ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

CyberSecurity

1 - “Introduction to Computer Security”

Gabriele D’Angelo <g.dangelo@unibo.it>

https://www.unibo.it/sitoweb/g.dangelo/en

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

CyberSecurity

1 - “Introduction to Computer Security”

Gabriele D’Angelo <g.dangelo@unibo.it>
https://www.unibo.it/sitoweb/g.dangelo/en

This block of slides does not refer to a specific chapter in the book.

It is a “gentle introduction”

Outline

A few basic principles of computer security:

A • principle **1**: "*non-existence of secure systems*"

it is a consequence of a principle that
has already been demonstrated.

B • principle **1** (**corollary**): "*systems complexity*"

C • principle **2**: "*entities composing a system*"

D • principle **3**: "*security = knowledge*"

E **Security of a cryptosystem** (*i.e.*, ***security through obscurity***)

Principle 1: on "Secure Systems"

SECURE SYSTEMS DO NOT EXIST

- **The software is often far from perfect**. It is full of **implementation bugs** and **design errors**. It is the same for **communication protocols**
- **The "unbreakable system" is only a myth**. Just like the impenetrable bank vault or the unsinkable boat (*e.g., RMS Titanic*)

A rational attacker will only target a system if the value of the potential gain exceeds the cost of the attack. Consequently, to determine how much to invest in security, one must be able to assess this value. To put this assessment of value and risk into practice, the professor introduces three key questions posed in logical sequence:
"What are we protecting?": Identify physical or digital assets (hardware, software, data, and information).
"From whom?": Identifies potential threat actors, understanding who the enemy is (ranging from amateurs like script kiddies to professionals, malicious insiders, or state-sponsored hackers).
"Why?": Identifies the motivations behind the attack (financial gain, espionage, sabotage, ideology).

Principle 1: on "Secure Systems"

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

An attacker will attack a system if there is enough value to make the attack worth it

SECURE SYSTEMS DO NOT EXIST

Security is costly: so there are various level of security.
It is very difficult to accurately assess the level of security required for a system.

The **security level** of a given system is determined by the **amount of time** necessary to break the system, the **amount of money** (or resources) and the **probability of success**

(amount of time it takes an attacker to find a vulnerability, develop an exploit, and breach the system)

(How much does it cost an attacker to breach the system?)

necessary to break the system

of the attack

Vulnerabilities can be discovered (by a paid individual or a criminal)

Drastically reducing the probability of success (e.g., by implementing two-factor authentication) discourages attackers, because they know they will have to make a huge number of attempts (increasing both time and costs) before succeeding.

Principle 1: on "Secure Systems"

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

How much am I willing to invest in identifying and fixing vulnerabilities in my system?

SECURE SYSTEMS DO NOT EXIST

What Are We Protecting? (The Assets) Hardware, Software, Data

- The **security level** of a given system is determined by the **amount of time** necessary to break the system, the **amount of money** (or resources) and the **probability of success**

What are we protecting?

Principle 1: on "Secure Systems"

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

SECURE SYSTEMS DO NOT EXIST

From Whom? (Threat Actors) Script Kiddies (Amateurs using pre-made tools), Cybercriminals, Insider Threats, Hacktivists and State-Sponsored Actors.

- The **security level** of a given system is determined by the **amount of time** necessary to break the system, the **amount of money** (or resources) and the **probability of success**

Understanding who the enemy is, is important for determining the appropriate level of security for our system.

What are we protecting? From who?

Principle 1: on "Secure Systems"

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

SECURE SYSTEMS DO NOT EXIST

Why? (The Reasons for an Attack) Financial Gain, Espionage, Sabotage & Warfare and Ideology & Ego.

- The **security level** of a given system is determined by the **amount of time** necessary to break the system, the **amount of money** (or resources) and the **probability of success**

What are the reasons for a possible attack?

What are we protecting? From who? Why?

Principle 1: on "Secure Systems"

Think in terms of value because that is the right way to assess the system's security level

SEC[illegible]XIST

- The **security le**[illegible] the **amount of time** necessar[illegible]**ey** (or resources) an[illegible]

- **Most times the ~~adversary~~ attacker is rational**
- **Most times the adversary has an economic goal (that is... $$$)**
- **This is not always true when the adversary is a government**

What are we protecting? From who? Why?

The Proactive Approach (Threat Modeling): Take action before a security incident occurs. You put yourself in the attacker's shoes while the system is still in the design phase to identify potential vulnerabilities and fix them before writing the final code.

Principle 1: on "Secure Systems"

- This is an "informal way" for defining a THREAT MODEL

By defining the threat model, I can determine the level of security the system requires

creates

- Threat modeling is a structured process used in cybersecurity to identify and analyze potential threats and vulnerabilities in a system or application

O NOT EXIST

determined by the **amount of**

e **amount of money** (or

ccess

What are we protecting? From who? Why?

