Disposition 5: System Security Mechanisms

Mathias Ravn Tversted

January 6, 2020

Table of contents

Trusted Computing Base

A *Trusted Computing Base* is a part or parts of a system that is trusted to perform according to specification. This can be established using hardware, or using software security depending on the threat model. It can also be in several places. An example of this is SGX. A system on chip that is protected is called a *secure enclave*. An enclave can be used to handle keys and or sign things. It can also run code. It will often be a part of the OS.

Firewalls

Even if a computer insists on authentication, it still has to perform untrusted communication momentarily. This opens it up to attack. Generally we have the following roles:

- **Packet filters**: Sits on network, filters communication, blocks packages to machine et cetera. TCP packages have flags that can be recognised etc.
- Proxies:
- Stateful Firewalls

Packet filtering

Packet filters can look at packages can determine if they should be allowed through. There is a tradeoff between being too nazi, and being *too* permissive. The packet filter needs to be smart and do thorough analysis in order to be effective

Proxies

A proxy firewall does not allow clients to make connections, but rather it handles connections *on behalf of the machines*. This hides the local network from the outside. Downside is that programs need to be configured to use proxies. Proxy firewalls are regarded as being quite secure.

Stateful firewalls

A *stateful firewall* keeps track of all the current connections that goes throuh it. It checks to see if the traffic going through it is allowed or not. Flexibility is needed for UDP packets. A stateful firewall can also translate local addresses to something else, this is known as a *masquerading firewall*, can help protect against IP address scanning. It can also scan for suspicious behavior, and block that traffic altogether.

Malware

Malware comes in different flavours, such as

- **Trojan Horses**: Appear to be useful, but have hidden intentions. Such as being spyware or destroying or stealing data
- **Viruses**: Infect programs and spread themselves to other machines
- **Worms**: Similiar to viruses, but are stand-alone programs
- **Ransomware**: Encrypts your files or similiar, and demands money, usually in cryptocurrencies. May initially act as trojans and may want to spread themselves (see WANNACRY)

Virus Scanners

Virus scanners scan your machine for malicious software. Many users do not update their scanners and attacks happen due to old scanners, software/hardware and not necessarily because of new attacks. (See WANNACRY infecting old computers). Malware may try to encrypt itself, or have barebones infrastructure that loads the actual malicious code into memory.

Intrusion Detection

Intrusion Detection is the practice of detecting malicious behaviour rather than particular pieces of malicious software. This happens by monitoring processes, and or users. An example of this, is a stateful firewall. There are two approaches: *Rule-based* and *statistical* intrusion detection. **Rule-based**: System sets up rules for normal behaviour **Statistical**: First gather statistics on normal behaviour, detect outliers. Intrusion detection is also a tradeoff between being eager and being permissive. A system may use *honey pots*.

Security Policies

We usually organise the way we think of security in the following ways

- **Security Policy**: Specification of the system in question. Description of security objects, and maybe high-level strategy for achieving the objectives
- **Threat Model**: Descriptions of the attacks we want to protect ourselves against
- **Security mechanisms**: The technical solutions we use to reach our objectives

Lattice Model

A lattice consists of a finite set S and a relation $\leq$. For any elements $a, b, c \in S$ it must hold that

- $a \leq a$
- $a \leq b \wedge b \leq a \implies a = b$
- $a \leq b \wedge b \leq c \implies a \leq c$

S is the set of privilees and rights. A relation $a \leq b$ means that a has at least as many rights as b. A lattice does not require that the relation is defined for all pairs. For it to be a lattice, it is also required that $a, b \in S$ we have a *greatest lower bound*, for $a, b, c \in S$, such that $c \leq a \wedge c \leq b$ and for any other c', we have $c' \leq c$. There exists some *least upper bound*, for $a, b, d \in S$ such that $a \leq d, b \leq d$, for any other d', we have $d \leq d'$.

Using the Lattice model

There are two ways of using the Lattice model to organise security policy: **Subjects**: Define *subjects* and *objects*. Classify subjects and objects and position them in the lattice. A subject s with classification $C(s)$ may do a certain operation on object o iff $C(s) \geq C(o)$ or iff $C(s) \leq C(o)$. **Information flow**: Assign a position to different parts of the system, and let the relation be information flow. Higher position means "more secret". You could also do this for authenticity, then it becomes "more reliable"

Bell-Lapadula Model and Biba model

An example of the Lattice model, is the *Bell-Lapadula Model* convinced for military applications. Consists of security levels that are linearly ordered, such as $public \leq secret \leq topsecret$. The model then defines two rules (sometimes known as simple security property, and the * property). May be unwieldy in practice.

- **No read up**: Subject s may read from object o iff $C(s) \geq C(o)$
- **No write down**: Subject s may write to object o iff $C(s) \leq C(o)$

Another model is the *Biba model*, which focuses on integrity. The higher levels are the ones that contain the most reliable inforamtion. In order to maintain

Other models

- **Chinese Wall**: Creates a wall between different subjects. There is defined a relation which says if someone is allowed to access something at all. Such as people who compete in a business setting.
- **Prevent-Detect-Recover**: Splits policy into 3 parts. One describes how to prevent attacks, one describes how to detect them, and one describes how to recover
- **Seperation of Duty**: Comes in two flavours, *Dual Control* and *Functional Separation*. Dual Control says that actions are possible if multiple people have approved it. Functional seperation is where an action requires multiple people at

Access Control

The most generic way is an *Access Control Matrix*. Let A be a matrix with a row for each subject s and a column for each object o, then $A[s, o]$ is a list of permitted operations for s on o. There are two approaches to doing this in practice

- **Access Control Lists**. Stored with each object, is a column assigned to the object. Called an access control list. Makes it hard to determine the total permissions of each subject. This is the approach of UNIX systems.
- **User Capabilities**: We store for each subject the row assignedto the subject. Called user capabilities. Makes it easy to determine permissions for users, but not for objects.

Dynamic Access Control

Who can update the ACM? There is *mandatory access control*, where subjects cannot the matrix. There is *discretionary access control*, where subjects have the power to update parts of the matrix. We can talk about *primitive operations* on access control matrices. These are create/delete subject, add/delete access right r. Assume a fixed finite number of access rights. We can now define a *command* with the following format Let $C(\text{parameter list } l) = \{r_1 \in A[s_1, o_1], ..., r_n \in A[s_n, o_n]\}$ followed by a list of primitive operations. This corresponds to the list of operations will be executed if the if the conditions are satisfied.

Undecidability and safety

Consider a fixed set of commands $com_1, ..., c_t$ and an initial ACM A. For a given access right r, does there exist a set of commands that will transform A into A' where $r \in A'[s, o]$ for s, o but $r \notin A[s, o]$? If not, then A is *safe* w.r.t to r. The following properties are provable

- If commands contains more than one operation, then it is undecidable if A is safe w.r.t to r
- If all commans only contain a single primitive operation, it is decidable if A is safe w.r.t to r
- If there a fixed upper bound ahead of time on the number of subjects and objects, then it is decidable if A is safe w.r.t to r

The undecidability, is that the states of A can be