Federal Office
for Information Security

A study on behalf of the
German Federal Office for Information Security (BSI)

Complementary and Alternative Technologies to Trusted Computing (TC-Erg./-A.) - Part 1 -

An Analysis of Security Techniques and Technologies
Complementary to Trusted Computing

Version 1.0 / 27. January 2010

Sirrix AG
security technologies

Summary:

The goal of this study is to explain and analyze contemporary security concepts, security models and trust models used in information processing systems. These are categorized and compared according to their differences and possible combinations. One focus of this study is to integrate the emerging concept of Trusted Computing into well-known structures, examining relations between Trusted Computing and these state-of-the-art concepts and further discussing meaningful combinations.

Authors:

Lothar Fritsch, Rani Husseiki, Ammar Alkassar

Sirrix AG security technologies
Im Stadtwald, Building D3
66123 Saarbrücken – Germany
Web: http://www.sirrix.com

Florian v. Samson

Bundesamt für Sicherheit in der Informationstechnik (BSI)
Postfach 200363
53133 Bonn – German
Web: http://www.bsi.de

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This study will present an introduction to the topic of security concepts, security models and Trusted Computing. It targets computer-literate readers with very little IT security knowledge. Readers of this study will be instructed about basic terms and concepts of information security, access control, and security goals. An overview of current security models and their trust assumptions will be given. The technology of Trusted Computing will be explained and set in relation to these security and trust models.

For practical purposes, current operating systems supporting the security technologies that are reviewed by this study will be briefly mentioned.

The study will provide references to technical and scientific literature supporting and elaborating the study's content.

The relevance of "Secure Operating Systems" as a base for application programs has strongly increased in recent years. As more complex information workflows establish themselves in private and public environments, they become exposed to internet connectivity. This leaves these workflows vulnerable to malware attacks (viruses, worms, spyware, etc.). Therefore, the need for securing and controlling information flow between interconnected platforms has increased. To achieve access control in unified ways, operating systems have implemented various security concepts over the last decades (e.g., Multi-Level Security (MLS), Compartmented Mode Security (CMS), Mandatory Access Control (MAC), and Discretionary Access Control (DAC)). Additionally, security models have been specified and implemented (e.g., Bell-LaPadula, Role-based Access Control (RBAC)). Current research proposes further concepts, e.g., Multilateral Security and the sHype Security Model. Hardware extensions such as the "Trusted Platform Module" of the Trusted Computing Group provide a modified perspective of trust in computer systems. These hardware extensions have great influence on the practical use and effectiveness of the above-mentioned security concepts.

Currently, many of the current developments in the areas of security models and Trusted Computing are available as FLOSS Programs („Free, Libre, Open Source Software"), but are mostly used for special-purpose application programs only. For this reason, this study researches the following topics:

1. The study analyzes existing security concepts, security models and trust models. These are compared to each other concerning their differences and their combination in practice.

2. The study researches potential application areas of the security concepts, security models and trust models as well as their respective security levels. It compares the effectiveness of their usage and possible combinations.

3. The study reviews free[1] implementations of the above security concepts, security models and trust models. The practical usability and effectiveness of these implementations is the major focus of this part of the study.

[1] "Free" refers here to FLOSS-Licensing as defined by the Open Source Initiative http://www.opensource.org/docs/definition.php.

Chapter 2

Definitions and Taxonomy

In this chapter, the terminology and the taxonomy of security will be clarified as is relevant for this study. There is indeed considerable potential for confusion in the fields of security concepts, security models and trust models. These three fields and other relevant vocabulary will be defined, clarified and differentiated from each other. As a result, this chapter acts as a brief reference while the details will be defined and explained in the following chapters.

This study focuses on security of data that is kept on computer systems. For the sake of simplicity, only three security properties, namely confidentiality, integrity and authenticity[2] are considered in this study. The other security property, availability [Rann1994], is based on a distinct approach that is beyond the scope of this study.

2.1 Definitions

This section defines the important terms and relates them to each other, with references given where necessary. Terms are introduced alphabetically, cross-referencing other terms where needed. Terms that are defined here are printed in boldface. For a quick start, the reader is invited to examine Figure 2.1 and read the definition of the terms security concept, security model and trust model to get familiarized with the basics of IT security. Please refer to the following sections on conflicts and misunderstandings in IT security as well.

The basic setting for IT security involves data objects that are protected from the actions of a subject according to security goals. One basic protection approach is called the security concept. Within the concept, one or more particular security models implement the mechanisms that enforce IT security.

Access control: Access control is a technique to manage the rights of a subject to access data or computer system functions.

Action: See **Operation.**

Confidentiality: Data remains confidential against unauthorized reading or observation. Sometimes, even the fact that data is there shall be kept confidential.

Integrity: Data remains in its intended state. It cannot be modified or deleted without proper authorization. Further elaboration on this term can be found in [PfKo2001].

Object: A portion of data (e.g., a file, data base entry or web object) that is stored on a computer system. The handling of objects is the focus of **security concepts** and **security models.**

[2] Authenticity can also be called accountability/non-repudiation with slightly different interpretations.

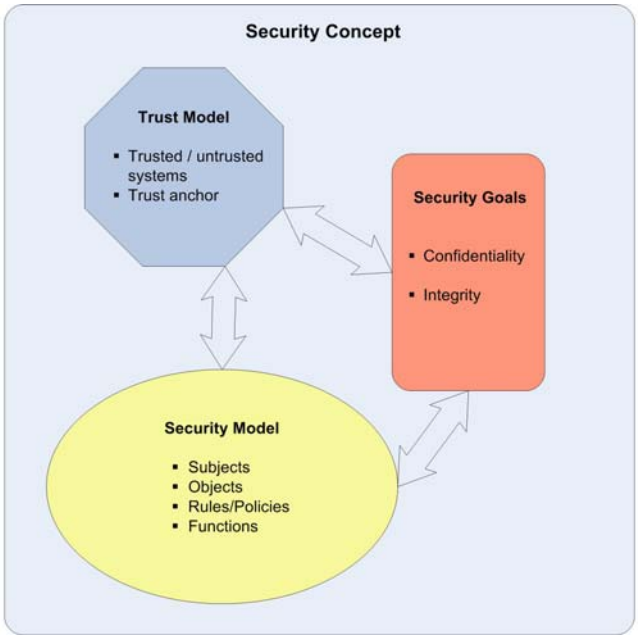

Figure 2.1: Basic IT security terms and their relation to each other.

Operation: Any operation on an object that is relevant to the security of the object. Example operations: read, write, delete, overwrite, copy, move, mail, and print.

Policy: see **Security Policy.**

Security anchor: A security anchor is a mechanism in a security model implementation that establishes a secure and trustworthy starting point for trust relationships. For example, this can be a manipulation-proof chip with secure startup software for a computer.

Security concept: An approach on how to structure a computer system in a way that it has control over the security of data.

Security goal: A specification of the security properties that a computer or database system should have. This is an important part of a **security concept.** Typical security goals are confidentiality, integrity and availability of data. For operations, the goal of accountability can be added. Upon specification of security goals, a more detailed analysis with consideration of the use cases is performed.

Security model: A detailed model on how **security policies** can be defined, enforced and managed. A security model usually involves models of **subjects, objects, operations** and **security goals.**

Security policy: A set of rules that specify which **subjects** are allowed to perform which security-relevant **operations** on which **objects**.

Subject: A person (e.g., user of a computer) or program acting on behalf of a person. A subject makes operations on **objects** guarded by **security policies**. Subjects must not be confused with "data subjects" as used in [PfKo2001], which is a data protection term.

Trusted Computing: A platform and security protocol specification made by the Trusted Computing Group [TCG0001]. Its purpose is to provide to computer systems a hardware-secured component and a set of secure algorithms that can establish a **security anchor** for secure computer system startup and trust enforcement.

Trust model: A specification of the trustworthiness of the technical components in a **security model.** Its purpose is the definition of important components that are not supposed to fail, or to point out assumptions that are of general importance to the security of the computer system. For example, on a computer system's authentication based on passwords, the program that checks the passwords against the database must be trusted. If it fails, the security of the computer system is compromised.

Unobservability: The inability to discover the presence of particular data, the fact that communication has taken place, or the identity of subjects who have corresponded with one another.

2.2 Conflicts

Many of the terms introduced above are ambiguous in practice. The definitions are selected to match the scope of this study. Concerning security terminology, the terms refer to the usage within the information security context. There is another security field often referred to as "Safety" which focuses on reliability, prevention of failures, stability of computer systems and the like. Here, the same terms or similar expressions can be used with different meanings.

Furthermore, in the context of data protection and online privacy, some terms might be used differently. This should not be a concern for readers of this study, but for reference in case of doubt, [PfKo2001] provides a terminology from the data protection world.

Carefulness is required when using the terms security concept, security model, trust model, and discussing their implementations in operating systems and application programs. Some sources confuse these terms with each other. When articles additionally discuss attacker models, threat models and other terms, careful interpretation is necessary for understanding the respective meanings.

2.3 Common Misunderstandings

This section lists and resolves common misunderstandings that are part of the public debate about information security. Some of these misunderstandings greatly influence the perception of IT security. For clarification and awareness, those will be explained below.

- **Digital Rights Management versus Trusted Computing:** A recent concept in the computer security filed is Trusted Computing [Pear2002]. Trusted Computing refers to the specification of the Trusted Computing Group [TCG0001] for hardware support in secure computer systems. This approach requires installing a secure chip into computers, which enables operating systems to check whether loaded software has been manipulated or not. Much debate evolved around Trusted Computing when Microsoft and others discussed further potential usages of Trusted Computing that support media control with Digital Rights Management. The concern was that Trusted Computing can be used to ensure that specific media player software is used that will enforce media policies. This concept has received a vast amount of negative press, with people fearing that computers non-conforming to certain policies could be disconnected from the internet, and that corporations could dictate the capabilities of people on their computers. All these negative speculations seem to have stuck to the term "Trusted Computing", without tangible proofs. Trusted Computing can ensure the correct loading of any security software. Whether it enforces any oppressive measures or not depends on the operating system – and its administrator. Section 6.1 elaborates on this topic.

- Trust is a frequently-used term in the security business. Unfortunately, trust in people is frequently confused with trust in security models or computer system components. When IT security people discuss trust, they consider its relevance in technical systems. One example is the trust that an automatic update of virus definitions by an anti-virus program will include meaningful and useful virus definitions – rather than sabotage the anti-virus tool. One usually assumes that the vendor of the anti-virus software has enough incentive to develop business where security in the program is a high priority. Hence, the anti-virus trust model supports the assumption that the update server is trustworthy because the vendor wants his business to survive. Trust in people is a problematic perspective. Usually one must trust people despite that. They can always talk, photograph screens, steal printouts or manually enter data into a different computer system. Hence, trust models deal with trustworthiness of the technical infrastructure. More on trust models can be found in Chapter 5. Please note that sometimes trust is confused with "reputation management", which focuses on the users of a web platform (see Section 5.1).

- "The only truly secure system is one that is powered off, cast in a block of concrete and sealed in a lead-lined room with armed guards.", says Gene Spafford [Spaf1989]. Indeed, security is relative. Researchers and Hollywood movies both consider "unbreakable codes" from different perspectives. In practice, however, if one has enough money and time, she will find ways to achieve most security measures. However, this greatly depends upon her own security goals whether she needs an absolutely unbreakable security system or not. In practical security, the question often is "How long shall my data be kept secret?", which is more of an insurance perspective. Security measures are there to reduce damage of an IT system – or to make misuse by accident or intent so difficult and expensive that the gain from misuse will be much smaller than the effort spent to access data. Have you ever asked yourself why there are no military-style barbed-wire twin fences around strawberry fields? Because the work of creeping on your knees, bending your back for hours to steal considerable amounts of strawberries causes most people to prefer buying strawberries somebody else picked.

  So the important insight from this is: know the value of your information, and the risks from unauthorized access to it. Then, come to a decision on how hard it should be for someone to break your security barriers.

Chapter 3

Security Concepts

This chapter presents an overview of generic security concepts that have evolved and gained relevance over time. It presents security concepts, their basic assumptions and application scenarios, and relevant implementations in operating systems. Furthermore, the changing IT landscape – away from centralized, bunkered servers to distributed computer systems and floating information – has a thorough impact on security concepts. This will be discussed as part of this chapter.

3.1 Multi-Level vs. Multilateral Security Concepts

This section presents and compares multi-level security and multilateral security concepts. Multi-level security concepts originate from military and other hierarchical organizations where confidentiality or security levels are used for security decisions. Multilateral security concepts define security policies according to rule sets. They can express security rules between individuals or roles along the same "level". This section will introduce both concepts and will present the most important security models that represent the two concepts.

3.1.1 Multi-level security (MLS)

Multi-level security implements one of two goals: either confidentiality or integrity. Its name is derived from various levels of privileges needed by subjects to access data. MLS is a long-established and well-researched security concept that was developed for public and military administration and their confidentiality needs. Its basic assumption is that there are security levels from "public" to "top secret" that are assigned to documents or data sets. Access to documents is only allowed for users, programs or computer systems that have at least the same or higher security clearance than the object that is being accessed. Such classifications exist in many areas of government. An example is shown in Table 3.1.

TOP SECRET	STRENG GEHEIM
SECRET	GEHEIM
CONFIDENTIAL	VS-VERTRAULICH
RESTRICTED/FOR OFFICIAL USE ONLY	VS-NUR FÜR DEN DIENSTGEBRAUCH
(Unclassified)	(Öffentlich)

Table 3.1: Example of security levels: German Safety Review Act (Sicherheitsüberprüfungsgesetz, SÜG), §4

MLS aims at integrating security levels into computer systems to achieve one of two goals:

Confidentiality: no information of a higher security level can be read from a lower level, and no information can be written from a higher to a lower security level.

Integrity: no information of a lower security level can be read from a higher level, and no information can be written from a lower to a higher security level.

There are several security models that specify MLS. The most important examples are Bell-LaPadula [BeLa1973], Biba [Biba1977] and Lomac [Fras2000]. They are explained in Chapter 4, Security Models.

3.1.2 Multilateral security

Multilateral security, in contrast to multi-level security, is not concerned about maintaining order according to security levels. Multilateral security is concerned with the implementation of security between various actors (users, computer systems, processes) that might very well be on the same MLS clearance level. Multilateral security models the relationships of computer systems to each other with respect to security issues. The goals of multilateral security can be complex and very different from each other. Thus, multilateral security is sometimes referred to as "policy-based security".

Important models based on multilateral security are Clark-Wilson [ClWi1987], Compartment/Lattice [Sand1993], Chinese Wall [BrNa1989] and BMA [Ande1996]. These models are explained and discussed in Chapter 4, Security Models.

3.2 Mandatory vs. Discretionary Access Control Concepts

This section introduces basic access control concepts. It focuses on the traditional distinction between rule-based access control (also known as "mandatory access control") and person-based access control ("discretionary access control"). Historically, the two concepts implemented distinct security approaches. Today, most access control systems consider rules as well as the personal identity of the subject. The two concepts are presented independently from each other.

Mandatory access control (MAC) is defined by the Trusted Computer System Evaluation Criteria [TCSEC1985] as "a means of restricting access to objects based on the sensitivity (as represented by a label) of the information contained in the objects and the formal authorization (i.e., clearance) of subjects to access information of such sensitivity".

Mandatory access control therefore defines access policies to objects according to the security state of an object. Access will be granted if the access permissions given by the clearance of the subject match the access restrictions mandated by the object’s label.

Discretionary access control (DAC) is defined by the Trusted Computer System Evaluation Criteria [TCSEC1985] as "a means of restricting access to objects based on the identity of subjects and/or groups to which they belong. The controls are discretionary in the sense that a subject with certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject (unless restrained by mandatory access control)".

Discretionary access control is commonly defined in opposition to mandatory access control (which is sometimes termed non-discretionary access control).

3.3 Security Concepts in Current Operating Systems

This section provides a discussion of general secure operating system architectures (e.g., FLASK or sHype). It will provide a table indicating the usage of security concepts in current operating systems, including SE Linux, Trusted Solaris, etc. It will discuss SELinux versus Linux. It will also

mention other operating systems. The resulting list serves as a quick reference to security technologies in practice.

3.3.1 Multi-level security

- Freely available implementations of operating systems with limited MLS applicability include Security-Enhanced Linux and TrustedBSD.

- Sun Microsystems offers Trusted Solaris, a commercial version of the Solaris Operating System that has data labeling functions but is not MLS approved. Early versions were evaluated at the TCSEC B1 level (the lowest allowed for MLS) but more recent versions were evaluated under the Common Criteria (version 2.1) at EAL4 (augmented), such as the Controlled Access (CAPP) and Labeled Security (LSPP) protection profiles.

- BAE Systems offers XTS-400, a commercial operating system that supports MLS at what the vendor claims is "high assurance". Early versions were MLS capable as evidenced by their evaluation at the TCSEC B3 level, but more recent versions were evaluated under the Common Criteria (version 2.1) at EAL5 (augmented). The protection profile used (CAPP and LSPP, both EAL3 protection profile that are not MLS-capable as discussed above) do not warrant MLS use of this product.

- IBM z/OS Version 1 Release 5 and later versions provide multi-level security support in z/OS. Designed together with DB2 Version 8, z/OS provides a solution for multi-level security on System Z mainframes. This support provides row-level security labeling in DB2. The German Federal Office for Information Security (BSI) awarded IBM EAL4+ certification (according to the CC version 2.1) for z/OS 1.7 with the RACF optional feature on March 2, 2006. The certification encompasses CAPP and LSPP both at the EAL 4+ level.

3.3.2 Mandatory Access Control

- SELinux supplements mandatory access control architecture to the Linux kernel, which was merged into the mainline kernel in August 2003. Red Hat Enterprise Linux (RHEL) version 4 (and later versions) comes with a SELinux-enabled kernel. Although SELinux is capable of restricting all processes in the computer system, the supported policy in RHEL only targets the most vulnerable programs (thus the name, the Targeted Policy). SELinux utilizes a Linux 2.6 kernel feature called LSM (Linux Security Modules interface).
- SUSE Linux has added a MAC implementation called AppArmor, which is based on LSM.

- Sun’s Trusted Solaris uses a mandatory and operating system-enforced access control mechanism (MAC), where clearances and labels are used to enforce a security policy.

3.3.3 Discretionary Access Control

DAC has been implemented in many operating systems, namely the UNIX and Windows operating systems families.

Chapter 4

Security Models

This chapter introduces security models. The first section deals with basic approaches to data security. The following sections summarize security models from scientific literature and practice. At the end of this chapter, the implementation of these models in relevant operating systems will be summarized.

4.1 Access Control vs. Information Flow Control vs. Type Enforcement

This section presents and discusses basic access control approaches and their differences. The three approaches of access control (guarding access rights to data), information flow control (guarding where data can move) and type enforcement (where objects belong to types that may only be manipulated by subjects from certain groups with rights to this "type" of data) will be summarized and discussed. The section presents established models for all three approaches and discusses their properties.

4.1.1 Access Control

In computer security, access control is a technique to manage access rights to data or computer system functions. Usually it implies the management of a subject's right to use a computer resource or to access data. More technically, access control can also be the right of processes to access data, start programs or to use remote services of other computer systems, web pages or data bases on a computer network.

Access control is one of the oldest concerns of computer security. It usually assumes that the computer is physically protected against unauthorized persons' direct access. Thereon, a nearly unlimited number of technologies for access control have been invented and are deployed on all levels of detail in computer systems. For a better understanding, a mental model of protection spheres that views an IT system from the outside to the inside is suggested to have. Figure 4.1 shows an example of this model.

On the outside sphere, access control methods such as passwords, Single Sign-On, smartcard access, firewalls, secure screens and further measures on the interfaces to the "outside" of the computer system are deployed. Deeper toward the inner spheres, access control measures get more detailed and more oriented toward the computer system and hardware resources.

Please note that some of the protected resources might be managed at several spheres in different ways, e.g., memory is managed on a per-user or per-program basis from the operating system level and further outside. But on the hardware level, memory management is the enforcement of appropriate electronic access to data flows between chips.

