Cybersecurity - Vulnerabilità

Paolo PRINETTO

Direttore
CINI Cybersecurity
National Laboratory

OLICYBER
OLIMPIADI ITALIANE DI CYBERSECURITY
POWERED BY CYBER CHALLENGE.IT

CYBERSECURITY
NATIONAL
LABORATORY

https://cybersecnatlab.it

License & Disclaimer

License Information

Disclaimer

- We disclaim any warranties or representations as to the accuracy or completeness of this material.
- Materials are provided “as is” without warranty of any kind, either express or implied, including without limitation, warranties of merchantability, fitness for a particular purpose, and non-infringement.
- Under no circumstances shall we be liable for any loss, damage, liability or expense incurred or suffered which is claimed to have resulted from use of this material.

Obiettivo della presentazione

- Fornire:
  - una tassonomia delle vulnerabilità
  - alcuni esempi ritenuti significativi
  - alcuni suggerimenti per l'attivazione di opportune contromisure.

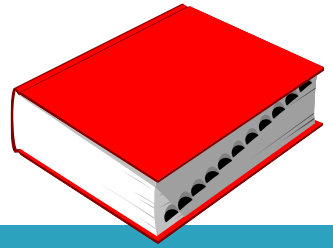
Prerequisiti

- Lezione:
  - *CS_1.03 - I Pilastri della Security*

Approfondimenti

- Alla lezione, dopo l'ultima slide, sono state aggiunte delle slide che presentano degli *Approfondimenti* relativi ad alcuni degli argomenti trattati.

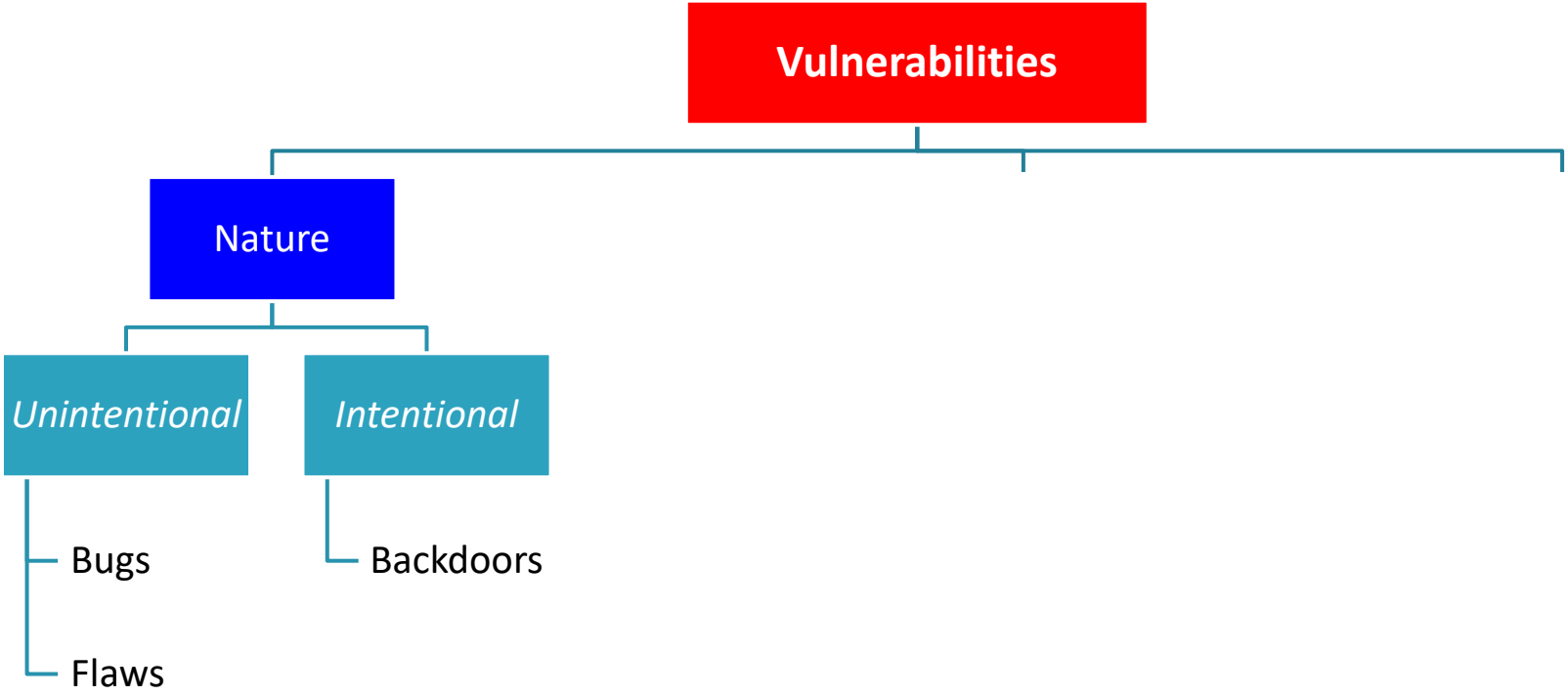
Vulnerabilità

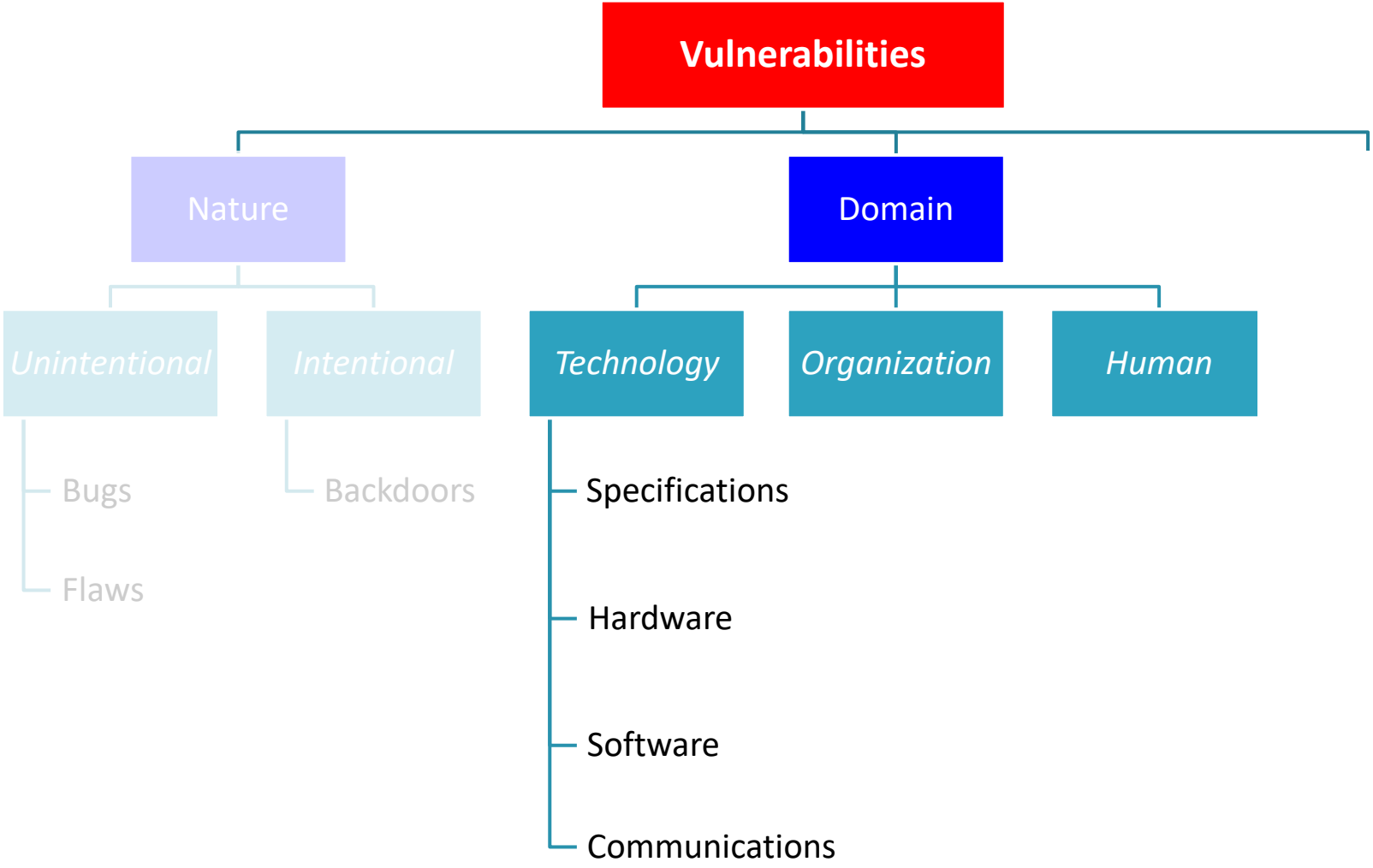


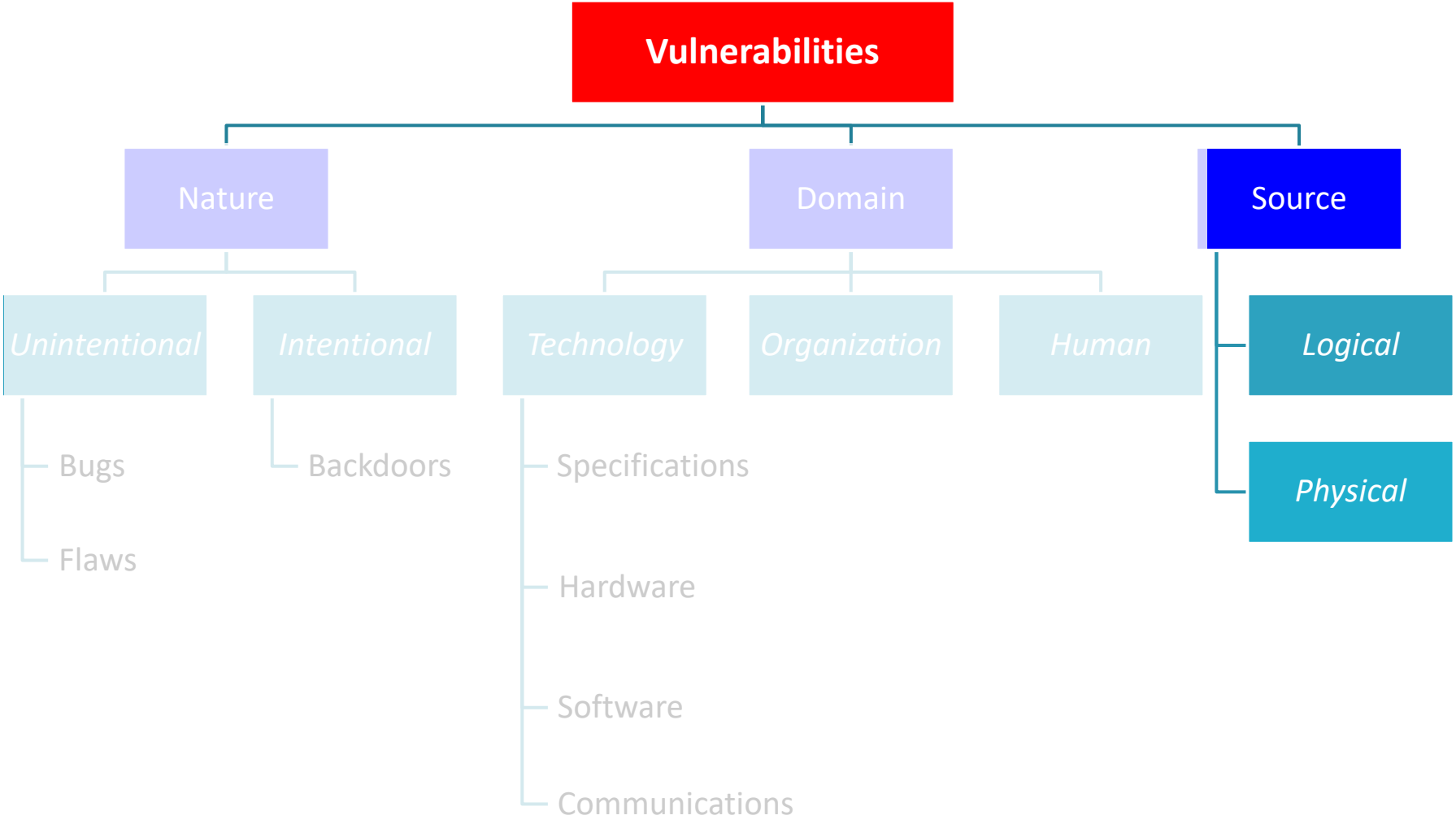
- Particolare debolezza, presente in una componente specifica di un sistema, che può essere *sfruttata* da un aggressore per sferrare un *attacco.*

