was given a chance to run. The process continued until all programs finished running.

1.7.4 Multithreading

Multithreading computers have hardware support to efficiently execute multiple threads. These are distinguished from multiprocessing systems in that the threads have to share the resources of single core: the computing units, the CPU caches and the Translation Lookaside Buffer (TLB). Where multiprocessing systems include multiple complete processing units, multithreading aims to increase utilisation of a single core by leveraging thread-level as well as instruction-level parallelism.

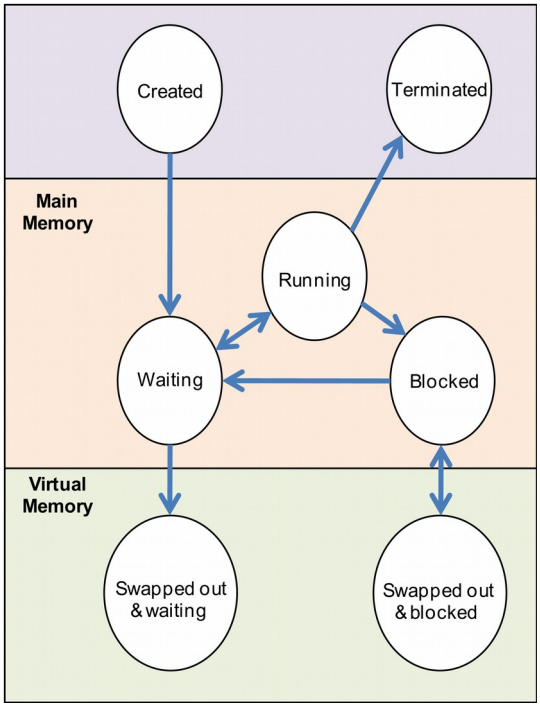
1.8 Protection Mechanisms



Ring 0: OS Kernel / Memory
Ring 1: Other OS components
Ring 2: Drivers
Ring 3: User-level programs and applications

Rings 0 – 2 run in privileged mode
Ring 3 runs in user mode

1.9 Process States



An OS kernel that allows multi-tasking needs processes to have certain states. Names for these states are not standardised, but they have similar functionality.
First, the process is "**created**" - it is loaded from a secondary storage device (hard disk or CD-ROM...) into main memory. After that the process scheduler assigns it the state "**waiting**".
While the process is "**waiting**" it waits for the scheduler to do a so-called context switch and load the process into the processor. The process state then becomes "**running**", and the processor executes the process instructions.

If a process needs to wait for a resource (wait for user input or file to open ...), it is assigned the "**blocked**" state. The process state is changed back to "**waiting**" when the process no longer needs to wait.

Once the process finishes execution, or is terminated by the OS, it is no longer needed. The process is removed instantly or is moved to the "**terminated**" state. When removed, it just waits to be removed from main memory.

1.10 Security Modes

Security modes refer to information systems security modes of operations used in Mandatory Access Control (MAC) systems. Often, these systems contain information at various levels of security classification. The mode of operation is determined by:

- The type of users who will be directly or indirectly accessing the system.
- The type of data, including classification levels, compartments, and categories, that are processed on the system.
- The type of levels of users, their need to know, and formal access approvals that the users will have.

Mode	Signed NDA for	Proper clearance for	Formal access approval for	A valid need to know for
Dedicated security	ALL information	ALL information	ALL information	ALL information
System high security	ALL information	ALL information	ALL information	SOME information
Compartmented security	ALL information	ALL information	SOME information	SOME information
Multilevel security	ALL information	SOME information	SOME information	SOME information

2. Security Architecture

2.1 Zachman Enterprise Architecture framework

	Why	**How**	**What**	**Who**	**Where**	**When**
Contextual	Goal list	Process list	Material list	Organisational unit and roles list	Geographical locations list	Event list
Conceptual	Goal relationship	Process relationship	Entity relationship model	Organisational unit and roles relationship model	Location model	Event model
Logical	Rules diagram	Process diagram	Data model diagram	Role relationship diagram	Location diagram	Event diagram
Physical	Rules specification	Process functional specification	Data entity specification	Role specification	Location specification	Event specification
Detailed	Rules details	Process details	Data details	Role details	Location details	Event details

The Zachman Framework is an Enterprise Architecture framework which provides a formal and highly structured way of viewing and defining an enterprise. It consists of a two dimensional classification matrix based on the intersection of six communication questions (What, Where, When, Why, Who and How) with six rows according to reification transformations.