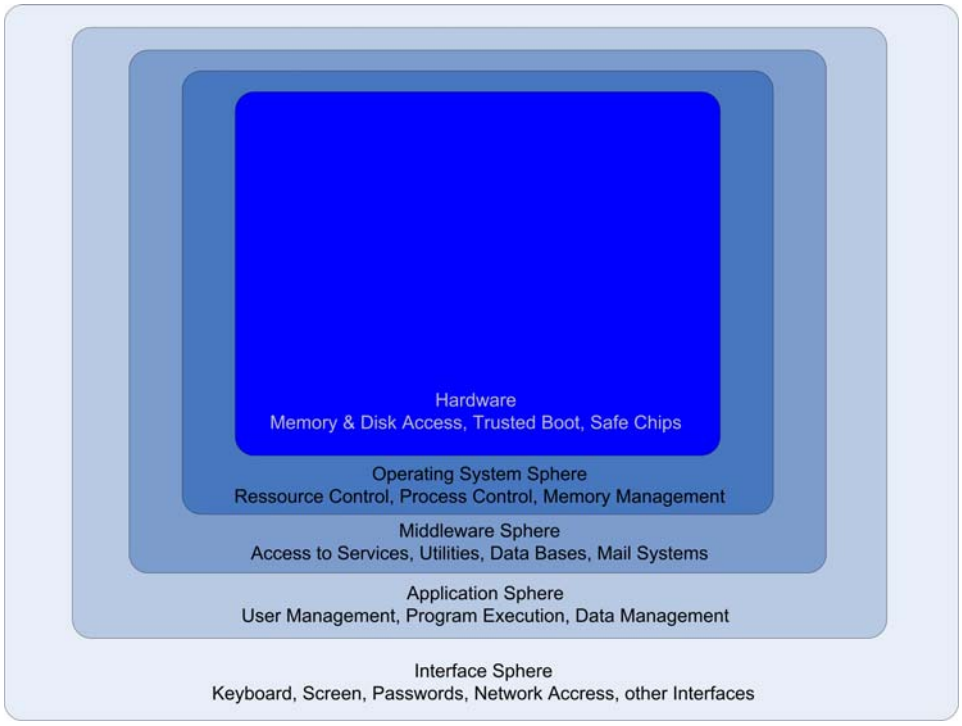

Figure 4.1: Access control spheres on computer systems.

Additionally, some of the functions can move to other layers as soon as the computer system is part of a distributed computer installation in a network.

However, most access control activities deal with the measures to grant access for a subject. Access control measures are composed of three features:

1. **Authentication:** A measure to identify the subject. It can range from simple secrets (e.g., passwords or knowledge about an IP-address) up to scans of biometric features.

2. **Authorization:** A step in which the access to data or resources is granted. Here, some software switch, temporary token, cryptographic key or other credential is issued to authorize access.

3. **Audit:** The generation of log files or other traces for various audit purposes (responsibility checking, security audits, intrusion detection).

Some security models presented in this chapter rely mainly on access control techniques to manage rights, e.g., access control lists or matrices. They will be explained below.

4.1.2 Information Flow Control

Contrarily to access control, some security models focus on the flow of information in and out of an information system or a particular state of it. This is a major difference from access control's focus on the elaboration of access policies. Here, the policies are formulated to precisely define the security-relevant actions that affect input or output of data. Policies regulate the possible moving of data between information systems or different security areas. The models presented apply different approaches for information flow control.

Information flow control often assumes that there are different "zones" that have associated security levels. Often, control is sought for the movement of data from one zone to another zone. A network firewall is an example of an information flow control technology that enforces security rules upon network data moving to and from the "public" zone to the "internal" zone. There are many more examples, e.g., mail systems that read confidentiality labels on data before passing on e-mails.

However, in information flow control systems there is a problem called "covert channels". Adversaries might find ways to send data out of a zone using a method that is allowed by the information flow control system – e.g., reading of external web pages. A simple covert channel can be to synchronize access to the external web page with radio clock signals. The pattern of access – no access vs. access – can create a Morse code that sends out information.

Covert channels are hard to control. Today, one assumes that a valuable – and dangerous – covert channel has to be fast to crack something of value (although passwords are rather short pieces of data). In practice, protection against covert channels seeks to limit the possible bandwidth of covert channels.

The Rushby Model

One example of an early information flow model is the policy model of John Rushby [Rush1992] which supports security policies concerning:

1. Noninterference: a formally proven state ensuring that from one zone, no observation of activities in the other zone is possible. If this is true, even covert channels would be avoided. Noninterference is represented as a relation between groups of users and commands. Users in group G do not interfere with those in group G' if the state seen by G' is not affected by the commands executed by members of G.

2. Transitivity: refers to the transitive flow of information from object A to object B and then to object C based on a relationship between the three objects. If information can flow from object A to B and from object B to C, it necessarily can from objects A to C.

3. Channel-Control: refers to the control of a set of commands (seen as a channel) resulting in information flow (or communication) between non-interfering groups.

The Rushby Model's view on a computer system refers to the state machine model. All access to information is modeled as the movement of data from one domain to another. The computer system is secure when all data moves according to the security policies and when the noninterference state is valid. The Rushby model is important in every operating system or program environment that is based upon virtual machines.

Digital Rights Management

Information flow control gained wide attention with the spread of broadband internet connections. Particularly the media industry developed a strong interest in the establishment of control over the usage of digital media, resulting in the **Digital Rights Management (DRM)** model. Here, a technological base that includes media production, media distribution (e.g., on web servers) and media players (DVD players, personal computers, mobile phones and MP3 players) is meant to enforce the vendor's policy and the consumer's license terms when consuming digital media. In particular, DRM is meant to enforce copy protection and pay-per-view policies.

DRM requires a number of sophisticated cryptographic technologies. It usually builds on top of tamper-resistant hardware. It uses protocols that incorporate watermarking and fingerprinting technologies, identity management and encryption with key management.

DRM spawned a wide public debate about many facets of fair use, information control, and copyright. It created fears of total control of media producers over personal computers and media consumption.

In practice, however, many DRM schemes have been broken. CD copy protection hardly works for a long period of time; DVDs are copied to personal computers and distributed digitally; and the DRM-protected files distributed by Apple's iTunes server (the commercially most successful on-line music store) can now be cracked by free software available on the internet. The deployment of secure DRM platforms on all levels imposes great cost on the value chain of media sales – as remarked in an analysis by Lewis [Lewi2003].

Sticky Policies

A contemporary approach to information flow control is the model of Sticky Policies [CPB2003]. The basic idea is that a security policy is firmly attached to the data object it is intended for. Processed on computers that have secure hardware and secure operating and application software, nothing can be done to the data object that is not allowed by the attached policy. To ensure an intact operating system and the correct application software, the sticky policy paradigm relies on Trusted Computing [TCG0001] as a secure hardware platform. To enforce security, sticky policies must be processed on every step of data handling, as shown in Figure 4.2.

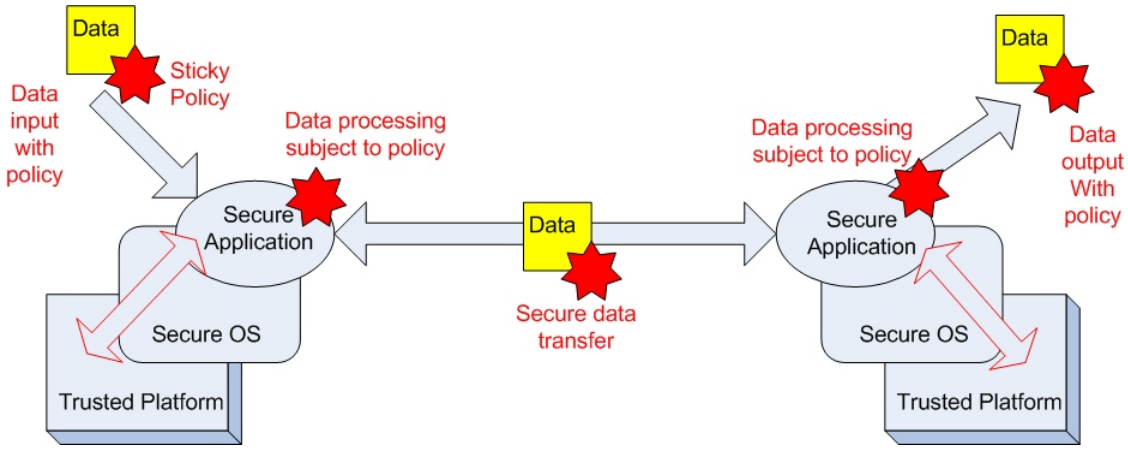

Figure 4.2: How sticky policies protect data.

Sticky policies face some challenges. Making the policies stick to data so they cannot be removed requires advanced cryptographic technology. The whole infrastructure must be based on Trusted Computing. If a data owner ever wants to update policies, a certain effort has to be spent to update the objects. The underlying cryptographic processes face severe key management challenges.

4.1.3 Type enforcement

Type enforcement is a security method that groups computer system resources and data objects into Meta groups called types. At the same time, subjects are assigned into domains. The security policy then defines which domains can perform particular actions on certain types.

Type enforcement organizes a computer system similarly to a company. Financial files are assigned to a type "finance", bookkeeping and controlling staff is assigned to the "financial department" domain. The policy states that only the domain "financial department" will have access to the "finance" type.

History

Type Enforcement was introduced in 1985 [BoKa1985] as a method of implementing integrity protecting computer systems without relying on a trusted user. It labeled objects as well as subjects, and specified access from subjects to objects and subjects to subjects in two matrices. Subject labels were called domains, and object labels were called types. Subjects to object access can be read, write, and execute. Type Enforcement (TE) was implemented first in the Secure Ada project (LOCK), and later by TIS in Trusted XENIX. Domain and Type Enforcement (DTE) was first presented in 1991 [BrRo1991] and is an extension of TE that was adapted by many researchers [BSSW1995]. It differs mainly in specifying policies in an intuitive policy language rather than using two matrices. Domain-to-domain transitions are allowed by the execution of binaries designated as entry points to the target domain.

Researchers in the Information Assurance Research Group of NSA worked with Secure Computing Corporation (SCC) to develop a strong, flexible and mandatory access control architecture based on Type Enforcement, a mechanism first developed for the LOCK system. NSA and SCC developed two Mach-based prototypes of the architecture: DTMach and DTOS [Smal2000]. NSA and SCC then worked with the University of Utah’s Flux research group to transfer the architecture to the Fluke research operating system. During this transfer, the architecture was enhanced to provide better support for dynamic security policies. This enhanced architecture was named FLASK [Smal2000]. NSA has now integrated the FLASK architecture into the Linux operating system as SELinux to transfer the technology to a larger developer and user community.

Discussion

Domain and Type Enforcement (DTE) is based on an enhanced version of Type Enforcement (TE). The main additions to the original model are a high level policy specification language and a human readable format of attribute values in the runtime policy database. Type Enforcement is a table-based access control model. Active entities, the subjects, have an attribute Domain, while passive entities, the objects, have a Type attribute. Possible accesses by subjects to objects are grouped into the access modes read, write, execute and traverse.

A global Domain Definition Table (DDT) contains the allowed interactions, where domains and types form rows and columns, and each cell holds a set of access modes.

Subject-to-subject access control is based on a global Domain Interaction Table (DIT) with subjects as both descriptors and, again, a set of access modes, e.g., signal, create or destroy, in the cells.

In contrast to the original TE model, DTE supports implicit attribute maintenance. This means that attribute values (whether for Domains or Types) assigned e.g., to a certain directory in a directory structure would also apply to all sub-directories as well (if those do not have explicit attribute

values). Moreover, the policy language allows assigning the type to a certain object according to its path prefix.

The first process on an operating system (e.g., the init process on a UNIX system) gets a predefined initial domain assigned. Each process can enter another domain by executing a program bound to it, a so-called entry point. An entry point may be executed to explicitly enter one of its associated domains, if the subject's current domain has "execute" right on the target domain. The auto access right to a domain automatically selects this domain, if one of its entry points gets executed.

The user-domain relationship is entirely built on entry points like command shells, etc. However, a DTE aware login program can select from all domains associated with an entry point to avoid individual copies for each domain.

Relevance

Type enforcement's main contribution to information security is its utility in reducing the risk of high-privileged processes, e.g., on UNIX systems when they are captured by an attacker. Type enforcement can be used to force such an attacking process to behave according to the data objects type policy – although the process can have more power over the data object.

Type enforcement is implemented in FLASK[3], Microsoft Active Directory and SELinux.

Type Enforcement is a registered trademark of Secure Computing Corporation who evaluated a firewall product with type enforcement to Common Criteria level EAL4+ with Basic and Medium PP Compliance: the Sidewinder G2 Security Appliance Models.

4.2 Basic security models

The models presented in this section are well-known and easily understandable. They have been implemented in some form on nearly all operating systems from mainframe computers to mobile phones. They are presented for the purpose of contrasting the more sophisticated security models.

4.2.1 Owner-Based Policies

Owner-based policies are a security scheme where the owner of a data object can define a policy that grants or denies access rights to other subjects.

History

Most multi-user operating systems deployed some form of owner-based policies to enable group work.

Discussion

Owner-based policies can be expressed in security models like HRU (see Section 4.3.3).

[3]FLASK is an operating system security architecture that provides flexible support for security policies. It is an acronym for Flux Advanced Security Kernel [SSLHAL1999]. Type enforcement is a core framework in security-focused operating systems such as NSA's Security-Enhanced Linux (SELinux) and TrustedBSD.

Relevance

Owner-based policies are found in many contemporary computer systems, collaborative work environments or file access in local ad-hoc networks. However, as the policies can be expressed within other models, they can be embedded into one of the larger security models.

4.2.2 Access Control Lists

Access control lists (ACL) are one of the oldest forms of security models. Here, for each object, a list defines the actions on this object allowed by the respective subjects.

Discussion

Access control lists grant access rights. They express a specific policy for the objects, subjects and actions listed. An access control list attached to file “salaryfile” might look like this:

“salaryfile”	John	„read“
“salaryfile”	Jane	„read“, „write“
“salaryfile”	Ben	„read“, „write“, „move“, „delete“

Evidently, Ben is the boss with permission to move or delete the salary file. Jane is allowed to write to it, so she is a secretary or salary accountant. John as a bookkeeper is just allowed to read it.

This form of access control is very maintenance-intensive upon changes in data, staff or access rules. For instance, if Ben in the above example deletes the salaryfile data, then John, Jane and Ben will all have ACL rules that refer to a non-existing file. ACL by itself does not support the maintenance of meaningful ACL structures.

Relevance

ACLs are ubiquitous. One can make a safe assumption that they will never disappear from computer systems. The reason for this is their simple and intuitive concept that can be easily defined and implemented, e.g., on prototype computer systems or software products that are delivered with low security relevance.

Examples of access control lists in contemporary computer systems are:

- Web server access control: the .htaccess and .htpasswords files that are used in many web servers for access control decisions are ACLs or ACLs enhanced by the possibility of group definition. But with web servers, the limits of ACLs are very visible too. A large on-line publisher will manage its thousands of subscribers’ access control permissions with other mechanisms than .htaccess files.
- Firewall and routing equipment: Here, simple lists of allowed data traffic source and destination addresses (subjects) and protocols (objects) are entered to configure basic network security.

But wherever ACL gets deployed, usually the corresponding management tools for the lists turn up to be required soon afterward. Hence it can be said that ACLs are practically functional mostly through the policy management tools that can create and maintain complex lists.

4.3 Confidentiality-Oriented models

This section provides an overview of security models focusing on confidentiality of data. Such models enforce strict confidentiality policies concerning subjects' access to data objects on a computer system. Strict confidentiality here implies that no data is revealed to unauthorized subjects at any time. The models may introduce detailed rules on how to express access policies for different subjects or groups of subjects.

Several confidentiality-oriented approaches have been developed. Their perspective is explained, and their major properties will be summarized. First, the classic models of static access control techniques are explained and then more dynamic approaches and more complex policies are mentioned.

4.3.1 Access Matrix Model / Access Control Lists

The access matrix model and its instance, the access control list (ACL) is one of the oldest security models. It assumes a separation of a computer system into data objects, subjects and operations. To determine the permission of a subject to execute a particular operation on an object, the computer system looks up the permissions in an access matrix or access control list.

There, either the listing of all objects and all subjects as a matrix with all possible operations is represented, or a list of all objects and subjects with their specific policies. ACL really is a memory-preserving representation of a matrix that only contains "positive" policy entries for allowed operations while neglecting the unnecessary entries that are not allowed anyway.

History

Access matrix security was introduced in 1971 [Lamp1971]. Being an intuitive scheme, it was used widely in operating systems such as UNIX and in database access control. It partly persists until today in firewall configuration tables or spam filters.

Discussion

Although being an intuitive model, the access matrix has an inherent problem. The matrix represents the state of the security model at one particular time only. Every time new objects are introduced to the computer system, a user changes, or some policy changes, the whole matrix must be created anew. This requires considerable effort and leads to the potential of introducing errors.

Relevance

Parts of the access matrix model still persist in UNIX systems. Some filtering equipments, such as firewalls and SPAM filters, allow the configuration of "blacklists" and "whitelists" which are ACLs.

With the spread of instant messaging, IP telephony and web-based communities, ACLs return as "buddy lists" or "contact lists" that determine the degree of communication and information the participants to such computer systems can share with other participants.

In mobile telephony, existing implementations of location-based services of the mobile phone network impose the management of an access control matrix on the phone users [Snek2001]. Here, the user is expected to update and maintain his matrix about which services within the mobile phone network are allowed to access his location. The vendors claim this is a secure privacy model, but the flaw of vast management overhead stated above remains for the users.

4.3.2 Lattice-Based Access Control Model

In computer security, lattice-based access control (LBAC) is a complex method for limiting information access based on any combination of objects (such as resources, computers, and application programs) and subjects (such as individuals, groups or organizations) [Sand1993].

In this type of control model, a lattice is used to define the levels of security that an object may have, and that a subject may have access to. That is, a partial order is defined on the security levels, in such a way that any two security levels always have a greatest lower bound (meet) and least upper bound (join). If two objects A and B are combined to form another object C, that object is assigned a security level formed by the "join" of the levels of A and B, and if two subjects need to jointly access some secure data, their access level is defined to be the "meet" of the subjects' levels. A subject is allowed to access an object only if the security level of the subject is greater than or equal to that of the object, in the partial order defining the lattice.

History

Lattice based access control models were first formally defined in 1976 in [Denn1976]. They were developed for the expression of complex rights management.

Discussion

LBAC is known as a more specific set of access control restrictions and is more general than role-based access control (RBAC). It allows the definition of complex policies. LBAC is one of the few models that explicitly take care of creating security states for joined data and thus introduces the idea of data life cycle management into IT security. A lattice model is a mathematical structure that defines greatest lower-bound and least upper-bound values for a pair of elements, such as a subject and an object.

However, the lattice model requires complex lattice structures that must be free of errors. Additionally, the lattice has to be newly generated upon changes on the computer system (leaving users, change of security levels).

Relevance

The true power of the lattice model is its capability to model complex flows of information and their respective security properties. In the research community, two application areas for the lattice model have been identified and published:

- Document access control: With the development of the World Wide Web technology, the issue of document access on web servers came to the security community [DrNe1998]. Strategies for managing access control in reading, writing, moving and joining documents were needed and modeled as a lattice policy.
- Computer supported cooperative work (CSCW) policy management: In a generalization of the above theme, CSCW systems are platforms for computer-based communications, document management and group collaboration (e.g., including calendars, data, files,

messages). The modeling of access policies has been done successfully with the lattice model.

4.3.3 Dynamic Access Control Model (HRU)

Dynamic access control enhances the static concepts of access control lists, matrices or lattice-based models by the capability of dynamically granting or removing rights. In static computer systems, for a temporary delegation of rights, a computer system administrator will modify the definitions within the lists or tables. Once the condition for the delegation ceases, the computer system administrator is once again needed to remove the definitions from the subject. On computer systems with many users, this is inefficient and can also lead to human errors that compromise security.

Dynamic access control thus introduces mechanisms to dynamically delegate security privileges to other subjects without the need of a computer system administrators' intervention or an update in the computer system security configuration. This model is called HRU according to its inventing team (Harrison, Rurro and Ulman) [HaRU1976].

History

Dynamic access control was introduced in 1976 by Harrison, Rurro and Ulman in [HaRU1976]. The core concept of the model is the use of mathematical reduction to check whether a new rule would compromise the computer system security.

Discussion

The HRU primitive commands are used to express access control matrix changes. The six HRU primitive commands are: ENTER rule, DELETE rule, CREATE subject, DESTROY subject, CREATE object, DESTROY object. Using HRU, subjects can enter rules with rights delegation into the access control matrix. Harrison, Rurro and Ulman modeled HRU as a state machine. They showed that under certain conditions, the machine can always decide whether the computer system stays secure. However, their main contribution was also to show that in certain conditions under which a delegation is performed, an algorithm cannot effectively decide on behalf of the access control matrix system whether the computer system is still secure or not. The reasons for this are the complexity of the resulting rules.

Relevance

HRU has been an important contribution to computer security. Its proven security functions and known limitations lead to HRU-like implementations in operating systems, e.g., in the Type Enforcement concept where temporary privileges are needed.

4.3.4 Simple Security Stages

Security stages reflect a simple, stage-based classification of data, e.g., according to a multi-level scheme as shown in Figure 3.1. The levels are combined with access rules that define how documents at certain levels are accessed, e.g., a certain military rank implies a certain access level.

History

Security levels originate from military document handling. The levels usually correspond to some hierarchy where the position of a subject in the hierarchy also determines the access level for documents.

Discussion

Numerous access policies can be defined in the security stages model. Reading (confidentiality) and writing (integrity) are usually treated separately. Problems arise when the level of a document changes due to its joining with information from other levels. Security stages do not manage the document life cycle.

Relevance

While most governments still classify documents along security levels, many problems have been found in the original security stages model. According to Anderson [Ande2001], documents flow "upward" in security levels over time as they are processed in higher security levels. Also, the senior employees seem to accumulate high-security-access permissions over the years and may turn into a high security risk when they have access to too many classified documents. A separation of duty between different work-groups cannot be modeled with the simple security stages either. This is sought to be solved with separated compartments (see Section 4.5.2). The Bell-LaPadula model (Section 4.3.5) aims at improvement of the security stages model.