Vulnerabilità

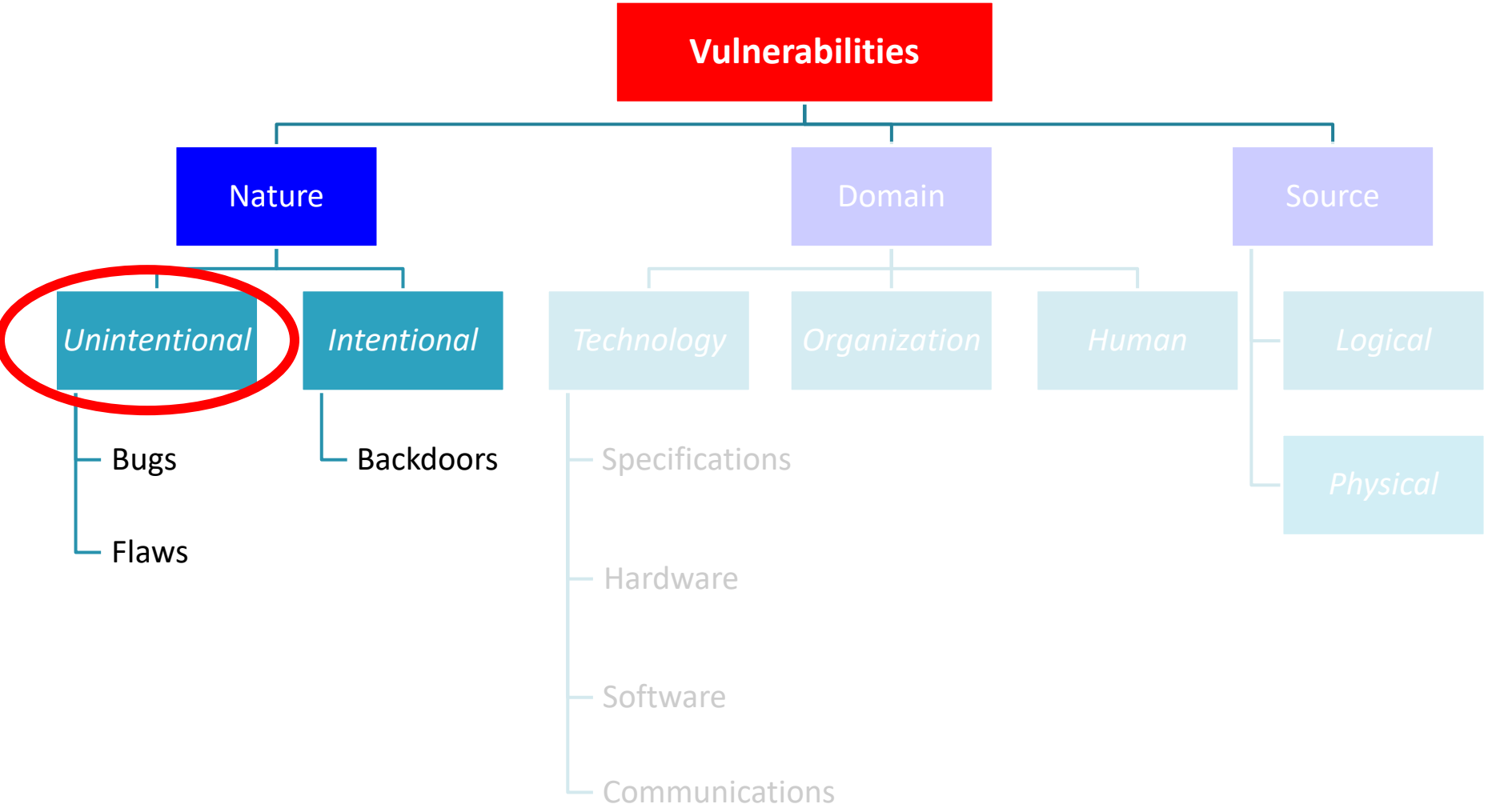
- Possono essere raggruppate secondo diverse dimensioni ortogonali
- Nel seguito ci concentreremo sulle seguenti tre:
  - *Natura* della vulnerabilità
  - *Dominio* della vulnerabilità
  - *Sorgente* della vulnerabilità

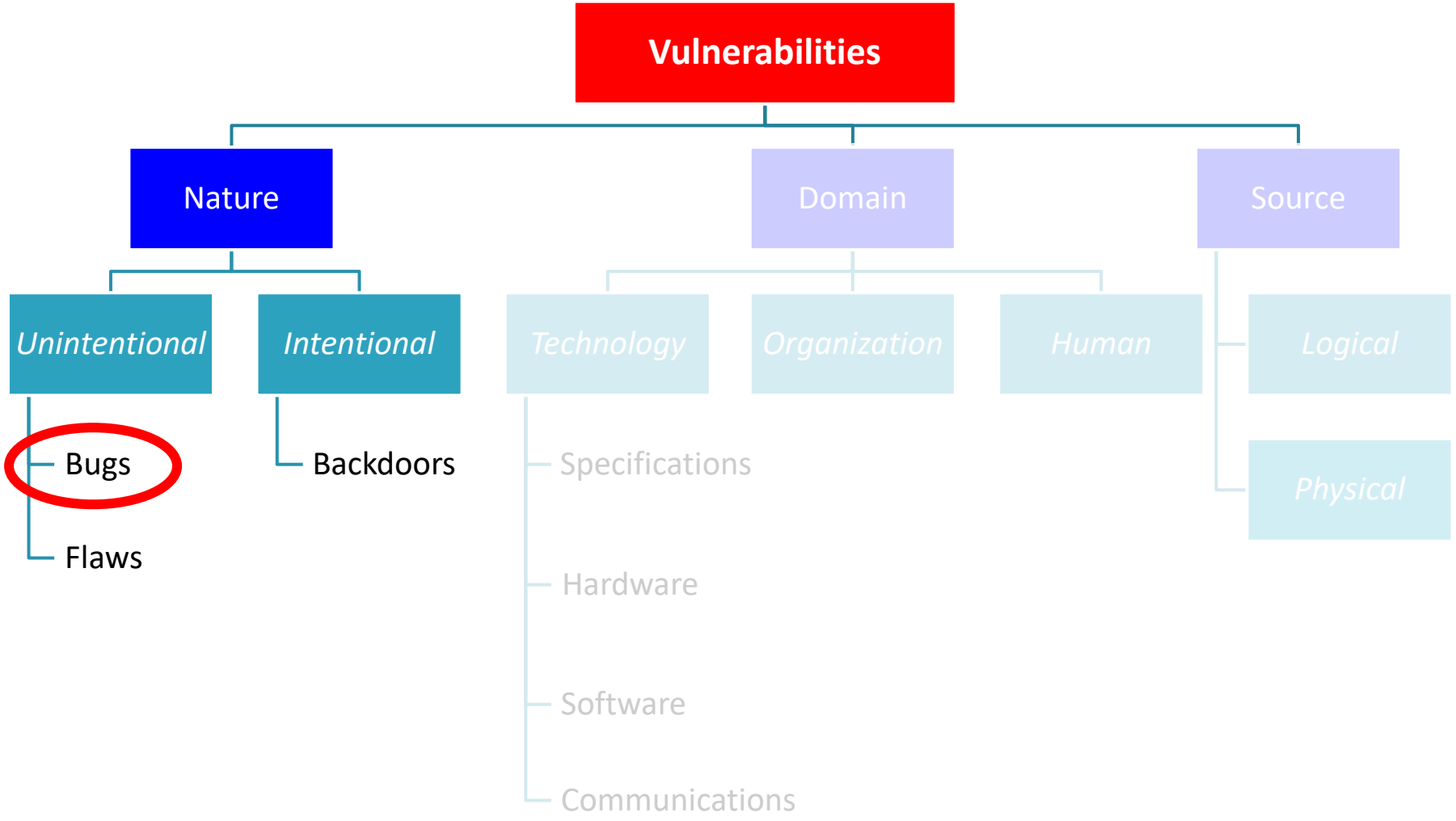

Vulnerabilities
Nature
Unintentional
Intentional
Bugs
Flaws
Backdoors


Vulnerabilities
Nature
Domain
Unintentional
Intentional
Technology
Organization
Human
Bugs
Flaws
Backdoors
Specifications
Hardware
Software
Communications

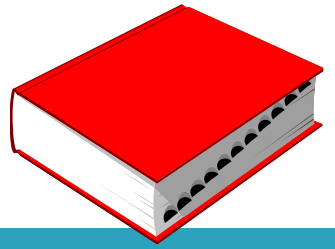

Vulnerabilities
Nature
Domain
Source
Unintentional
Intentional
Technology
Organization
Human
Logical
Physical
Bugs
Backdoors
Specifications
Flaws
Hardware
Software
Communications

Alcuni casi significativi


Vulnerabilities
Nature
Domain
Source
Unintentional
Intentional
Technology
Organization
Human
Logical
Physical
Bugs
Backdoors
Specifications
Flaws
Hardware
Software
Communications


Vulnerabilities
Nature
Domain
Source
Unintentional
Intentional
Technology
Organization
Human
Logical
Physical
Bugs
Backdoors
Specifications
Flaws
Hardware
Software
Communications

Bug (Errore)



- Errore commesso in una delle fasi di sviluppo di un prodotto che produce un risultato indesiderato o involontario.

Bug: possibili sorgenti

- I bug possono essere generati da fonti diverse, tra cui:
  - Persone
  - Procedure
  - Strumenti

Bug: possibili sorgenti

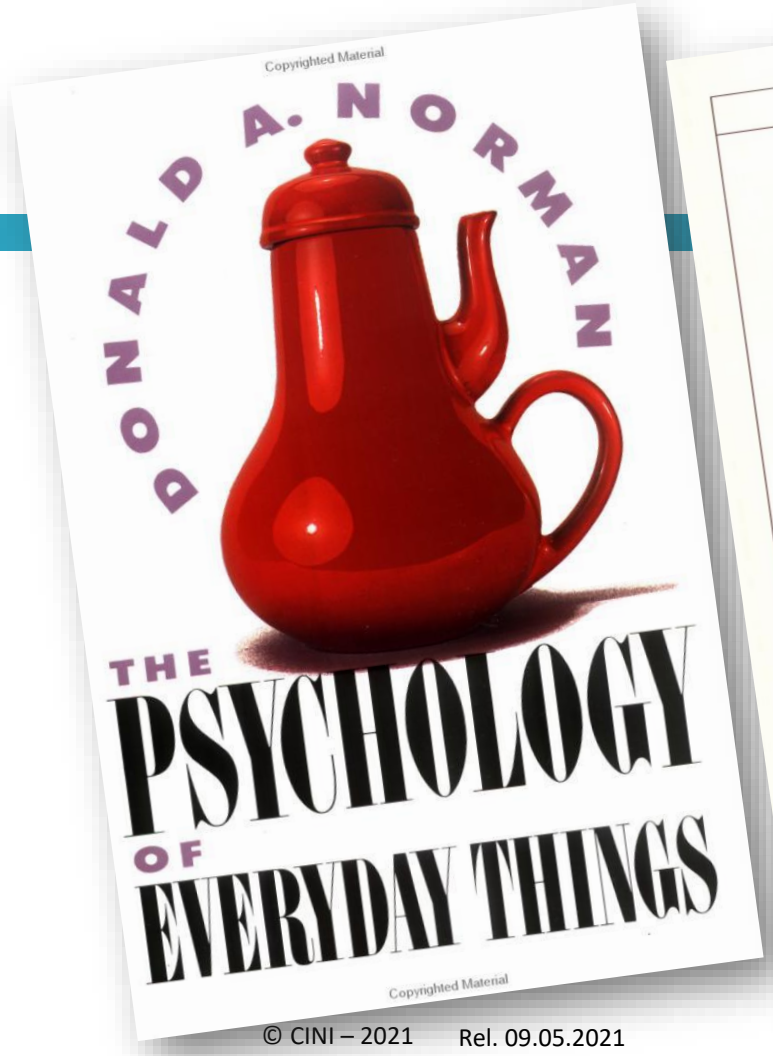
- I bug possono essere generati da fonti diverse, tra cui:
  - Persone
  - Procedure
  - Strumenti

- Durante la fase di progettazione e produzione:
  - Inesperienza di progettisti, responsabili di produzione e di collaudo:
    - Errori di progettazione
    - Carenze nelle fasi di convalida e verifica (V&V : Validation & Verification)
    - Insufficienti coperture da parte dei processi di collaudo
- Sul campo:
  - uso improprio

A proposito di “uso improprio”

- L'errore dell'operatore è una delle cause di failure più comuni
- Tuttavia molti errori attribuiti agli operatori sono in realtà causati da progetti che richiedono a un operatore di eseguire operazioni di correzione e di ripristino senza un adeguato addestramento e senza adeguato supporto nelle decisioni

