Computer Security and Privacy
Principles of computer security (part I)

Carmela Troncoso
SPRING Lab
carmela.troncoso@epfl.ch

Some slides/ideas adapted from: Philippe Oechslin, George Danezis, Emiliano de Cristofaro, Gianluca Stringhini

Basic principles to build Security Mechanisms

READING: J. Saltzer and M. Schroeder. *The Protection of Information in Computer Systems*. Fourth ACM Symposium on Operating Systems Principles (October 1973)
(Intro & Section 1)

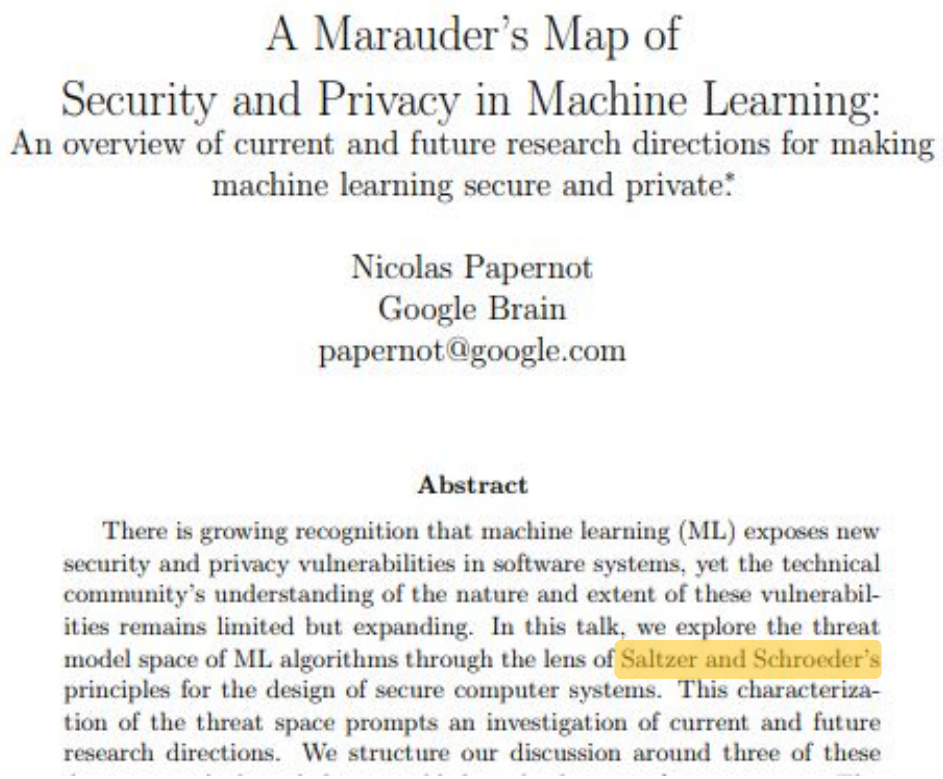
8 + 2 principles at the core of security engineering practices

*"Principles **guide** the design and contribute to an implementation without security flaws"*

https://www.acsac.org/secshelf/papers/protection_information.pdf

Why should you care about principles from 1973

READING: J. Saltzer and M. Schroeder. *The Protection of Information in Computer Systems*. Fourth ACM Symposium on Operating Systems Principles (October 1973)
(Intro & Section 1)



A Marauder's Map of
Security and Privacy in Machine Learning:
An overview of current and future research directions for making machine learning secure and private*

Nicolas Papernot
Google Brain
papernot@google.com

Abstract

There is growing recognition that machine learning (ML) exposes new security and privacy vulnerabilities in software systems, yet the technical community's understanding of the nature and extent of these vulnerabilities remains limited but expanding. In this talk, we explore the threat model space of ML algorithms through the lens of Saltzer and Schroeder's principles for the design of secure computer systems. This characterization of the threat space prompts an investigation of current and future research directions. We structure our discussion around three of these

Keynote at Workshop on Artificial Intelligence and Security

2018

1 - Economy of mechanism

“Keep the [security mechanism / implementation] design as simple and small as possible”

Why?