4.3.5 Bell-LaPadula Model

The Bell-LaPadula model (BLP) [BeLa1973] is a formal state transition model of a computer security policy that describes a set of access control rules which use security labels for objects and clearances for subjects. The model assumes that all access to data on a computer system is performed with security access functions that are part of the secure computer system.

History

The Bell-LaPadula model was introduced in 1973. The authors formalized computer systems using system theory and provided a complete model that included subjects, objects, security levels, and actions.

Discussion

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 computer system satisfies the security objectives of the model. The Bell-LaPadula model is built on the concept of a state machine with a set of allowable states in a computer system. The transition from one state to another state is defined by transition functions.

This security model is directed toward access control and is characterized by the phrase: no read up, no write down. With Bell-LaPadula, users can create or modify content only at or above their own security level: no write-down. Subjects can view content only at or below their own security level.

Therefore, the Bell-LaPadula model defines a Simple Security Property and a * (star) property as follows:

1. The Simple Security Property states that a subject at a given level of security must not read an object at a higher security level (no read-up).
2. The * (star) Property states that a subject at a given level of security must not write to an object at a lower security level (no write-down).

Relevance

The Bell-LaPadula model has greatly influenced IT security modeling since it was presented. A customized implementation into the mainframe timesharing operating system MULTICS[4] starting 1965 brought the model into practice. Soon Bell-LaPadula was referenced by the TCSEC security evaluation criteria[5].

Contemporary operations systems such as the Windows NT/Vista family and Trusted Solaris with TCSEC evaluations implement versions of the model (see [Bell2005] for a list of operating systems and evaluations).

Contemporary research still builds upon the Bell-LaPadula model. In access control to location information in location-based mobile network services, a model for an access matrix combined with the BLP model is constructed in [Snek2001].

4.3.6 Role-Based Access Control (RBAC)

Role based access control (RBAC) has a unique approach. Its principal concept is the regulation of object access through well-defined roles. Thus, not a particular user identity is of relevance for the security decision, but its role. Security management based on the RBAC model is the management of subjects' capabilities to take up certain roles. Subjects are assigned into groups. Access rights are grouped by role name, and the use of resources is restricted to individuals authorized to assume the associated role. For example, within a hospital system the role of doctor can include operations to perform diagnosis, prescribe medication, and order laboratory tests; and the role of researcher can be limited to gathering anonymous clinical information for studies.

A role hierarchy defines roles that have unique attributes and that may contain other roles; that is, one role may implicitly include the operations that are associated with another role.

History

Role based access control (RBAC) has been published in 1992 [FeKu1992].

Discussion

[4]For the history of MULTICS, see http://www.multicians.org/history.html (as of 11-Nov-2007).

[5]The "Trusted Computer System Evaluation Criteria" (TCSEC) were developed in the United States of America and published in 1983 to evaluate security properties of computer systems. They alternatively are called "Orange book" due to the color of the booklet they were published in. http://csrc.nist.gov/publications/history/dod85.pdf. The TCSEC evolved into the initial version of the Common Criteria (ISO standard 15408)

RBAC allows a subject to incorporate several roles. A role can be incorporated by many subjects. A constraint places a restrictive rule on the potential inheritance of permissions from opposing roles, thus it can be used to achieve appropriate segregation of duties. For example, the same person should not be allowed to both create a login account for someone, and also be allowed to authorize the procedure.

In an organization with requirements that span dozens or hundreds of systems and application programs, using RBAC to manage roles and assign role memberships becomes extremely complex without hierarchal creation of roles and privilege assignments.

The model has evolved since its first publication. It gained relevance in modern operating systems and other application programs since. A current reference is the original inventors' book "Role Based Access Control" [FKC2003].

Relevance

The use of RBAC to manage user privileges within a single system or application program is widely accepted as a best practice. Systems including Microsoft Active Directory, SELinux, FreeBSD, Solaris, Oracle DBMS, PostgreSQL 8.1, SAP R/3 and many others effectively implement some form of RBAC.

Several organizations are experimenting with the inclusion of provisions for RBAC in open consensus specifications. RBAC is an integral part of the security models for Secure European System for Applications in a Multi-vendor Environment (SESAME) distributed system and the database language SQL3. In addition, the Object Management Group's (OMG) Common Object Request Broker Architecture (CORBA) Security specification uses RBAC as an example of an access control mechanism which can be used with the distributed Object Technology defined by the OMG.

Ongoing development of RBAC and its implementations are summarized and maintained on a dedicated web page at the United States National Institute of Standards and Technology [NIST0001].

4.3.7 Chinese Wall Model (Brewer-Nash)

The Chinese Wall security model [BrNa1989] is designed to account for commercial security interests in business data bases. The assumption is that for example, e.g., within the financial industry, there can be conflicts of interest, insider knowledge and business secrets when a company does business with several customers that compete with each other. The Chinese Wall model proposes to build a "Chinese Wall" between sensitive information from competing customers. It therefore assigns customer files to business segment classes, and assigns subjects to data sets. With a set of enforced rules, it is guaranteed that no subject can access data from a competing company data set once it has been assigned to one of the data sets.

History

Chinese Walls are separation measures that have been implemented in companies to separate persons who make investment decisions from persons who are knowledgeable of information that might influence the investment decisions.

In the United States, the expression was used after the stock market crisis of 1929, when the U.S. government enacted separation between investment bankers and brokers. Rather than prohibiting a

company from doing both businesses, the government permitted the implementation of Chinese wall procedures.

The term "Chinese Wall" may, according to Wikipedia[6], refer to the Great Wall of China, and its scale and effectiveness at separating one side from the other – or alternatively, the term may originate from a reference to Chinese standing screens which allow for the temporary installation of a wall in a room lacking the permanent architectural feature. This origin seems fitting for its usage in organizations such as law firms that create ad hoc barriers between offices or lawyers in order to protect against a conflict of interest.

In computer security, the Chinese Wall Model is a security model where read/write access to data is protected through classification of data in conflict-of-interest classes. This is the basic model used to provide both privacy and integrity for data.

The first publication of the Chinese Wall model was in 1989 [BrNa1989].

Discussion

The separation-of-duty property of the model was found to be incompatible with the Bell-LaPadula model in [BrNa1989]. However, the authors suggest an extension to the Clark-Wilson-model that implements the Chinese Wall security policy.

The model includes:

- Rules to prevent conflict of interest
- Basically a formal description of normal commercial non-disclosure practice
- The requirement that a professional worker should not deal with different clients who are competitors with each other so that he cannot be accused of using "inside knowledge"

Like the Bell-LaPadula model, Chinese Wall model has two access control rules that can be expressed formally. Informally, the rules can be stated as:

1. **The simple security property:**

   a subject S can access company C's data only if

   a. S has already accessed C's data or

   b. S has not accessed any of C's competitors' data

2. **The *-property:**

   S can write to C's data only if S cannot read any other company's sensitive data.

Relevance

Security researchers claim that the Chinese Wall model is rather a policy to be expressed in the implementation of a secure computer system than a model of its own. However, the function is vital to many business environments, e.g., the financial industry, consulting, and the health care market.

The implementation of the Chinese Wall policy is specific to the application field.

[6] See http://en.wikipedia.org/wiki/Chinese_wall.

4.4 Integrity-Oriented models

This section presents and discusses integrity-oriented security models. Besides confidentiality, protection of data against unauthorized manipulation, deletion or other forms of handling is required in many application scenarios. For this purpose, integrity-oriented security models have been developed. Most models deal with writing permissions according to a hierarchy in an organization, for example public administration or military. Below, the three integrity-oriented models "BiBa", "Low-Watermark" and "Clark-Wilson" are introduced and compared to each other. However, their basic assumption is that someone "higher up the hierarchy" is less dangerous for document integrity than a person with a lower position.

4.4.1 Biba Model

Biba's security model ensures the integrity of data according to a write policy [Biba1977]. It is a formal state transition system that describes a set of access control rules designed to ensure data integrity.

History

The Biba model was introduced in 1977 by Ken Biba. Its purpose has been the protection of data against unauthorized manipulation.

Discussion

The Biba model is generally compared to the Bell-LaPadula model. While BLP ensures confidentiality by only allowing the movement of data up or into the same security level, the Biba model ensures that no lower security classified subject can move data into higher security levels.

As with the Bell-LaPadula model, the Biba model defines a Simple Security Property and a * (star) property. In this case they are reversed to Bell-LaPadula:

1. The Simple Security Property 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 * (star) Security Property 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)

The Biba model thus is often called a "write-down" model.

Relevance

The Biba model was relevant to government and military fields of application. Today, in some particular use-cases such as firewall configuration tables or software in the health care domain, Biba policies can be relevant. When a computer system goes through a reset, it is not allowed to read configuration data from a lower-security file. When a user writes a configuration file, he is not allowed to write it into a higher-security configuration file.

4.4.2 Low-Watermark Access Control

Low-Watermark is a name for many integrity-oriented models that consider as a common rule of thumb that writing to data degrades the data to the lowest security level involved. The low-watermark-model is related to the Biba Model. However, in contrast to Biba, where “no-write up” and “no-read down” rules are enforced, in the low-watermark-model, read down is permitted, but the object will be temporarily degraded to the lower subject level. A differentiation between subject-low-watermark and object-low-watermark can point out which of the involved entities will be downgraded to the needed low integrity level.

History

The Biba model is considered a subject-centric low-watermark model. Following this, many extensions have been published. Recent implementations for the Linux operating system family include a form of low watermark.

Discussion

Low-watermark approaches can limit security problems to a lower security level or avoid these problems. The resulting concept is somehow similar to type enforcement, which also reduces security privileges to a reasonably low level. Low-watermark approaches, however, are usually oriented toward a multi-level approach and thus are simpler in terms of policy complexity.

Low-Watermark models have the same practical problems the Biba model has. Once data is classified into levels, the change of levels of subjects or objects and any adaption of security policies is not a simple task.

Relevance

Low-Watermark approaches are used in operating systems as a protection mechanism against malicious software. For example, LOMAC is a dynamically-loadable security module for Linux operating systems that uses Low Water-Mark Mandatory Access Control (MAC) to protect the integrity of processes and data from viruses, Trojan horses, malicious remote users, and compromised network server daemons.

According to SPARTA [Spar2000], LOMAC separates data and processes into two distinct levels: "Once loaded, LOMAC divides the system into two conceptual levels of integrity: high and low. The high-integrity side contains all processes and files that should be protected from malicious software and remote users: the operating system kernel servers (such as “kflushd”), the system binaries (bin, lib), the system configuration files (etc), and any mission-critical data (your web pages). The low-integrity side contains the processes that must interact with remote users or system (remote login sessions, web clients and servers, mail delivery agents) and the files they download from the net (web content, mail, attachments)." This is – compared to Biba – a simple policy.

For a further review on the availability of low watermark implementations, please refer to Safford and Zohar’s essay at [SaZo2004].

4.4.3 Clark-Wilson (Commercial Integrity)

The Clark-Wilson integrity model allows specifying and analyzing an integrity policy for an information processing system. The model was developed in the context of banking systems and has

been formalized in 1987 by David Clark and David Wilson in [ClWi1987]. The objective was to develop a model that formalizes the notion of information integrity and prevents the corruption of data items in a computer system due to either error or malicious intent. In the Clark-Wilson model, Integrity is defined by a set of constraints in which data is considered to be in a consistent or valid state when it satisfies these constraints. Well-formed transactions move the computer system from one consistent state to another and an integrity policy describes how the data items in the computer system should be kept valid from one state of the computer system to the next.

History

The origins of Clark-Wilson, which has been in practical use in accounting systems even before being formalized by Clark and Wilson in their article [ClWi1987], are described in [ClWi2007] as follows: "The paper develops the model as a way to formalize the notion of information integrity, especially as compared to the requirements for multi-level security (MLS) systems described in the Orange Book. Clark and Wilson argue that the existing integrity models such as Bell-LaPadula (read-down/write-up) and Biba (read-up, write-down) were better suited to enforcing data confidentiality rather than information integrity. The Bell-LaPadula and Biba models are more clearly useful in, for example, military classification systems to prevent the theft of information and the tainting of information at higher classification levels, respectively. In contrast, Clark-Wilson is more clearly applicable to business and industry processes in which the integrity of the information content is paramount at any level of classification (although the authors stress that all three models are obviously of use to both government and industry organizations)."

Discussion

This model is an integrity model based on well known, time tested accounting practices. Data is categorized into two levels:

1. Constrained data items (CDI)
2. Unconstrained data items (UDI)

CDIs can exclusively be accessed through special-purpose transformation procedures (TP). Integrity of CDIs is checked with integrity verification procedures (IVP). The model is not a formal model, but rather a collection of rules, such as:

- Subjects have to be authenticated
- TPs have to preserve integrity
- There must be a separation of duty policy

Clark-Wilson is not a multi-level security model, but can be combined e.g., with Biba. It drew attention to different kinds of security models, not just military-style MLS. Its problems are:

- Implementing IVPs and TPs for complex computer systems is difficult
- Internal integrity of a transaction does not guarantee that the transaction was correct (but there is auditing)

At the heart of the model is the notion of a relationship between an authenticated principal (i.e., user) and a set of programs (i.e., TPs) that operate on a set of data items (e.g., UDIs and CDIs). The model must also ensure that different entities are responsible for manipulating the relationships between principals, transactions, and data items. As a short example, a user capable of validating or creating a relation should not be able to execute the programs specified in that relation.

The model consists of two sets of rules: Certification Rules (C) and Enforcement Rules (E). The nine rules ensure the internal integrity of the data items (i.e. prevention from modification of data by

users) as well as their external integrity (i.e. consistency of the data ensuring that data in the computer system reflects the real-world data it represents). A TP is said to be "certified" if it has been checked to comply with the Certification Rules.

The model's Enforcement and Certification Rules define data items and processes that provide the basis for an integrity policy. 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 computer 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 requires that the "certifier" of a transaction and the implementer are different entities. The model contains several constructs that represent both data items and processes that operate on those data items. The central data type in the Clark-Wilson model is a Constrained Data Item (CDI). An Integrity Verification Procedure (IVP) ensures that all CDIs in the computer system are valid at a certain state. Transactions that enforce the integrity policy are represented by Transformation Procedures (TPs). A TP takes as input a CDI or Unconstrained Data Item (UDI) and produces a CDI. A TP must transition the computer system from one valid state to another valid state. UDIs represent computer system input (e.g., provided by a user or adversary). A TP must guarantee (via "certification") that it transforms all possible values of a UDI to a "safe" CDI.

Comparison of the Biba model with Clark-Wilson

Both models do have integrity levels, though the Clark-Wilson model only has two levels for objects (CDIs and UCIs) and two for subjects (certified TPs and uncertified procedures).

The Biba model has no concept of "certification". There is no mechanism for checking the trusted entities or their actions.

The Biba model cannot emulate the Clark-Wilson model but the Clark-Wilson model can emulate the Biba model (by choosing appropriate TPs in the allowed triples).

Relevance

The Clark-Wilson approach has been configured to work on UNIX and Windows NT systems. Several models for database integrity are based on the Clark-Wilson model.

4.5 Other Models

Some security models do not fit into the categories that have been introduced so far – or extend, modify or combine these in ways that make them defiant to categorization. This section will therefore describe important security models beyond the above discussed categories. Outstanding features of the models are highlighted.

4.5.1 BMA Model

The British Medical Association's security model.

History

The BMA model came into existence after a long debate between the British government, health professional associations and security experts. A long communication process was necessary to move from military perimeter security views to a multilateral security approach in medical information systems. BMA extensively analyzes roles, responsibilities and the flow of (medical) information along computer systems that are used by different subjects in the health care environment. The model was described in 1996 in [Ande1996].

Discussion

The BMA model follows nine principles, quoted from [Ande2001]:

1) Access control: each identifiable clinical record shall be marked with an access control list naming the people or groups of people who may read it and append data to it. The system shall prevent anyone not on the access control list from accessing the record in any way.

2) Record opening: a clinician may open a record with herself and the patient on the access control list. Where a patient has been referred, she may open a record with herself, the patient and the referring clinician(s) on the access control list.

3) Control: one of the clinicians on the access control list must be marked as being responsible. Only she may alter the access control list, and she may only add other health care professionals to it.

4) Consent and notification: the responsible clinician must notify the patient of the names on his record's access control list when it is opened, of all subsequent additions, and whenever responsibility is transferred. His consent must also be obtained, except in emergency or in the case of statutory exemptions.

5) Persistence: no-one shall have the ability to delete clinical information until the appropriate time period has expired.

6) Attribution: all accesses to clinical records shall be marked on the record with the subject's name, as well as the date and time. An audit trail must also be kept of all deletions.

7) Information flow: information derived from record A may be appended to record B if and only if B's access control list is contained in A's.

8) Aggregation control: there shall be effective measures to prevent the aggregation of personal health information. In particular, patients must receive special notification if any person whom it is proposed to add to their access control list already has access to personal health information on a large number of people.

9) Trusted Computing Base: computer systems that handle personal health information shall have a subsystem that enforces the above principles in an effective way. Its effectiveness shall be subject to evaluation by independent experts.

Relevance

The BMA model was one of the first extensively specified models that changed perspective away from a military-oriented protection model. It provided an application-centric multilateral security model that considered work flows, a specific threat model, and specific protection mechanisms for the application area.

The BMA policy is implemented in practical and research secure computer systems. According to Anderson [Ande2001], it is being used in clinical systems. The formulation and elaboration of its security policies, according to the same source, is under development.

4.5.2 Compartments / Multi-Category Security (MCS)

The concept of compartments is based on a separation-of-duties philosophy. Here, objects are categorized into compartments of equal accessibility. A subject is assigned to a compartment, and is only allowed to access objects within the same compartment.

History

According to Anderson [Ande2001], compartments have probably been in use with secret services for centuries. The strict separation of knowledge into the responsible departments protected agents and other secrets.

Discussion

Compartments create very strict borders between data object groups. The groups have to be defined. Problems arise when either the data separation cannot be kept exact due to an overlap in data, e.g., in several projects, or due to the assignment of a subject to several compartments.

Relevance

Sole compartments are hardly found today. The Chinese Wall policy can be viewed as an algorithm that dynamically creates new compartments and defines a subject's membership to them. However, some researchers enhance the lattice security model with compartment labels and use the term "compartment security".

4.5.3 sHype-Model

sHype [sHype] is a hypervisor security architecture developed by IBM Research. A hypervisor is a virtualization platform that allows multiple operating systems to run on a host computer at the same time [Wiki0004]. Hypervisors have a long tradition in operating systems development. According to IBM[7], sHype's main goal is to provide a secure foundation for server platforms, providing functions such as (citation from [sHype]):

- Strong isolation, mediated sharing and communication between Virtual Machines[8]:

  These properties are all strictly controlled by a flexible access control enforcement engine. This engine can enforce mandatory policies such as Multi-level Security, Role-based Access Control, and Type Enforcement.

- Attestation and integrity guarantees for the hypervisor and its virtual machines:

  We are extending the Trusted Computing Group's specification to include hypervisor-based server platforms. Our goal here is secure boot or authenticated boot guarantees for the hypervisor platform, Virtual Machines, and optionally the guest operating systems and software running on Virtual Machines. To support a large number of Virtual Machines, we have developed a virtual TPM architecture which we have applied to the XEN open source hypervisor.

[7] See http://www.research.ibm.com/secure_systems_department/projects/hypervisor/.

[8] One of the design principles here is that security is reached through the isolation of applications within their own, secure virtual machines.

- Resource control and accurate accounting guarantees:

  All resources are strictly accounted for and may be constrained. Simple resources include memory and CPU cycles. More elaborate resource management is needed to control network bandwidth, e.g., to limit the network bandwidth to a Virtual Machine.

- Secure Services

  sHype provides the base infrastructure for disaggregation of operating system services, such as security policy management or distributed auditing, into smaller and more manageable protected execution environments, thereby enabling their system-wide utilization and potentially enhancing the assurance of these services.

In the area of FLOSS-Software, IBM has developed a small security extension to XEN, a FLOSS hypervisor. It allows administrators to define simple policies (currently: Chinese Wall and Type Enforcement) that govern the control and sharing capabilities of Virtual Machines that run simultaneously on a single XEN system.

History

The first hypervisor was IBM's CP-40, a research operating system that began production use in January 1967, and which became the first version of IBM's CP/CMS operating system. Throughout the mainframe computing history, implementations and enhancements of this hypervisor lead to the core of IBM's high-availability mainframe server machines, e.g., S/360 and S/370.

Discussion

sHype results from decades of experience at IBM on reliable server systems, operating systems and security issues. The full virtualization of operating systems supports many security functions, ranging from strict separation of application and data areas up to a higher robustness of computer systems against failure due to a single failing program.

sHype provides security functions and interfaces that enable the implementation of security models based on the reliability of hypervisor systems. The resulting computer systems are generally considered to have a high degree of security.

However, sHype might still need strong protection against very skilled attackers, and additional hardware support to protect the operating systems.