Lettura consigliata



Bug: possibili sorgenti

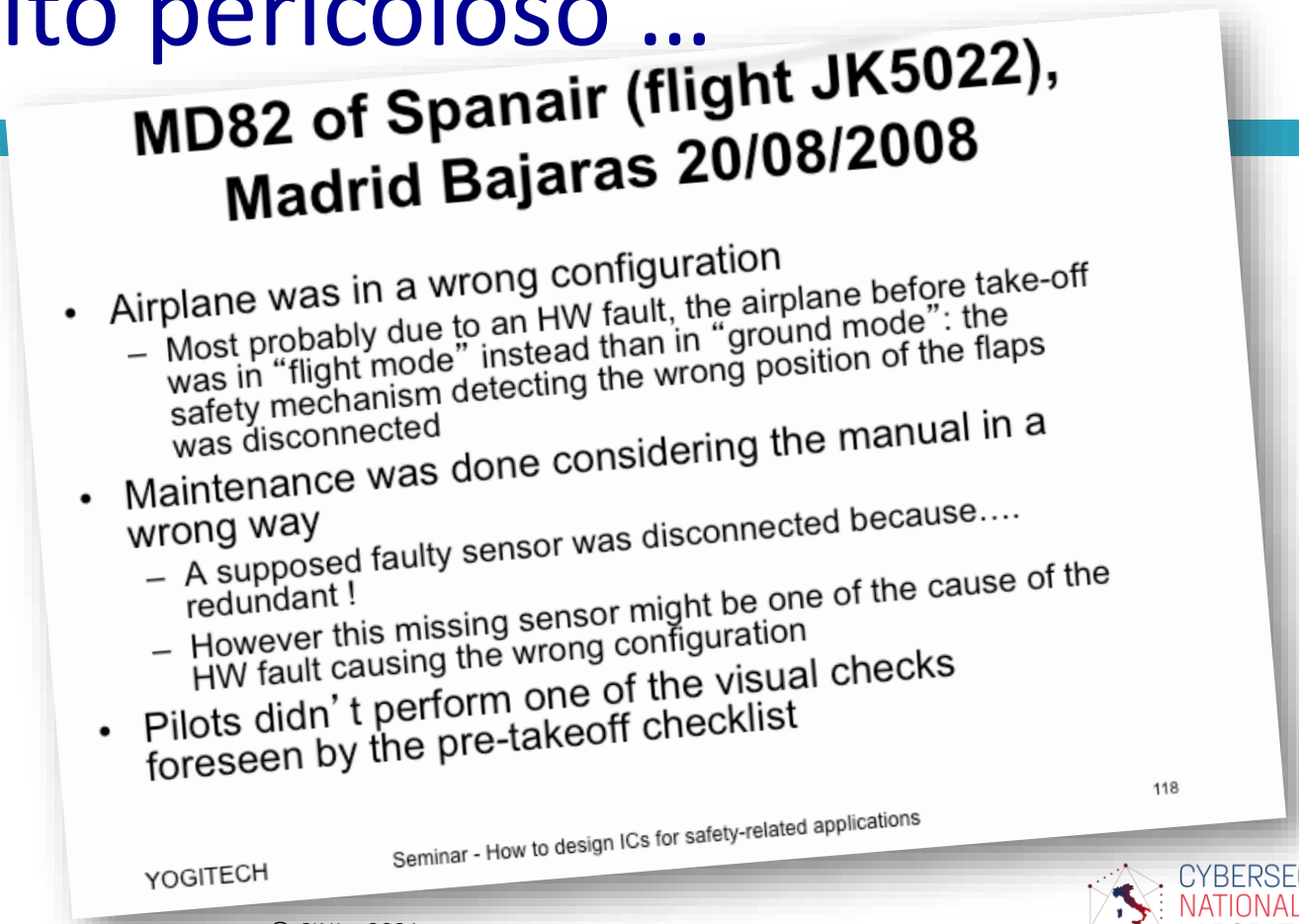
- I bug possono essere generati da fonti diverse, tra cui:
  - Persone
  - Procedure
  - Strumenti
- Errori all'interno delle:
  - Regole di progettazione
  - Metodologie di progettazione
  - Metodologie di V&V
  - Metodologie di collaudo
- Mancanze di controlli di conformità rispetto a:
  - Metodologie di progettazione e V&V
  - Standard adottati ai vari i livelli

Bug: possibili sorgenti

- I bug possono essere generati da fonti diverse, tra cui:
  - Persone
  - Procedure
  - Strumenti
- Gli strumenti adottati nelle varie fasi possono essere
  - Inappropriati
  - Bacati

Mix molto pericoloso …

Mix molto pericoloso …



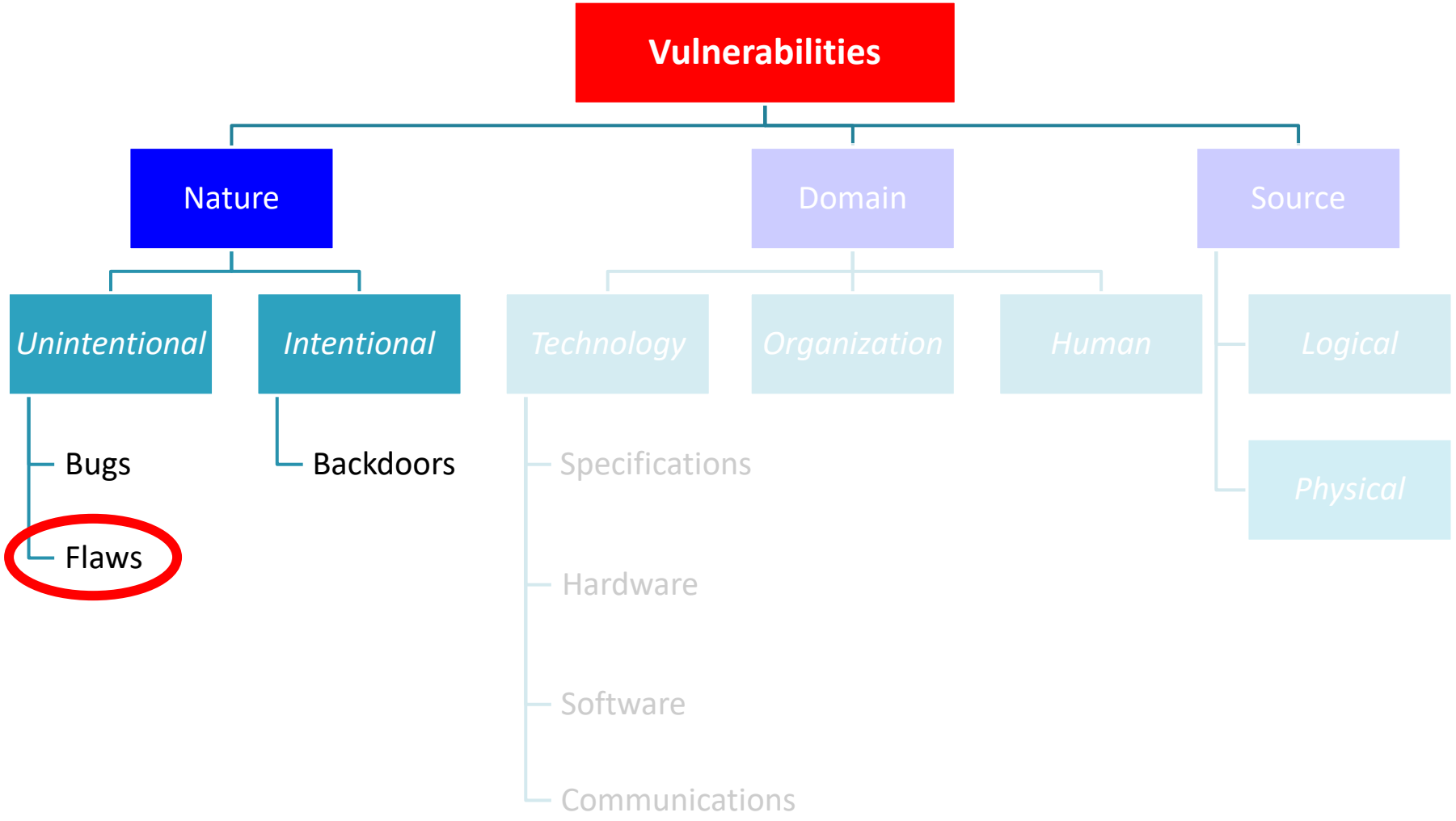
MD82 of Spanair (flight JK5022), Madrid Bajaras 20/08/2008

- Airplane was in a wrong configuration
  - Most probably due to an HW fault, the airplane before take-off was in "flight mode" instead than in "ground mode": the safety mechanism detecting the wrong position of the flaps was disconnected
- Maintenance was done considering the manual in a wrong way
  - A supposed faulty sensor was disconnected because….redundant !
  - However this missing sensor might be one of the cause of the HW fault causing the wrong configuration
- Pilots didn't perform one of the visual checks foreseen by the pre-takeoff checklist

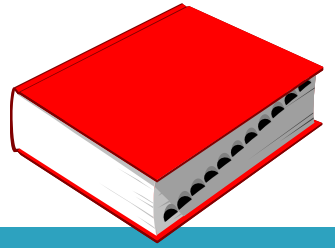
YOGITECH Seminar - How to design ICs for safety-related applications 118

Bug: come rimediare

- Sono necessari ulteriori investimenti in termini di:
  - Persone
  - Procedure
  - Strumenti


Vulnerabilities
Nature
Domain
Source
Unintentional
Intentional
Technology
Organization
Human
Logical
Physical
Bugs
Backdoors
Specifications
Hardware
Software
Communications
Flaws

Flaw (Difetto)



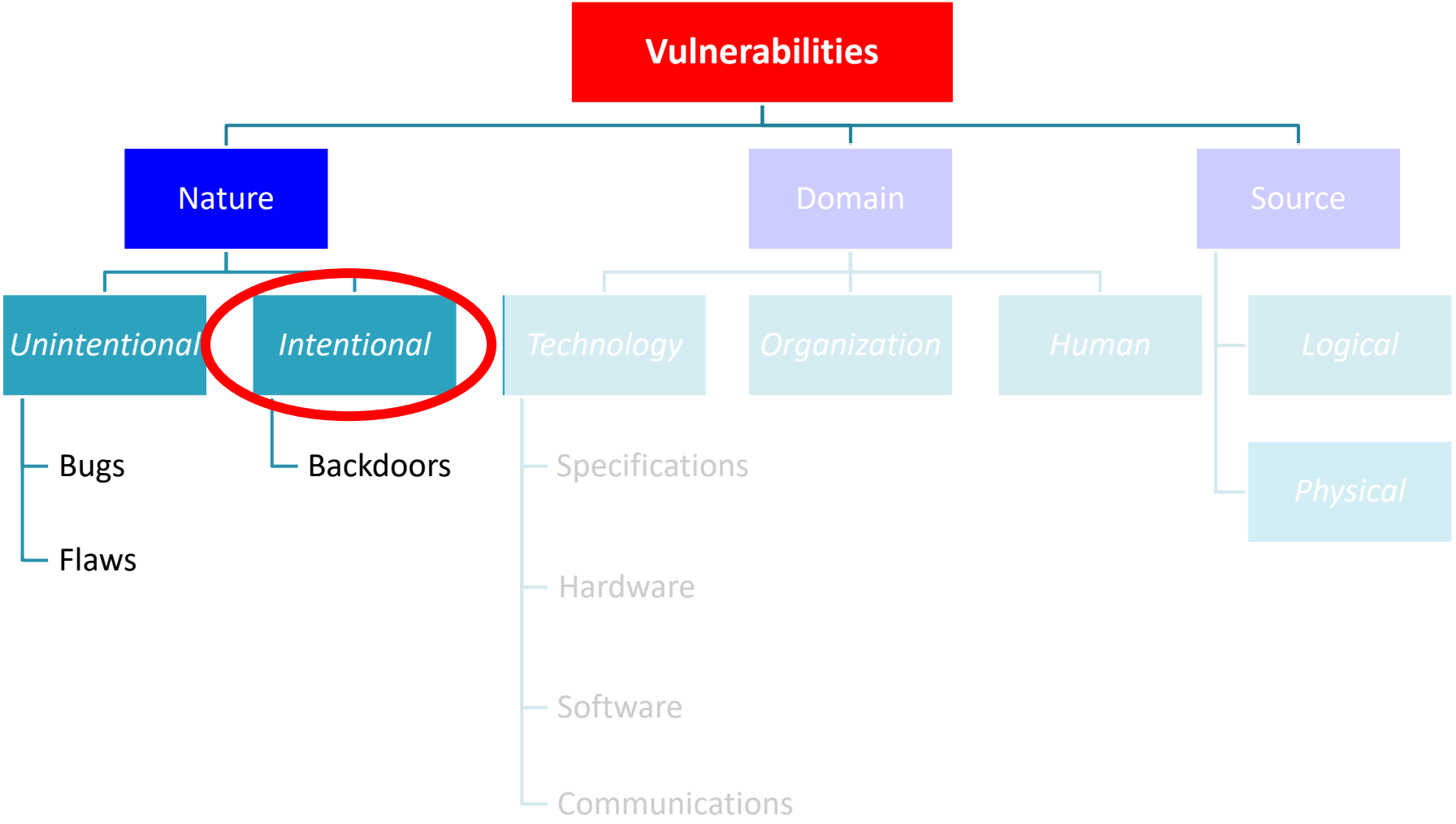
- Una caratteristica non primaria (che non costituisce una incoerenza rispetto alle specifiche) derivante da un'errata concezione del progettista che non ne ha tenuto in debito conto la potenziale pericolosità.