2.2 Sherwood Applied Business Security Architecture (SABSA)

	Assets (WHAT)	Motivation (WHY)	Process (HOW)	People (WHO)	Location (WHERE)	Time (WHEN)
Contextual	The business	Business risk model	Business process model	Business organisation and relationships	Business geography	Business time dependencies
Conceptual	Business attributes profile	Control objectives	Security strageties and architectural layering	Security entity model and trust framework	Security domain model	Security related lifetime and deadlines
Logical	Business information model	Security policies	Security services	Entity schema and privilege profiles	Security domain definitions and associations	Security processing cycle
Physical	Business data model	Security rules, practices and procedures	Security mechanisms	User applications and user interface	Platform and network infrastructure	Control structure execution
Component	Detailed data structures	Security standards	Security products and tools	Identities, functions, actions and ACLs	Processes, nodes, addresses and protocols	Security step timing and execution
Operational	Assurance of operational continuity	Operational risk management	Security service management and support	Application, user management and support	Security of sites and platforms	Security operations schedule

SABSA is a framework and methodology for Enterprise Security Architecture and Service Management. It was developed independently from the Zachman Framework, but has a similar structure.

SABSA is a model and a methodology for developing risk-driven enterprise information security architectures and for delivering security infrastructure solutions that support critical business initiatives. The primary characteristic of the SABSA model is that everything must be derived from an analysis of the business requirements for security, especially those in which security has an enabling function through which new business opportunities can be developed and exploited.

The model is layered, with the top layer being the business requirements definition stage. At each lower layer a new level of abstraction and detail is developed, going through the definition of the conceptual architecture, logical services architecture, physical infrastructure architecture and finally at the lowest layer, the selection of technologies and products (component architecture).

2.3 ISO 7498-2

Basic Reference Model - Part 2: Security Architecture

Part 2 of ISO 7498 provides a general description of security services and related mechanisms, which may be provided by the Reference Model and defines the positions within the Reference Model where the services and mechanisms may be provided.

Part 2 of the ISO 7498 extends the field of application of ISO 7498, to cover secure communications between open systems.

2.4 ISO/IEC 42010:2007

ISO/IEC 42010:2007 addresses the activities of the creation, analysis and sustainment of architectures of software-intensive systems, and the recording of such architectures in terms of architectural descriptions. It establishes a conceptual framework for architectural description and defines the content of an architectural description.

2.5 The Open Group Architecture Framework (TOGAF)

TOGAF is a framework for enterprise architecture which provides a comprehensive approach to the design, planning, implementation, and governance of enterprise information architectures.

The architecture is typically modelled at four levels or domains:

1. Business
   - defines the business strategy, governance, organisation, and key business processes of the organisation.

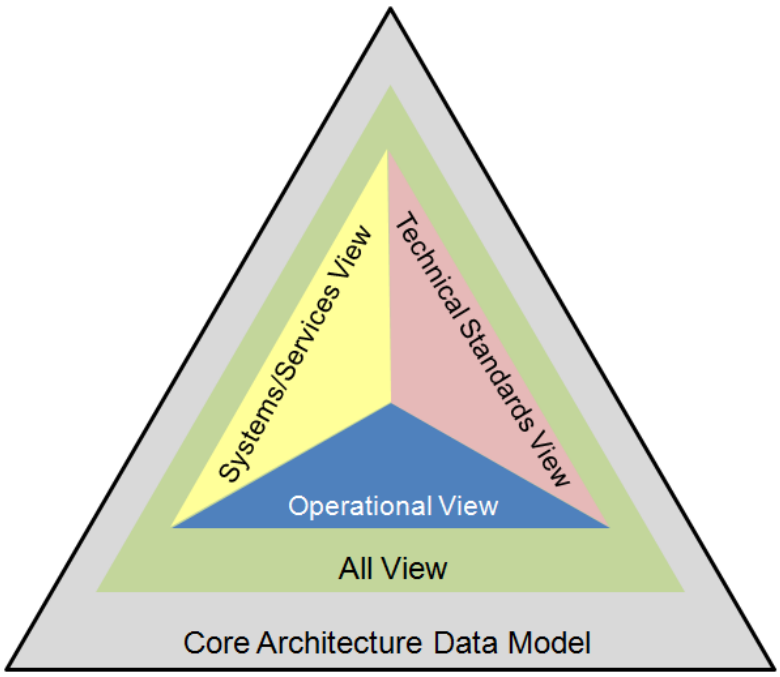
2. Application
   - provides a blueprint for the individual application systems to be deployed, the interactions between the application systems, and their relationships to the core business processes of the organisation with the frameworks for services to be exposed as business functions for integration.

