Computer Security and Privacy
Principles of computer security

Slides made by Carmela Troncoso, SPRING Lab, additions by Thomas Bourgeat
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*. Proceedings of the IEEE, Invited Paper, (1975)
(Intro & Section 1)

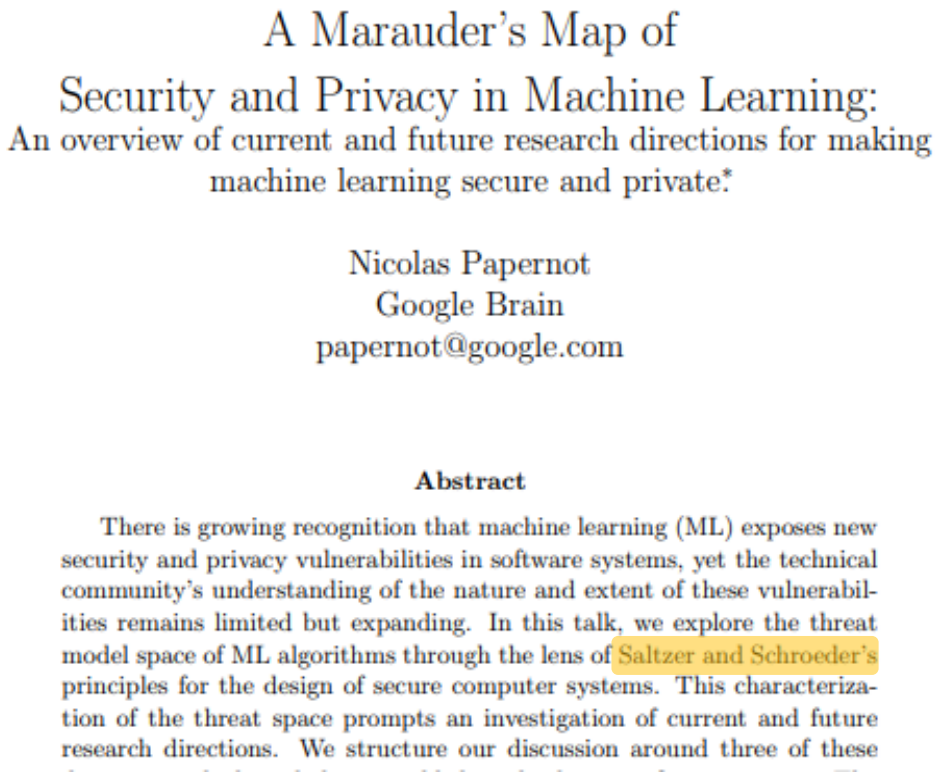
8 (+ 2) principles at the core of security engineering practices

*"[…], experience has provided some useful principles that can **guide** the design and contribute to an implementation without security flaws"*

https://www.cl.cam.ac.uk/teaching/1011/R01/75-protection.pdf

Why should you care about principles from 1975

READING: J. Saltzer and M. Schroeder. *The Protection of Information in Computer Systems*. Proceedings of the IEEE, Invited Paper, (1975)
(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

Heartbleed

The Heartbleed Bug: A "Simple" Flaw

- A critical vulnerability (CVE-2014-0160) reported in 2014 in the OpenSSL library - used to encrypt traffic (HTTPS).

- A feature called the "Heartbeat" allows a client to send a **small piece of data** and ask the server to send it back, just to prove the connection was alive.

- Issue: The attacker could **lie** about the size of their data.
  - Attacker sends 1 byte of data.
  - Attacker *claims* they sent 64,000 bytes.
  - The server, **without checking the actual length**, would reply with the attacker's 1 byte *plus* the next 63,999 bytes of whatever was in its private memory.

The Heartbleed Bug: A "Simple" Flaw

- Feature Complexity:
  - The "Heartbeat" extension was a (very useful) non-essential feature added to the TLS protocol.
  - This *added complexity* created a *new attack surface*. The core function of TLS was secure without it.
- Codebase Complexity:
  - The OpenSSL codebase is massive, and difficult for anyone to understand:

Language	files	blank	comment	code
C	1617	85678	86364	559054
Perl	720	38736	32396	251473

  - The bug was a single missing line of code: a bounds check

```
if (request_size < actual_size) { error; }
```

  - This simple-but-deadly flaw hidden for *two years* because the overall complexity of the code made audit difficult.