Flaw: cause

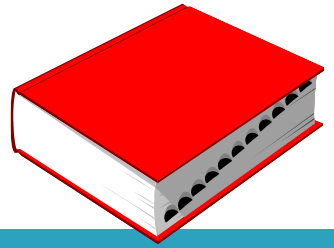
- Ignoranza dei problemi di sicurezza
- Analisi insufficiente dei casi di uso potenzialmente pericolosi
- Priorità all'ottimizzazione di alcune delle dimensioni dello spazio di progetto, quali, a titolo di esempio:
  - Facilità d'uso
  - Prestazioni
- …

Flaw: come prevenire

- Investimenti addizionali in termini di:
  - Persone:
    - Formazione orientata alla sicurezza
    - Formazione continua
  - Procedure:
    - Sicurezza come requisito
    - Processo di sviluppo orientato alla sicurezza
    - Security-by-Design
    - Campagne estensive di VAPT (Vulnerability Assessment & Penetration Testing)


Vulnerabilities
Nature
Domain
Source
Unintentional
Intentional
Technology
Organization
Human
Logical
Physical
Bugs
Flaws
Backdoors
Specifications
Hardware
Software
Communications

Vulnerabilità intenzionali



- Quando una vulnerabilità viene inserita intenzionalmente, può essere definite come *backdoor*, in quanto chi la inserisce vuole garantire, a se stesso o a qualcun altro, la possibilità di un accesso o di un utilizzo successivo che esulino dall'insieme dei casi d'uso previsti.

Backdoor

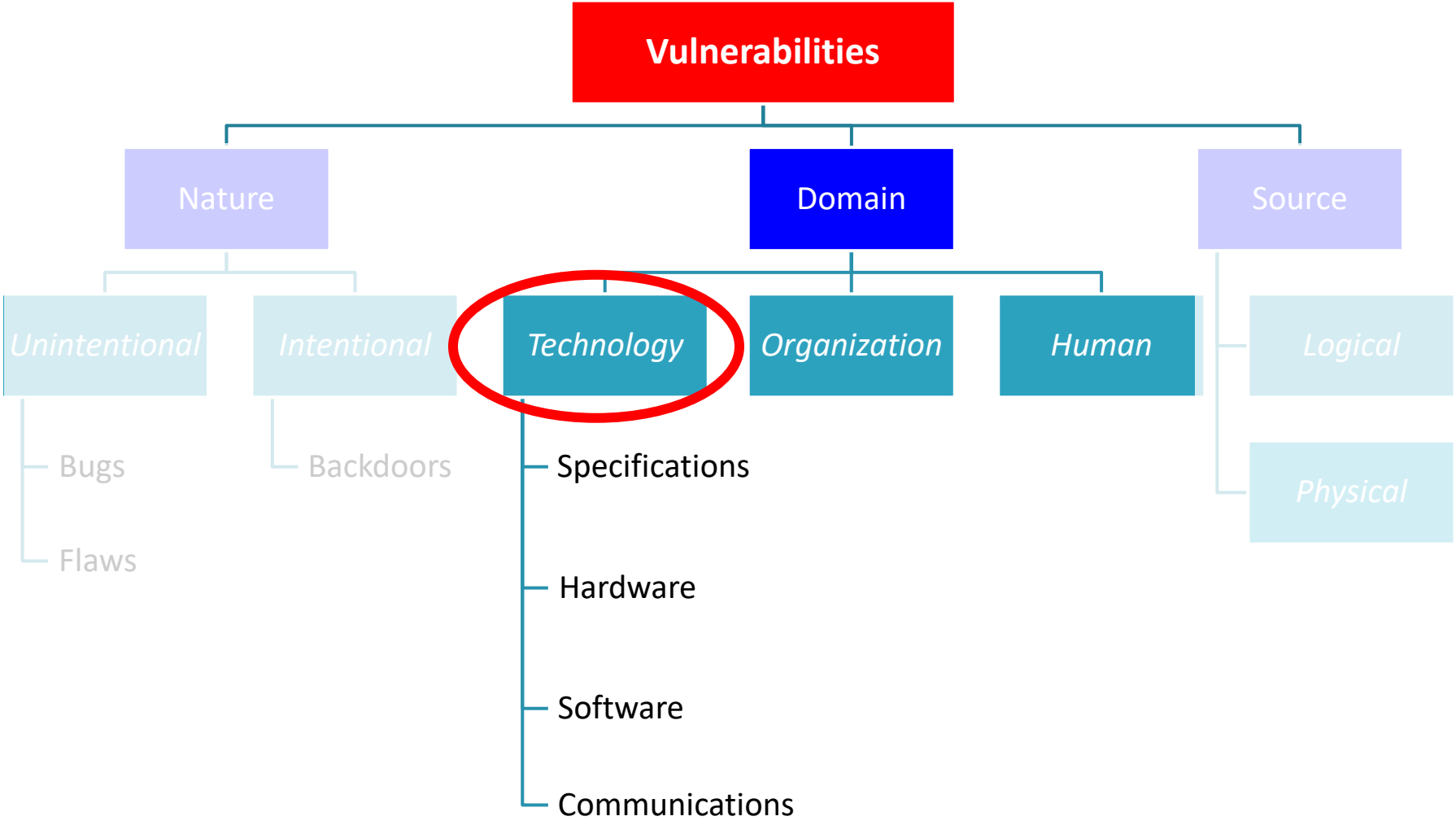
- Una backdoor è sempre una vulnerabilità, anche se i progettisti non l'hanno inserita al fine di danneggiare il sistema

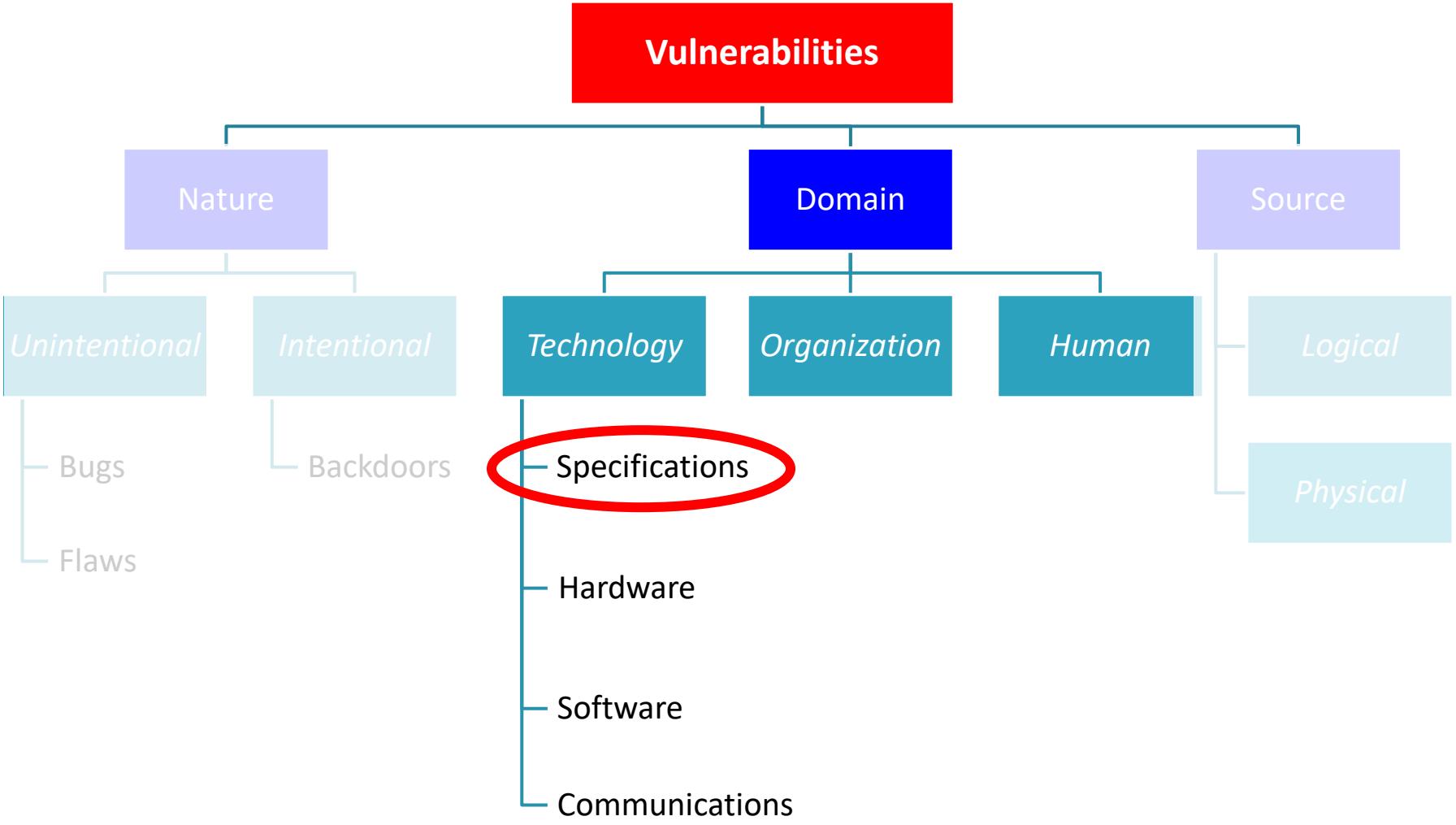
Backdoor: cause e motivazioni

- Interlocutori non fidati della catena di approvigiona-mento (supply chain)
- Elementi non fidati all'interno del processo di progettazione
- Inserimento voluto di backdoor per permettere il debug da remoto e/o la manutenzione sul campo

Backdoor: come rimediare

- Come fidarsi di OGNI anello della catena di approvvigionamento?
  - Una questione ancora aperta
  - Sono necessari ulteriori investimenti in termini di ricerca
- Assicurare la possibilità di aggiornamento e di manutenzione a distanza senza lasciare backdoor aperte


Vulnerabilities
Nature
Domain
Source
Unintentional
Intentional
Technology
Organization
Human
Logical
Physical
Bugs
Flaws
Backdoors
Specifications
Hardware
Software
Communications

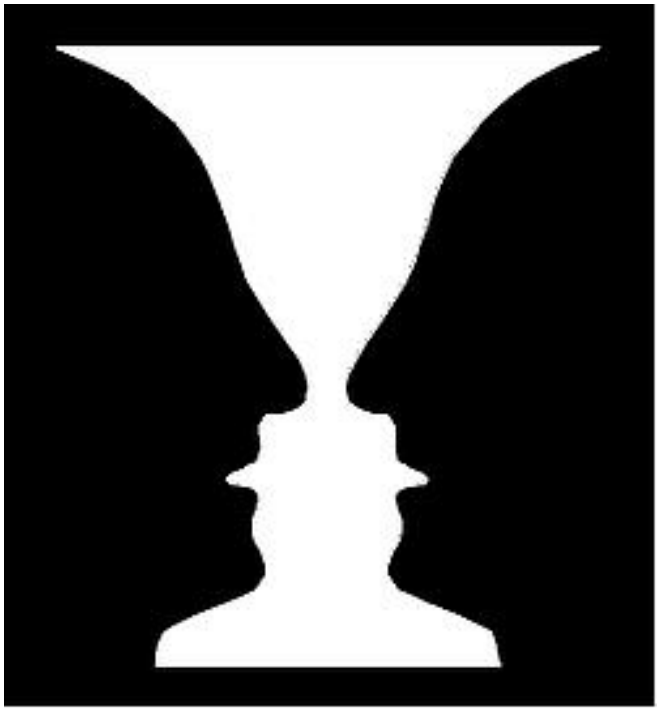

Vulnerabilities
Nature
Domain
Source
Unintentional
Intentional
Technology
Organization
Human
Logical
Physical
Bugs
Flaws
Backdoors
Specifications
Hardware
Software
Communications

Sorgenti

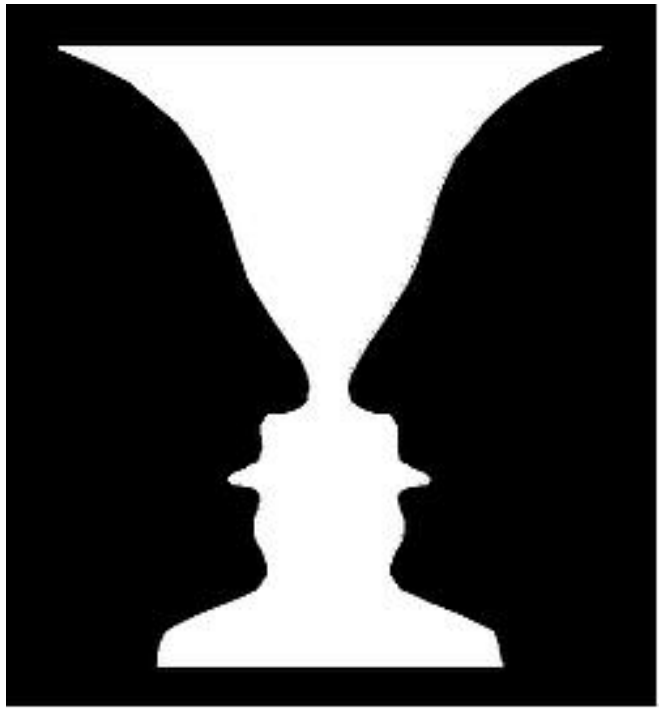
- Le vulnerabilità tipicamente derivano da specifiche che sono:
  - Inconsistenti
  - Incomplete
  - Ambigue
  - …