It needs to be easy to audit and verify.

(operational testing is not appropriate to evaluate security)

[Penetration testing is valuable]

“Trusted Computing Base” (TCB): Every component of the system on which the security policy relies upon

The "Trusted Computing Base" (TCB)

Every component of the system on which the security policy relies.

Hardware / firmware / software

The TCB is **trusted** to operate correctly for the security policy to hold

The only proper use of the verb "to trust" in Security Engineering:

"X trusts Y will do Z"

If something goes wrong within the TCB **the security policy may be violated**

…and if something goes wrong outside the TCB?

The TCB must be kept small to ***ease verification*** (economy of mechanism) and ***diminish the attack surface***

2 – Fail-safe defaults

***"Base access decisions on permission rather than exclusion"*[SS75]**

If something fails, be as secure as if it does not fail

→ errors / uncertainty should err on the side of the security policy

Do not try to fix!! (e.g., automated doors: if they cannot close, stay open)

Whitelist, do not blacklist

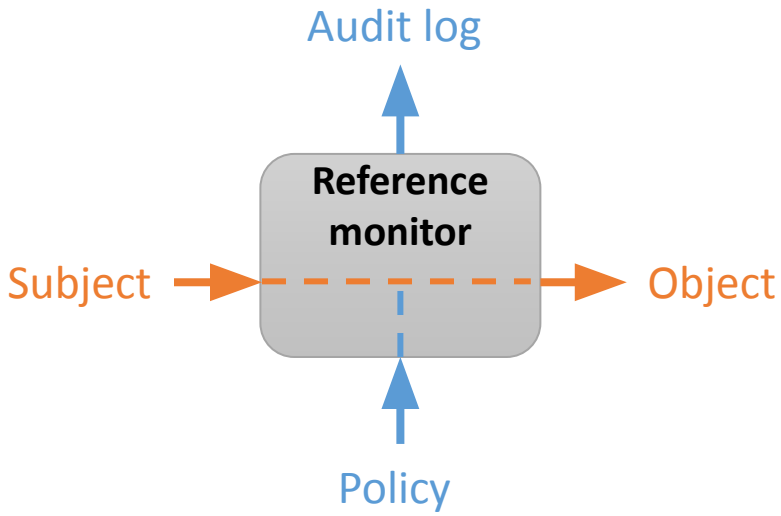
→ lack of permission is easy to detect and solve

Examples:

- Security door: if no permission, do not open
- Form input: if no permission to write in X, do not write anywhere

3 – Complete mediation

“Every access to every object must be checked for authority” [SS75]



mediates ALL actions from subjects on objects and ensures they are according to the policy

Difficult to implement

- Performance?
  - Checking everything is slooooooow
- Time to check vs. time to use
- Modern distributed systems
  - You can only check what you see!

4 – Open design

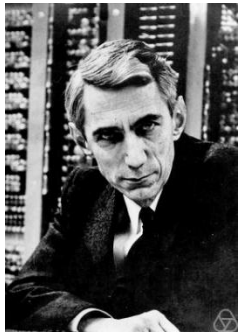
“The design should not be secret” [SS75]

Kerckhoff
La Cryptographie Militaire
(*1883*)

“The design of a system should not require secrecy”

"The enemy knows the system"

"one ought to design systems under the assumption that the enemy will immediately gain full familiarity with them”



Shannon
Communication Theory of Secrecy Systems
(*1949*)

Baran
Security, secrecy, and tamper-free considerations
(*1964*)

“The Paradox of the Secrecy About Secrecy”

“Without the freedom to expose the system proposal to widespread scrutiny' by clever minds of diverse interests, is to increase the risk that significant points of potential weakness have been overlooked”

https://www.rand.org/pubs/research_memoranda/RM3765/RM3765.chapter2.html

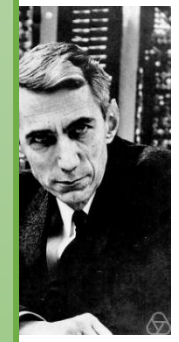
4 – Open design

“The design should not be secret” [SS75]