Security enthusiasts, check the code! https://github.com/openssl/openssl/commit/731f431497f463f3a2a97236fe0187b11c44aead

1 - Economy of mechanism

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

Why?

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]

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"

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 "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!!

Use **Whitelist**, do not use 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

A 2017 ransomware epidemic

Some of it due to fail-safe defaults!

Ransomware – What is it?

- A Ransomware **denies you access to your own data**, most commonly by **encrypting your files**
- Attacker then demands a **ransom payment** in exchange for the private key required to decrypt your files

- **Some History of Ransomware - 1989**
  - Someone mailed 20,000 infected **5.25-inch floppy disks** (This thing: 💾) to attendees of a WHO AIDS conference. The disks were labeled "AIDS Information Introductory Diskette"
  - After 90 boot, the malware activate. It used simple (broken) encryption to encrypt all *filenames* on the C: drive and hide directories
  - A message appeared on the screen demanding the user send $189 via **postal mail to a P.O. Box in Panama** to receive the decryption key
  - Leaflet with floppy said that the software could *"adversely affect other program applications"* and, *"you will owe compensation and possible damages to PC Cyborg Corporation and your microcomputer will stop functioning normally"*

- Prevalence of ransomware exploded with the arrival of cryptocurrencies

Overall story is disturbing, source: https://cyberalarm.police.uk/blog/?article=2024-12-17-the-first-ransomware, more sources in the notes

The case of MongoDB – Ease of Use Vs Security

- Database software (You will learn about databases in CS-300!). MongoDB is flexible and scalable, storing data in JSON-like documents (CS-214, Week 9).

- For a while, the "out-of-the-box" default installation was wide open on some systems:
  - Authentication was turned OFF by default (no admin password was required).
  - It defaulted to **binding to all network interfaces (0.0.0.0) (CS-202)**, not just the secure localhost (127.0.0.1).

- Consequences: attackers ran scripts that **scanned the entire internet** for open MongoDB instances. The scripts would connect without a password, steal the data, delete/encrypt the originals, and leave a ransom note demanding Bitcoin.
- "[…]on some systems the default configuration has the database listening on a publicly accessible port as soon as it's installed. Users are *supposed to read the manual** and set up access control and authentication after installing the software *but it seems that plenty of them don't***. The result is an internet-connected database with no access control or authentication."

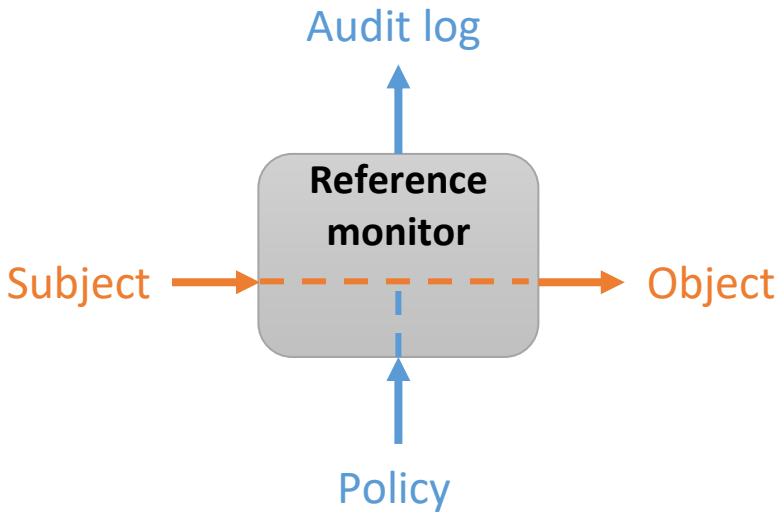
(*) 😂

(**) 

Source: https://news.sophos.com/en-us/2017/01/11/thousands-of-mongodb-databases-compromised-and-held-to-ransom/