A hypervisor system also increases the computer system administration effort, as several instances of one or more operating systems and their respective security environments have to be administered.

Relevance

In general, hypervisor technology is well established on the market. The major UNIX vendors, including Sun Microsystems, HP, IBM, and SGI, have been selling virtualized hardware since before 2000. These have generally been large computer systems.

There are two types of virtualization: full-virtualization and paravirtualization. In full virtualization, a guest operating system is fully unaware that it is running in a virtualization environment. In paravirtualization, the guest operating systems is aware of the virtualization layer, and works together with the underlying hypervisor to achieve higher performance. However, in paravirtualization, the guest operating system needs to be modified, as opposed to full-virtualization.

Therefore, for instance, multiple host operating systems have been modified to run as guest operating systems on Sun's Logical Domains Hypervisor. As of late 2006, Solaris, Linux (Ubuntu and Gentoo), and FreeBSD have been ported to run on top of Sun's Hypervisor (and can all run simultaneously on the same processor, as fully-virtualized independent guest operating systems). Wind River "Carrier Grade Linux" also plans to run on Sun's Hypervisor. Full virtualization on SPARC processors was not difficult because the SPARC architecture, since its inception in the mid-1980s, was deliberately kept clean of artifacts that would have impeded virtualization. (Compare with virtualization on x86 processors, described in the next paragraph.)

Similar trends have been seen with Linux x86/x64 server platforms, where virtualization efforts have been led by FLOSS-Software projects such as XEN, VirtualBox and KVM (Kernel Virtual Machine). Since these technologies span from large computer systems down to desktops, they are described in the next section.

On desktop and portable computers, virtual machine monitors like VMware follow a Hypervisor-approach. With the security functionality of sHype added, some of these concepts can provide strong security.

4.6 Security Models in Current Operating Systems

This section presents an annotated table with relevant implementations of security models. The implementations can either be independent products or part of an operating system. While some historically important operating systems may be mentioned, the table will focus on current, publicly available operating systems that are either commercial or FLOSS software.

Generally, implementation of security models can be distinguished in certified and non-certified operating systems. Certified operating systems were successfully examined to be conformant with a particular catalog of IT security criteria (e.g., TCSEC [TCSEC1985], ITSEC [ITSEC1991] or Common Criteria [CC1996]).

Vendor / Implementation	Related Model	Operating System	Accreditation
NSA, Red Hat / SELinux	Bell-LaPadula	GNU/Linux	None
FreeBSD Foundation / TrustedBSD	BiBa, LoMAC	TrustedBSD	None
N.A. (Amon Ott) / Rule Set Based Access Control (RSBAC)	Bell-LaPadula	GNU/Linux	None
Sun Microsystems/ Trusted Solaris	Bell-LaPadula	Trusted Solaris	EAL4+
Microsoft / Windows Vista	BiBa	Windows Vista	None
Unisys / OS 2200 **Argus Systems/ Pitbull LX**	BiBa, Bell-LaPadula, Lattice, Clark-Wilson Lattice	OS 2200 AIX, Sun Solaris, Linux	TCSEC B1 ITSEC E3, TCSEC F-B1

Table 4.1: Implementation of security models according to [Wiki0001]

Chapter 5

Trust Models

This chapter introduces the purpose of a trust model, its features, requirements and, use-cases. A basic notion is the model of centralized trust versus distributed trust. This will be elaborated in the section.

5.1 Taxonomy of Trust Models

This section introduces trust models. Trust is a central, yet ambiguous term in IT security[9]. A trust model is an important part in any IT security concept. It defines the trusted part of a computer system, and the rules according to which trust can be propagated to other parts of the computer system. This section will deal with questions like: What is trust in IT security? Who or what is to be trusted? What are the categories of trust models? What are examples of models that fit in each category? Trust transitivity will be introduced as well as ranking of trust levels, and some examples will illustrate the implication of different trust models.

5.1.1 Trust versus Reputation

Trust in IT security *always* reads as "trust in the correct function of a technical component which is important for the system security". In this definition, the 1990ies brought some scientific work on trust management in IT systems and distributed computer systems. Focus of the work was mostly the authentication of computer systems or software as a trust-enabling measure.

With the E-Commerce boom in the late 1990ies and into the 21st century, many findings about trust in E-Commerce were published. Here, the interpretation of trust was either the confidence of e-commerce customers in a web shop system, or the confidence of a web shop company in the relationship with its customers. Here, trust issues moved away from the technology focus, into risk management, financial security and fraud management.

Furthermore, the emerge of peer-to-peer computer and network services and "Web 2.0" community platforms redefined trust as a peer-controlled, community-based confidence in the honest or honorable behavior of people in the community. This is what scientists call "reputation management" rather than trust management.

For our study, we consider trust as "trust in the correct function of a technical component which is important for the system security".

5.1.2 Trust Models

The terminology used in trust models is similar to that of security models. There are objects and subjects. Subjects can be individuals or organizations. Trust is "propagated" through a computer system. Propagation is done by means of technology, for example authentication technology such as

[9] See definition of "trust model" in Section 2.1

cryptographic authentication protocols. This introduces cryptographic keys into trust models. These keys are sometimes used as part of certificates.

Trust metrics are methods that deliver a measure of trust in certain computer system properties. Numerous methods and approaches exist, ranging from simple scoring up to probabilistic calculations. Trust metrics are mentioned for completeness, but will not be presented in depth in this text.

Trust models

Trust models are models that describe how security is used to counter threats to a computer system. According to the ITU-T X.509 standard, Section 3.3.54, trust is defined as follows: "Generally an entity can be said to trust a second entity when the first entity makes the assumption that the second entity will behave exactly as the first entity expects."

Trust models thus support the decision about the behavior of another computer system component. Every security model will have some trust assumptions in the functioning of the security features built into it. For example, usually the correct functioning of cryptographic algorithms according to their correct use is assumed. The cryptography is trusted to function. Another example is the CPU of a computer. Normally the assumption is that the CPU will not maliciously modify or remove source code from program execution. It is trusted to function correctly.

The basic approach of a trust model is the establishment of trust in important computer system functions. To come to a basic idea about how trust is established in a computer system, its designers ask questions like:

- What are the specific mechanisms that are necessary to respond to a specific threat profile?
- How does implicit or explicit validation of an entity's identity – or the characteristics necessary for a particular event or transaction – occur?
- Where is the trust anchored? How strong is the anchor?

Trust modeling is the process performed by the security architect to define a complementary threat profile and trust model based on a use-case-driven data flow analysis. The result of the exercise integrates information about the threats, vulnerabilities, and risk of particular information technology architecture. Further, trust modeling identifies the specific mechanisms that are necessary to respond to a specific threat profile.

The purpose of a trust model is to respond to a specific threat profile. A threat profile is the set of threats and vulnerabilities identified through a use-case-driven data flow analysis that is particular to an organization. Essentially, a threat profile identifies likely attackers and what they want. The level of trust necessary for one organization or circumstance may be different from the level of trust required by another organization or circumstance. For example, the level of assurance that an organization needs regarding the authentication of a user may be different in particular use cases. Trust is evaluated on a gradient – there are no one-size-fits-all solutions.

- Trust requirements must be matched to the specific kinds of threats or vulnerabilities facing an organization and to the degree of risk that the threats will occur.
- There must be a starting point in establishing credentials for identity. A common electronic form of such a credential is called "certificate". A certificate is created using cryptography. It contains a public key that matches the certificate's owner's secret key. A certificate is validated by a certificate authority with electronic signatures.
- Trust does not happen spontaneously. It requires a methodical process of credential establishment and consistent validation.

Implicit trust can be found where the simple existence of data, access rights or the availability of a resource is present. For example, implicit trust is assumed for a word processor that is installed on office computers. It is there and available to users because it is trusted to do its job – and not cause harm to the computer system and violate its security policy.

"Trust propagation" is an approach where trust is not explicit in a trust model, but can implicitly be assured to another entity. A very simple form of trust propagation is the use of passwords to log on to a computer. The possession of the password shows to the authentication system that the user is trusted by the computer system administrator.

Generally, there are three basic ways to propagate trust through entities in a computer system:

1. Direct trust: An entity A and an entity B "know" each other (e.g., because they were programmed by the same software vendor). This establishes a direct trust link.

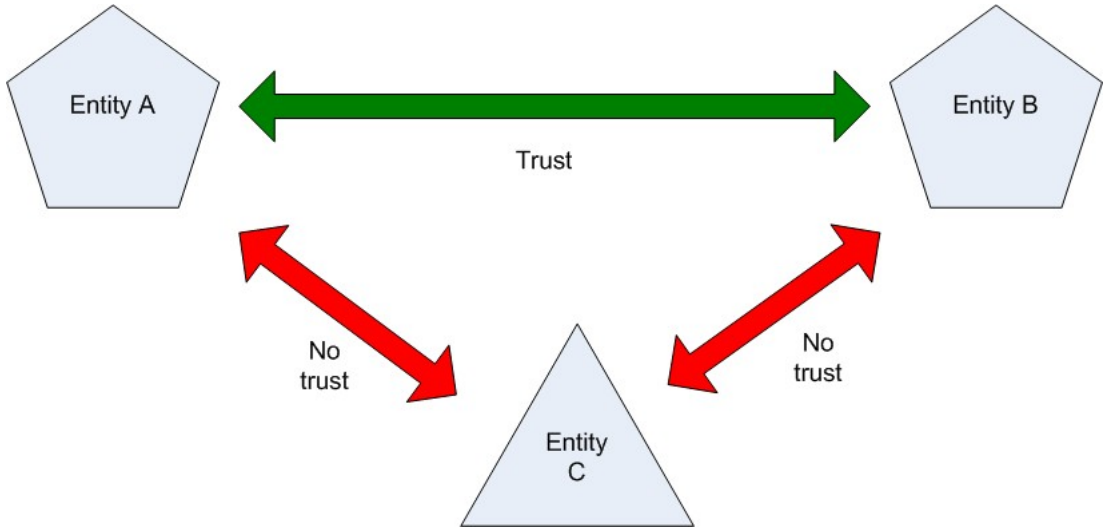

Figure 5.1: Direct trust: Both entities know each other and establish a trust relationship.

2. Hierarchical trust (sometimes, transitive trust): A hierarchy of trust is created. A centralized entity, the certificate authority (CA), creates credentials for all trusted entities. Members of the same trust hierarchy can recognize each other by the analysis of their certificates. This approach resembles the idea behind ID cards or club cards. If you have one, you are let in.

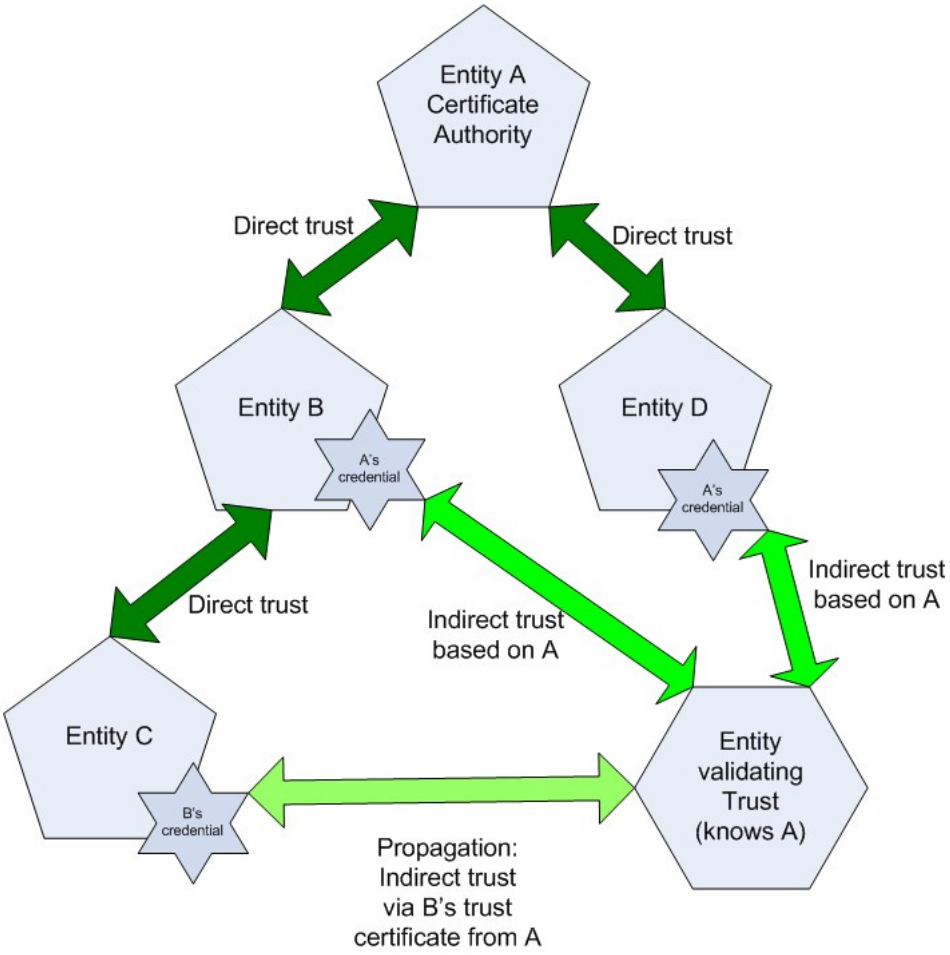

Figure 5.2: Hierarchical trust: Entities trust each other based on a hierarchy defined among them they can rely on.

3. Indirect trust or "Web of trust" (sometimes, spontaneous trust): Indirect trust somehow resembles the social approach of people toward trust. Here, entities create each other's credentials when they trust each other. As each entity can create many credentials for many different other entities, this creates a "web of trust". An entity seeking trust in another entity can check the list of credentials of the other entity for an issuer it trusts. It can also accumulate the different credentials and issuers.

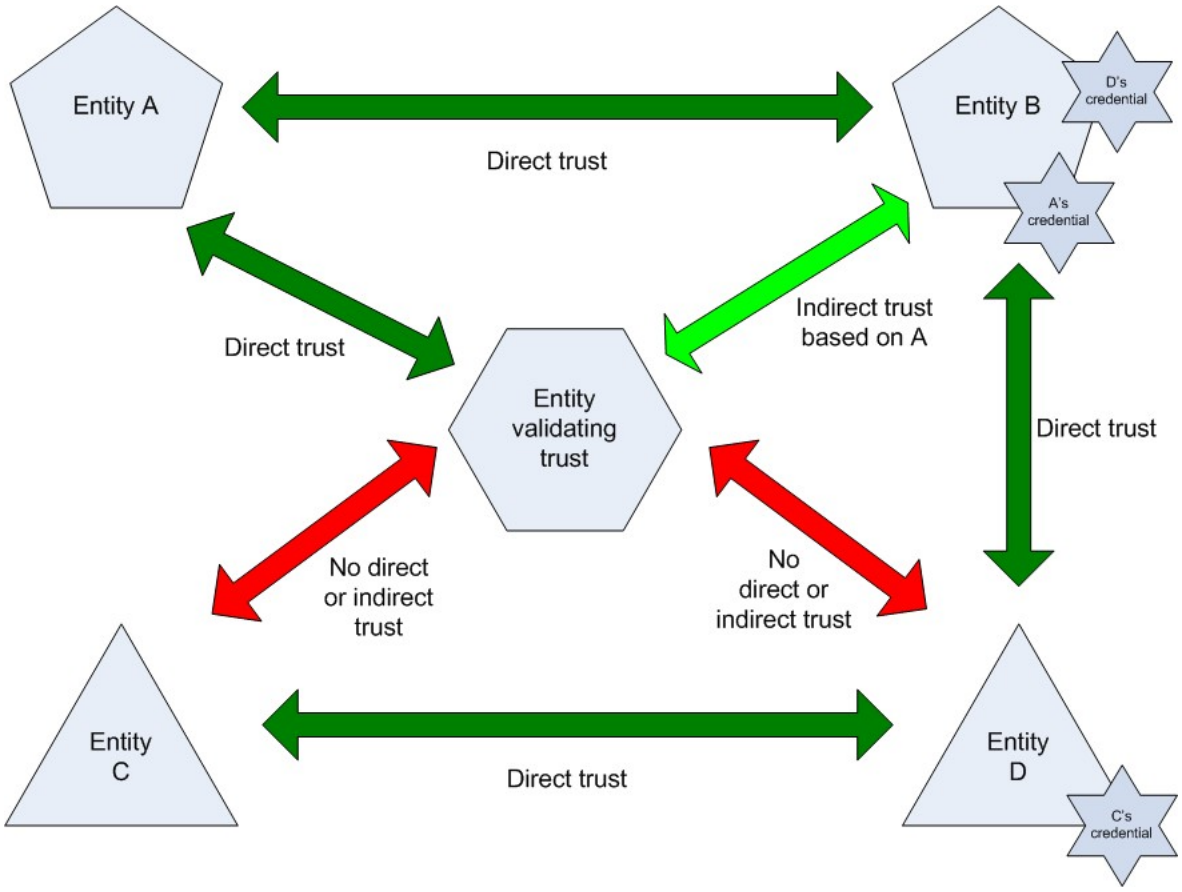

Figure 5.3: Indirect or "web of trust": A network of relationships establishes trust through multiple indirect relationships and individual trust preferences.

The distinction between hierarchical trust and indirect trust is not always clear. Hierarchical trust systems can "cross-validate" each other, creating bridges between the hierarchies. Just as well, individual parts of a web of trust can look like a hierarchy if there is one strongly connected entity that many others directly trust. One important feature of hierarchical trust systems is that there exists, by definition, a central entity.

Local trust versus distributed trust

Another important distinction in the area of trust models is the difference between local trust and distributed trust.

Local trust refers to an information system where all trusted entities are within the information system boundaries. Here, many trust relationships can be established with direct trust links. Operating system components can check each other with checksums. Users can be required to have a – locally presented – finger ready for a biometric fingerprint scan. With local trust models, the most important question is how the components are locally secured against manipulation or malfunction. Local trust models usually care mostly about:

- A secure storage of important keys and certificates
- Protection of programs, configuration files, password lists, etc.
- Access control privileges on the computer system

- Separation of distinct user or application software spaces on the computer system
- Interfaces in and out of the computer system

Distributed trust faces other challenges the models have to consider. In a distributed network, possibly with mobile components such as laptops, the entities establish data connections through untrusted networks. Communication partners, remote user log-ins, data transfers and remote login data have to be specially treated. Attackers can exploit the remote communication opportunities rather than breaking into a computing center to access a computer system. In distributed trust models, the major focus is put on:

- The validation of a remote entity's trustworthiness
- The securing of communication through untrusted communication channels
- The administration of keys and credentials and their secure distribution to the remote or mobile entities
- The authentication of remote users

More about the topic of trust in distributed computer systems will be included in Section 5.2 below.

5.1.3 Objects of Trust Models

A trust model covers many entities of a computer system. Objects in a trust model are all kinds of components, software modules, hardware modules or credentials that are important parts of the security model. The presentation here is oriented along the spheres that were introduced in the access control section in Figure 4.1. In the following, the innermost sphere, the hardware, and then the outer spheres are described.

Hardware sphere

In the hardware sphere, the trust model defines which hardware objects are trusted to what levels, and how risks are dealt with. Questions to consider are for example:

- Will a hard disk actually "delete" data upon file deletion or should the data be physically destroyed upon its removal from a computer system due to the value of the data?
- Can the processor, the main board function or the interfaces be trusted? Can, for example, the graphics card BIOS contain malicious software that manipulates data on the screen? Can sophisticated radio interception methods possibly pick up screen content?
- Which weaknesses can be prevented by using special-purpose security hardware, e.g., cryptographic modules, smartcards, a trusted platform?
- What will happen to the trust relationships in the computer system if one of the hardware components does not function as trustworthily as expected?

For assurance about hardware functionality, there exist several certification schemes. Evaluation and certification according to ITSEC or Common Criteria can be performed according to various levels of investigation depth. However, the evaluation is usually done for one particular application context with its specific trust and security assumptions, e.g., whether a cryptographic module can be worthless to trust in when authorized users give computer access to untrusted users resulting in the untrusted users' ability to execute the software using the cryptographic module. Secure user authorization will be required in the certificate's conditions.

Problems arise upon hardware repairs, updates, and the introduction of new hardware. The old trust relationships in a computer system can be changed by this without proper planning. Maintenance introduces maintenance technicians that have access to the core security hardware. Maintenance

personnel have proven to be notoriously untrustworthy when there is enough profit involved in untrustworthy behavior [Ande1999].

Trust models and security policies therefore should not only specify the "best case" hardware setup, but also consider fallback solutions upon failure, the maintenance case, and other measures.

Software sphere

After the hardware of a computer system is ready to do its work, the operating system needs to be started up for the users and programs. A boot loader brings the initial starting program into the computer system's memory (it is stored on a chip, a hard disk sector or e.g., a USB-stick). The boot loader ensures an initial state of the hardware, and loads the operating system into the computer. Boot loaders, BIOSes and such software are critically important in the trust model: they are the first pieces of software that can determine the remainder of the startup process. They can have exclusive access to the computer system's resources. They should be much protected from manipulation.