ZERODIUM Payouts for Mobiles*

Price tags

Vulnerability Broker collects and sells vulnerabilities discovered and undisclosed by researchers (entities that buy and sell information about software vulnerabilities, usually zero-days, from security researchers, acting as intermediaries.)

chain of vulnerabilities to gain root privileges (e.g., vulnerabilities in WhatsApp and the kernel)

FCP: Full Chain with Persistence
RCE: Remote Code Execution
LPE: Local Privilege Escalation
SBX: Sandbox Escape or Bypass

4 types of exploit

When the device is restarted, the system remains compromised.

iOS
Android
Any OS

code that runs automatically without user interaction

Zerodium then resells this information and the associated attack methods (exploits) to its select clients, which are primarily government agencies, intelligence services, and law enforcement agencies (for espionage or cyberwarfare purposes).

Up to $2,500,000									1.001 Android FCP Zero Click Android	1.002 iOS FCP Zero Click IOS
Up to $2,000,000										
Up to $1,500,000									2.001 WhatsApp RCE+LPE Zero Click IOS/Android	2.002 iMessage RCE+LPE Zero Click IOS
Up to $1,000,000									2.003 WhatsApp RCE+LPE IOS/Android	2.004 SMS/MMS RCE+LPE IOS/Android
Up to $500,000	3.001 Persistence IOS	2.005 WeChat RCE+LPE IOS/Android	2.006 iMessage RCE+LPE IOS	2.007 FB Messenger RCE+LPE IOS/Android	2.008 Signal RCE+LPE IOS/Android	2.009 Telegram RCE+LPE IOS/Android	2.010 Email App RCE+LPE IOS/Android	4.001 Chrome RCE+LPE Android	4.002 Safari RCE+LPE IOS	
Up to $200,000	5.001 Baseband RCE+LPE IOS/Android		6.001 LPE to Kernel/Root IOS/Android	2.011 Media Files RCE+LPE IOS/Android	2.012 Documents RCE+LPE IOS/Android	4.003 SBX for Chrome Android	4.004 Chrome RCE w/o SBX Android	4.005 SBX for Safari IOS	4.006 Safari RCE w/o SBX IOS	
Up to $100,000	7.001 Code Signing Bypass IOS/Android	5.002 WiFi RCE IOS/Android	5.003 RCE via MitM IOS/Android	6.002 LPE to System Android	8.001 Information Disclosure IOS/Android	8.002 [k]ASLR Bypass IOS/Android	9.001 PIN Bypass Android	9.002 Passcode Bypass IOS	9.003 Touch ID Bypass IOS	

The price paid by Zerodium to purchase the vulnerability from researchers

The chart shows how much money Zerodium is willing to pay for a single working exploit targeting mobile devices (Android and iOS). On the vertical axis, we find the severity of the attack:
- RCE (Remote Code Execution): it allows the attacker to execute malicious code on the victim's phone remotely, without having physical access to the device.
- LPE (Local Privilege Escalation): the attacker already has limited initial access to the device and exploits this vulnerability to obtain maximum system privileges (e.g., Root or Administrator permissions).
- SBX (Sandbox Escape): allows the attacker to break out of the "security sandbox" in which apps run in isolation to access data from other applications.
- FCP (Full Chain with Persistence): a combined attack that merges RCE, SBX, and LPE, allowing the phone to be completely and permanently compromised.

* All payouts are subject to change or cancellation without notice. All trademarks are the property of their respective owners.

2019/09 © zerodium.com

ZERODIUM Payouts for Mobiles*

FCP: Full Chain with Persistence
RCE: Remote Code Execution
LPE: Local Privilege Escalation
SBX: Sandbox Escape or Bypass

iOS
Android
Any OS

While RCE allows attackers to execute commands from a remote location to a device, LPE allows a low-privileged user to gain administrative or root-level access.

A full-chain vulnerability on Mobile is a sophisticated cyberattack that links together multiple individual software flaws, often zero-day vulnerabilities, to achieve complete, unauthorized control over a device. A full chain links, for example, a browser exploit, a sandbox escape, and a kernel privilege escalation to move from an external attack to full "root" or kernel-level control.

Is not a full-chain vulnerability

WhatsApp RCE example:

CVE-2022-36934

Exploit Whatsapp to be able to run unauthorized code on the device and then doing something on the device itself (maybe there is something after that maybe another privilage escalation)

Up to $2,500,000									1.001 Android FCP Zero Click (Android)
Up to $2,000,000									1.002 iOS FCP Zero Click (IOS)
Up to $1,500,000								2.001 WhatsApp RCE+LPE Zero Click (IOS/Android)	2.002 iMessage RCE+LPE Zero Click (IOS)
Up to $1,000,000								2.003 WhatsApp RCE+LPE (IOS/Android)	2.004 SMS/MMS RCE+LPE (IOS/Android)
Up to $500,000	3.001 Persistence (IOS)	WeChat RCE+LPE (IOS/Android)	iMessage RCE+LPE (IOS)	FB Messenger RCE+LPE (IOS/Android)	Signal RCE+LPE (IOS/Android)	Telegram RCE+LPE (IOS/Android)	2.010 Email App RCE+LPE (IOS/Android)	4.001 Chrome RCE+LPE (Android)	4.002 Safari RCE+LPE (IOS)
Up to $200,000	5.001 Baseband RCE+LPE (IOS/Android)	The WiFi HW use a Driver running in the Kernel Level	6.001 LPE to Kernel /Root (IOS/Android)	2.011 Media Files RCE+LPE (IOS/Android)	2.012 Documents RCE+LPE (IOS/Android)	4.003 SBX for Chrome (Android)	4.004 Chrome RCE w/o SBX (Android)	4.005 SBX for Safari (IOS)	4.006 Safari RCE w/o SBX (IOS)
Up to $100,000	7.001 Code Signing Bypass (IOS/Android)	5.002 WiFi RCE (IOS/Android)	5.003 RCE via MitM (IOS/Android)	6.002 LPE to System (Android)	8.001 Information Disclosure (IOS/Android)	8.002 [k]ASLR Bypass (IOS/Android)	9.001 PIN Bypass (Android)	9.002 Passcode Bypass (IOS)	9.003 Touch ID Bypass (IOS)