3. Data
   - describes the structure of an organisation's logical and physical data assets and the associated data management resources.

4. Technology
   - describes the hardware, software and network infrastructure needed to support the deployment of core, mission-critical applications.

A set of foundation architectures are provided to enable the architecture team to envision the current and future state of the architecture.

2.6 Department of Defence Architecture Framework (DoDAF)



DoDAF is a reference model to organise the enterprise architecture (EA) and systems architecture into complementary and consistent views.

The DoDAF defines a set of products that act as mechanisms for visualising, understanding, and assimilating the broad scope and complexities of an architecture description through graphic, tabular, or textual means.

It is especially suited to large systems with complex integration and interoperability challenges, and is apparently unique in its use of "operational views" detailing the external customer's operating domain in which the developing system will operate.

DoDAF Views

- All View (AV)
  - provides overarching descriptions of the entire architecture and define the scope and context of the architecture.
- Operational View (OV)
  - provides descriptions of the tasks and activities, operational elements, and information exchanges required to accomplish DoD missions.
- Systems and Services View (SV)
  - a set of graphical and textual products that describe systems and services and interconnections providing for, or supporting, DoD functions.
- Technical Standards View (TV)
  - define technical standards, implementation conventions, business rules and criteria that govern the architecture.

3. Security Models

3.1 Trusted Computing Base (TCB)

The TCB of a computer system is the set of all hardware, firmware, and/or software components that are critical to its security, in the sense that bugs or vulnerabilities occurring inside the TCB might jeopardise the security properties of the entire system. By contrast, parts of a computer system outside the TCB must not be able to misbehave in a way that would leak any more privileges than are granted to them in accordance to the security policy.

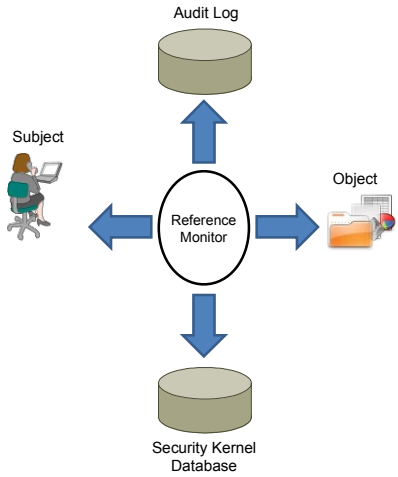
The careful design and implementation of a system's trusted computing base is paramount to its overall security. Modern OSs strive to reduce the size of the TCB so that an exhaustive examination of its code base (by means of manual or computer-assisted software audit or program verification) becomes feasible.

3.1.1 Security Perimeter

This is an imaginary boundary that separates the TCB from the rest of the system.

3.1.2 Reference Monitor

This is a tamperproof, always-invoked, and small enough to be fully tested and analysed module of the TCB that controls all software access to data objects or devices. The reference monitor verifies the nature of the request against a table of allowable access types for each process on the system.



3.1.3 Security Kernel

This is the hardware and software of the TCB that implements the reference monitor.

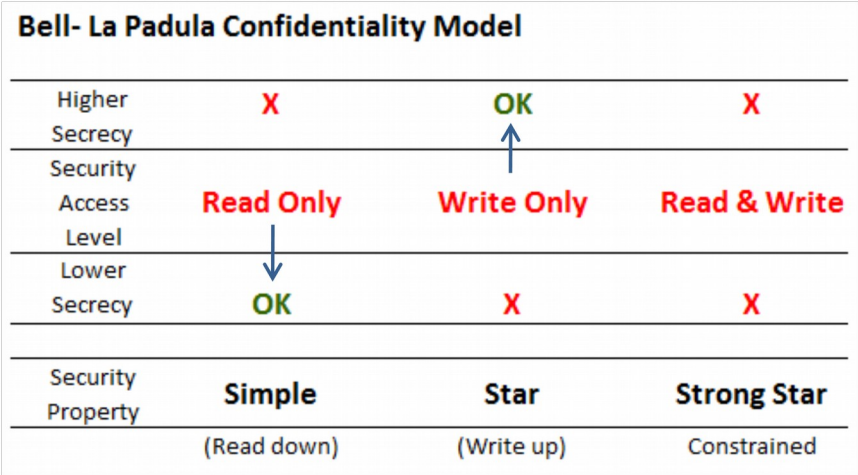
3.2 Model Summary

Integrity	Confidentiality
Clark Wilson	Bell-la Padula
Biba	Chinese Wall
Graham Denning	Brewer & Nash
G&M	
Sutherland	