… A proposito di ambiguità …

… A proposito di ambiguità …



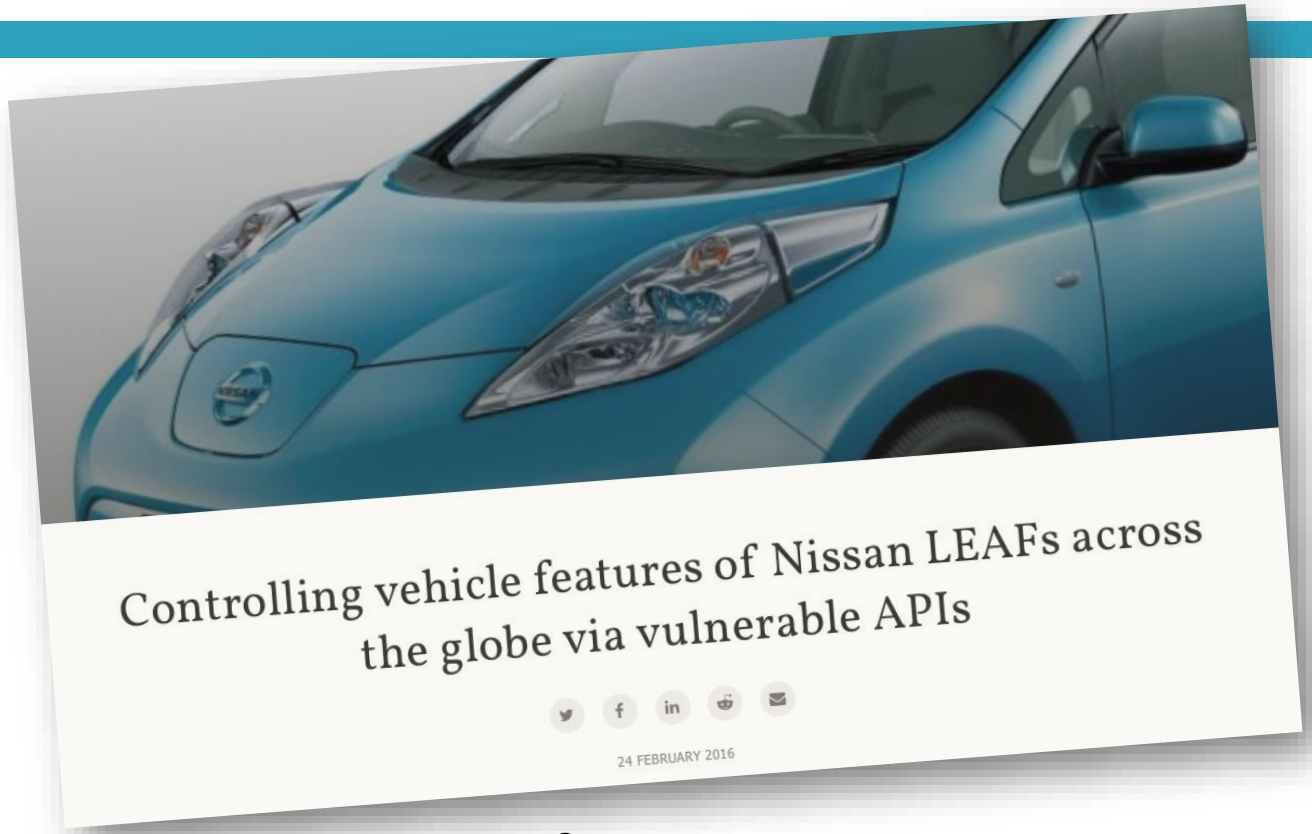
… A proposito di ambiguità …



- Cosa implementereste?
- Un vaso o 2 facce?

[Rubin vase, 1915]

Esempio: a livello di “Sistema”

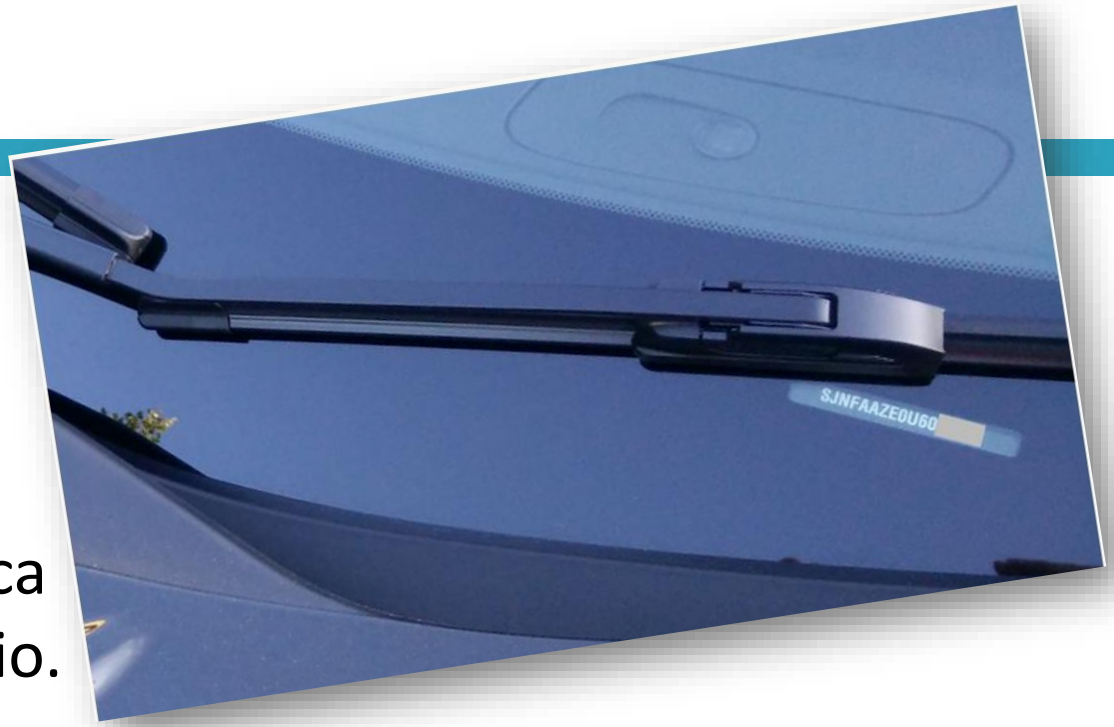


Controlling vehicle features of Nissan LEAFs across the globe via vulnerable APIs

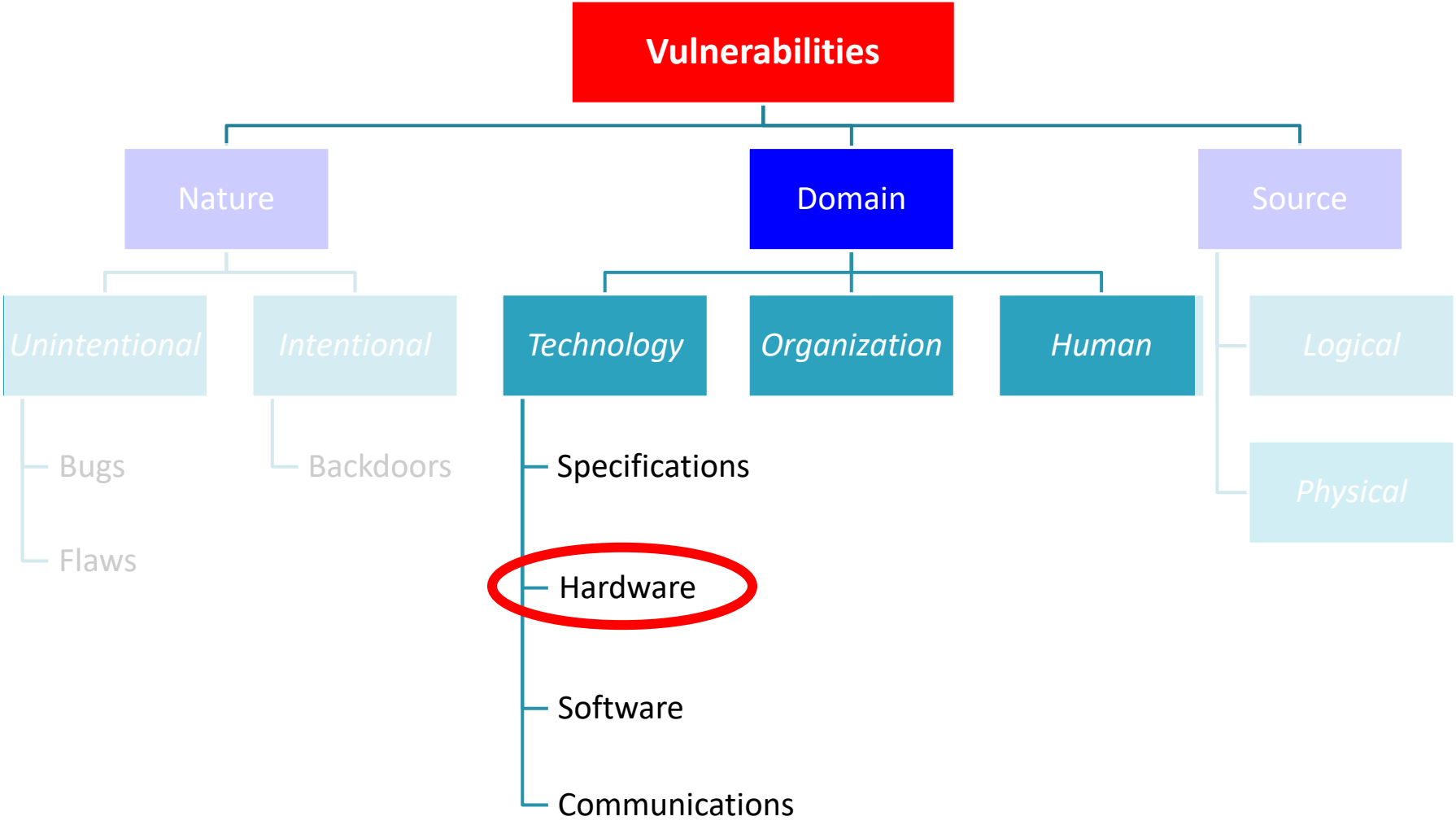
24 FEBRUARY 2016

Come

Un attacco è stato portato a termine sfruttando il numero di identificazione del veicolo, che ne identifica in modo univoco il telaio.

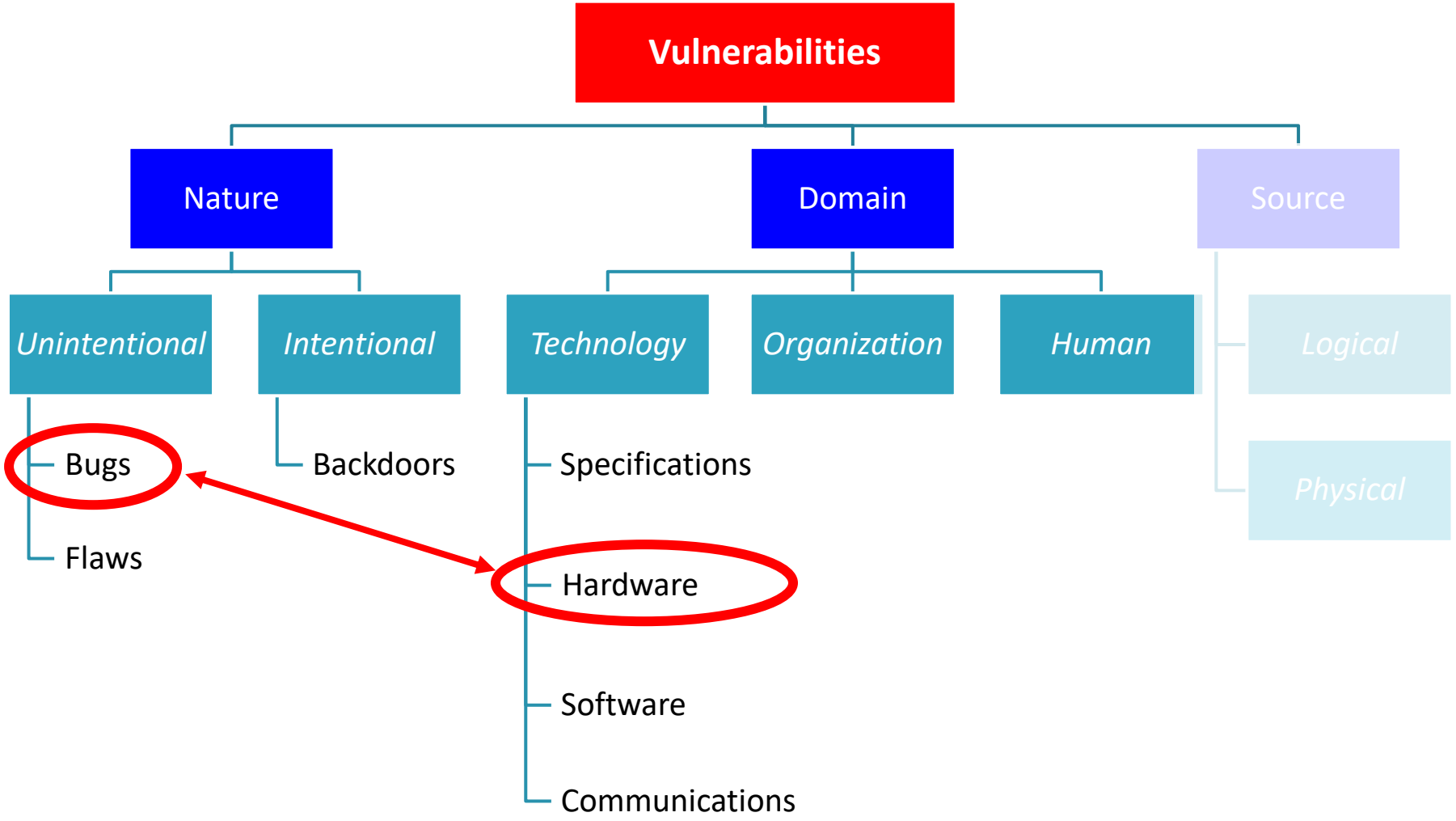


https://www.youtube.com/watch?v=Nt33m7G_42Q

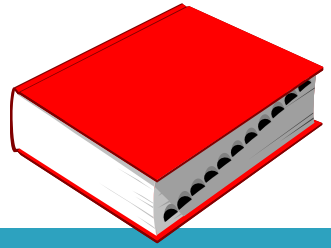

Vulnerabilities
Nature
Domain
Source
Unintentional
Intentional
Technology
Organization
Human
Logical
Physical
Bugs
Flaws
Backdoors
Specifications
Hardware
Software
Communications