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

Scenario

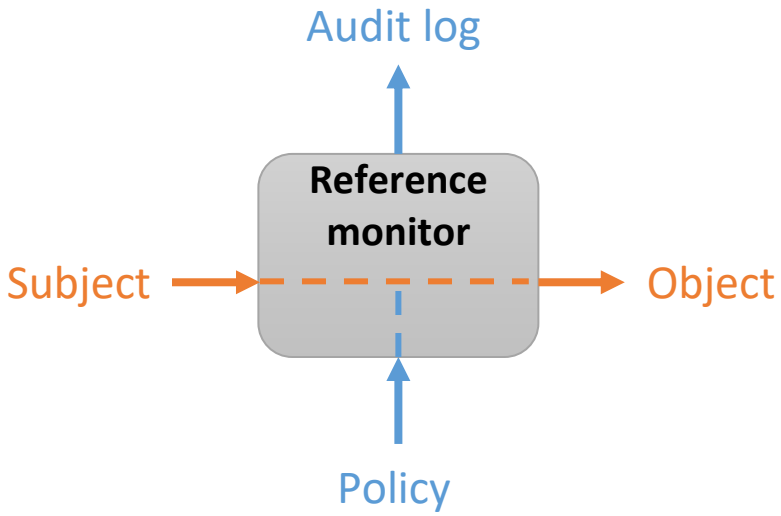
- Imagine you're building a new service like Netflix/Youtube/generic content platform. Users can subscribe or unsubscribe to the "Gold Member“ with one click.
- **Your Goal:** Protect your gold feature/content so only "Gold Members" can see the gated content
- **The "Easy" Logic:** To be fast, your code checks the user's status *only once* at login
  - User logs in
  - Code checks database: "Is user a Gold Member?" -> Yes
  - Code creates a session cookie that says: {'user': 'Alice', 'status': 'paid'}
- **The Flaw:** For every video request after this, your code *only* looks at the cookie, it doesn't re-check the database. *This is a violation of Complete Mediation*

Missing Mediation - An Attack

- Alice pays a Gold Membership for one day, logs in as a "Gold Member." Her session cookie says 'status': 'paid'
- She goes to her account page and clicks "Unsubscribe." The database is correctly updated to "unpaid”
- She then tries to access the privileged content. Your code **only checks the cookie**.
- **Result:** Alice, now an unpaid user, can still watch all premium content until her original session expires.

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

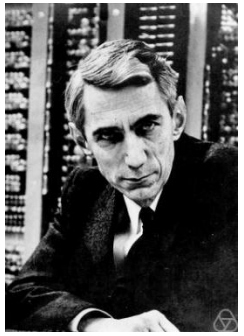
“The design should not be secret” [SS75]

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

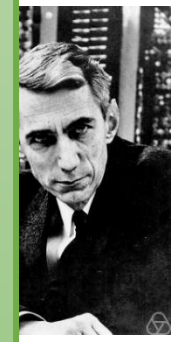
4 – Open design

“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



non
ry of Secrecy Systems
(*1949*)

“The Paradox of the Secrecy About Secrecy”

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

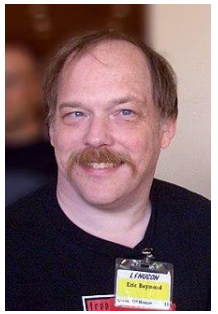
4 – Open design

“The design should not be secret” [SS75]

Open design results in better & easier auditing

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

(More of a mantra, need many eyeballs, defect density is high)



Raymond

The Cathedral and the Bazaar (*1997*)

Secrecy is unrealistic!!

Way to build a bad threat model!

Famous failure of a closed design:

- GSM encryption

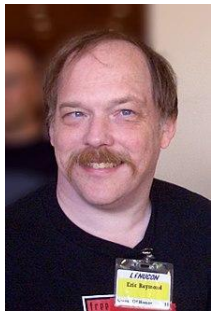
4 – Open design

“The design should not be secret” [SS75]

Open design results in better & easier auditing

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

(More of a mantra, need many eyeballs, defect density is high)



Raymond

The Cathedral and the Bazaar (*1997*)

Secrecy is unrealistic!!

Way to build a bad threat model!