Automatic Privilage Excalation

Idea:
Exploit a Vulnerability in the Application and then exploit a Kernel Vulnerability to take control of Root Privilages

* All payouts are subject to change or cancellation without notice. All trademarks are the property of their respective owners.

2019/09 © zerodium.com

Zerodium @Zerodium · Jun 1

We're looking for #0day exploits affecting Pidgin on Windows and Linux.

Bounty: $100,000

Read more: zerodium.com/temporary.html

9 129 213

Gary Kramlich
@rw_grim

Replying to @Zerodium

This really shows the sad state of funding in open source software.. I managed to find 25k USD last year to work on @impidgin full time, but if you can find a security exploit in my and other's non paid work you can make 4 times as much as I did... 🤦

9:59 PM · Jun 1, 2021 · Twitter Web App

Most of the time, the attacker is rational: the developer can introduce vulnerabilities and selling them. A lot of software products rely on open-source libraries

Idea Cloud Computing:
Exploit App Vulnerability, Exploit Virtual OS Kernel Vulnerability and Exploit Virtualization Software Vulnerability (to escape the VM) to arrive on the HW or OS of the Machine

Vulnerability Brokers are against the Fix of the Vulnerability (the Manifacture wants to fix it)

So, a researcher that has found a vulnerability can:
1) Sell the Vuln to Manufacture (participating to Bug Bounty Program: get way less money than from Vuln. Brokers)
2) Sell the Vuln to Vuln. Broker

(COROLLARY)

Principle 1: on "Complexity of Systems"

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

MORE COMPLEXITY → LESS SECURITY

As a system grows in size and functionality, maintaining a high level of security becomes exponentially harder. Why?
- Larger Attack Surface: More features mean more potential entry points for attackers.
- The Human Limit: No single engineer can understand the entire codebase and its dependencies.

- (***Empirical rule***) **Increasing the complexity** of a systems leads to an **insecure system**

Vulnerabilities rarely emerge from a single isolated component; rather, they arise from complex and unexpected interactions between multiple components.

- **Linear increase** of complexity means **super-linear increase** of insecurity

Principle 1: on "Complexity of Systems"

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

MORE COMPLEXITY → LESS SECURITY

- Do you want to build a "***reasonably secure***" system?

Reduce the Complexity of the System

KISS RULE:

avoid unnecessary complications.

- ***Keep It Simple and Stupid***
- *Keep It Simple, Stupid*
- *Keep It Short and Stupid*

If you don't understand the whole system, it is very difficult to have that system secure

Principle 1: on "Complexity of Systems"

MORE COMPLEXITY → LESS SECURITY

- Does this mean that the systems to be secured can **not** be **complex** or **complicated**?

Principle 1: on "Complexity of Systems"

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

MORE COMPLEXITY → LESS SECURITY

- Does this mean that the systems to be secured can **not** be **complex** or **complicated**?

High complexity is inevitable in modern systems. While the underlying system might be complex, the protection system must remain as simple as possible.

NO!

Principle 1: on "Complexity of Systems"

Example:
- Complex Scenario (Wrong): To secure access to servers, you create a custom proprietary system with 5 different encryption steps, tokens that expire every 12 seconds, which requires three different databases to verify identity, and which changes rules depending on the time of day. Result: A nightmare to manage. A minor bug in any of these steps will block legitimate users or create a huge logical flaw that a hacker will exploit.
- Simple Scenario (KISS - Correct): To secure the exact same complex server, use a standard, open, and widely tested protocol like SSH with public/private cryptographic keys and simple two-factor authentication (2FA). Result: The mechanism is straightforward, clean, very easy to verify, and leaves very little room for human error.

MORE COMPLEXITY → LESS SECURITY

It is the most important part of the whole system (example: Authentication of the device to use it)

- The **protection system** must be "***as simple as possible***". If possible... "***very simple***"

in security systems

- **Unnecessary complexity** can lead to {**design**, **implementation**, **usage**} errors

Every added complexity -> Potential Error -> Exploitable Vulnerability.

- Each **error** is a potential "**vulnerability**"

Principle 2: on “Entities composing a system”

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

These are the zones of the system where we can have a
Vulnerability: a system consists of three interconnected components.