3.3 Bell-La Padula Model (BLP)

The Bell-La Padula Model is a state machine model used for enforcing access control in US government and military applications. It was developed by David Elliott Bell and Leonard J. La Padula, subsequent to strong guidance from Roger R. Schell to formalise the U.S. DoD Multi-Level Security (MLS) policy.

The model focuses on data confidentiality and access to classified information. In this formal model, the entities in an information system are divided into subjects and objects. The notion of a "secure state" is defined, and it is proven that each state transition preserves security by moving from secure state to secure state, thereby inductively proving that the system satisfies the security objectives of the model.



The model defines two Mandatory Access Control (MAC) rules and one Discretionary Access Control (DAC) rule with three security properties:

The **Simple Property** - a subject at a given security level may not read an object at a higher security level (no read-up).

The ***-property** ("star"-property) - a subject at a given security level must not write to any object at a lower security level (no write-down). The *-property is also known as the Confinement property.

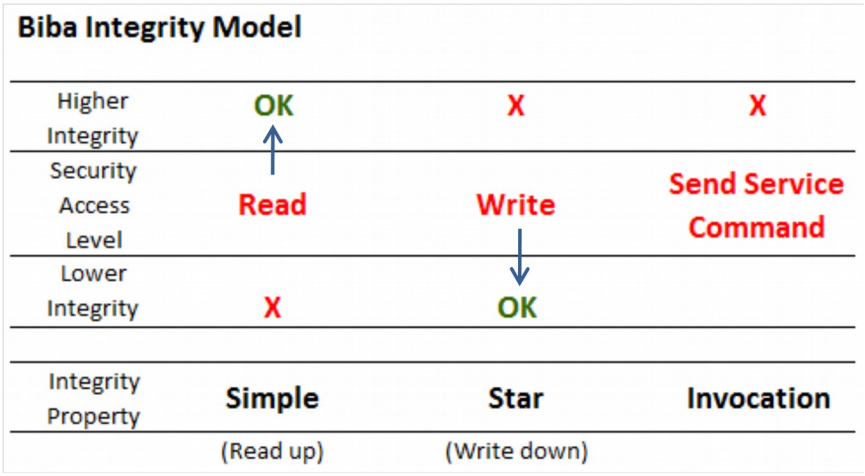
The **Strong *-property** ("strong star"-property) – If you have Read and Write properties then you are restricted to read and write at your level of security but you cannot read or write at higher or lower levels of security.

A **Discretionary Access Control** (DAC) which provides a specific access matrix to specify the discretionary access permissions.

	Application A	Application B	File A	Device A
Role A	read, write, execute, owner	execute	read	write
Role B	read, execute	read, write, execute, owner		

3.4 Biba Integrity Model

The Biba Integrity Model developed by Kenneth J. Biba in 1977, is a formal state transition system of computer security policy that describes a set of access control rules designed to ensure data integrity.



Data and subjects are grouped into ordered levels of integrity. The model is designed so that subjects may not corrupt data in a level ranked higher than the subject, or be corrupted by data from a lower level than the subject.

In general the model was developed to circumvent a weakness in the Bell-La Padula Model which only addresses data confidentiality. The Biba model defines a set of security rules similar to the Bell-La Padula model. These rules are the reverse of the Bell-La Padula rules:

1. The **Simple** Integrity axiom states that a subject at a given level of integrity must not read an object at a lower integrity level (no read down).
2. The ***-property** (star) Integrity axiom states that a subject at a given level of integrity must not write to any object at a higher level of integrity (no write up).
3. With the **Invocation** property a process from below can not request access at a higher integrity level.

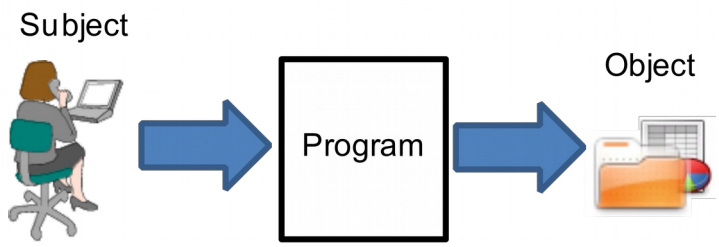
3.5 Clark-Wilson model

This model provides a foundation for specifying and analysing an integrity policy for a computing system.