Famous failure of a closed design:

- GSM encryption

Key principle behind the academic discipline devoted to understanding computer security

The “hidden” instance

- In June 2017, the personal data (names, addresses, birthdates, and *political profiling*) of 198 million US voters was found exposed online.
- The 1.1 TB database was compiled by Deep Root Analytics, a data firm contracted by the RNC.
- The server was publicly accessible.
  - It had no password and no authentication required to access it
  - The only thing “protecting” this data was that the URL (dra-dw.s3.amazonaws.com) was "semi-secret“ (“**D**eep **R**oot **A**nalytics, **D**ata **W**arehouse”)
  - Security through Obscurity did not work
- A security researcher discovered the URL, and disclosed the issue and later wrote a detailed report, for a while the entire 1.1 TB database was technically open for anyone on the internet to download.
  - It is unknown if this data was stolen by malicious actors.

Back-of-the-envelope calculation – addresses are not safe

- How many IPv4 addresses are there on the internet (CS-202)?
  - 32-bits addresses (8*4) = 4B possibilities
  - How long does it take to ask these 4B how many of them are hosting a website/running a service on a given port?
    - 3 minutes in 2013*
- Similarly, if you run your own service (website, database, your next great startup idea) :
  - picking a random port (CS-202 ports are numbers between 0-65535) **is not hiding the service**

Source *: https://web.archive.org/web/20250215174330/https://blog.erratasec.com/2013/09/masscan-entire-internet-in-3-minutes.html

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!

Scenario- IT system in a typical business

System: Payment system for a business (order stuff from vendors)

Security Goal: Attacker must not be able to steal money

Principals: Employees (submit order request), Vendors, Managers (Approve new vendor, approve orders, inherits Employee)

Threat Model: Attacker can corrupt *at most one* principal (No collusion threat model)

A *single* corrupted employee can execute the entire attack:

- **Create Vendor:** They add a fake shell company (e.g., " M.T. Pockets Consulting" which the adversary secretly own) as a "New Vendor“
- **Create Order:** Submit fake order requests for this vendor
- **Approve:** Use their own authority to "Approve Expenses"

Solution is to split roles:

- **Role A (Procurement):** Can **ONLY** create/approve new vendors. Cannot approve payments
- **Role B (Finance Manager):** Can **ONLY** approve payments for vendors *already in the system*. Cannot create new vendors

Separation of Privileges is a bit reminiscent of the “Separation of Power” principle or the idea of “checks and balances” in political sciences

Recap

Economy of mechanism. Keep it simple!

Fail-safe defaults. Safe defaults, and 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

“Need-to-know” principle

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

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

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. **The problem is the isolation of data**

CS-200 Throwback

- Who remembers what are symbolic links?

- **Refresher (What is a Symlink):** A symbolic link (or "symlink") is simply a "shortcut" file. When the operating system tries to access the symlink, it is automatically redirected to the *real* file or directory the link is pointing to.

Exercise – Find the vulnerability

Principal 1: A "root/admin" program that predictably writes a log file to a `/tmp/starting.log`. Something like:

```
echo "Service is starting..." > /tmp/starting.log
```

Principal 2: The “root/admin” logging subsystem.

Principal 3: A user that can run nonroot programs. Has write accesses to `/tmp` too.

Security Goal: Preserve log integrity (integrity of the files written by Principal 2)

Threat model: Adversary can corrupt Principal 3

Solution – Violation of the Least common mechanism

The Common Mechanism:

- The `/tmp` directory on a Linux/Unix machine. This directory is shared between low-privilege users and high-privilege 'root' processes. Both can (and do) write temporary files here.

Attacker simply creates a symlink:

```
ln -s /var/log/dmesg /tmp/starting.log
```

Actually it does not work anymore on modern Linux, there is a mitigation for /tmp! But it works if the symbolic link is in /home/user

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

Amazon S3 Buckets

- **S3** stands for **S**imple **S**torage **S**ervice, a cloud storage service from AWS
- Think of it as a limitless "hard drive in the cloud." You store data as "Objects" (files, videos, backups) inside containers called "Buckets“
- It's the backbone of the modern internet, used for everything from hosting website images to storing data and application logs
- Access to S3 buckets is controlled by a sophisticated system of permissions

The Confusion

- Psychological acceptability is not just for end users!
  System administrators are also "users" of a configuration UI

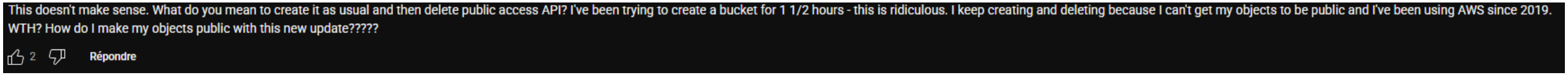
- The S3 bucket permission UI included a potentially misleading option: grant access to **"Any Authenticated AWS User"**
  - **What admins *thought* it meant:** "Any authenticated user *in my account*."
  - **What it *actually* meant:** "ANY person on Earth with ANY AWS account (which is free to create)."