(MOST) SYSTEMS ARE COMPOSED OF 3 TYPES OF ENTITIES

- Software
- Hardware
- Humanware

Humanware represents the people who interact with the system:
network administrators, developers, managers, and customers.

Principle 2: on "Entities composing a system"

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

SOFTWARE

- Example of a software attack (Difficult): Spend weeks analyzing the binary code of a banking application, find a memory bug (buffer overflow), write a complex exploit to bypass the operating system's defenses, and inject malicious code. This requires extraordinary skills and a great deal of time.
- Example of an alternative attack (Easy): Send a targeted phishing email to the bank's administrator (Humanware attack) pretending to be the director requesting an urgent check, prompting them to enter their credentials on a clone site. Within 10 minutes, the hacker is inside with full privileges, without having touched a single line of the banking software's code.

- It is a critical part of the system with a **large exposed attack surface**

- remember that... **often other parts of the system** (hardware or humanware) **are "easier to attack"**

Princ ing a system”

- Many, very complex programs
- Each program made of thousands or millions of lines of source code (*i.e. computer instructions*)

RE

- It is a critical part of the system with a **large exposed attack surface**
- remember that… **often other parts of the system are “easier to attack”**

Principle 2: on "Entities composing a system"

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

SOFTWARE

Air-Gapped Computer: This refers to a computer system that is physically isolated from all insecure networks, including the internet. The name implies there is a "gap of air" between the machine and any network, offering maximum security against remote, network-based, or malware attacks.

- It is a critical part of the system with a **large exposed attack s**

- r the system are "**easier to attack**

That can be **accessed remotely** (i.e. from the Internet, from a wireless network or from a user interface)

(example: cars)

Principle 2: on "Entities composing a system"

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

SOFTWARE

doing a well attack suface analysis isn't so easy

- It is a critical part of the system with a **large exposed attack surface**

tech or not (example: a paper document containing information about a company is considered a significant part of an organization's physical attack surface)

- remem[illegible] ***Everything* that can be used by an attacker to violate the system** [illegible]tem are "**easier to attack**"

Principle 2: on "Entities composing a system"

SOFTWARE

- It is a critical part of the system with a **large exposed attack surface**
- remember that... **often other parts of the system are "easier to attack"**

Principle 2: on "Entities composing a system"

Attacking the software often requires some technical skills… attacking the human in the system requires skills that are not about computing (psychological skills. Example of attack: phishing)

OFTWARE

stem with a **large exposed attack**

- remember that… **often other parts of the system are "easier to attack"**

Principle 2: on "Entities composing a system"

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

When it is not possible to update the firmware to fix vulnerabilities, organizations must shift from applying patches to implementing mitigation measures, such as removing the hardware or reducing its attack surface, to manage the risks. Firmware vulnerabilities have a significant impact because they operate below the operating system level, often allowing persistent and invisible access.

HARDWARE

- All the modern hardware is composed of **hardware AND software** (that is called **firmware**)

  Firmware is a specific type of software that is built directly into a hardware component. Firmware can contain bugs and vulnerabilities just like standard software.

- The hardware can be **tampered** by an attacker

  Examples: Inserting hardware keyloggers, stealing unencrypted hard drives, or exploiting supply chain vulnerabilities by altering components before delivery.

- Even the **physical security** of systems and devices (**e.g. to deny unauthorized access**) is hard to guarantee

  If an attacker has unrestricted physical access to a device, it can no longer be trusted. Why it is hard to guarantee: Modern infrastructure is decentralized, making physical perimeters insufficient to protect all assets.

Principle 2: on "Entities composing a system"

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

HARDWARE

- All the modern hardware is composed of **hardware AND software** (that is called **firmware**)
- acker
- nd devices (**e.g. to deny**
  tee

This means that an attacker can modify the hardware or ~~act on~~ compromise the software (e.g. flashing a modified firmware)

if the firmware is updatable and not well secured, the attacker can install a firmware with a backdoor

Principle 2: on "Entities composing a system"

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

To interfere with a system (by touching it, altering it, or being in the middle) with the intent to cause harm or make unauthorized changes. "something" is any Asset (Hardware, Software, or Data) whose Integrity or Availability is compromised because an unauthorized entity stepped in and altered its original, intended state.

Manomettere/Alterare

Definition: "*interfere with (something) in order to cause damage or make unauthorized* ~~*alterations*~~ changes"

Because tampering is a huge risk, engineers design sensitive systems using two defensive approaches:
- Tamper-Resistance: Creating physical or logical barriers that make it extremely difficult to alter the system.
- Tamper-Evident: If I cannot prevent a hacker from accessing the system, I ensure that any tampering is immediately visible. In the physical world, security seals (sigilli di sicurezza) are used (stickers that break if you try to open them); in the software world, encryption is used (e.g., hash functions or digital signatures) to instantly detect if even a single bit of a file has been altered.

ARE

nposed of **hardware AND software**

Tampering requires that the attacker have physical access to the hardware.

- The hardware can be **tampered** by an attacker

How can you know if a hardware device has been tampered with? To ensure that a hardware device has not been tampered with, you must ensure the physical security of the device (since, if a hardware device has been tampered with, it is very difficult to detect).