The model is primarily concerned with formalising the notion of information integrity. Information integrity is maintained by preventing corruption of data items in a system due to either error or malicious intent. An integrity policy describes how the data items in the system should be kept valid from one state of the system to the next and specifies the capabilities of various principals in the system. The model defines enforcement rules and certification rules.

The core of the model is based on the notion of a transaction:

- A well-formed transaction is a series of operations that transition a system from one consistent state to another consistent state.
- In this model the integrity policy addresses the integrity of the transactions.
- The principle of Separation of Duty (SoD) requires that the certifier of a transaction and the implementer be different entities.

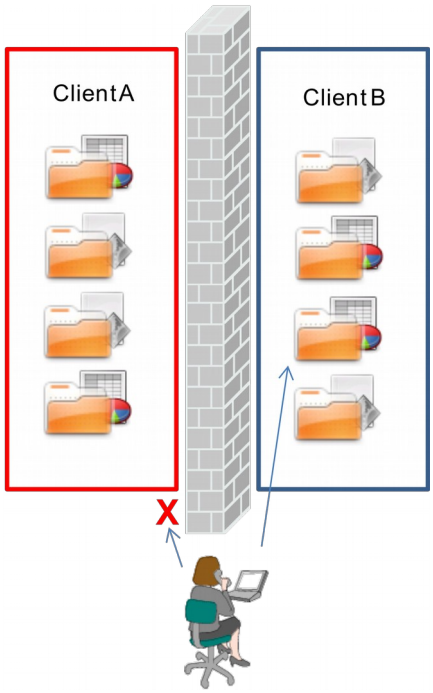


Access Triple **– SUBJECT – PROGRAM – OBJECT** relationship. The subject making a change to the object must comply with the restrictions built into the program which prevents the subject making an inappropriate adjustment of the object. A **well formed transaction** is one where the changes requested by the subject meet the integrity rules.

3.6 Brewer and Nash model

This model was constructed to provide information security access controls that can change dynamically. This security model, also known as the **Chinese wall** model, was designed to provide controls that mitigate conflict of interest in commercial organisations, and is built upon an information flow model.

In the Brewer and Nash Model no information can flow between the subjects and objects in a way that would create a conflict of interest.

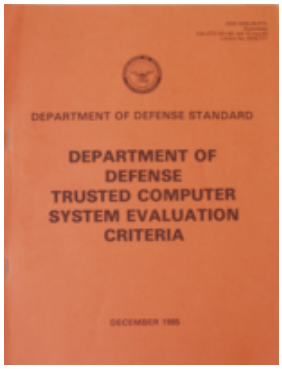


4. Evaluation Models

The following are evaluation standards which can be used to evaluate systems security.

- Trusted Computer System Evaluation Criteria (TCSEC)
  - US DoD
  - Orange book
- Information Technology Security Evaluation Criteria (ITSEC)
  - European Union (EU)
- Common Criteria for Information Technology Security Evaluation
  - ISO/IEC 15408

4.1 Trusted Computer System Evaluation Criteria



Trusted Computer System Evaluation Criteria (TCSEC) is a United States Government Department of Defence (DoD) standard that sets basic requirements for assessing the effectiveness of computer security controls built into a computer system. It essentially is an implementation of the Bell-La Padula security model. The TCSEC was used to evaluate, classify and select computer systems being considered for the processing, storage and retrieval of sensitive or classified information.

The TCSEC, frequently referred to as the Orange Book, is the centrepiece of the DoD Rainbow Series publications. Initially issued in 1983 by the National Computer Security Centre (NCSC), an arm of the National Security Agency, and then updated in 1985, TCSEC was replaced by the Common Criteria international standard originally published in 2005.

4.1.1 TCSEC Classes

The TCSEC defines four divisions: D, C, B and A where division A has the highest security. Each division represents a significant difference in the trust an individual or organisation can place on the evaluated system.