Following the boot loader, the operating system is started. As the operating system controls most of the data handling and application program execution, measures for trust in the operating system must be taken, too. These, however, are not simple to achieve.

On the application layer, a certain amount of trust is required, too. Application programs should not be able to compromise security, e.g., by communicating with remote computer systems or loading new programs onto the computer and thereby installing viruses, as some web browsers can do. It is generally advisable to analyze the various software layers according to their relevance to the threat and trust models. Example questions for trust model specification are:

- Can any program manipulate the BIOS?
- Who, and under which conditions, may patch or update the operating system?
- How long may the operating system stay without updates?
- Where and how do application software store and protect application-specific cryptographic keys? Can they be accessed from a malware agent?

Cryptographic Keys

Cryptographic keys play a vital role in trust management. The keys are small pieces of data, so they can be easily leaked and transported, e.g., using a covert channel. The secure installation, storage and use of cryptographic keys are essential to a trust model. Cryptographic keys can be deployed in hardware as well as in software. Determined by the risk at stake or the security level required, hardware-based protection of keys is unavoidable. Generally, one can assume a hierarchy of key storage security (higher numbers indicate less security), although there can be variations depending on the particular technology:

1. Keys will be created, used and stored only on hardware specialized for key management (Hardware Security Module (HSM), e.g., TPM, smartcard)
2. Keys stored in secure hardware storage areas, but used by the CPU and thus temporarily loaded into memory
3. Keys that are kept on disks in encrypted form, but decrypted by the CPU and temporarily loaded into memory
4. Keys kept unencrypted on persistent storage

However, naive usage of certified cryptographic hardware modules does not help when other security policies allow easy access to the modules, as described in [Ande1999] for the case of compromised cash machines.

5.1.4 Subjects of Trust Models

Subjects in trust models are users of a computer system, or abstract entities that act according to a particular role. Here, a general distinction in the local and the distributed case is helpful.

When dealing with subjects, a trust model is concerned with the authentication and authorization of users that access a computer system. Access in a trust model does not only refer to "use of the system", but can also refer to the ability of particular roles, such as the computer system administrator, the maintenance personnel, or any person, to directly access hardware components with hardware tools.

If the assumption is made that the computer with the smartcard that contains a company's secret key is secure because it is locked away into a safe, then the assumption that the people who have the safe keys are trustworthy is often made implicit, but should be made explicit.

A helpful distinction is that of good user versus malicious user of a computer system. A malicious user will try to abuse privileges and opportunities to violate the security policy. Typical protection against malicious intent would be the deactivation of USB ports on a PC to prevent data being leaked on USB sticks. However, if a trusted, usually well-behaving user simply forgets to pick up his printouts from the printer, how will this be considered in the trust model?

Entities are remote computers, computer system processes or computer services that are part of a computer system. Entities do not necessarily act directly on behalf of a real user. They might simply pursue a certain important task on a system. Regular back-ups of hard disk files are done by an entity called "backup process" that is automatically started on many systems. However, if this entity has "read" privileges of a portion of the hard disk with secrets, these secrets can easily be written out to a tape drive with a removable – and easy-to-steal – tape. The entity "backup process" needs similar attention like users that access the computer system.

In case the entity is the remote part of an authorization system or an encryption system, the establishment of trust is of particular importance to the overall computer system's trustworthiness.

5.2 Trust in Distributed Computer Systems

The transition from a local network to interconnected, distributed computer systems, e.g., on the Internet, requires thorough refinement of the trust models used for security models. While all trusted components in a local network are subject to physical security as well as local administrative control, the remote computer systems in the distributed scenario cannot be trusted by definition. Some form of trust evidence is needed to distinguish trusted from untrusted computer systems. Classic and new mechanisms for remote trust establishment will be introduced in this section, such as server certificates and remote attestation.

5.2.1 Introduction to Distributed Trust Requirements

As mentioned above, trust in distributed computer systems requires different approaches than local trust in a single computer system. Distributed computer systems are made up of several machines and have some properties such as remote users, untrusted communication networks, and mobility.

Distributed computer systems can be seen as extending the access control problem to a network of collaborating computer systems. Therefore, distributed trust models are based on vital building blocks such as secure network connections, remote credentials and remote integrity.

5.2.2 Vital Techniques in Distributed Trust

Key management is the core measure for trust in distributed computer systems. Cryptographic keys and algorithms fulfill many tasks in trust establishment. They are used for the secure connections, for user authentication, integrity checking and remote computer system's identification.

Key management is based on a Public Key Infrastructure (PKI) which is a setting where public keys can be openly distributed, but can be used to perform encryption or signature operations involving a secret key. A PKI has at least one certificate authority (CA) which is responsible for the issuance of certificates. The certificates contain approved public keys, and possibly additional data, signed by the CA. Certificates can be used in cryptographic protocols to establish secure communication channels, encrypt and sign data or to authenticate entities.

Certificates can be used in many ways to add trust to a distributed computer system, for example to:

- Establish secure communication channels
- Authenticate remote entities (e.g., user identities using tokens)
- Authorize actions
- Check integrity of data or program code
- Create secure audit trails

However, securing a distributed computer system that uses secure and trusted hardware, runs trusted (certified) operating systems and application programs that can all be authenticated based on certificates is an enormously complicated task. CAs have to be synchronized, and every installation and patch that might happen on any component in the distributed computer system might need new certificates. In turn, their certificates have to be distributed or made available to the other entities.

Remote attestation is another important technology for establishing trust in distributed computer systems. Remote attestation (RA) provides the capability to validate with certainty the hardware and software configuration of a remote host. It is a cryptography-based approach that was introduced with Trusted Computing. Its base is a hardware support for the validation of of loaded software, and software that allows for remote inquiries about the state of the computer system. Using certificates and signatures, a remote computer system cannot fake its software configuration.

While this capability has received much criticism of concerned anti-DRM activists, its benefits are hardly expressed with equal passion. A remote attestation system can check that a remote computer indeed has a particular version of a web shop system installed that is known for being privacy-conformant.

5.3 Trust Relationships

Trust can be expressed as a relationship between computer systems, just as trust between human beings often is established either via social networks and mutual responsibilities, or through the authority of a position in a hierarchy.

This section introduces the two basic approaches for the establishment of trust relationships: Web of Trust (where cross approval of trusting entities' certificates is performed) and Hierarchical Trust (where trust relationships are built according to central entities in a hierarchy). The approaches' implications will be analyzed and discussed.

5.3.1 Hierarchical Trust Model

The hierarchical trust model requires all creation of certificates and their validation to be coordinated by a certificate authority. It is required to trust in the CA to function correctly. This is the reason for the hard security requirements that are expected from CAs that issue electronic signature certificates that are legally binding in the European Union according to its signature legislation.

The CA issues signed certificates, thus establishing a trust relationship with the owner of the certificate. Usually the hierarchical approach is used for centralized computer systems and reflects an organizational hierarchy. All members that seek trust in a certificate holder check whether they get presented a valid certificate from their CA. Trust can be propagated in this model by introducing sub-CAs that are allowed to create certificates for entities (e.g., a department can issue its own certificates). The sub-CA will have a signed certificate from the central CA, thus establishing a trust relationship to the central CA. A certificate from the sub-CA can be linked to the sub-CA and, using the sub-CA's certificate, directly to the CA.

One open problem remains: How does the validating party know that there exists a sub-CA, and how can he check the link to the central CA? For this purpose, CAs have a directory of certificates that is used to look up certificates. If a validating party gets a certificate with an unknown CA signature, he can look up the directory and will find whether this is a sub-CA known to his central CA. If so, he can also look up the certificate that establishes the trust relationship between the sub-CA and the CA.

One last important topic is certificate revocation. To cancel a trust relationship, the corresponding certificate need to be marked as "revoked" in the CA's directory, as well as in the CA's certificate revocation list (CRL).

5.3.2 Web of Trust

The web of trust model aims to be a de-centralized approach. It requires "introducers" as a starting point of trust; hence an introducer is a trusted entity. A web of trust has many introducers. The role of an introducer is to declare trust in a different entity inside the web of trust. Say A is the introducer and B is the introduced entity. Assuming the trust in B will be declared through A, this will be done through the attestation of B's public key by A. Therefore, B has a certificate signed by A. If one imagines A as a central CA with no other introducers around, then it would first look like the hierarchical trust model. But since every entity can act as an introducer, many introducers are expected to exist. The result is a network of entities with cross-validated certificates, and with many direct and indirect trust links in it.

However, lacking a central CA with the authority to create trust, now every participant in the web of trust has to decide who the trustworthy introducers are. This is done by setting trust levels to introducers in each entities personal directory of keys.

The revocation of certificates inside the "web of trust" is done through special revocation signatures which the certificate creator (normally the entity itself) can create and publish through the OpenPGP-Keyserver.

The web of trust concept was introduced by the popular PGP encryption software. Many descendants such as PGPi, OpenPGP and GnuPG / GPG (GNU privacy guard [GNUP2004]) re-implemented it.

5.3.3 Discussion of Hierarchical Trust versus Web of Trust

The hierarchical trust model establishes a strong, centralized control over the trust relationships. The CA determines all subsequent trust relationships. By using directories, certificates of very large organizations or collections of entities can be validated and maintained by the CA.

The advantage for validating parties in hierarchical trust models is the fact that they simply need to check the directory to find the trust path to the CA by linking certificates to each other. Usually, there are few ambiguities in this process. The link to the central CA has to be established, and it has to be checked that none of the certificates on the path to the CA is invalid.

The drawback of the hierarchical approach is its centrality, too. There is a single point of failure – the CA. If the CA certificate is compromised or the directory manipulated, the whole trust relationships of the organization become worthless. All certificates have to be re-issued.

If a directory server is not available, then no validation is possible. Also, the management of revocation can be an enormous effort in large application scenarios with many certificates. Imagine software for which a certificate has been issued by a CA and that is installed on 12000 laptops. If there is a software problem that causes the organization to cancel trust in a program and to revoke its certificate, then each of the 12000 laptops has to go on-line and query the CA before the infrastructure of computer systems in total can be considered safe again.

With the web of trust, the centrality aspect is avoided. This approach has advantages as well as disadvantages. There is little to no central management possible, as the web of trust maintains trust levels at each participant's introducer database. The web of trust thus has no single point of failure like a CA. However, the decision about trustworthy introducers and their revocation in case of lost trust is no simple issue. Some information channel is required – with all its confidentiality and integrity problems. The web of trust works best with many direct trust relationships – and a central directory for revocation.

5.4 Implementing Trust Models, Trust Anchor

This final section concerning trust in IT systems deals with implementation issues of trust models. As the trust model defines the trusted parts of a computer system, its implementation is critical for the quality of the complete secure computer system. Of particular interest is the trust anchor, which is the "first secure part" of the computer system that all other security relies on (compare this to a large houses' foundation – it has to be stable, strong enough and durable to support a house).

5.4.1 Central Trusted-Third Party

This approach to practical security is based on the hierarchical trust model. Some centralized CA is responsible for issuing certificates, directory management and revocation. All validating parties check certificates upon using them.

A widely available example for a centralized approach is the use of SSL server certificates for web servers. Web browsers have a built-in list of CAs (more CAs can be installed later). If the operator of a web page asks for a SSL certificate from one of the built-in CAs, the web browser will automatically check the certificate and provide encrypted web surfing.

Certificates are usually data structures in the X.509 certificate format. X.509 is a very common certificate format. All X.509 certificates comply with the ITU-T X.509 international standard; thus X.509 certificates created for one application program can be used by any application program complying with X.509. In practice, however, different companies have created their own extensions to X.509 certificates, and not all of them work together.

Validating a certificate requires someone to validate that a public key and the name of the key's owner go together. With X.509 certificates, the validator is always a CA or someone designated by a CA. An X.509 certificate is a collection of a standard set of fields containing information about a user or device and its corresponding public key. The X.509 standard defines what information is included in the certificate, and describes how to encode it.

Critical to the practical security of the centralized approach is the security and reliability of the CA and its directory. The CA must also be reachable all the time for validation purposes. Where high security demands are expected based on certificates, the CA is put into a high-security computing center. It is made redundant, and replicated to several locations to be available as back-up if one of the computer centers fails.

Practical examples of this are the CAs for legally binding electronic signatures.

5.4.2 Distributed Trusted Third Parties

A distributed trust approach in practical security is based on the distribution of secure hardware with certificates and keys to different ends of the network. Here, more or less direct trust relationships are established by the installed certificates or security modules. A security module is a special piece of hardware for cryptography and secure storage. It is usually protected against manipulation, e.g., by containing chemicals that will destroy an important part of the module when it is opened or by including the electrical characteristics of the case into the functionality.

The modules are then installed into the entities. The cooperation of the distributed entities is then established using the security modules and their certificates. Practical examples are bank cash machines (teller machines), which are equipped with security modules, and digital TV receiver boxes for encrypted pay-TV. However, if any problem occurs that requires the revocation of keys, or the change of security-relevant elements, usually the security module has to be exchanged or updated by the vendor. In bank cash machines, the module then is replaced. TV receivers receive a new smartcard with fresh keys and algorithms.

5.4.3 The Trust Anchor

Trust anchors are public keys and associated information that are directly trusted by an application program to validate digital signatures, including signatures covering other public keys.

The original entity's authentication permeates all trust models. This refers to a situation where, before trust can be established, relying entities must be convinced of the identity of all other entities with which they communicate or conduct transactions. The level of satisfaction required to convince the relying entity should be specified in a published security policy. As the name implies, original entity authentication occurs only once, at the beginning of a trust relationship.

The original entity authentication establishes a credential that can be evaluated, tested, or referenced by an authenticator or relying entity.

To ensure a reliable validation or authentication process, tokens or credentials must be unique and bound to a specific entity. Furthermore, there should be an agreed upon and standardized format for credentials, as well as for the protocols used to test those credentials.

The original entity's certificate is a critical part of the trust chain, and a valuable target for attackers. As explained above, the central CA key, as well as its certificate, is critical in a hierarchical trust model. Additionally to a reasonable protection of the CA key, establishing a trust anchor in the relying entity must be taken care of. The CA public key certificate must be protected; otherwise an attacker might exchange that certificate and create a fake CA.

A common technique for protecting a trust anchor is to store it into a high security module (HSM) such as a smartcard or to embed the trust anchor into a computing platform (e.g., using a TPM).

Common security issues in the context of trust anchors are:

- Management of a trust anchor life cycle (how to update, distribute, and revoke it)
- Maintaining several trust anchors in a distributed infrastructure
- The hardware that the trust anchor is stored in becomes insecure with the progress of technology

Chapter 6

Alternatives and Combinations

This chapter presents alternative concepts and emerging security techniques. Particularly, it focuses on the recent approach of "Trusted Computing" – an architecture composed of hardware and software providing a trustworthy platform – which is introduced and explained below.

Furthermore, possible uses of Trusted Computing for combinations and enhancements that elaborate the security concepts and models that have been reviewed above are discussed in this chapter. Many of the security models presented in Chapter 4 can be anchored to a trusted computer system using Trusted Computing. Examples and limitations of this approach are covered in this chapter.

6.1 Emerging Concept: Trusted Computing and its Classification

Being a recent IT security development, "Trusted Computing" (sometimes referred to as "Trusted Platforms", or "TCPA") has hit the news with controversial debates about freedom on computers. As mentioned in Section 2.3, the potential of Trusted Computing for the establishment of security anchors and working security models has somehow remained in the shade of the highly emotional media debate. This section thoroughly explains the concept of Trusted Computing according to the Trusted Computing Group's specification [TCG0001]. It will introduce basic models, functionality and provide examples on meaningful usage. A short overview of operating systems in support of Trusted Computing will be provided.

6.1.1 The Trusted Computing Group (TCG)

The Trusted Computing Group (TCG) evolved from the Trusted Computing Platform Alliance (TCPA) which was an industry working group focused on the development of trust and security mechanisms in computer platforms. It was formed by Compaq (today part of Hewlett-Packard), Hewlett-Packard, IBM, Intel and Microsoft in January 1999.

In October 1999 the TCPA announced a draft specification and opened the possibility for other companies to join under a non-disclosure agreement. In August 2000 the first public version of the TCPA Specification was released for comments and has been published as TCPA Specification 1.0 in February 2001. This specification was platform independent and basically defined functions that must be provided by a Trusted Platform Module (TPM) from the viewpoint of a hardware manufacturer.

The TPM Work Group, which has been formed in February 2001, revised the specification regarding practical implementation issues and error correction. This led to the TCPA Specification 1.1 which has been published in August 2001. Many specifications of a "Trusted Computing Platform", which cannot be implemented in hardware, have been deferred to other TCPA Specifications.

In September 2001 the TCPA PC Specific Work Group published its first specification. This working group was set up to design a special specification for the PC platform. The next milestone was the TCPA Specification 1.1b which has been released in May 2002. In April 2003, the TCPA

was transformed into a non-profit organization, called Trusted Computing Group (TCG). The TCG adopted all TCPA Specifications and continued their development.

In addition to the PC Specific Work Group and the TPM Work Group, which have been adopted from the TCPA, the TCG established several other working groups. These are concerned with the development of specifications for mobile devices, PC clients, servers, storage systems, infrastructure for Trusted Computing, TCG Software Stack (TSS) and Trusted Network Connect (TNC).

In November 2003 the last major change to the TCG Specification has been published as TPM Main Specification 1.2. It essentially describes the platform-independent functionality that must be provided by a TPM.

As of 2007, the TCG has more than 130 members, including hardware and software vendors, software developers and network and infrastructure companies.

Activities

TCG Work Groups elaborate the TCG Specifications. At present there are eight working groups considering different aspects of Trusted Computing. The Infrastructure Work Group works on the adoption and integration of TCG platform-specific specifications into Internet and enterprise infrastructure technologies. Following are the main working groups of the TCG as quoted from the TCG website:

TCG: The Trusted Computing Group (TCG) is a not-for-profit organization formed to develop, define, and promote open standards for hardware-enabled trusted computing and security technologies, including hardware building blocks and software interfaces, across multiple platforms, peripherals, and devices. TCG specifications will enable more secure computing environments without compromising functional integrity, privacy, or individual rights. The primary goal is to help users protect their information assets (data, passwords, keys, etc.) from compromise due to external software attack and physical theft.

TPM: The TPM Work Group is chartered to create the Trusted Platform Module (TPM) specification. The definition of the TPM architecture comes from the TC and the TPM Work Group defines the implementation of that architecture.

TSS: The purpose of the TSS Work Group is to provide a standard set of APIs for vendors of application programs who wish to make use of a TPM. The work group will produce a vendor neutral specification which will provide an abstraction of the hardware differences so that vendors of application programs can write application software that will work regardless of the hardware, Operating System, or environment that is used. The TSS will also provide means for applications to talk to TPM's either locally or remotely.

PC Client: The PC Client Work Group will provide common functionality, interfaces, and a minimum set of security and privacy requirements for PC client that use TCG components to establish their root of trust. This work group shall serve an advisory role by providing information to the TPM Work Group and other TCG Work Groups on possible architectural and design issues that may impact their work. The PC Client Work Group's deliverables SHALL NOT address any functionality, interface (except those interfaces between the operating system and the pre-operating system environment), security or privacy issues for the Operating System(s) that are hosted by the platform.

Server: The purpose of the Server Work Group is to provide definitions, specifications, guidelines, and technical requirements as they pertain to the implementation of TCG technology in servers. The work group will endeavor to produce a spec that allows compatibility with currently specified API's and further enables current TCG infrastructures.

Mobile Phone: The Mobile Phone Work Group will work on the adoption of TCG concepts for mobile devices to enable different business models in market environment of open terminal platform. The work group will enhance TCG as needed to address specific features of mobile devices like their connectivity and limited capability as analyzed through various usage scenarios that may demonstrate added value of mobile devices in TCG.

Infrastructure: The Infrastructure Work Group will work on the adoption and integration of TCG platform specific specifications into Internet and enterprise infrastructure technologies to enable various business models in a mixed environment of open platform architectures. Conventions for representing and exchanging information useful in making trust decisions will be established by leveraging existing Internet and related infrastructure standards. Considerations shall be made for representing platform roots of trust, trust chaining, key lifecycle services and the relationship these may have to owner policies. The work group will define an architectural framework, interfaces and metadata necessary to bridge infrastructure gaps.

TNC: The Trusted Network Connect (TNC) Work Group has defined and released an open architecture and a growing set of standards for endpoint integrity. The TNC architecture enables network operators to enforce policies regarding endpoint integrity at or after network connection. The standards ensure multi-vendor interoperability across a wide variety of endpoints, network technologies, and policies.

Storage: The Storage Work Group will build upon existing TCG technologies and philosophy, and focus on standards for the security of dedicated storage systems. One objective is to develop standards and practices for defining the same security services across dedicated storage adapter interfaces, including but not limited to ATA, Serial ATA, SCSI, FibreChannel, USB Storage, IEEE 1394, Network Attached Storage (TCP/IP), and iSCSI. Storage systems include disk drives, removable media drives, flash storage, and multiple storage device systems. Act as TCG liaison to other storage industry standards groups that have jurisdiction over the interface standards to promote adoption of TCG technology. Interaction with other standards groups will be with the approval of the Technical Committee.