Kerckh
La Cryptograp
(*188*

When you design… algorithms are public! Only key elements are kept secret

Crypto: only keep the key secret
Authentication: only keep password secret
Obfuscation: only keep the used noise secret



nnon
ry of Secrecy Systems
(*1949*)

“The Paradox of the Secrecy About Secrecy”

Baran
Security, secrecy, and tamper-free considerations
(*1964*)

https://www.rand.org/pubs/research_memoranda/RM3765/RM3765.chapter2.html

4 – Open design

“The design should not be secret” [SS75]

Open design results in better & easier auditing

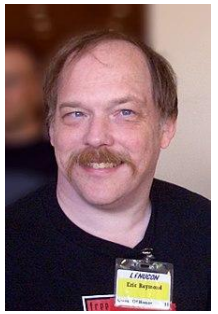
Linus’ law: *"given enough eyeballs, all bugs are shallow"*

Secrecy is unrealistic!!

Way to build a bad threat model!

Famous failures closed design:

- DVD encryption
- GSM encryption



Raymond

The Cathedral and the Bazaar (*1997*)

Key principle behind the academic discipline devoted to understanding computer security

5 – Separation of privilege

“No single accident, deception, or breach of trust is sufficient to compromise the protected information” [SS75]

*A **privilege** allows a user to perform an action on a computer system that may have security consequences, e.g., create a file in a directory, access a device, write to a socket for communicating over the Internet.*

Require multiple conditions to execute an action improves security

Examples: two keys to open a safe, two-factors to authenticate

Problems

- Availability?
- Responsibility?
- Complexity!

Recap

Economy of mechanism. Keep it simple!

Fail-safe defaults. If there is a problem, your move should comply with the policy

Complete mediation. Verify *every* action

Open Design. Make the design (and implementation) of your mechanism available

Separation of Privilege. Try to never rely on *only one* entity or action

6 – Least privilege

“Every program and every user of the system should operate using the least set of privileges necessary to complete the job” [SS75]

Rights added as needed, discarded after use

Damage control

- Minimize high privilege actions & interactions

What principle is this related to?

“Need-to-know” principle

- Examples
  - Guest accounts @ EPFL
  - Data minimization principle (Data Protection)

7 – Least common mechanism

“Minimize the amount of mechanism common to more than one user and depended on by all users” [SS75]

“Every shared mechanism represents a potential information path between users and must be designed with great care to be sure it does not unintentionally compromise security”

Remember “Economy of mechanism”

- (Design) Interactions make it hard to validate the security design
- (Implementation) Interactions may lead to unintentional leaks of information
  - Unintended channels: use of /tmp, shared cache

Note that this refers to mechanisms! Not to the code. Reusable code that has been tested many times is helpful for security

8 – Psychological acceptability

“It is essential that the human interface be designed for ease of use, so that users routinely and automatically apply the protection mechanisms correctly” [SS75]

Hide complexity introduced by security mechanisms

- Security mechanisms should not make the resource more difficult to access than if it was not present

Mental model of the (honest) users must match security policy and security mechanisms

Cultural acceptability – not all mechanisms are acceptable everywhere

- (Authentication) Face recognition not suitable in cultures that cover their face
- (Safety) Register of everyone who sleeps in a dorm

Extra principles from physical security
9 - Work factor

DIFFICULT TO TRANSPOSE TO COMPUTER SECURITY!!

"Compare the cost of circumventing the mechanism with the resources of a potential attacker" [SS75]

It helps **refining** the threat model!

Quantifying **cost** is hard**?**

- cost of compromising insiders?
- cost of finding a bug?
- monetization?