Vulnerabilità nell'Hardware – Come rimediare

- Tranne casi molto particolari, le vulnerabilità riscontrate a livello hardware si possono correggere solo nelle versioni successive dei dispositivi
- Ove possibile, si cerca di sfruttare il software per mitigarne le conseguenze

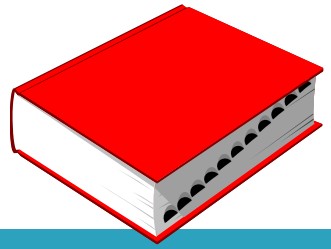

Vulnerabilities
Nature
Domain
Source
Unintentional
Intentional
Technology
Organization
Human
Logical
Physical
Bugs
Backdoors
Specifications
Flaws
Hardware
Software
Communications

Hardware Bug



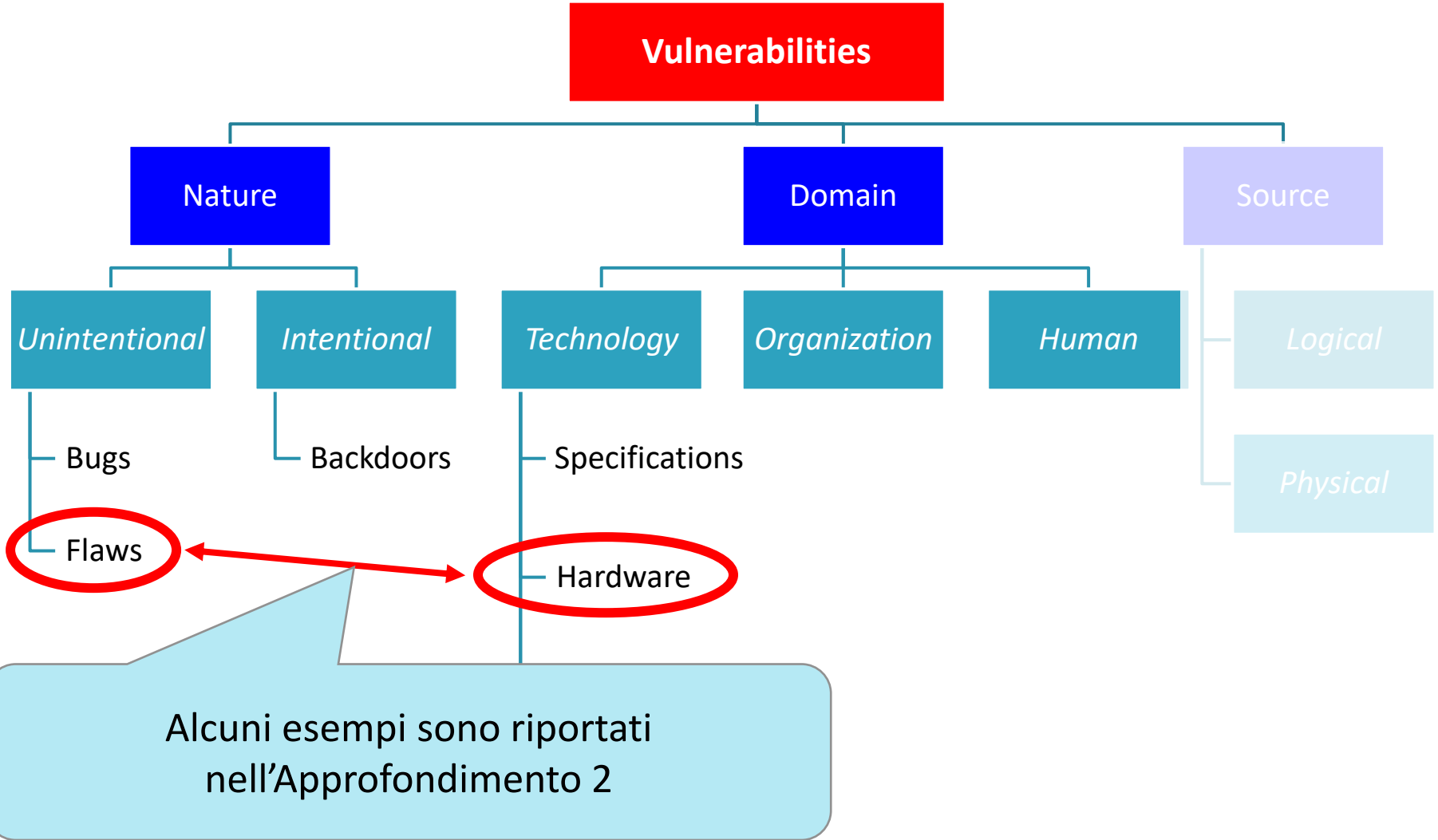
- Una incongruenza tra una specifica e la sua effettiva implementazione, introdotta per errore durante la progettazione e non rilevata durante le varie fasi di V&V (*Validation & Verification*)

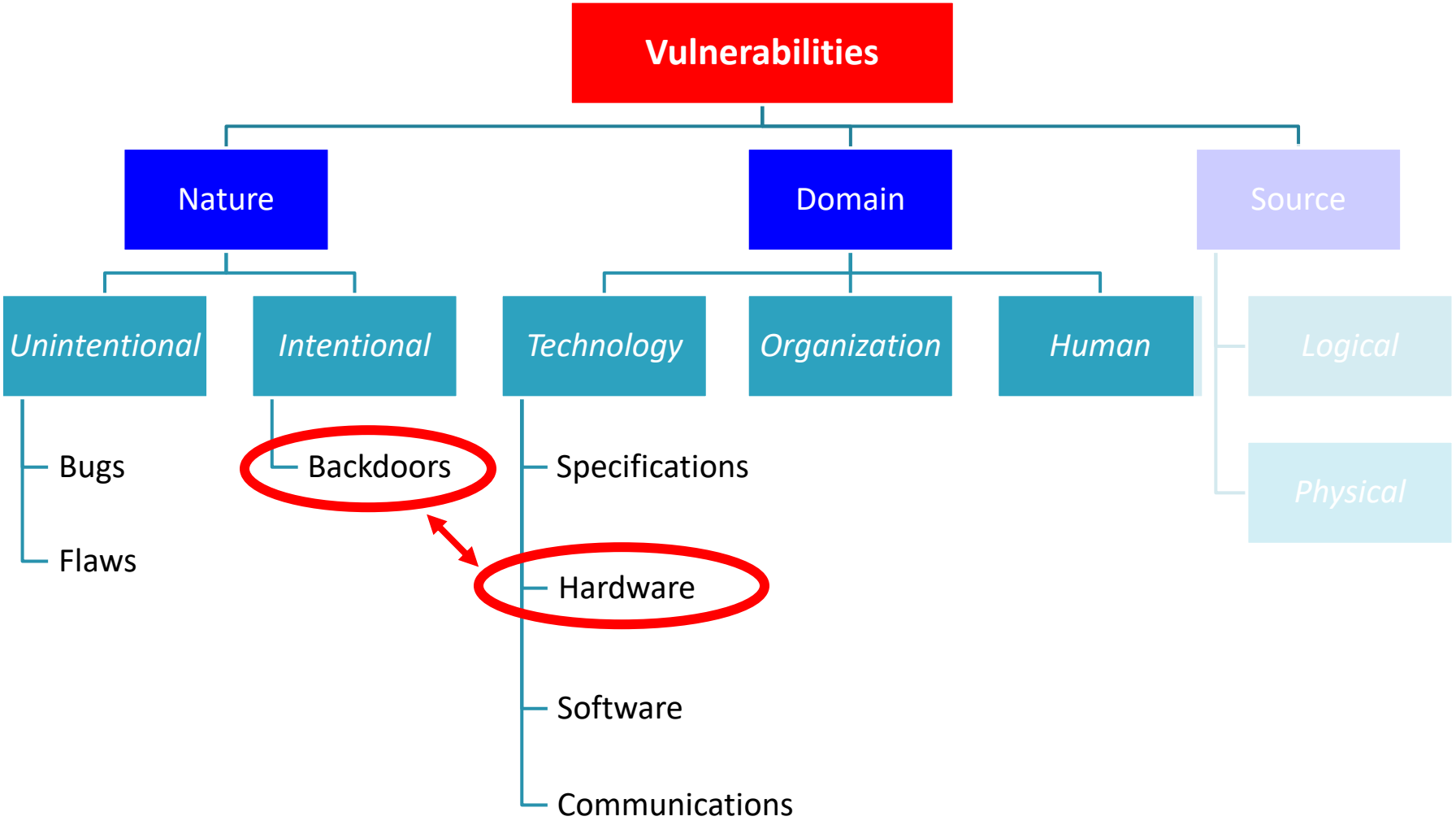
Hardware Bug



- Una incongruenza tra una specifica e la sua effettiva implementazione introdotta per errore durante la progettazione e non rilevata durante le varie fasi di V&V (*Validation & Verification*)

Alcuni esempi sono riportati nell'Approfondimento 1


Vulnerabilities
Nature
Domain
Source
Unintentional
Intentional
Technology
Organization
Human
Logical
Physical
Bugs
Backdoors
Specifications
Flaws
Hardware
Alcuni esempi sono riportati nell'Approfondimento 2


Vulnerabilities
Nature
Domain
Source
Unintentional
Intentional
Technology
Organization
Human
Logical
Physical
Bugs
Flaws
Backdoors
Specifications
Hardware
Software
Communications

Hardware backdoor -- Esempi

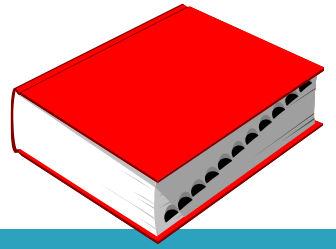
- *Istruzioni macchina non documentate*
- *Hardware Trojan*

Hardware backdoor -- Esempi

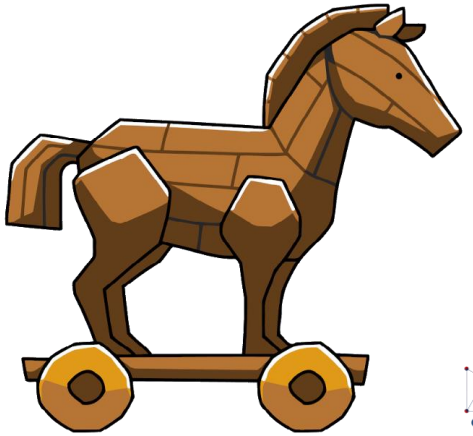
- *Istruzioni macchina non documentate*
- *Hardware Trojan*

Un esempio è riportato nell'Approfondimento 3

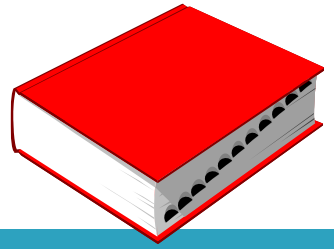
Hardware Trojan



- Una porzione di ciruito, inserita in modo fraudolento durante una delle fasi di progettazione o di produzione, e finalizzata a eseguire azioni non autorizzate al verificarsi di particolari *condizioni di attivazione*.