6.1.2 Trusted Computing Functions

Integrity Measurement, Verifiable and Secure Booting

Integrity measurement is one of the basic and probably most fundamental mechanisms of Trusted Computing. Integrity measurement means to calculate and store the state of a computer or device under secured conditions. The process of measurement while booting is called trusted booting (sometimes also referred to as authenticated booting; however, this term has been used recently for a subset of trusted booting where the measurement is reported to a remote third party). It is a method that securely logs which software is booted on a computing device but does not influence the boot sequence (e.g., in deciding which software to execute and which not). After finishing the boot sequence, these logs can be used for checking the state of the computer system’s components. An important remark is that trusted booting by no means ensures that the computing device is in a "secure" state. With the help of the log entries, it is also possible to report the state of the computing device to remote entities. By analyzing the logs, remote entities can decide about the trustworthiness of a computer system or sub-components of it. This approach is particularly suitable for secure open platforms, which can be modified in many ways.

The technology of trusted booting has been introduced by Arbaugh et. al. in [ArFaSm97] (there, they call it authenticated boot vs. secure boot) long before the TCG specification and is already used in other designs. Currently available designs of trusted booting require new hardware or support untrusted application software insufficiently. There exists another type of booting a computer system in a trusted way called secure booting. Secure booting checks the program code before

executing it according to a set of security policies and thereby avoids that malicious or unauthorized program code is running on the computing device. The security policy comprises identifiers of authorized modules. Hash values can be used as identifiers. If an identifier or hash value is not found in the list of the security policy, the process of booting is discontinued. Usually this technology is used to ensure locally that the platform is in a secure state, but not to prove this fact to a remote party. The technique of secure booting can be used to construct secure closed platforms, which have a limited number of executable software. Designs for secure booting have been known for over 15 years. Both mentioned methods can be combined. After measurement of integrity the results are sent to a remote validation entity that checks the results. If the computing platform is in an invalid state the remote validation entity may initiate a remediation. So the platform has to be updated. After that, it starts the integrity measurement again until it is in a valid state. When a boot sequence can be validated (remote or locally) it ensures that components of the platform are not emulated, so that a specific hardware with a specific operating system with a specific GUI and a specific application software is indeed running in the identified device.

Binding, Sealing, and Attestation

The hardware integrated root of trust can provide methods to bind data, licenses and user authentication to a specific platform that consists of specific hardware and software executed on that hardware. This can be realized by cryptographic operations which are executed and stored in a protected hardware environment. Within the TCG specification, a unique key, for which a certificate is issued by the manufacturer and which is stored protected in the microcontroller, is essential for approving the platform as trusted and can also be used to authenticate a special user accurately. This key is the initial point for certificates and further keys.

Sealing allows tying data to a specific computing device and thus, to restrict the access to sensible data which is stored on its hard disk to computer systems with a dedicated hardware and software configuration. A major security problem of computing systems is storing cryptographic keys securely. Keys or passwords that protect private documents are often retained locally on a hard drive of a PC, side by side with the encrypted documents themselves. Everyone who gains access to the computing system can also access stored cryptographic keys and passwords. But keys should be kept secure so that only legitimate users can access them.

The technique of sealed storage is based on a key that is partially generated by the identity of the software requiring the key. Furthermore, the identity of the computing device that executes the software represents the second part of the key. So, these keys need not be stored on the hard drive but can be created whenever they are required. If a program different from the program which initiated the encryption (or sealing of sensible information) tries to decrypt or unseal this data, this fails because the generated key is not equal to the original one. That follows from the different identifiers of the software that seal and unseal the data, and consequentially the generated keys are different as well. A similar use case is that encrypted data is transferred to another computing device that tries to decrypt the data. This will also fail. So, for example, emails that one can read on onehis computer are unreadable on other computer systems. With the help of sealed storage one cannot prevent that confidential data is copied to another computer system, but can prevent others from reading it on this computer system. By using attestation it is possible to check the hardware and software states of a remote platform. Therefore, the results of integrity measurements, which characterize the software and hardware environment of a computing system, are signed with a key. The signed outcomes can be checked by a remote party and need no physical presence. Together with the signed outcomes, certificates that accredit the used key as trusted are sent. A remote attestation can be conducted directly or through a trusted third party that approves the remote platform as trusted.

A trusted third party (TTP) checks keys and certificates of a computing device. If they are valid, the trusted third party issues a certificate that attests a computing device as trusted.

Direct anonymous attestation (DDA) without using a third party validating entity is a further attestation technique. It can be proven that a valid certificate exists without disclosing it. So certificates can be generated anonymously. Group signature schemes allow that every member of a group can sign messages on behalf of the whole group. This supports the anonymity of the group members and provides a valid signature.

A third technique to check that executable code is trustworthy is provided by code-signing. Most of the code-signing implementations provide a digital signature mechanism to attest the identity of the author or the producer. A checksum is also attached to the code in order to check that the code has not been altered. Therefore,, a user or a computer system can decide whether the source is trustworthy and whether the code has been modified. Proof-Carrying Code is a technique that guarantees safe execution of untrusted code. The author of the code adds information (called annotation) about the security policy that is fulfilled by the code. The receiver compares the security policy of the code with a set of safety rules established by him. The code will only be executed by the receiver if the examination of the security policy of the code conforms to requirements of the user. Each tampering of the code or the annotation can be noticed by the receiver.

6.1.3 Trusted Computing Architecture

In this section, the main functionalities of the specifications version 1.1b and 1.2 of the Trusted Computing Group (TCG), including binary attestation and binary sealing, are briefly reviewed.

The main components of the TCG proposal are the hardware component Trusted Platform Module (TPM), the protected pre-BIOS[10] called the Core Root of Trust for Measurement (CRTM), and the support software called TCG Software Stack (TSS) which performs various functions like communicating with the TPM, the rest of the platform or with other platforms.

Trusted Platform Module (TPM) Specification Overview

The TPM Specification is the main part of the TCG Specifications. It defines all platform independent aspects and functions that must be provided by a trusted platform. All computer system-specific aspects have been sourced out to computer system specific documents like the PC Specific Specification.

Components

The TCG Specification does not prescribe that TPM devices have to be implemented in hardware, but to provide the degree of security arrogated by the TCG Specifications with a software implementation may be an infeasible task. Thus, most TPM implementations are in hardware. A TPM provides an RSA key generation algorithm, cryptographic functions such as signature and validation with asymmetric cryptography, a secure random number generator (RNG), non-volatile tamper-resistant storage, and the hash function SHA[11].

Hardware-based TPM devices can be compared to integrated smartcards containing a CPU, some memory, and special application software. The assumption is that the chip is tamper-evident and mounted on (or integrated in) the motherboard such that removal is evident to visual inspection. The

[10]The BIOS is the software that manages the start of a computer system by initializing hardware, and loading the operating system.

[11]See http://en.wikipedia.org/wiki/SHA-1 for an overview, or http://tools.ietf.org/html/rfc4634 for the RFC 4634.

main chip contains a special security controller with some internal, non-volatile ROM for the firmware, non-volatile EEPROM for the data and RAM. Furthermore, it contains a cryptographic engine for accelerating RSA encryption and decryption processes, a hash accelerator and a random number generator that is needed to generate secure cryptographic keys.

The components of a TPM must be trusted to work properly which is intended to be guaranteed by Common Criteria evaluation.

TPM Key Types and Credentials

A TPM contains a Root of Trust of Storage (RTS), or Storage Root Key (SRK), that protects data and keys entrusted to a TPM. The SRK manages a small amount of volatile storage inside the TPM device that is used to hold currently used keys (key slots). Unused keys may be encrypted with a storage key and moved off a TPM, e.g., to a hard disk drive. The storage key might be encrypted with another storage key which leads to a key hierarchy with the SRK being the root. The key slots of the TPM are managed by a trusted operating system service outside a TPM that is called Key Cache Manager (KCM).

Each TPM protected key is stored with several attributes that identify the type of the key and what it is intended to be used for. These attributes are set during the generation of the particular key and cannot be altered later.

Storage Root Key

The Storage Root Key (SRK) is used to wrap TPM protected keys which can be stored outside a TPM. This builds a hierarchy of keys on an external storage device like a hard disk drive. The SRK cannot be extracted out of a TPM (it is a non-migratable key (NMK)), and is generated during the process of creating a platform owner. It can be re-generated by creating a new platform owner which destroys the previous key hierarchy and all the keys it contains.

Endorsement Key

Each TPM device is shipped with an embedded non-migratable Endorsement Key (EK) that has been generated as a part of the manufacturing process in or outside a TPM. The EK uniquely identifies a TPM and the surrounding platform. The entity that generates the EK issues an Endorsement Credential which should provide evidence that the EK has been properly created and embedded into a valid TPM.

Besides the two special keys described above, a TPM can create four different types of asymmetric keys:

Migratable Keys (MK): Migratable keys are cryptographic keys that are only trusted by the party that generated them (e.g., the user of the platform). A third party has no guarantee that such a key has indeed been generated on a TPM.

Non-Migratable Keys (NMK): Contrarily to a migratable key, a non-migratable key is guaranteed to reside in a TPM-shielded location. A TPM can create a certificate stating that a key is an NMK.

Certified-Migratable Keys (CMK): This type of key, introduced in version 1.2 of the TCG specification, allows a more flexible key handling. Decisions to migrate and the migration itself are delegated to two trusted entities, chosen by the owner of a TPM upon creation of the CMK: The Migration-Selection Authority (MSA) controls the migration of the key, but does not handle the migrated key itself. In contrast, the Migration Authority (MA) handles the migration of the key: toIn

order to migrate a CMK to another platform, a TPM expects a certificate of an MA stating that the key to be migrated can be transferred to another destination. Furthermore, the certificate of the CMK that the owner/user uses to prove that it was really created by a TPM contains information about the identity of the MA resp. MSA.

Attestation Identity Keys (AIK): These non-migratable signature keys provide pseudonymity and anonymity of platforms including a TPM. AIKs are locally created by a TPM. A certificate is issued for the public part by a Privacy Certification Authority (Privacy CA) stating that this signature key is really under control of a secure TPM. In order to overcome the problem that this party can link transactions to a certain platform, version 1.2 of the TCG specification defines a cryptographic protocol called Direct Anonymous Attestation (DAA), eliminating the Privacy CA. AIKs can be used to attest to specific platform configuration states.). A platform can have multiple AIKs to avoid correlation of platform identities. In order to generate AIKs, the Endorsement, Conformance and Platform Credentials (see below), the EK and the authorization by the platform owner to use them is required.

TPM Credentials

A trusted platform is delivered with several credentials (digital certificates) that should provide assurance that its components have been constructed to meet the requirements of the TCG Specifications.

Endorsement Credential: As already mentioned, the Endorsement Credential should provide evidence that the EK has been properly created and embedded into a valid TPM. It is issued by the entity that generated the EK. This credential contains the name of the TPM manufacturer, the TPM part model number, its version and stepping and the public part of the EK. The EK is a RSA encryption/decryption key pair which is used along with the Endorsement, Platform and Conformance credentials to provide evidence of the platform's identity in a protocol to establish AIKs.

Conformance Credentials: The Conformance Credential is issued by an evaluation service (e.g., the platform manufacturer, vendor or an independent lab) with sufficient credibility to properly evaluate TPMs or platforms containing a TPM. It should indicate that the Trusted Building Block (TBB) design and implementation has been accepted according to the evaluation guidelines. A single platform may have multiple Conformance Credentials for multiple TBBs. They typically contain the name of the evaluator, platform manufacturer, the model number and version of the platform, the TPM manufacturer name, TPM model number and version or stepping and a pointer to the location of the TPM and platform conformance documentation. The Conformance Credential does not contain privacy sensitive information or information that can be used to uniquely identify a specific platform.

Platform Credential: The Platform Credential is issued by the platform manufacturer, vendor or an independent entity. It should provide evidence that the platform contains a TPM as described by the Endorsement Credential. The Platform Credential contains the name of the platform manufacturer, the platform model number and version and references to the Endorsement Credential and the Conformance Credentials. The Platform Credential is privacy sensitive since it contains information that can be used to uniquely identify a specific platform.

TCG Software Stack (TSS)

The TCG Software Stack (TSS) provides a platform independent software interface for accessing TPM functions. The TSS enables the creation of interfaces for existing cryptographic APIs such as MS-CAPI8 or PKCS#119. This enables TPM support for current application programs that isis

using those APIs. In order to take full advantage of a TPM's attestation functions, however, application software will have to support TSS directly.

6.1.4 Operating Systems supporting Trusted Computing

Microsoft Windows Vista

Microsoft [Micr2007] is an Initial Promoter Member of the TCG. Its operating system, Windows Vista, formerly known as Longhorn or NGSCB, provides native support for TCG 1.2 compliant TPM hardware for -file and folder encryption (called BitLocker[12]).[13]). In the 64-bit-version, additional operating system kernel integrity checking and driver signing is performed [Muel2007].

In May 2007, Microsoft and the Trusted Computing Group announced interoperability of a set of protocols that enable remote validation of "system health" (called "TNC" by the TCG and "NAP" by Microsoft). According to a press release of TCG, Microsoft will ship Vista with this option in 2008[14].

Below application software level, Vista contains according to Microsoft[15] a TPM management API called TPM Base Services (TBS). Their functionality is described by Microsoft[16] as:

- Provide efficient sharing of limited TPM resources, such as key slots, authorization sessions slots, and transport slots
- Provide prioritized and synchronized access to TPM resources between multiple instances of TPM core services (TCS)
- Provide appropriate management of TPM resources across power states
- Prevent TPM software stacks from accessing TPM commands that should be restricted, either because of platform limitations or administrative requirements

Many of the security features that were announced for the Palladium or NGSCB platforms (e.g., listed in the outdated description at [Bund2007]) are not part of Vista as of 2007.

Hypervisor: XEN

XEN, a FLOSS virtual machine monitor (VMM, also referred to as Hypervisor), which has been developed by the University of Cambridge and is currently distributed under the GNU General Public License (GPL). XEN is able to execute multiple guest operating systems in a virtual machine (VM) by providing an abstract layer above the computer's hardware. It supervises and manages the distribution of resources such as CPU time or I/O cycles to the guest operating systems. In order to be able to work with the XEN architecture, the guest operating system's kernel has to be modified ("Paravirtualization"). IBM Research integrated sHype's hypervisor security architecture into XEN.

---

[12]See http://technet2.microsoft.com/WindowsVista/en/library/c61f2a12-8ae6-4957-b031-97b4d762cf311033.-mspx

[13]See http://technet2.microsoft.com/WindowsVista/en/library/c61f2a12-8ae6-4957-b031-97b4d762cf311033.-mspx

[14]See https://www.trustedcomputinggroup.org/news/Industry_Data/TNC_NAP_white_paper_final_may_-18_07.pdf.

[15]See http://technet2.microsoft.com/WindowsVista/en/library/29201194-5e2b-46d0-9c77-d17c25c56af31033.-mspx.

[16]See http://msdn2.microsoft.com/en-us/library/aa446796.aspx

sHype provides flexible access control enforcement to strongly isolate and control the sharing of hardware resources and the communication between different VMs.

Since a TPM is not a device that was designed to be accessed by multiple operating systems at the same time, IBM Research is about to develop a virtual TPM architecture in order to provide TPM support to all operating systems running on XEN. To realize this, the TPM command set defined in the TCG 1.2 Specification has been extended by virtual TPM management commands which enable virtual instances of TPMs that can be transparently used by all guest operating systems. Thus software intended to work with hardware TPMs is supposed to continue to work without any changes when executed on XEN.

PERSEUS Architecture and TURAYA security kernel

The PERSEUS architecture provides an open computing platform offering a basis for the realization of multilateral security based on Trusted Computing. It is intended to support a wide range of hardware platforms like PC, PDA and embedded computer systems. PERSEUS uses a micro-kernel which only includes elementary functions such as process management, memory management and inter-process communication and thus minimizes the security-relevant part of a computer system which allows formal verification of its implementation. The PERSEUS architecture realizes security critical application software like digital signatures or DRM application software and conventional operating system as separate processes which are only able to communicate to each other or to the computer system's hardware by involving the PERSEUS security kernel.

PERSEUS will enable the realization of policy enforcement, e.g., enforcing license agreements (if accepted by the user) or permitting access to information only for payment but prevent content providers to gather more private information about the user as needed to provide their service. Thus PERSEUS may serve as basis for DRM solutions and provides compartment mode security to prohibit access to documents outside a desired workflow. It can also be used to realize a secure multi server system which is able to run different isolated processes like a web server, database and firewall in parallel on the same hardware.

The security software layer of PERSEUS consists of a micro-kernel based on the L4 micro-kernel family which has been developed at the University of Karlsruhe and the Technical University Dresden. Above this kernel, a resource management and access control layer has been placed, and it controls the distribution of the computer system's hardware resources to the conventional operating systems and all security relevant application software above it. PERSEUS also provides a secure boot loader which has been implemented as a TCG-enabled version of GRUB (Grand Unified Bootloader) to assure that a specific operating system configuration is booted. A secure user interface (Secure GUI) will provide trusted paths between the user and secure application programs, as well as between secure application programs and hardware. PERSEUS' application manager will ensure the controlled installation and update of software since these tasks offer the possibility to infiltrate malicious software.

Conventional operating systems usually run application software with the rights of the user who started them. This results in running application software with more privileges then they actually need. PERSEUS' application manager will care about privileges of application software since these should be minimal.

TURAYA Security Kernel

TURAYA is the implementation of a security architecture using the security software layer of PERSEUS and available under commercial and open-source licenses. The TURAYA Security Kernel has been developed according to the HASK Common Criteria Protection Profile (see [HASK-PP])

6.2 Combining Trusted Computing With Related Security Concepts

The section above introduced Trusted Computing (TC) as a technology for security and trust anchor establishment. These properties are an important foundation for the security models that were introduced in Chapter 4. Hence, this section will illustrate and discuss how security models can be combined with Trusted Computing to achieve higher security levels than the currently available ones. Remaining challenges of this approach like the border of the trusted computer system, and the particularities of key management in Trusted Computing will complete the section.

6.2.1 Trust Anchor with Trusted Computing

The use of Trusted Computing for securing a trust anchor in a computer system is one of the major use-cases of TC. Using a TPM along with the TSS software stack, trust anchors can be maintained with high levels of security.

The principal trust anchor in Trusted Computing is the TPM. This piece of hardware hosts the security core functions such as the algorithms and the secure storage to be used in the trusted environments.

To establish a meaningful trust anchor, not only TPM must be present in a computer system, but also its integration into the computer system startup routine (CRTM) must be established. From there, the trust anchor can be linked to the rest of the local computer system.

6.2.2 Software Integrity with Trusted Computing

TC's property of verifiable and secure booting creates an environment where the loading and starting of software on a computer system can be enforced and verified. A setup with this property ensures that a computer will run a particular software configuration when started. If the operating system designers and application software designers set up the computer systems in secure ways, the computer system administrators will be able to install them properly and ensure that the software will not be manipulated.

Software integrity is a concept that (until recently), strongly relied on the assumption that only the computer system administrator can install and change critical software on the computer system – and that he will always do so in the right way. Additionally, users can be allowed to install software. In the latter case, the integrity of the installation is completely at the mercy of the user's skills and intents. While some approaches such as type enforcement (see Section 4.1.3) and hypervisors (see Section 4.5.3) try to isolate application programs from each other, the introduction of integrity control based on Trusted Computing can add another dimension to application integrity.

With the help of a secure loader (that in turn is checked by the boot loader using a TPM trust anchor), all operating system parts and applications are checked upon loading. Even configuration files, policies or application data can go through the secure loader function. This way, the computer system detects the integrity state of all components, application programs and their data if necessary. Trust decisions can be made based upon the things that were found upon loading of the application programs.

A practical example of this would be a personal computer in a private household. As of today, all applications usually share most of the data with each other. Word processing, home banking, e-mail, gaming, downloading software and media consumption happen all with access to most of the data. Many of the malware attacks such as viruses, Trojan horses, spyware and automated phishing happen due to a manipulated piece of software running within the application programs or operating

systems. In a protected computer system with integrity control, one can imagine the following scenario: Using a hypervisor approach, the computer would upon boot-up start several virtual machines. One of these virtual machines is the trusted machine which allows only application programs with absolute integrity to execute and access the critical data. Here, one would install home banking software, signature software, and other critical software and their data. A second machine would be a place to install software, try out new application programs, and lower the trust level to "medium". This machine will only allow access to critical data upon explicit user confirmation. Here, application programs can be experimented, but possible security problems are reported by the system. A third virtual machine would be the "anything goes" sandbox. Here, no access to local data or application program outside of the machine would happen. Active content, downloaded software, computer gaming and the like can be done. By using a TPM and the connected secure loader, the computer system can always switch to the trustworthy virtual machine when the user performs actions that need high security levels.

6.2.3 Distributed Trust with Trusted Computing

Remote attestation based on the Trusted Computing Trust Anchor can be of great significance for trust management in distributed computer systems. Note that a distributed computer system ranges from a collection of removable peripherals (e.g., USB-sticks and hard drives) up to a whole network of collaborating GRID computers or web servers and data bases.