- Even the **physical security** of systems and devices (**e.g. to deny unauthorized access**) is hard to guarantee

Principle 2: on "Entities composing a system"

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

HARDWARE

The "evil maid attack" is an example of unauthorized alteration of personal devices

composed of **hardware AND software**

- The hardwa e **tampered** by an attacker
- Even the **physical security** of systems and devices (**e.g. to deny unauthorized access**) is hard to guarantee

Principle 2: on "Entities composing a system"

Possible to put a backdoor in a binary firmware without knowing the source code (it is easy to place it in the middle of the firmware)

To prevent unauthorized modifications, modern CPUs rely on hardware-based tamper-resistance, which cryptographically signs and verifies the firmware before execution.

HARDWARE

The Physical security of an HW must be guaranteed across the full hardware lifecycle, from initial purchase to final disposal (because if control were lost even for just a few seconds, it could be tampered). We do this because it is very difficult to detect if the HW it was tempered

It is an attack that exploits temporary physical access to compromise a device so that the attacker can control it remotely (by inserting a backdoor into the device's firmware).

The "evil maid attack" is an example of unauthorized alteration of personal devices

Evil Maid Attack Demo

A "nice demo": the time required to flash the tampered firmware is very limited

comp

- The hardwa e **tampered** by acker
- Even the **physical security** of syst ns and devices (**e.g. to deny unauthorized access**) is hard to guarantee

Principle 2: on "Entities composing a system"

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

- Lack of knowledge: The user or administrator simply does not know that a certain action is dangerous (e.g., reusing the same password for work and personal websites).
- Overload & Fatigue: A tired employee, overwhelmed by countless emails and deadlines, has a much lower attention span. It is in that moment of fatigue that they will click on the phishing email without checking it carefully.
- Bad design: If a security system is poorly designed, frustrating, and too complicated to use, the user will try to bypass it just to get their work done.
- Goodwill: Humans are inclined to help others, so an attacker can exploit this "goodwill" by posing as a colleague in trouble who urgently needs a file or access to avoid losing a client.
- Lack of attention: A one-second distraction can lead to sending a confidential document to the wrong recipient.
- Stupidity (in both users and administrators): attitudes of total laziness or unjustifiable indifference. For example, a user who deliberately ignores antivirus security warnings, or a system administrator who leaves the main server password as "admin/admin" or "123456."

HUMANWARE

- The "**human in the loop**" is often the **weakest link in the system** (e.g. **social engineering**)

Attackers exploit human psychology (fear, urgency, trust) rather than technical flaws to bypass defenses (e.g., Phishing).

- For **many reasons**: lack of knowledge, overload, fatigue, bad design, goodwill, lack of attention… stupidity (*in both **users** and **system administrators***)

Principle 2: on "Entities

Definition: "the use of deception to manipulate individuals into divulging confidential ~~or personal~~ information that may be used for fraudulent purposes" (*e.g. phishing emails*)

inganno

to push him/her

(that goes against security policies)

Social engineering is widely recognized as the most common and effective method for cyberattacks in the history of cybersecurity, making the "human element" the weakest link in security.

- The "**human in the loop**" is often the **weakest link in the system** (e.g. **social engineering**)
- For **many reasons**: lack of knowledge, overload, fatigue, bad design, goodwill, lack of attention… stupidity (*in both **users** and **system administrators***)

Principle 2: on "Entities composing a system"

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

HUMANWARE

Correct way to mitigate this problem is throgh procedures (properly designed) to identify if i'm talking with a collegue or attacker

Is "goodwill" a problem?

(example: faking to be a collegue and ask for confidential info)

Yes, when abused by attackers

- The "**human in the l** … **he system** (e.g. **social enginee**
- For **many reasons**: lac… knowledge, overload, fatigue, bad design, goodwill, lack of attention… stupidity (*in both **users** and **system administrators***)

We are stupid when we don't understand something

Principle 2: on "Entities composing a system"

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

OPSEC (Operational Security) is a risk management process that it aims to prevent the exposure of sensitive information that attackers could exploit to launch attacks. It involves analyzing one's daily activities from the perspective of a potential attacker. The 5 phases of the OPSEC process:
- Identify critical information: Determine which data (intellectual property, passwords, customer data, locations) would cause harm if exposed.
- Analyze threats: Understand who the potential adversaries are (cybercriminals, competitors, hackers) and their motivations.
- Analyze vulnerabilities: Identify weaknesses in processes, human habits, or systems that could reveal information.
- Assess risks: Determine the likelihood that a vulnerability will be exploited and the extent of the damage.
- Apply countermeasures: Implement practices or tools to protect critical information.

SECURITY AS A PROCESS

Cybersecurity isn't a product

- Securing a system is a [**never ending**] **process** and **not a product**

(Cybersecurity is a day-by-day investment)

- **Reasonably secure system** requires **an endless work** on the system and **education**

secure

- The system that today is "secure", tomorrow will not be. **Everyday new faults are discovered**
- The **lack of updates** leads to **fragile** and **insecure** systems

Principle 2: on "Entities composing a system"

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

EVEN UPDATING IS NOT EASY…

- Are the updates **available** for all your devices? (e.g. **smartphone**)
- In other words, **is your devices supported**? **Until when**?