Difficult to quantify

Extra principles from physical security
10 - Compromise recording

DIFFICULT TO TRANSPOSE TO COMPUTER SECURITY!!

"Reliably record that a compromise of information has occurred [...] in place of more elaborate mechanisms that completely prevent loss" [SS75]

Keep **tamper-evidence logs,**

they may enable recovery (integrity)

Logs **are not magic**:

What if you cannot recover? (if confidentiality mechanisms were in place)

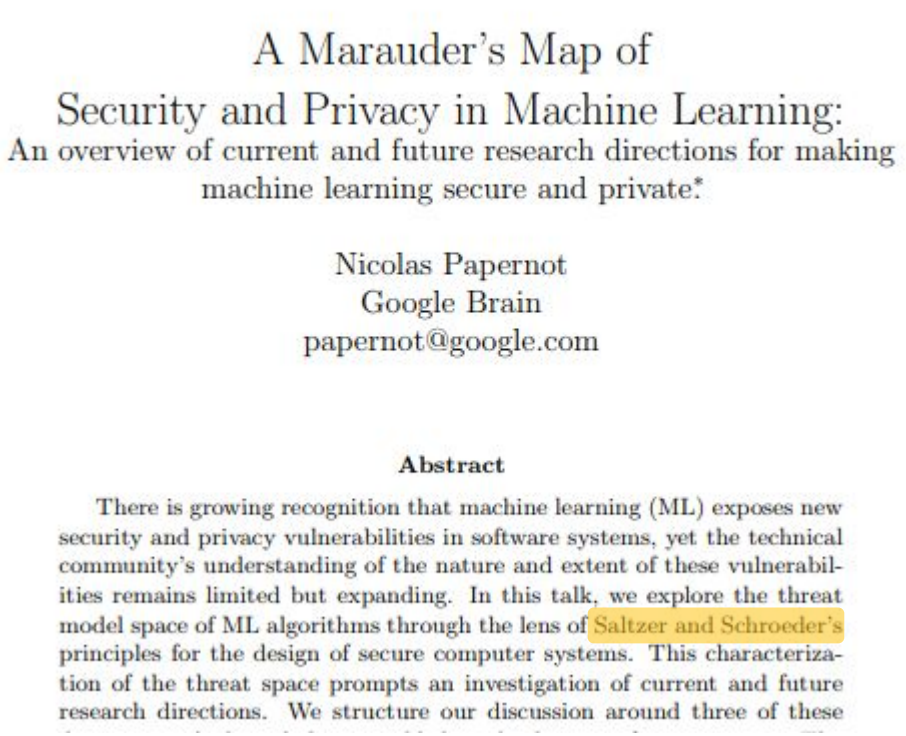
How to keep integrity?

Logs may be a vulnerability (Privacy)?

Logging the log? (Availability)

Logging is not a guarantee that the compromise is detected.

Why principles are important?



A Marauder's Map of
Security and Privacy in Machine Learning:
An overview of current and future research directions for making machine learning secure and private*

Nicolas Papernot
Google Brain
papernot@google.com

Abstract

There is growing recognition that machine learning (ML) exposes new security and privacy vulnerabilities in software systems, yet the technical community's understanding of the nature and extent of these vulnerabilities remains limited but expanding. In this talk, we explore the threat model space of ML algorithms through the lens of Saltzer and Schroeder's principles for the design of secure computer systems. This characterization of the threat space prompts an investigation of current and future research directions. We structure our discussion around three of these

Least privilege. Let the ML learn as little as possible so that information cannot be extracted

Least Common Mechanism. Get samples labelled from different origins

Psychological acceptability. Users must be able to understand why models classify or misclassify an input

Work factor. The cost of the attack, e.g., in terms of number of calls to an API, matters for its relevance

Compromise recording. Ideally we would like to be able to log all steps inside the algorithm

Summary of the lecture

Principles allow us to identify safe and unsafe *patterns* in when designing security mechanisms

- Do not use principles as a blind checklist!
- Use principles as tools to weight design decisions.