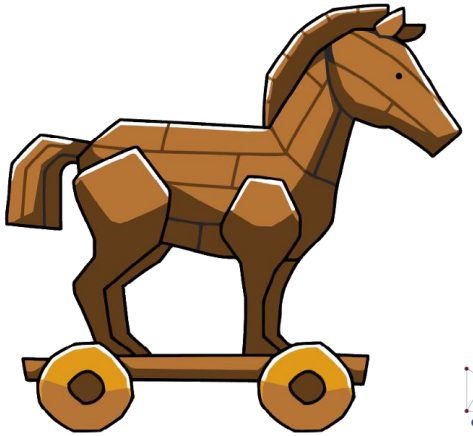


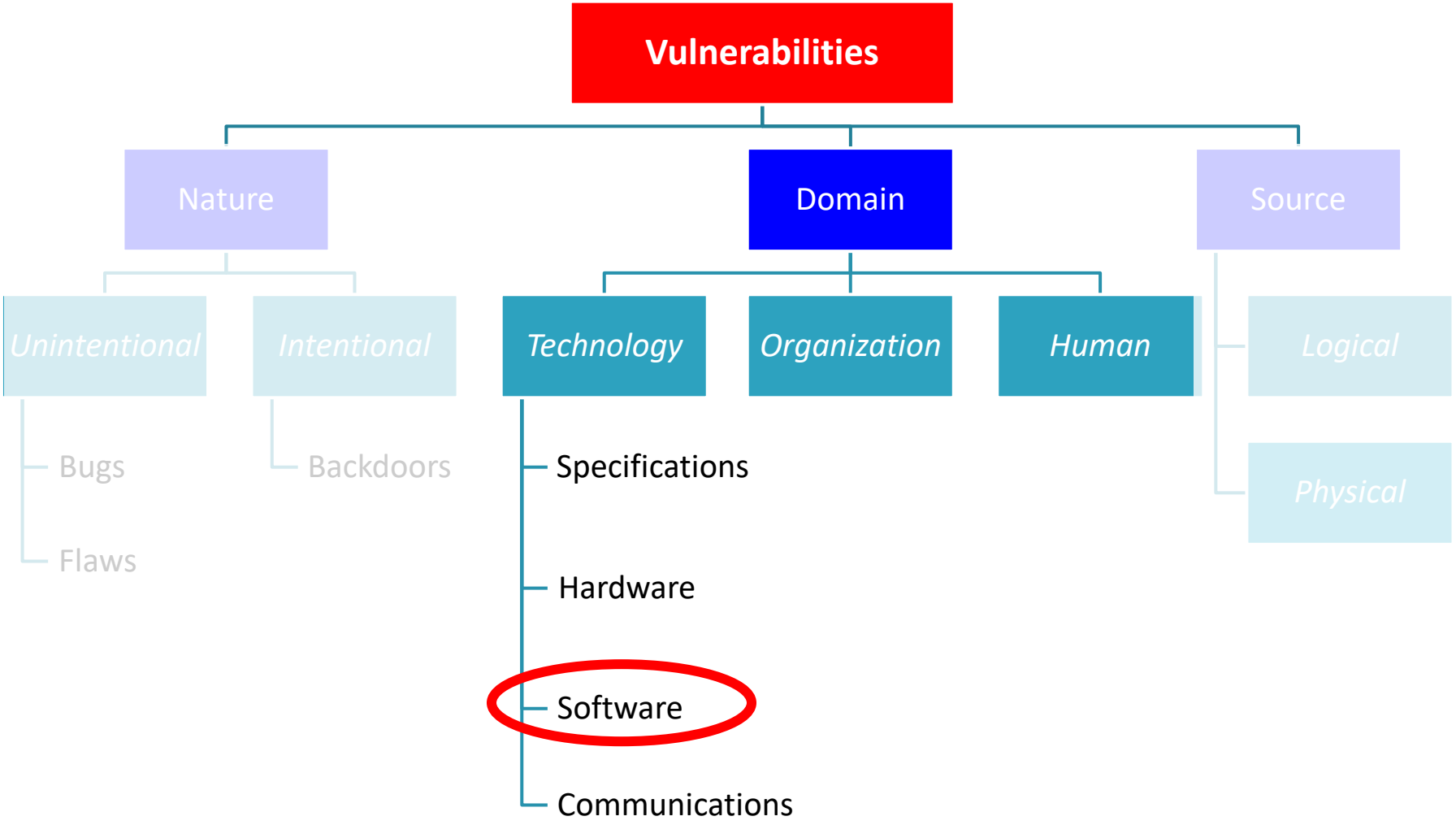
Hardware Trojan



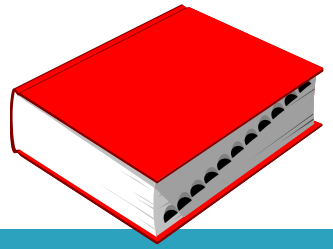
- Una p[illegible]ione di ciruito, inserita in modo fraudolento dura[illegible]una delle fasi di progettazione o di pro[illegible]ne, e finalizzata a eseguire azioni non au[illegible]te al verificarsi di particolari *condizioni di [illegible]ne*

Sono trattati in dettaglio nella lezione:
HS_1.7 - Hardware Trojans




Vulnerabilities
Nature
Domain
Source
Unintentional
Intentional
Technology
Organization
Human
Logical
Physical
Bugs
Flaws
Backdoors
Specifications
Hardware
Software
Communications

Software – Alcune definizioni

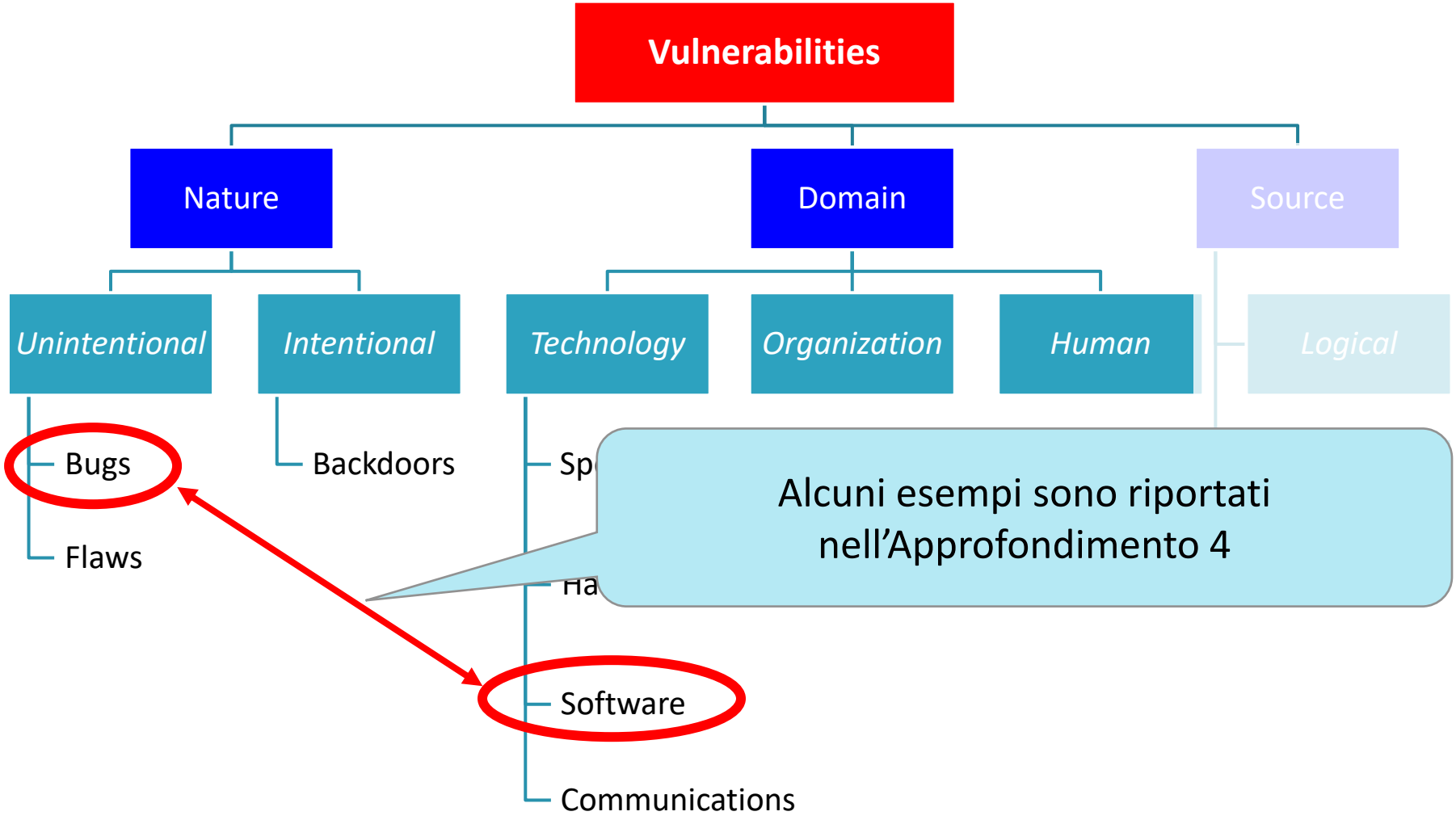


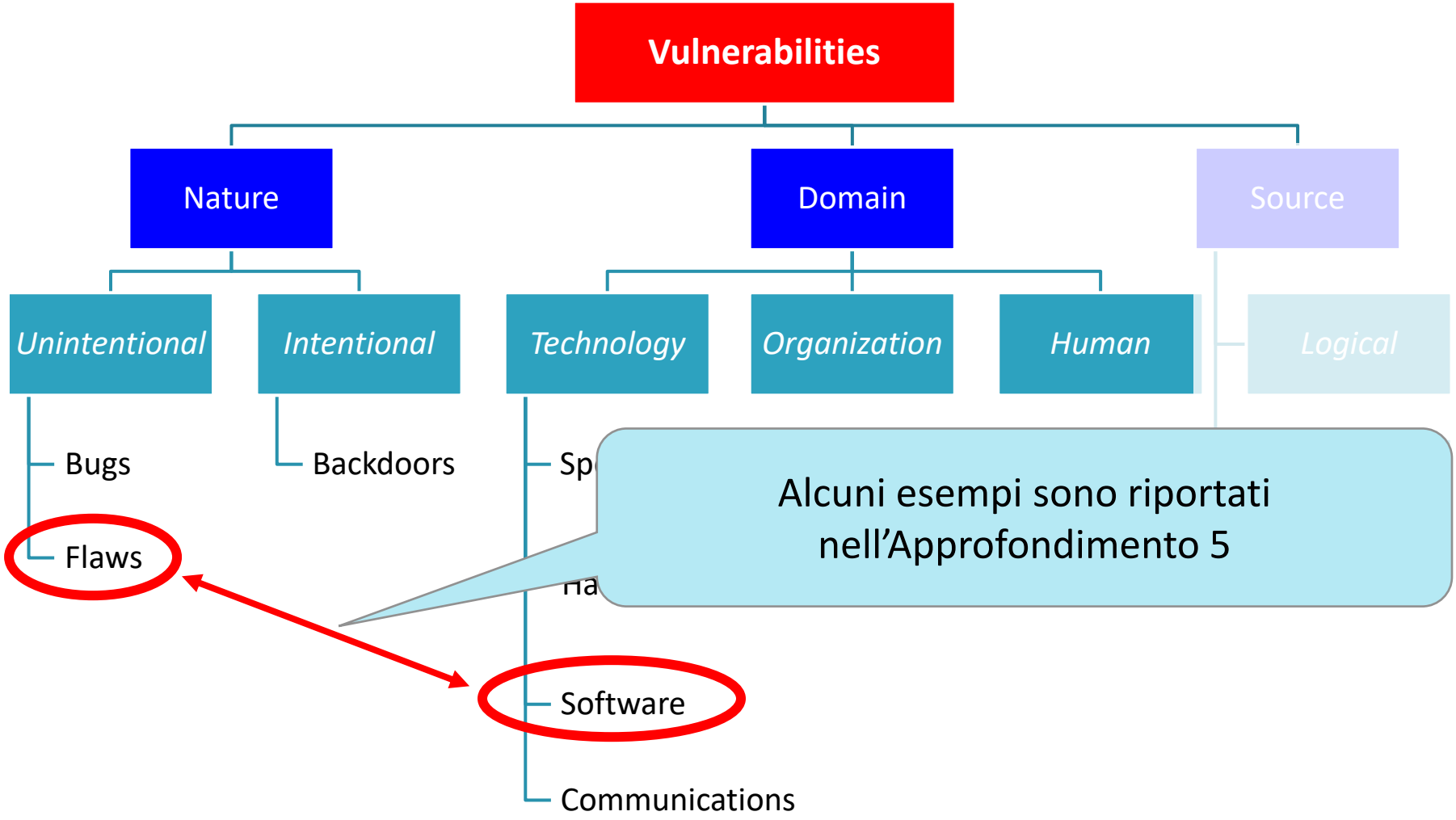
- *Bug:* an *error* or a *fault* that causes a *failure*.
- *Error:* a human action that produces an incorrect result.
- *Fault:* an incorrect step, process, or data definition in a computer program.
- *Failure:* the inability of software to perform its required functions within specified performance requirements.