Auto-Updates -> Pro: install automatically the fix for a detected vuln. Cons: install updates automatic with backdoors or other vuln.

Principle 2: on "Entities composing a system"

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

System Administrators focus on Updates (when to do it, how, etc..)

EVEN UPDATING IS NOT EASY...

- Are the updates **available** for all your devices? (e.g. **smartphone**)
- In other words, **is your devices supported**? **Until when**?

if the device is no longer supported, the device is vulnerable if they discover a vulnerability and you cannot update it

The tech guy need to define the UPDATE POLICY, based on Company Context

- If the updates are available...
  - Updates can break existing system dependencies, cause software conflicts, or introduce new bugs (Choose which part of the update to do).

    what *update to apply?* (***All of them***... *are you sure of that?*)
  - Timing is a trade-off: updating live systems (in production) causes downtime. Scheduling updates during off-peak hours (e.g., night) minimizes impact but requires on-call engineering support (a rotation where engineers are tasked with monitoring systems and mitigating production incidents outside of normal business hours).

    when *do you apply updates?* (*During the* ***night****? In "****production****"?*)

    if the vulnerability is public known, update ASAP, considering the time (example: customer peeks, etc..)
  - You must analyze the patch note. Security-critical hotfixes require immediate deployment, while feature-rich updates might be delayed to prioritize system stability.

    why*?* (*Is that for* ***security*** *or to add* ***new functionalities****?*)
  - ***how****?* (*What is the best way to apply the update?* ***Unattended****?*) (Automatic Update)

Choosing the update method:
- Unattended (Automated): High efficiency but risks catastrophic, unmonitored failures.
- The Staging Strategy: Patches should be tested in an isolated staging environment before being pushed to production.

Principle 2: on "Entities composing a system"

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Since updating involves risks, a system administrator cannot blindly apply patches as soon as they are released. They must weigh the pros and cons:
- The benefits: What security vulnerabilities am I fixing? What risks do I face if I don't update?
- The costs: How much does it cost to shut down the machines for the update? What is the financial cost if something goes wrong and the system stops working?

EVEN UPDATING IS NOT EASY…

- The **updating process** is always a dangerous operation
  - for example: ***fixing bugs*** *can introduce* ***new bugs***, ***instability***, ***side-effects***… due to complexity of the system we are dealing with
- What we need is a **costs/benefits evaluation**
  - that is **costly** (*e.g. time consuming*)

    Analyzing the impact of an update requires time and financial resources. Companies often need to test updates in a staging environment that mirrors the production environment before deploying them to production. This process requires hours of work and qualified personnel.
  - it is often based on **partial information** (*i.e. often there is a lack of detailed information on the* ***updates content***)

    to take decision on a specific update

Without knowing exactly what has been changed in the code, those responsible for managing the system are forced to make critical decisions based on incomplete information, increasing uncertainty and the risks involved in the operation.

Principle 3: on "Security = Knowledge"

KNOWLEDGE OF THE SYSTEM

Trust is even more important than Knowledge

- Real security is based on **deep knowledge** (of the system to be protected)
- What are the ~~**effects**~~ implications of this principle **on software**?
  - **Open Systems** vs. **Closed Systems**
  - **Open Source** vs. **Closed Source**

Principle 3: on "Security = Knowledge"

KNOWLEDGE OF THE SYSTEM

- Real security is based on **dee** be protected)
- What are the **effects** of this pri ple **on software**?
  - **Open Systems** vs. **Closed Systems**
  - **Open Source** vs. **Closed Source**

The specifications of the system are

OPEN / CLOSED

owledge”

Prof. Opinion

I really prefer OPEN specifications (e.g. for interoperability with other system) but this does not guarantees that the system is secure (e.g. due to design or implementation faults)

This means that the system's operating rules are a business secret: no one outside that company knows exactly how the data is processed or packaged for transmission. If you want to create something that interacts with that system, you either can't do it, or you have to pay a huge amount of money for a license.

E OF THE SYSTEM

If they have specs open/closed, doesn't guarantee security by definition

The specifications of the system are OPEN / CLOSED

- Real securit ed on dee be protected)
- What are t effects of this pri ple on software?
  - Open Systems vs. Closed Systems
  - Open Source vs. Closed Source

This means that the details of how the system is designed, how it works, and how it communicates are public, free, detailed, and accessible to anyone (often standardized by international organizations). Anyone can read the technical manuals and build software or hardware that is fully compatible with that system, without having to ask permission or pay royalties to anyone.