- **D — Minimal protection**
- **C — Discretionary protection**
  - C1 — Discretionary Security Protection
  - C2 — Controlled Access Protection
- **B — Mandatory protection**
  - B1 — Labelled Security Protection
  - B2 — Structured Protection
  - B3 — Security Domains
- **A — Verified protection**
  - A1 — Verified Design

4.2 Information Technology Security Evaluation Criteria

Information Technology Security Evaluation Criteria (ITSEC)

The Information Technology Security Evaluation Criteria (ITSEC)[1] is a structured set of criteria for evaluating computer security within products and systems. The ITSEC was first published in May 1990 in France, Germany, the Netherlands, and the United Kingdom based on existing work in their respective countries. Following extensive international review, Version 1.2 was subsequently published in June 1991 by the Commission of the European Communities for operational use within evaluation and certification schemes.

Since the launch of the ITSEC in 1990, a number of other European countries have agreed to recognise the validity of ITSEC evaluations.

The ITSEC has been largely replaced by Common Criteria, which provides similarly-defined evaluation levels and implements the target of evaluation concept and the Security Target document.

4.2.1 ITSEC Classes

ITSEC defines two ratings:

- Assurance (E0 – E6)
- Functionality (F-C1, F-C2, F-B1, F-B2, F-B3)

The product or system being evaluated, called the target of evaluation, is subjected to a detailed examination of its security features culminating in comprehensive and informed functional and penetration testing. The degree of examination depends upon the level of confidence desired in the target. To provide different levels of confidence, the ITSEC defines **evaluation levels**, denoted E0 through E6. Higher evaluation levels involve more extensive examination and testing of the target.

Unlike earlier criteria like TCSEC, ITSEC did not require evaluated targets to contain specific technical features in order to achieve a particular assurance level. For example, an ITSEC target might provide authentication or integrity features without providing confidentiality or availability. A given target's security features are documented in a Security Target document, whose contents have to be evaluated and approved before the target itself is evaluated. Each ITSEC evaluation is based exclusively on verifying the security features identified in the Security Target.

[1] http://www.iwar.org.uk/comsec/resources/standards/itsec.htm

4.3 Common Criteria

The Common Criteria for Information Technology Security Evaluation[1] is an international standard (ISO/IEC 15408) for computer security certification.

Common Criteria is a framework in which computer system users can specify their security functional and assurance requirements, vendors can then implement and/or make claims about the security attributes of their products, and testing laboratories can evaluate the products to determine if they actually meet the claims. In other words, Common Criteria provides assurance that the process of specification, implementation and evaluation of a computer security product has been conducted in a rigorous and standard manner.

4.3.1 Evaluation Assurance Level (EAL)

The EAL (EAL1 through EAL7) of an IT product or system is a numerical grade assigned following the completion of a Common Criteria security evaluation, an international standard in effect since 1999. The increasing assurance levels reflect added assurance requirements that must be met to achieve Common Criteria certification. The intent of the higher levels is to provide higher confidence that the system's principal security features are reliably implemented. The EAL level does not measure the security of the system itself, it simply states at what level the system was tested to see if it meets all the requirements of its Protection Profile.

4.4 Comparison of Evaluation Levels

CC	TCSEC	ITSEC
	D	F-D E0
EAL 1		
EAL 2	C1	F-C1 E1
EAL 3	C2	F-C2 E2
EAL 4	B1	F-B1 E3
EAL 5	B2	F-B2 E4
EAL 6	B3	F-B3 E5
EAL 7	A1	F-B3 E6

[1] http://www.commoncriteriaportal.org/

4.5 Flaw remediation

Additional to the EAL level an evaluation of flaw remediation can be added to the assurance level which further provides assurance as to the quality of the product. There are three levels.

ALC_FLR.1

This level provides for the identification of security measures where developers provide documented flaw remediation procedures.

ALC_FLR.2

Further to documenting flaws, a flaw reporting procedural mechanism to ensure that any reported flaws are corrected.

ALC_FLR.3

This flaw remediation demonstrates that a systematic flaw remediation mechanism exists.

4.6 Certification and Accreditation of Systems [1]

Certification is the evaluation of both technical and non-technical security features of an information technology system.

Accreditation is the acceptance by management that the certified system meets the security needs of the organisation.