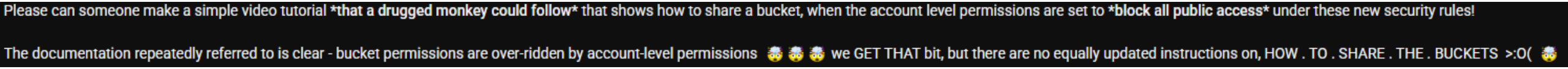
- Consequences: several leaks starting in 2017 (RNC, DowJones, …)

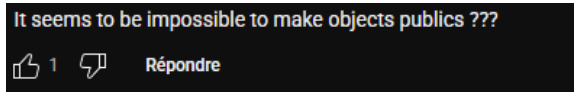
- New layer of default failsafe in 2023
  - https://aws.amazon.com/about-aws/whats-new/2022/12/amazon-s3-automatically-enable-block-public-access-disable-access-control-lists-buckets-april-2023

Safer default have tradeoffs

Youtube comments when Amazon advertised their new policy in a release video







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)

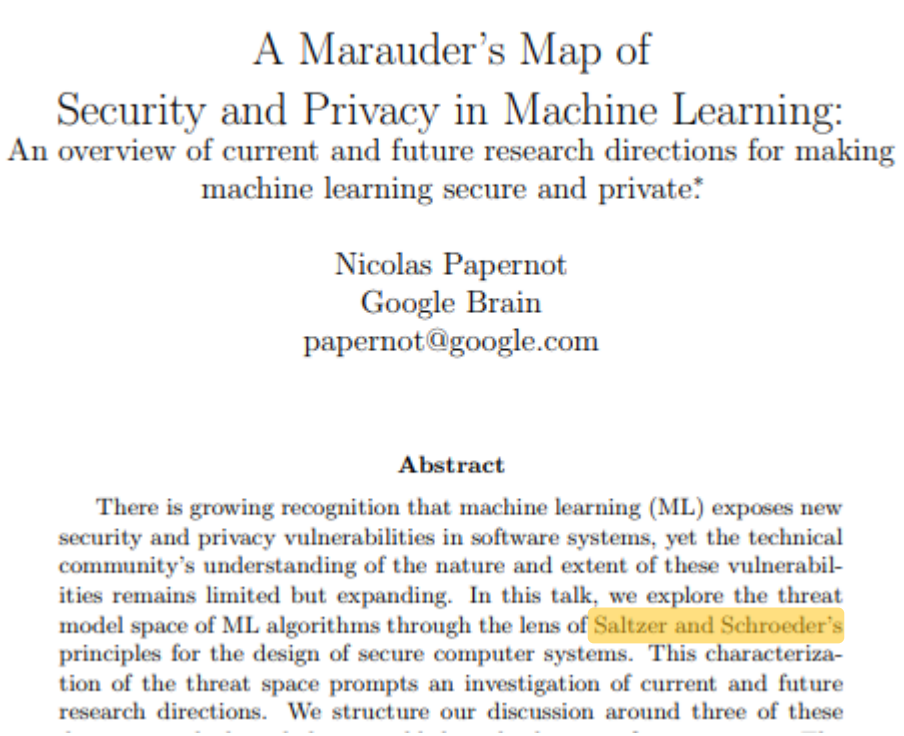
How to keep integrity?

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

"Since no one knows how to build a system without flaws, the alternative is to rely on eight design principles, which tend to reduce both the number and the seriousness of any flaws: Economy of mechanism, fail-safe defaults, complete mediation, open design, separation of privilege, least privilege, least common mechanism, and psychological acceptability" [SS75]

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.

The principles are deeper than they look, and they are easy to violate – typically caused by improperly evaluated tradeoffs.