Principle 3: on "Security = Knowledge"

KNOWLEDGE OF THE SYSTEM

- Real security is based on **deep knowledge** (of the system to be protected)
- What are the **effects** of this principle **on software**?
  - **Open Systems** vs. **Closed Systems**
  - **Open Source** vs. **Closed Source**

Principle 3: on "Security = Knowledge"

KNOWLEDGE OF THE SYSTEM

- Real security is based on **d** be protected)
- What are the **effects** of this prin **software**?
  - **Open Systems** vs. **Cl d Systems**
  - **Open Source** vs. **Closed Source**

The source code (i.e. computer instructions) used by the programmers to implement the system are

OPEN / CLOSED

Knowledge”

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Prof. Opinion

I really prefer OPEN Source (e.g. *it can be inspected, verified, validated etc.*) but this does not guarantees that the system is secure

GE OF THE SYSTEM

- Real secu sed on d be protectec
- What are effects of this prin software?
  - Open ystems vs. Cl d Systems
  - Open Source vs. Closed Source

The source code (i.e. computer instructions) used by the programmers to implement the system are

OPEN / CLOSED doesn't guarantee security by definition

- Closed Source (Security through Obscurity): Keeping source code secret under the assumption that attackers can't exploit what they can't see.
- Open Source (The Power of Crowdsourcing): Making source code public to maximize system knowledge.

Kn

G

I really prefer OPEN Source (*e.g. it can be inspected, verified, validated etc.*) but this does not guarantees that the system is secure

Even if it is theoretically possible (for me) to check all the code in a system (*e.g. an open source Operating System like GNU/Linux*), this is not practically possible (*since it is made of millions of lines of source code*)

Even if i have the source code of the program, it is difficult to know if the binary version of it contains a backdoor or malicious code that is different from the source code that is public available

- Real secu sed on m to be protectec
- What are effec of this principle on software?
  - **Open ystems** vs. **Closed Systems**
  - **Open Source** vs. **Closed Source**

Principle 3: on "Security = Knowledge"

Management of Trust is fundamental in Cybersecurity and based on a lot of different stuff (example: choosing the browser based on history of vuln. fixing)

KNOWLEDGE OF THE SYSTEM

baseline of CySec (define the allies and enemies)

- Real security is based o knowledge (of the system to be protected)
- What are the **effec**
  - **Open Systems**
  - **Open Source** v

In most cases it is a matter of

Security is based on deep knowledge of the system we want to protect, but

TRUST

is more important than knowledge (because it is very difficult to know/verify/check every single detail of the system)

Do you prefer to trust a community (i.e. Open Source) or a company (i.e. Close Source)?

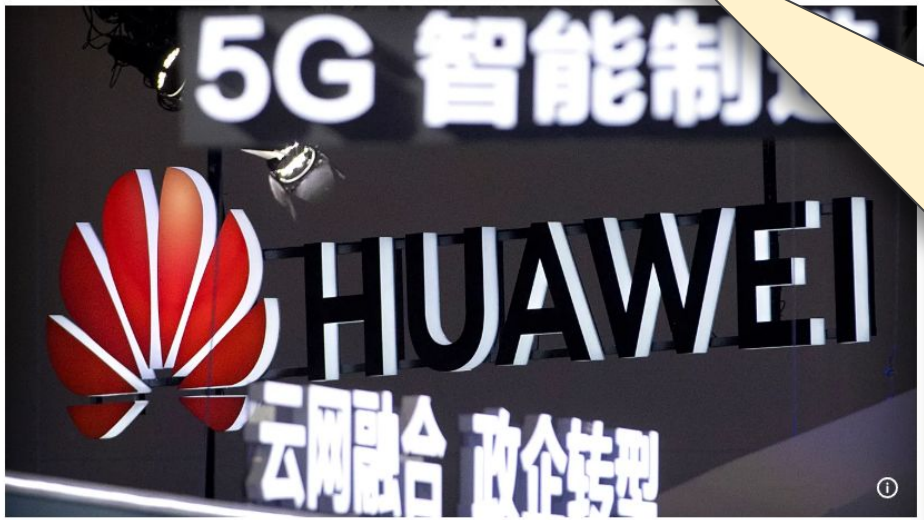
Modern products are built on a complex software supply chain, heavily relying on external libraries, third-party services, and open-source components. Ultimately, if a security breach occurs, the legal and financial responsibility lies solely with the manufacturer. To mitigate this risk and build trust with their customers, manufacturers must certify every component integrated into their product.

As systems grow in complexity, a model based on total verification (manually checking every single line of third-party code) is no longer viable. In the future, ensuring supply chain security will shift toward a scalable trust model. Organizations will increasingly rely on both internal and external formal certifications to validate their security posture, prove regulatory compliance (conformità), and guarantee security to their end customers.

Principle 3: on “Security = Knowledge”

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Eleven EU countries took 5G security measures to ban Huawei, ZTE



Copyright Mark Schiefelbein/Copyright 2018 The AP. All rights reserved

By Cynthia Kroet

Published on 12/08/2024 - 16:37 GMT+2

because they cannot go in deep on the software of these devices to check about vulnerabilities like backdoors (they restricted because they don't have trust on some companies: possible that China gather info through it (not verified))

«Eleven countries [...] have used legal powers to impose **restrictions** on telecom suppliers that are considered high-risk, such as Huawei and ZTE, for 5G network infrastructure, a European Commission spokesperson told Euronews»

Principle 3: on "Security = Knowledge"

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

KNOWLEDGE OF THE SYSTEM

- **Users' education** is a form of **knowledge**
- **Investing in education** is good but **not that simple**:
  - education is **costly**
  - **perception of security** [lack of] importance of system security
  - [wrong] **habits** (*i.e. "we have always done it this way"*)
  - **ideological resistance** (*i.e. "information anarchism"*)
  - **cost of security** procedures (*vs. "productive investments"*)

It is important but a few cares about it (increasing in these day)

Some users reject security rules for ideological reasons (for example: don't use passwords).