The establishment of distributed trust relies on binding the devices to each other. Basically, this is done by the deployment of a key or a certificate that is stored in each of the component's TPM. By executing remote attestation protocols, components or computers can convince each other of the fact that they belong to the same trust zone or that they have a common trust anchor.

This feature is actually an enhancement of contemporary protocols that use certificates and keys for building trust, e.g., X.509-certificates for authentication or smartcard certificates for single sign on. Trusted Computing adds a secure storage for the credentials and a hardware-protected, possibly standardized trust anchor for the certification scheme in all parts of the distributed computer system. The major benefit of using Trusted Computing for this is the access to a standardized platform for trust anchor establishment and attestation.

Many solutions for this challenge have been implemented using smartcards, CAs, PKI and security modules of quite spread compatibility and quality.

6.2.4 Classic Security Models and Trusted Computing

Many of the classic security models are implemented as software components in the operating systems or data base engines. A common concept is that of a reference monitor as a decision engine about access to data. The reference monitor is yet another software module that performs a very critical function in secure computer systems. Most of the security models presented in this study make use of a reference monitor at some point, e.g., upon access to data or upon granting of privileges.

In combining and enhancing the classic security models with Trusted Computing, the reference monitor is a key component to adapt to take advantage of a TPM. The reference monitor can cooperate with a TPM and the secure loader in various ways:

- enforce integrity policies of application programs
- enforce data integrity policies
- ensure the integrity of identity, credentials and keys

By enhancing the reference monitor, models like, e.g., type enforcement, can reliably check an application program's integrity before assigning it a higher security level.

By adding Trusted Computing to the classic security models, it is to be expected that the robustness and effectiveness of the classic security models increases. Also, their functionality can be enhanced by secure loading, integrity control and remote attestation.

6.2.5 Challenges

Although the functional specification and hardware integration of Trusted Computing is mostly in place today, challenges for its deployment and success remain. These challenges are summarized in the following sections.

Standardization

The effective deployment of any security-enhancing Trusted Computing platform base requires a strong standardization effort. Not without a reason, the Trusted Computing Group's activities have taken many years, and are still ongoing. As the effective use of Trusted Computing in combination with other security measures in operating systems or application programs requires the enhancement of these, the protocols and procedures for standardized usage of Trusted Computing have to be found as well.

Yet it remains to see how fast the vendors of different operating system families are ready to implement standardized, possibly certified security models, unified interfaces for remote attestation and generic methods for integrity or configuration control, key management and the like.

Critical Mass

For any new technology to succeed, it must reach a critical mass on the market. As described in diffusion theory [Roge2003], a high number of so-called 'early adopters' has to spread knowledge about the usefulness of the new security platform. Only when their – usually well cost-intensive – advance succeeds, a massive adoption can follow.

Usually, with products in the information economy, strong network effects support fast adoption. Network effects can only be established when major industry players join in to support the new platforms and thus create a cross-focus-group interaction. This can be reached by standardization and open interfaces.

Customization and Management of Keys

When many Trusted Computing platforms are deployed, the customization of the secure machines and the key management comes into focus. One of the core requirements from consumer organizations is the possibility to start a personal computer in a way that is not controlled by remote requirements that enforce a local configuration. This necessarily leads to computer systems with many alternative configurations. Instead of a single PC, the user or owner now has the administration effort of several virtual machines than are used in a serialized manner or in concurrently. This requires education and experience, which require user effort – and slow down adoption.

Another important question is that of key management in large infrastructures that are based on Trusted Computing platforms. To enable a global remote attestation infrastructure, not only a standardized or homogeneous computer and software base is necessary, but also a way to deal with

multitudes of attested configuration values of computer system's components, certificates, keys and other credentials that are being used within this infrastructure. It can be expected that PKI technology will become a key to manage these infrastructures.

Monopolies and Oligopolies

The final challenge is that of competition politics. Even very early in the advent of Trusted Computing, the danger of the establishment of monopoly power or the establishment of fatal lock-in into certain platforms was raised. Indeed the reliance on a single computer system that belongs to one or a few major players might create such a situation. This usually increases prices, lowers competition, and in the long run, prevents innovation.

The deployment of Trusted Computing technology should therefore be implemented with care. Several break points in the infrastructure, which allow the exchange of components, need to be considered, for example, whether there should be an interface to separate the operating system from the trusted platform (e.g., a hypervisor)? The same can be done with key management and integrity control.

A measure to help with this issue is the development of open interfaces and solutions, e.g., by standardization, FLOSS-Software development, or – in the worst case – by governmental antitrust regulation.

Chapter 7

Conclusion

This chapter concludes the study's findings concerning the combinations of classic security models with the Trusted Computing platform. It analyzes the applicability, maturity and availability of the technologies and provides an outlook on emerging developments.

Concluding the analysis of classic security methods and upcoming technologies, this chapter argues toward the increasing security and stability of computer systems due to the incorporation of hardware-based, standardized trusted computing platforms. This section mentions exemplary trends of potential development based on Trusted Computing. It also shows the place of Trusted Computing components and functionalities – along with other supporting or alternative concepts, models and mechanisms – within the scope of security-critical application programs.

7.1 Development paths of secure computer systems based on Trusted Computing

As explained in Section 6.2.1, Trusted Computing provides the possibility for establishing a Trust Anchor which can be at the root of the chain of trust between the tamper-resistant hardware (the TPM) within the execution system and the policy enforcement mechanisms. This can allow a range of application areas that are categorized in four areas. The development paths are shown in Figure 7.1 which illustrates the further evolution of secure computing that will be discussed below.

Figure 7.1: Possible development paths of Trusted Computing.

7.1.1 Classic Security Models

Mandatory Access Control (MAC) and Discretionary Access Control (DAC), Multi-level security (MLS), Chinese Wall, Type enforcement, Separation of duty and Role Based Access Control (RBAC) are all based on the concept of an access control matrix, with different properties and allowed operations. The matrix is usually at the basis of a security policy that needs special enforcement mechanisms. Typically, a program monitor running between the untrusted executable code and the executing system is responsible for checking that all operations are according to the security policy. The trustworthiness of the program monitor is hence crucial for the enforcement of such access control policies on the execution system.

Using a TPM as Trust Anchor can help establish trust in policy enforcement mechanisms such as a program monitor. In fact, the trustworthiness of a program monitor is guaranteed through its integrity, which can itself be achieved by signing the program monitor with a private key corresponding to a Trust Anchor. For instance, a user's platform having a TPM as Trust Anchor can sign a hash value of the program monitor (among other pieces of software) and send it in a remote attestation protocol to another party. The latter can check the integrity of the program monitor based on the Trust Anchor, which would enable him to trust the program monitor running on the user's platform, and hence trust the enforcement of the access control model.

7.1.1 Secure Viewer and I/O

This category include secure input and output to an application program, taking into consideration the different peripherals used for that purpose such as screens, keyboards and fingerprint scanners. An approach based on Trusted Computing would support an application program user interface which is completely under control of a trusted computing base (TCB). The TCB itself can be checked through integrity checking based on the Trust Anchor and the chain of trust. This would allow security features of the user interface to be implemented and checked for correct operation. Such features might include mechanisms for authenticating the application program to the user in a way to allow him to undoubtedly identify the application program he is communicating with.

7.1.2 Secure Policy-based dataflow

Also considered as information flow control, this category includes use-cases that can tremendously be enhanced through a TPM as a Trust Anchor. Similarly to the program monitor case, policy-based dataflow builds on policy enforcement mechanisms. Those can be specific usage rights enforcers as in the case of DRM and Sticky Policies, or even integrity checkers for secure software updating. Those enforcers or checkers can themselves be subject to remote attestation based on signatures generated by a TPM private key corresponding to a Trust Anchor. Therefore, a remote party can check the integrity of those mechanisms running on the user's platform, and hence be assured about the behavior of the software on the platform. This would allow the remote party to confidently send data or media files to the user's platform, being sure that the usage policy controlling the data flow is well enforced. Application areas of such schemes are Pay-TVs, software rental and media with stateful licenses. They give great advantage for content providers who would be able to control the usage of their content by consumers.

7.1.3 Remote Trustworthy Computer systems

This category of applications spans over all scenarios requiring attestation of the software stack running on a remote platform to another party. This would allow the other party to predict the

behavior of the corresponding computer system, but also to report any action made by the user on the software, whether maliciously or by mistake. Being the root of the Chain of Trust, a TPM can issue a valid attestation certificate reflecting the status of the consecutive pieces of executable code running on the user's platform. The trustworthiness of the remote computer system in terms of predictable behavior is therefore reduced to that of the Trust Anchor. Typical use-cases include all sort of software integrity checking on remote or mobile platforms, ensuring non-tampering with the software, correct and trustworthy platform configuration in addition to accountability of securely authenticated users for their actions on the computer system, reporting audit logs and supporting forensic analysis.

7.2 Trusted Computing within the Security Concept

Trusted Computing plays a big role in establishing trust in the security of a computer system. By providing the Trust Anchor, it ensures reliability and trustworthiness of the Trust Model implementation.

Speaking on an abstract level, Trusted Computing supports a Security Concept by directly enhancing the Trust Model implementation with the Trust Anchor notion based on a TPM, therefore indirectly supporting the Security Model and Security Goals. For example, the achievement of a Security Goal like Confidentiality is directly dependent on encryption mechanisms and on the Security Model defining where encryption should be applied. The Security Model itself depends on the Trust Model in terms of specification of the trustworthiness of its technical components. Therefore, the integration of Trusted Computing technology in current secure computer systems improves the Security Concept by integrating a cheap, multi-purpose and standardized Trust.

The development paths of secure computer systems based on Trusted Computing depicted in Section 7.1 give a practical perspective of this conclusion.

But Trusted Computing is not only about establishing Trust Anchor. It is also capable of extending the trust beyond the components of a local computer system. In fact, functionalities based on Trusted Computing, such as Attestation, Sealing and Trusted Boot (based on CRTM) play an important role in establishing trust in remote computer systems or at least part of their components or behavior.

For this reason, the range of application areas for Trusted Computing and its supported functionalities is wide since the trustworthiness is extended from local to remote computer systems components.

7.3 Application of techniques complementary to Trusted Computing

The next Figure 7.2 gives an overview of the applicability of security concepts, models, mechanisms and hardware components mentioned in this report. Those are categorized according to application areas versus Trusted Computing supported features. The application areas that seemed more relevant for Trusted Computing are "Information Flow Control", "Data Access Control", "Secure Electronic Signature", "Content Protection", "Non repudiable PC Forensics", "Secure Virtual Machines", "Secure Authentication" and "Reliable Software". The Trusted Computing supported features which are "Remote Trustworthiness", "Trust Anchoring", "Policy Enforcement" and "Secure Input / Output" might be relevant or not to each of the application areas. Security models, mechanisms or components are attributed to a certain Trusted Computing-supported feature within a relevant use-case.

However, the elements in the figure are not only Trusted Computing components or functionalities, but also independent models, mechanisms and concepts complementary to those components or functionalities.

For example, content protection as an application area can benefit from different features supported by Trusted Computing, as shown in the figure. Remote trustworthiness is an important feature based on which a content provider can trust the user platform about a certain behavior which is necessary for protecting the distributed content so that it does not get illegally distributed to unauthorized users. Particularly, the attestation functionality provided by Trusted Computing is of relevance to content protection. With attestation, a user can report the configuration and properties of his platform to the content provider who compares them to "known-good" values. The content provider will be assured of the user's platform behavior, and hence be confident in sending over the content. The attestation certificate itself can only be validated based on the Attestation Identity Key (AIK) which is linked to TPM and to its platform. Hence, this TPM is the Trust Anchor of the platform.

On the other hand, fine-grained access control over the content is well achieved with the enforcement of a certain usage policy on the user's platform. Policy enforcement is therefore necessary, and can be achieved in this case by means of multilateral security concepts, based on sticky policy or DRM implementations. A trusted viewer is necessary to protect against leakage of content by intercepting the output, and this can be guaranteed with the inclusion of the elementary security features of the trusted viewer in the trusted computing base (which is measured during the attestation process).

This combination of security mechanisms, models and components – whether Trusted Computing-related or not – is represented in the 'Content Protection' category of Figure 7.2.

Another example which is the 'Data Access Control' application field shows how the same Trusted Computing supported features can be achieved with alternative mechanisms. Establishment of Trust Anchors in this case can rely on the use of smartcards where authenticity and authorization of the user is the ultimate security goal requiring trustworthiness. Moreover, policy enforcement is realized through security models such as Bell-LaPadula or Chinese Wall.

Combinations of Trusted Computing components and other mechanisms or concepts can also achieve a high security level, as is the case with secure virtual machines (VMs) which are based on an operating system's separation kernel for isolating the VMs, TPM for Trust Anchor establishment, and attestation functionality to remotely check the VMs configuration.

Therefore, complementary solutions to Trusted Computing are mainly the ones that address the measurable integrity of its technical components. Hardware based security techniques are generally aimed at this goal, and Trusted Computing seems to be the most relevant, well specified and standardized technology within this scope and that explicitly targets commercial application scenarios.

Figure 7.2: Trusted Computing and complementary techniques

Bibliography

[Ande1996] Anderson, R. (1996) A security policy model for clinical information systems. In Proceedings of the 15th IEEE Symposium on Security and Privacy. IEEE Comp. Society Press, 1996.

[Ande1999] Anderson, R. (1999) Why cryptosystems fail, University Computer Laboratory, Cambridge, UK.

[Ande2001] Anderson, R. (2001) Security Engineering, Wiley Publishers, New York.

[ArFaSm97] Arbaugh W. A.; Farber D. J.; Smith J. M. (1997) A Secure and Reliable Bootstrap Architecture, Proceedings of the IEEE Symposium on Research in Security and Privacy, Oakland, May 1997.

[Bell2005] Bell, D. E. (2005) Looking back at the Bell-LaPadula security model, Proceedings of the Annual Computer Security Applications Conference (ACSAC) 2005, Marriott University Park, Tucson, Arizona.

[BSSW1995] Badger, L., Sterne, D. F., Sherman, D. L., Walker, K. M., Haghighat, S. A. (1995) Practical Domain and Type Enforcement for UNIX, Proceedings of the IEEE Symposium on Security and Privacy 1995.

[BeLa1973] Bell, D. E., LaPadula, L. J. (1973) Secure Computer Systems: Mathematical Foundations, MITRE Corp Bedford MA Report AD0770768, Nov. 1973, 42 pages.

[Biba1977] Biba, K. J. (1977) Integrity Considerations for Secure Computer Systems, MITRE Corp MA Report MTR-3153, April 1977.

[BoBo1995] Borcherding, B.; Borcherding, M. (1995) Covered trust values in distributed systems, In: Communications and multimedia security. Proceedings of the IFIP TC6, TC11 and Austrian Computer Society Joint Working Conference on Communications and Multimedia Security, 1995. Ed.: R. Posch. London 1995. pp. 24-31.

[BoKa1985] Boebert, W. E. and R. Y. Kain (1985) A Practical Alternative to Hierarchical Integrity Policies, Proceedings of the National Computer Security Conference, Vol. 8, Num. 18, 1985.

[BrNa1989] Brewer, D.F.C., Nash, M.J. (1989) The Chinese Wall Security Policy, In: Proceedings of the IEEE Symposium on Security and Privacy, IEEE Press, pp. 206–214.

[BrRo1991] O'Brien, R. and C. Rogers (1991) Developing Applications on Lock, Proceedings of the National Computer Security Conference, Vol 14, 147-156, October, 1991.

[Bund2007] Bundesamt für Sicherheit in der Informationstechnik (2007) Sichere Plattformen und die Trusted Computing Group (TCG), http://www.bsi.bund.de/sichere_plattformen/trustcomp/infos/tcgi.htm

[CC1996] Common Criteria Portal (1996) Common Criteria (CC), http://www.commoncriteriaportal.org/.

[ClWi1987] Clark, D., Wilson, D. (1987) A Comparison of Commercial and Military Computer Security Policies, In: Proceedings of the IEEE Symposium on Security and Privacy, 1987, pp. 184, http://doi.ieeecomputersociety.org/10.1109/SP.1987.10001.

[ClWi2007] Clark-Wilson Model, Wikipedia, 15-Nov-2007, http://en.wikipedia.org/wiki/Clark-Wilson_model.

[CPB2003] Cassa Mont, M.; Pearson, S. and Bramhall, P. (2003) Towards Accountable Management of Identity and Privacy: Sticky Policies and Enforceable Tracing Services, Proceedings of the 14th International Workshop on Database and Expert Systems Applications (DEXA'03), IEEE Computer Society, pp. 377.

[Denn1976] Denning, D. E. (1976) 4, Communications of the ACM, v.19 n.5, May 1976, pp.236–243.

[Dix2003] Dix, A. (2003) Trusted Computing und Datenschutz in Deutschland, DuD - Datenschutz und Datensicherheit 27(2003)), pp. 561–562.

[DrNe1998] Dridi, F., Neumann, G. (1998) Towards Access Control for Logical Document Structures, Proceedings of the Ninth International Workshop on Database and Expert Systems Applications, DEXA 98, pages 322-327, Vienna, Austria, August, 1998.

[FeKu1992] Ferraiolo, D. F., Kuhn, D.R. (1992) Role Based Access Control, Proceedings of the 15th National Computer Security Conference 1992.

[FKC2003] Ferraiolo, D. F., Kuhn, D. R., Chandramouli, R. (2003) Role Based Access Control, Artech House publishers, 338 pages.

[Fras2000] Fraser, T. (2000) LOMAC: Low Water-Mark Integrity Protection for COTS Environments, Proceedings of the IEEE Symposium on Security and Privacy, 2000.

[GNUP2004] GNU privacy guard, 15-Nov-2007, http://www.gnupg.org/

[GS2003] Günnewig, D. and Stüble, C. (2003) Trusted Computing ohne Nebenwirkungen - Spezifikationen der Trusted Computing Group, DuD – Datenschutz und Datensicherheit 27(2003)), pp. 556–560.

[HaRU1976] Harrison M. A., Ruzzo, W. L., Ullman, J. D. (1976) Protection in Operating Systems, Communications of the ACM 19/8, pp. 461-471.

[HASK-PP] Sirrix AG: "High Asurance Security Kernel EAL 5Protection Profile", certified by the BSI, 2008, online: http://www.sirrix.com/media/downloads/54500.pdf.

[ITSEC1991] Commission of the European Communities (1991) Information Technology Security Evaluation Criteria (ITSEC): Preliminary Harmonised Criteria, Document COM(90) 314, Version 1.2, Jun-1991.

[Lamp1971] Lampson, B. W. (1971) Protection, Proceedings of the 5th Princeton Conference on Information Sciences and Systems, pp. 437.

[Lewi2003] Lewis, S. R. (2003) How much is stronger DRM worth? 2nd Annual Workshop 'Economics and Information Security'; University of Maryland, May 2003.

[Micr2007] Microsoft Corporation, 15-Nov-2007, http://www.microsoft.com

[Muel2007] Müller, Thomas (2007) Trusted Computing mit Windows Vista und Windows Server 'Longhorn', 10. Deutscher IT-Sicherheitskongress des Bundesamtes für Sicherheit in der Informationstechnik, 22.-24. May 2007, Bonn-Bad Godesberg. In: Innovationsmotor IT-Sicherheit, ISBN 978-3-922746-98-0, Secumedia Verlag Wiesbaden, 2007, http://www.xnos.org/fileadmin/labs/tc/TrustedVistaTM06.pdf

[NIST0001] Role Based Access Control, National Institute of Standards and Technology, NIST, 08-Aug-2007, http://csrc.nist.gov/rbac/

[Pear2002] Pearson, S. (2002) Trusted Computing Platforms: TCPA Technology in Context, Prentice Hall International.

[PfKo2001] Pfitzmann, A. and Köhntopp, M. (2001) Anonymity, Unobservability, and Pseudonymity - A Proposal for Terminology,: , in: H. Federrath (Eds.): Designing

Privacy Enhancing Technologies, Heidelberg, Springer Verlag, pp. 1-9. Latest version at: http://dud.inf.tu-dresden.de/Anon_Terminology.shtml

[Rann1994] Rannenberg, K. (1994) Recent Development in Information Technology Security Evaluation - The Need for Evaluation Criteria for Multilateral Security, in: R. Sizer; L. Yngström; H. Kaspersen and S. Fischer-Hübner (Eds.): Security and Control of Information Technology in Society - Proceedings of the IFIP TC9/WG 9.6 Working Conference 1993, August 12-17, 1993, onboard M/S Ilich and ashore at St. Petersburg; Russia.

[Roge2003] Rogers, E. M., Diffusion of innovations. New York: Free Press, 2003.

[Rush1992] Rushby, J.(1992) Noninterference, Transitivity, and Channel-Control Security Policies, SRI Computer Science Laboratory Technical Report CSL-92-02,

http://www.csl.sri.com/papers/csl-92-2/ .

[Sand1992] Sandhu, R. S. (1992) The Typed Access Matrix Model, Proceedings of the 1992 IEEE Symposium on Security and Privacy, May 04-06, 1992 p.122.