- **Phase 1 : Definition**
  - Assignment of project personnel
  - Documentation of need
  - System Security Authorisation Agreement (SSAA)
- **Phase 2 : Verification**
  - Refinement of SSAA
  - Systems development activity
  - Certification analysis
- **Phase 3 : Validation**
  - Further refinement of SSAA
  - Certification evaluation of the integrated system
  - Development of a recommendation for Designated Approving Authority (DAA)
  - DAA's accreditation decision
- **Phase 4 : Post Accreditation**
  - Maintenance of SSAA
  - Systems operation
  - Change management
  - Compliance validation

[1] https://www.commoncriteriaportal.org/files/ppfiles/pp0067b_pdf.pdf

4.7 Certified products [1]

Name	Company	Assurance Level
Apple Mac OS X 10.6	Apple Inc.	EAL3+
Citrix XenServer 6.0.2 Platinum Edition	Citrix Systems, Inc.	EAL2+,ALC_FLR.2
Red Hat Enterprise Linux Ver. 5.3 on Dell 11G Family Servers	Dell, Inc.	EAL4+,ALC_FLR.3
HP HP-UX 11i v3 (using CCv3.1)	HP Company	EAL4+,ALC_FLR.3
Red Hat Enterprise Linux, v3 Update 3	HP Company	EAL3+,ALC_FLR.3
IBM AIX 7 for POWER V7.1 Technology level 7100-00-03 with optional IBM Virtual I/O Server V2.2	IBM Corporation	EAL4+,ALC_FLR.3
Microsoft Windows 8 and Windows RT	Microsoft Corporation	None
Microsoft Windows 8 and Windows Server 2012	Microsoft Corporation	None
Microsoft Windows Server 2008 R2 Hyper-V Release 6.1.7600	Microsoft Corporation	EAL4+,ALC_FLR.3
Microsoft Windows Mobile 6.5	Microsoft Corporation	EAL4+
Microsoft Windows Vista and Windows Server 2008	Microsoft Corporation	EAL1
Oracle Solaris 11.1	Oracle Corporation	EAL4+,ALC_FLR.3
Oracle Enterprise Linux v5 Update 1	Oracle Corporation	EAL4+,ALC_FLR.3
Oracle Enterprise Linux v4 Update 5	Oracle Corporation	EAL4+,ALC_FLR.3
Red Hat Enterprise Linux on 32 bit x86 Architecture, v6.2	Red Hat, Inc.	EAL4+,ALC_FLR.3
Red Hat Enterprise Linux v6.2 on IBM Hardware for Power and System z Architectures	Red Hat, Inc.	EAL4+,ALC_FLR.3
Red Hat Enterprise Linux, v3 Update 2	Red Hat, Inc.	EAL3+,ALC_FLR.3
SUSE Linux Enterprise Server 11 Service Pack 2 on IBM System z	SUSE Linux Gmbh	EAL4+,ALC_FLR.3
VMware ESX 4.0 Update 1 and vCenter Server 4.0 Update 1	VMware, Inc.	EAL4+,ALC_FLR.2
VMware ESXi 4.0 Update 1 and vCenter Server 4.0 Update 1	VMware, Inc.	EAL4+,ALC_FLR.2

[1] http://www.commoncriteriaportal.org/products/certified_products.csv

5. Bibliography

(2012). COBIT 5, An ISACA Framework. A Business Framework for the Governance and Management of Enterprise IT. ISACA.

Lahti C, Peterson R (2005). Sarbanes-Oxley Compliance Using COBIT and Open Source Tools. First Edition Edition. Elsevier / Syngress.

Faris C, Gilbert B, LeBlanc B. (2013). Integrating the triple bottom line into an enterprise risk management program. [ONLINE] Available at: http://www.coso.org/documents/COSO-ERM%20Demystifying%20Sustainability%20Risk_Full%20WEB.pdf. [Accessed 06 December 2013].

Henning D (2009). Tackling ISO 27001: A Project to Build an ISMS. [ONLINE] Available at: http://www.iso27001security.com/GIAC_GCPM_gold_henning.pdf. [Accessed 06 December 2013].

ITIL (2013). ITIL - Core Cabinet Office Material . [ONLINE] Available at: http://www.itil-officialsite.com/Publications/Core.aspx. [Accessed 06 December 2013]. APM Group Ltd.

SEL (2010). CMMI for Development, Version 1.3. Carnegie Mellon University, SEI.

SEI (2010). CMMI for Services, Version 1.3. Carnegie Mellon University, SEI.

NIST (2012). Information Security: Guide for Conducting Risk Assessments. SP 800-30. NIST.

(ISC)² (2012) Official (ISC)² Guide to the CISSP Common Body of Knowledge. Third Edition.