An investment in security does not generate a profit; it merely serves to prevent a loss. For a manager focused on short-term numbers, it is always difficult to choose to spend money on security rather than on productivity. So, because compliance is mandatory, regulations are the primary leverage used to force business leaders to budget for cybersecurity over productivity.

Addendum: "Security of a Cryptosystem"

Principle that is important for the Professor

KERCKHOFFS'~~S~~ PRINCIPLE

the security of a cryptographic system shouldn't rely on the secrecy of the algorithm

"A **cryptosystem** should be secure **even if everything about the system**, **except the key**, **is public knowledge**" (1883)

Addendum: "Security of a Cryptosystem"

KERCKHOFFS'S PRINCIPLE

- "A **cryptosystem** should be secure **even if everything about the system**, **excepi key**, **is public knowledge**" (1883)

It is a mechanism (i.e. hardware or software) used for encrypting information with the aim to preserve confidentiality

algorithm

(encryption to preserve this property)

Addendum: "Security of a Cryptosystem"

KERCKHOFFS'S PRINCIPLE

- "A **cryptosystem** should be secure **even if everything about the system**, **except the key**, **is public knowledge**" (1883)

Addendum: "Security of a Cryptosystem"

KERCKHOFFS'S PRINCIPLE

- "A **cryptosystem** should be secure **even if everything about the system**, **except the key**, **is public knowledge**" (1883)

SHANNON'S REFORMULATION

- "The **enemy knows the system**" not every secret of the system, but the design of the system

It is much better to base the security on the selection of the proper algo, not on the secrecy of it (maintaining private the key)

Addendum: "Security of a Cryptosystem"

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

SECURITY THROUGH OBSCURITY

Wrong principle: against Kerckhoffs' principle

- Means relying on the **design** or **implementation secrecy** as the **main method of providing security**

It is very difficult to maintain during the years

Why Obscurity Fails in Modern Cybersecurity:
- The Single Point of Failure: if the design/implementation of algorithm is leaked, security drops to zero. Fixing it requires rewriting the entire system.
- The Key-Based Solution (Kerckhoffs): If a secret key is compromised, the algorithm remains secure. You only need to change the key, a fast, zero-cost operation.
- The Power of Peer Review: Public algorithms are verified by global cryptographic communities: flaws are found and patched by friendly researchers before attackers can exploit them.
- The Reality of Reverse Engineering: Modern analysis tools ensure that an attacker will eventually discover how a system works. Secrecy is a temporary delay, not a security barrier.

Addendum: "Security of a Cryptosystem"

SECURITY THROUGH OBSCURITY

- Means relying on the **design** or **implementation secrecy** as the **main method of providing security**
- The **security through obscurity** is **WRONG**

Addendum: "Security of a Cryptosystem"

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

The **security through obscurity** is **WRONG**

- Does this means that I must **publish all the details about my system internals?**

When you develop a new application, there are two separate path that you can choose to adopt encryption:
1) Use standards (Good Approach: study papers and choose the perfect one)
2) Create a new algo for encryption (Stupid Approach: will do something with a lot of vulnerabilities and errors, if you are not an expert)

Addendum: "Security of a Cryptosystem"

Before deploying any security control, you must define a Threat Model:
1. WHAT are you securing? (Asset Identification) You cannot protect everything equally. You must catalog your high-value assets (e.g., source code, customer PII, financial databases, physical infrastructure) and map their boundaries.
2. WHY are you securing it and against WHO? (Threat Analysis) Understand the specific risks and adversary motives. Are you preventing data theft (Confidentiality), unauthorized changes (Integrity), or system downtime (Availability)?
So, a security mechanism is useless without a clear model. The threat model defines the mechanism, ensuring that the defense matches the actual risk and asset value: first, create a threat model, and only then decide which technology or defense to implement (the mechanism). If you skip this step, you risk spending millions to fortify a secondary door while leaving the front door wide open. Risk should guide your choice of tool, never the opposite.

The **security through obscurity** is **WRONG**

- Does this means that I must **publish all the details about my system internals?**
- **NO**, it would be very stupid! →

The security must not be based on secrecy, but this doesn't mean that you have to publish all details (you need to increase the cost for attacker to breach the system or gather info). Before disclosing information, evaluate whether it is truly necessary. Excessive disclosure fuels OSINT (Open Source Intelligence), allowing attackers to gather the data needed to refine and launch a more effective attack.

The security mechanism is useful if it is possible to enforce it (users will try to bypass it)

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

CyberSecurity

Gabriele D'Angelo

<g.dangelo@unibo.it>

https://www.unibo.it/sitoweb/g.dangelo/en

Attributions and Notes

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

- These slides are partially based on the work of Prof. **Renzo Davoli**
- **Ludovico Gardenghi** has worked on a previous version of these slides
- I wish to **thank all of them**, **any errors that remain are my sole responsibility**