[Sand1993] Sandhu, R. S. (1993) Lattice-Based Access Control Models, ACM Computer Journal, 26, 11, Nov. 1993, pp. 9-19.

[SaZo2004] Safford,D., Zohar, M.(2004) A Trusted Linux Client (TLC), T.J. Watson Research Center IBM, http://www.research.ibm.com/gsal/tcpa/tlc.pdf

[sHype] IBM sHype security hypervisor, 30-Aug-2007,

http://www.research.ibm.com/secure_systems_department/projects/hypervisor/

[Smal2000] Smalley, S.(2000) Flask: Flux Advanced Security Kernel, http://www.cs.utah.edu/flux/flask/

[Smal2000] Smalley, S.(2000) The Distributed Trusted Operating System (DTOS) HomePage, http://www.cs.utah.edu/flux/dtos/

[Snek2001] Snekkenes, E. (2001) Concepts for personal location privacy policies, 3rd ACM Conference on Electronic Commerce, Tampa, Florida, USA, pp. 48-57.

[Spaf1989] Spafford, G. (1989) Quotable Spaf, Gene Spafford's Personal Pages,

http://homes.cerias.purdue.edu/~spaf/quotes.html.

[Spar2000] SPARTA - information Systems Security Operations (2000) LOMAC: MAC You Can Live With, http://www.isso.sparta.com/opensource/lomac/index.html

[SSLHAL1999] Spencer, R., Smalley, S., Loscocco, P., Hibler, M., Andersen, D., Lepreau, J. (1999) The Flask Security Architecture: System Support for Diverse Security Policies, Proceedings of the Eighth USENIX Security Symposium, Aug. 1999, pp. 123-139.

[TCG0001] The Trusted Computing Group, 10-Aug-2007,

http://www.trustedcomputinggroup.org/.

[TCSEC1985] United States Department of Defense (1985) Trusted Computer System Evaluation Criteria, DoD Standard 5200.28-STD, Dec-1985.

[Wiki0001] Mandatory Access Control, Wikipedia, 08-Aug-2007,

http://de.wikipedia.org/wiki/Mandatory_Access_Control.

[Wiki0002] Discretionary Access Control, Wikipedia, 08-Aug-2007,

http://de.wikipedia.org/wiki/Discretionary_Access_Control.

[Wiki0003] Role Based Access Control, Wikipedia, 08-Aug-2007,

http://de.wikipedia.org/wiki/Role_Based_Access_Control

[Wiki0004] Hypervisor, Wikipedia, 09-Sep-2007, http://en.wikipedia.org/wiki/Hypervisor

[Wiki0007] Low watermark (computer security), Wikipedia, 15-Nov-2007, http://en.wikipedia.org/wiki/Low_watermark_(computer_security)

Index

Glossary

Access Right

The access right is the permission to access (read and/or write) data in a particular domain.

Attestation

When referring to the attestation of a compartment A, it is meant that the configuration of A and its trusted computing base (TCB) are proven to be rooted in hardware RTM. Note that attestation only makes sense if the challenger can check the cryptographic proof (e.g., a signature). This requires the challenger to either be located on a remote trusted platform or that its integrity can be checked locally by means of sealing.

Attestation Identity Key (AIK)

An AIK is a non-migratable signature key that is created by, and provides pseudonymity of, a Trusted Platform Module (TPM). The public portion of an AIK is certified by a Privacy Certification Authority (Privacy CA) stating that this signature key is truly under the control of a TPM. The version 1.2 of the TPM specification defines a cryptographic protocol called Direct Anonymous Attestation (DAA), allowing a TPM to create AIKs without interaction with a Privacy CA.

Attestation Kernel

An attestation kernel is a low-level operating system kernel that mainly provides an interface for integrity checks.

Authorization Data

Data that is solely used for authentication purposes (e.g., a password).

Basic Input Output System (BIOS)

The BIOS is the code on a PC platform that initializes memory and hardware devices.

Binding

Encrypting data to a platform configuration such that only the configuration defined during the binding process can be used to decrypt the data.

Certificate Authority (CA)

A CA is an authority that certifies particular statements.

Certified Migratable Keys (CMK)

Introduced in version 1.2 of the TCG specification, this type of encryption key allows for more flexibility in the handling of keys. The decision to migrate a key, and the migration itself, is delegated to two trusted entities chosen by the owner of the TPM upon creation of the CMK using a certificate of that trusted entity.

Compartment

A compartment is a software component that runs in logical, or even physical, isolation from other software components. Sometimes, the term *Partition* is also used for a compartment.

Container

An object, for example a file, used to store information;

Core Root of Trust for Measurement (CRTM)

The CRTM is a PC component specified by the TCG that measures the BIOS before executing it. See also Root of Trust for Measurement (RTM).

Direct Anonymous Attestation (DAA)

A cryptographic protocol developed in the context of the TCG specification to avoid third parties link transactions to a certain platform; DAA eliminates the need for a Trusted Third Party (TTP) by using a zero-knowledge protocol.

Domain

A domain is a set of compartments that are granted an equal level of trust.

Dynamic Root of Trust for Measurement (D-RTM)

The D-RTM is an RTM, supported by Intel's TXT or AMD's Presidio hardware extension. Dynamic loadable code, running in an isolated environment of the CPU, is used as root for a chain of trust. Integrity measurement results of D-RTM are securely stored into a TPM.

Endorsement Key (EK)

An EK is an asymmetric 2048-bit RSA-Encryption key, which is unique for every TPM. The EK resides inside the TPM permanently and can be used to authenticate a TPM and its platform.

Secure Communication Network

Secure network connections are using protocols such as SSL, TLS, or IPSec to fulfill security requirements and to realize virtual private networks (VPN).

General Public License (GPL)

The most widely used license for FLOSS (Free, Libre and Open Source Software).

GNU's Not UNIX (GNU)

The GNU Project was launched in 1984 to develop a complete Unix-like operating system that is based on FLOSS. Variants of the GNU operating system, which use the operating system kernel called Linux, are now widely used; though these systems are often referred to as "Linux", they are more accurately called GNU/Linux systems. GNU is a recursive acronym for "GNU's Not Unix"; it is pronounced 'guh-noo'. For more information, see http://www.gnu.org.

Grand Unified Bootloader (GRUB)

GRUB is a boot loader of the GNU project implementing the multiboot specification allowing users to have several different operating systems on their computer at once.

Hardware

The hardware includes all physical functions of a computer system (e.g., the CPU).

Hypervisor

See Virtual Machine Monitor.

Integrity Measurement Architecture (IMA)

IMA is an architecture developed by IBM that generates verifiable representative information about the software stack running on a GNU/Linux operating system. IMA can be used by remote parties to determine the integrity of a GNU/Linux System during runtime.

Channel

A channel is a means of communication between compartments. Our security model distinguishes between secure, trusted and plain channels. A plain channel does not provide any security over the data communicated. A secure channel ensures the confidentiality and integrity of the communicated data as well as the authenticity of the channel end point. A trusted channel is a channel that extends the verification of authenticity by also validating the configuration of the channel end point.

Configuration

An unambiguous description of the behavior of a software component that is based on its instruction set, internal state and the configuration of the underlying platform.

Linux

A term used to identify the Linux kernel; usually mentioned together with a version number. Example: Linux-2.6.22.2. Often, the term Linux includes the whole GNU/Linux operating system.

Mandatory Access Control (MAC)

Refers to a means of access control in computer security that is defined by the Trusted Computer System Evaluation Criteria (often referred to as "The Orange Book by the Department of Defense") as "a means of restricting access to objects based on the sensitivity (as represented by a label) of the information contained in the objects and the formal authorization (i.e. clearance) of subjects to access information of such sensitivity".

Migratable Key (MK)

Cryptographic encryption keys used by a TPM that are only trusted by the party that generates them (e.g., the user of a platform); offers no guarantee to a third party that such a key has indeed been generated on a TPM.

Migration Authority (MA)

Migration Authority Handles the migration of a CMK; see also MSA.

Migration Selection Authority (MSA)

The MSA is the party that controls the migration of a CMK; see also MA.

Non-Migratable Key (NMK)

Contrary to a migratable key, a non-migratable encryption key is guaranteed to reside in a TPM. A TPM can create a certificate stating that a key is a Non-Migratable Key.

Platform Configuration Register (PCR)

PCRs are the registers of a TPM that store the configuration of the software components.

Platform

The TCB’s underlying hardware.

Privacy Certification Authority (Privacy CA)

A Trusted Third Party (TTP) that certifies the trustworthiness of an AIK.

Program

A program is a binary representation (file) of an executable compartment.

Root of Trust for Measurement (RTM)

The Root of Trust for Measurement is the basis for creating the integrity measurements of a platform. The RTM is the first element in the chain of trust, mostly proceeding with boot loader and kernel, leading to the operating system and applications. On startup, the code of the RTM, which is part of the BIOS, is executed and generates measurement values covering the components of the boot process.

Sealing

Sealing means to encrypt data with the public portion of a TPM-protected storage key. In addition, it is possible to bind the data to a set of platform metrics, i.e. the platform can only unseal the data if it is in the defined state represented by the PCR values. Sealing and

unsealing can only be successful on the platform providing the TPM.

Secure Boot

Secure boot is a security property of a bootstrap architecture that only bootstraps a pre-defined configuration of software components. If the software has been modified, the bootstrap process is interrupted.

Secure Container

A secure container is a container providing certain security properties, such as integrity, confidentiality and freshness.

Smartcard

A card made from plastic which has an integrated chip providing cryptographic functions, storage, and a microprocessor.

Software

Software is the executable program code, for example an application.

Static Root of Trust for Measurement (S-RTM)

The Static Root of Trust for Measurement (S-RTM) is the static RTM. The integrity measurements of the S-RTM are stored in PCRs 0 - 15 and 23.

Storage Root Key (SRK)

A SRK is an asymmetric 2048-Bit RSA key stored inside the TPM, which is used to encrypt TPM-internal data. The SRK is created by taking ownership of the TPM and resides permanently until the owner is cleared.

TCG Software Stack (TSS)

The software stack specified by the TCG that allows accessing and using the TPM.

Trusted Channel

A trusted channel is a communication channel that provides integrity, confidentiality, and authenticity (see "Secure Communication Channel"). In addition, it includes information about the behavior of the communicating endpoints.

Trusted Computing (TC)

Components and mechanisms that have been defined in order to increase the trustworthiness of IT systems.

Trusted Computing Base (TCB)

The part of a computer system that can violate a defined security policy.

Trusted Computing Group (TCG)

An industry consortium defining several specifications required to build a trustworthy computing platform, including the TPM specification, the TCG “TCG Software Stack” (TSS) specification, and the “Trusted Network Connect” specification.

Trusted Computing Platform Alliance (TCPA)

TCPA is the former name of the TCG.

Trusted Network Connect (TNC)

The TNC architecture focuses on security solutions to network access control with the help of Trusted Computing. Integrity measurements are exchanged as security proofs between the communicating endpoints, and are verified before access is allowed to the network, or access to data in the network.

Trusted Platform Module (TPM)

A hardware device, protected against manipulation and designated for opt-in usage, providing protected capabilities and shielded

locations. The TPM is a passive component and contains engines for random number generation, calculation of hash values and key generation. A TPM generates and stores keys, signs or binds data to the platform and measures the platform's current state.

Trusted Third Party (TTP)

The TTP is a party that has to be trusted by all other participants of a protocol.

Insecure Communication Network

Insecure public or private networks used for communication of computer systems.

Virtual Machine Monitor (VMM)

A virtual machine monitor (also called hypervisor) generates and manages virtual machines (VMs). It distributes the resources of the underlying hardware such that the resources are available for the operation of each virtual machine.

Virtual Machine (VM)

A VM is a software implementation of a machine or a computer that behaves like a physical machine from the operating systems perspective. Virtual Machines need the presence of a software layer (Virtual Machine Monitor (VMM)) in order to access the multiplexed physical hardware.

Chapter 8

Appendix

8.1 Creative Commons License

Creative Commons

Creative Commons Legal Code

Attribution-NoDerivs 3.0 Unported

http://creativecommons.org/licenses/by-nd/3.0/legalcode

CREATIVE COMMONS CORPORATION IS NOT A LAW FIRM AND DOES NOT PROVIDE LEGAL SERVICES. DISTRIBUTION OF THIS LICENSE DOES NOT CREATE AN ATTORNEY-CLIENT RELATIONSHIP. CREATIVE COMMONS PROVIDES THIS INFORMATION ON AN "AS-IS" BASIS. CREATIVE COMMONS MAKES NO WARRANTIES REGARDING THE INFORMATION PROVIDED, AND DISCLAIMS LIABILITY FOR DAMAGES RESULTING FROM ITS USE.

License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED. BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. TO THE EXTENT THIS LICENSE MAY BE CONSIDERED TO BE A CONTRACT, THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions

a. **"Adaptation"** means a work based upon the Work, or upon the Work and other pre-existing works, such as a translation, adaptation, derivative work, arrangement of music or other alterations of a literary or artistic work, or phonogram or performance and includes cinematographic adaptations or any other form in which the Work may be recast, transformed, or adapted including in any form recognizably derived from the original, except that a work that constitutes a Collection will not be considered an Adaptation for the purpose of this License. For the avoidance of doubt, where the Work is a musical work, performance or phonogram, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered an Adaptation for the purpose of this License.

b. **"Collection"** means a collection of literary or artistic works, such as encyclopedias and anthologies, or performances, phonograms or broadcasts, or other works or subject matter other than works listed in Section 1(f) below, which, by reason of the selection and arrangement of their contents, constitute intellectual creations, in which the Work is included in its entirety in unmodified form along with one

or more other contributions, each constituting separate and independent works in themselves, which together are assembled into a collective whole. A work that constitutes a Collection will not be considered an Adaptation (as defined above) for the purposes of this License.

c. **"Distribute"** means to make available to the public the original and copies of the Work through sale or other transfer of ownership.

d. **"Licensor"** means the individual, individuals, entity or entities that offer(s) the Work under the terms of this License.

e. **"Original Author"** means, in the case of a literary or artistic work, the individual, individuals, entity or entities who created the Work or if no individual or entity can be identified, the publisher; and in addition (i) in the case of a performance the actors, singers, musicians, dancers, and other persons who act, sing, deliver, declaim, play in, interpret or otherwise perform literary or artistic works or expressions of folklore; (ii) in the case of a phonogram the producer being the person or legal entity who first fixes the sounds of a performance or other sounds; and, (iii) in the case of broadcasts, the organization that transmits the broadcast.

f. **"Work"** means the literary and/or artistic work offered under the terms of this License including without limitation any production in the literary, scientific and artistic domain, whatever may be the mode or form of its expression including digital form, such as a book, pamphlet and other writing; a lecture, address, sermon or other work of the same nature; a dramatic or dramatico-musical work; a choreographic work or entertainment in dumb show; a musical composition with or without words; a cinematographic work to which are assimilated works expressed by a process analogous to cinematography; a work of drawing, painting, architecture, sculpture, engraving or lithography; a photographic work to which are assimilated works expressed by a process analogous to photography; a work of applied art; an illustration, map, plan, sketch or three-dimensional work relative to geography, topography, architecture or science; a performance; a broadcast; a phonogram; a compilation of data to the extent it is protected as a copyrightable work; or a work performed by a variety or circus performer to the extent it is not otherwise considered a literary or artistic work.

g. **"You"** means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

h. **"Publicly Perform"** means to perform public recitations of the Work and to communicate to the public those public recitations, by any means or process, including by wire or wireless means or public digital performances; to make available to the public Works in such a way that members of the public may access these Works from a place and at a place individually chosen by them; to perform the Work to the public by any means or process and the communication to the public of the performances of the Work, including by public digital performance; to broadcast and rebroadcast the Work by any means including signs, sounds or images.

i. **"Reproduce"** means to make copies of the Work by any means including without limitation by sound or visual recordings and the right of fixation and reproducing fixations of the Work, including storage of a protected performance or phonogram in digital form or other electronic medium.

2. Fair Dealing Rights. Nothing in this License is intended to reduce, limit, or restrict any uses free from copyright or rights arising from limitations or exceptions that are provided for in connection with the copyright protection under copyright law or other applicable laws.

3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

a. to Reproduce the Work, to incorporate the Work into one or more Collections, and to Reproduce the Work as incorporated in the Collections; and,

b. to Distribute and Publicly Perform the Work including as incorporated in Collections.

c. For the avoidance of doubt:

i. **Non-waivable Compulsory License Schemes**. In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme cannot be waived, the Licensor reserves the exclusive right to collect such royalties for any exercise by You of the rights granted under this License;

ii. **Waivable Compulsory License Schemes**. In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme can be waived, the Licensor waives the exclusive right to collect such royalties for any exercise by You of the rights granted under this License; and,

iii. **Voluntary License Schemes**. The Licensor waives the right to collect royalties, whether individually or, in the event that the Licensor is a member of a collecting society that administers voluntary licensing schemes, via that society, from any exercise by You of the rights granted under this License.

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats, but otherwise you have no rights to make Adaptations. Subject to Section 8(f), all rights not expressly granted by Licensor are hereby reserved.

4. Restrictions. The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

a. You may Distribute or Publicly Perform the Work only under the terms of this License. You must include a copy of, or the Uniform Resource Identifier (URI) for, this License with every copy of the Work You Distribute or Publicly Perform. You may not offer or impose any terms on the Work that restrict the terms of this License or the ability of the recipient of the Work to exercise the rights granted to that recipient under the terms of the License. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties with every copy of the Work You Distribute or Publicly Perform. When You Distribute or Publicly Perform the Work, You may not impose any effective technological measures on the Work that restrict the ability of a recipient of the Work from You to exercise the rights granted to that recipient under the terms of the License. This Section 4(a) applies to the Work as incorporated in a Collection, but this does not require the Collection apart from the Work itself to be made subject to the terms of this License. If You create a Collection, upon notice from any Licensor You must, to the extent practicable, remove from the Collection any credit as required by Section 4(b), as requested.

b. If You Distribute, or Publicly Perform the Work or Collections, You must, unless a request has been made pursuant to Section 4(a), keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or if the Original Author and/or Licensor designate another party or parties (e.g., a sponsor institute, publishing entity, journal) for attribution ("Attribution Parties") in Licensor's copyright notice, terms of service or by other reasonable means, the name of such party or parties; (ii) the title of the Work if supplied; (iii) to the extent reasonably practicable, the URI, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work. The credit required by this Section 4(b) may be implemented in any reasonable manner; provided, however, that in the case of a Collection, at a minimum such credit will appear, if a credit for all contributing authors of the Collection appears, then as part of these credits and in a manner at least as prominent as the credits for the other contributing authors. For the avoidance of doubt, You may only use the credit required by this Section for the purpose of attribution in the manner set out above and, by exercising Your rights under this License, You may not implicitly or explicitly assert or imply any connection with, sponsorship or endorsement by the Original Author, Licensor and/or Attribution Parties, as appropriate, of You or Your use of the Work, without the separate, express prior written permission of the Original Author, Licensor and/or Attribution Parties.

c. Except as otherwise agreed in writing by the Licensor or as may be otherwise permitted by applicable law, if You Reproduce, Distribute or Publicly Perform the Work either by itself or as part of any Collections, You must not distort, mutilate, modify or take other derogatory action in relation to the Work which would be prejudicial to the Original Author's honor or reputation.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability

EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Collections from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.

b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

a. Each time You Distribute or Publicly Perform the Work or a Collection, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.

b. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

c. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.

d. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

e. The rights granted under, and the subject matter referenced, in this License were drafted utilizing the terminology of the Berne Convention for the Protection of Literary and Artistic Works (as amended on September 28, 1979), the Rome Convention of 1961, the WIPO Copyright Treaty of 1996, the WIPO Performances and Phonograms Treaty of 1996 and the Universal Copyright Convention (as revised on July 24, 1971). These rights and subject matter take effect in the relevant jurisdiction in which the License terms are sought to be enforced according to the corresponding provisions of the implementation of those treaty provisions in the applicable national law. If the standard suite of rights granted under applicable copyright law includes additional rights not granted under this License, such additional rights are deemed to be included in the License; this License is not intended to restrict the license of any rights under applicable law.

Creative Commons Notice

Creative Commons is not a party to this License, and makes no warranty whatsoever in connection with the Work. Creative Commons will not be liable to You or any party on any legal theory for any damages whatsoever, including without limitation any general, special, incidental or consequential damages arising in connection to this license. Notwithstanding the foregoing two (2) sentences, if Creative Commons has expressly identified itself as the Licensor hereunder, it shall have all rights and obligations of Licensor.

Except for the limited purpose of indicating to the public that the Work is licensed under the CCPL, Creative Commons does not authorize the use by either party of the trademark "Creative Commons" or any related trademark or logo of Creative Commons without the prior written consent of Creative Commons. Any permitted use will be in compliance with Creative Commons' then-current trademark usage guidelines, as may be published on its website or otherwise made available upon request from time to time. For the avoidance of doubt, this trademark restriction does not form part of this License.

Creative Commons may be contacted at http://creativecommons.org/**.**