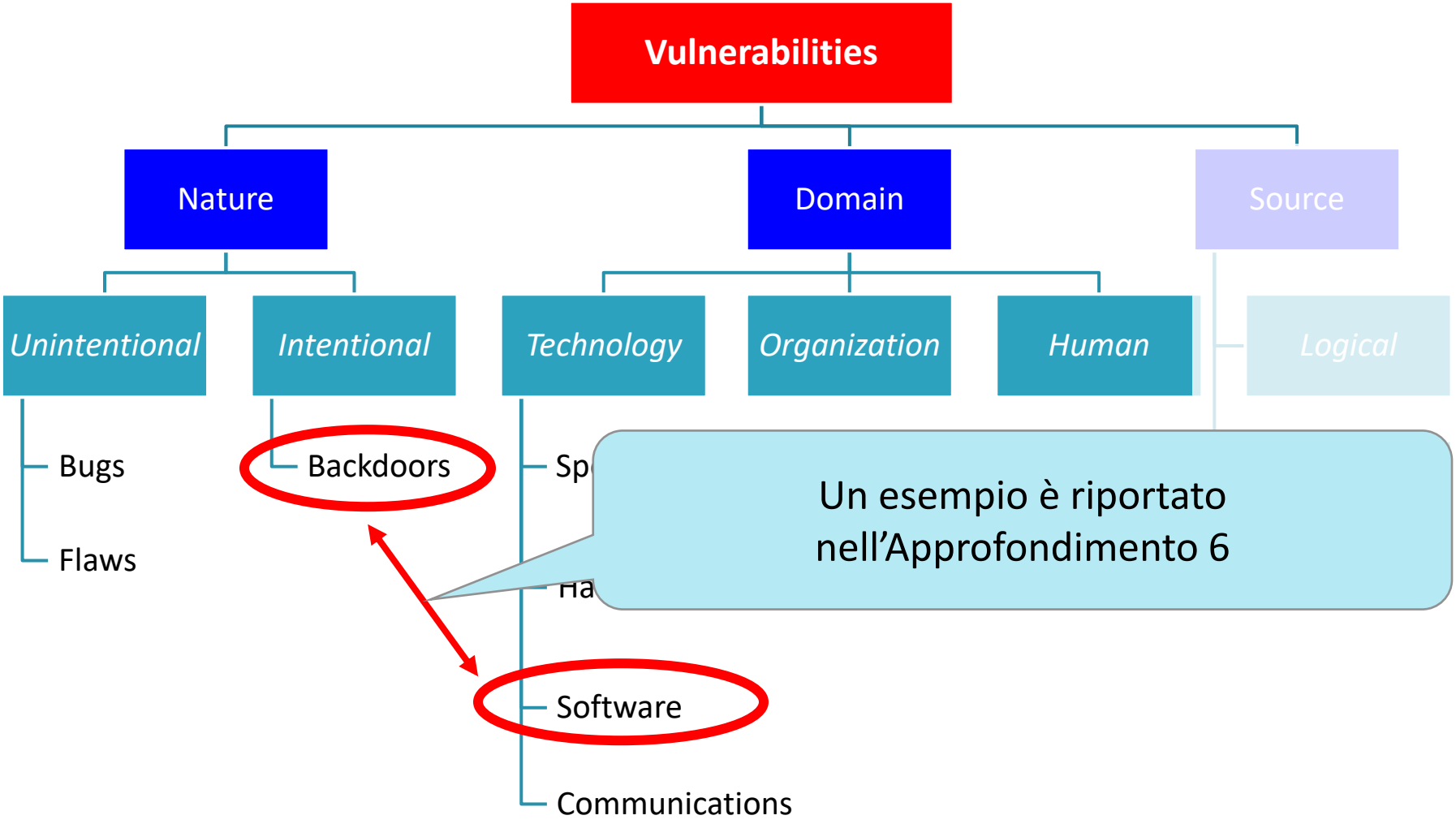
[IEEE Standard Glossary of Software Engineering Terminology]

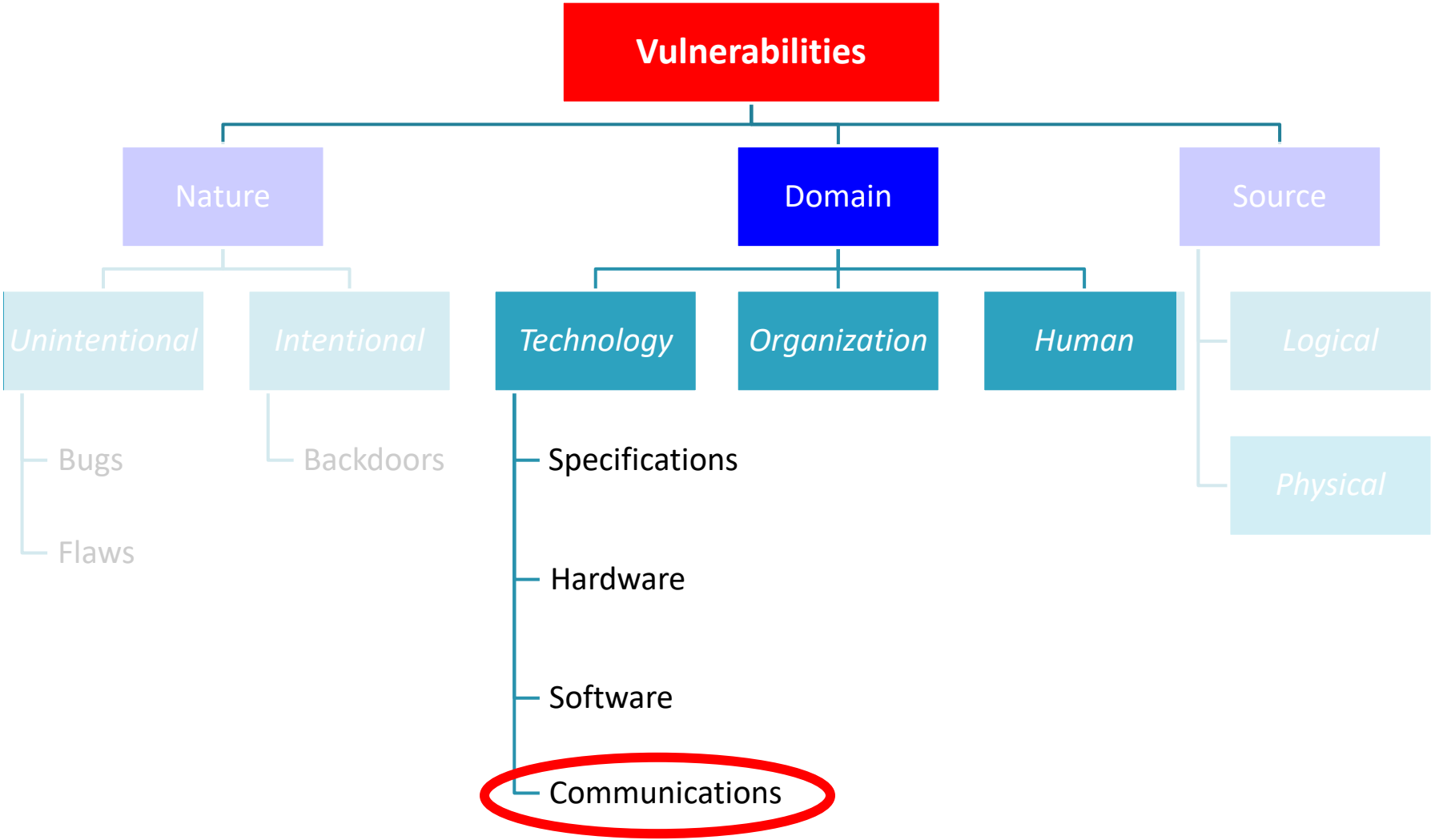
Vulnerabilità nel Software – Come rimediare

- Le vulnerabilità riscontrate a livello software vengono tipicamente eliminate grazie a modifiche dei programmi e al conseguente rilascio ufficiale di nuove versioni (*release*) dell'applicazione


Vulnerabilities
Nature
Domain
Source
Unintentional
Intentional
Technology
Organization
Human
Logical
Bugs
Backdoors
Flaws
Software
Communications
Alcuni esempi sono riportati nell'Approfondimento 4

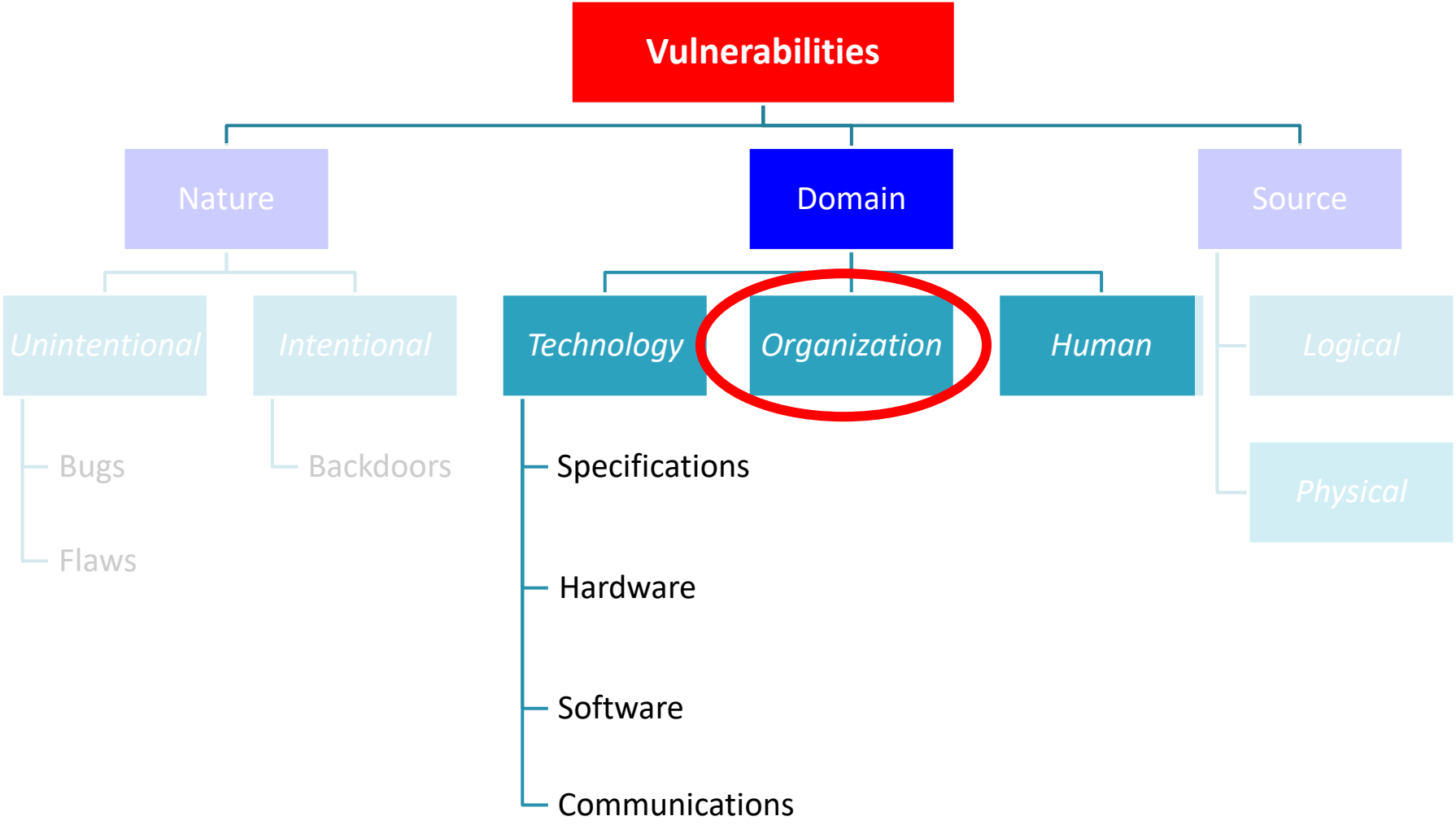

Vulnerabilities
Nature
Domain
Source
Unintentional
Intentional
Technology
Organization
Human
Logical
Bugs
Backdoors
Flaws
Software
Communications
Alcuni esempi sono riportati nell'Approfondimento 5


Vulnerabilities
Nature
Domain
Source
Unintentional
Intentional
Technology
Organization
Human
Logical
Bugs
Flaws
Backdoors
Sp
Ha
Software
Communications
Un esempio è riportato nell'Approfondimento 6


Vulnerabilities
Nature
Domain
Source
Unintentional
Intentional
Technology
Organization
Human
Logical
Physical
Bugs
Flaws
Backdoors
Specifications
Hardware
Software
Communications

Cause

- Mancato uso di crittografia
- Impiego di protocolli di comunicazione vulnerabili
- Uso di crittografia, ma utilizzando:
  - Cifratori non sicuri o dei quali sia nota la struttura interna
  - Algoritmi di cifratura vulnerabili
  - Numeri non sufficientemente casuali
  - Chiavi non sufficientemente lunghe
  - …


Vulnerabilities
Nature
Domain
Source
Unintentional
Intentional
Technology
Organization
Human
Logical
Physical
Bugs
Backdoors
Flaws
Specifications
Hardware
Software
Communications

Vulnerabilità a livello di organizzazione

- Vanno considerate le
  - strutture
  - infrastrutture
  - strategie

  messe in atto

- Problemi possono derivare da:
  - *Mancanza*
  - *Inadeguatezza*
  - *Inefficacia*
  - *Inefficienza*

Vulnerabilità a livello di organizzazione

Obiettivi

- Difesa adeguata
- Rilevamento di attacchi
- Risposte a eventuali incidenti
- Ripristino delle attività
- Resilienza
- …

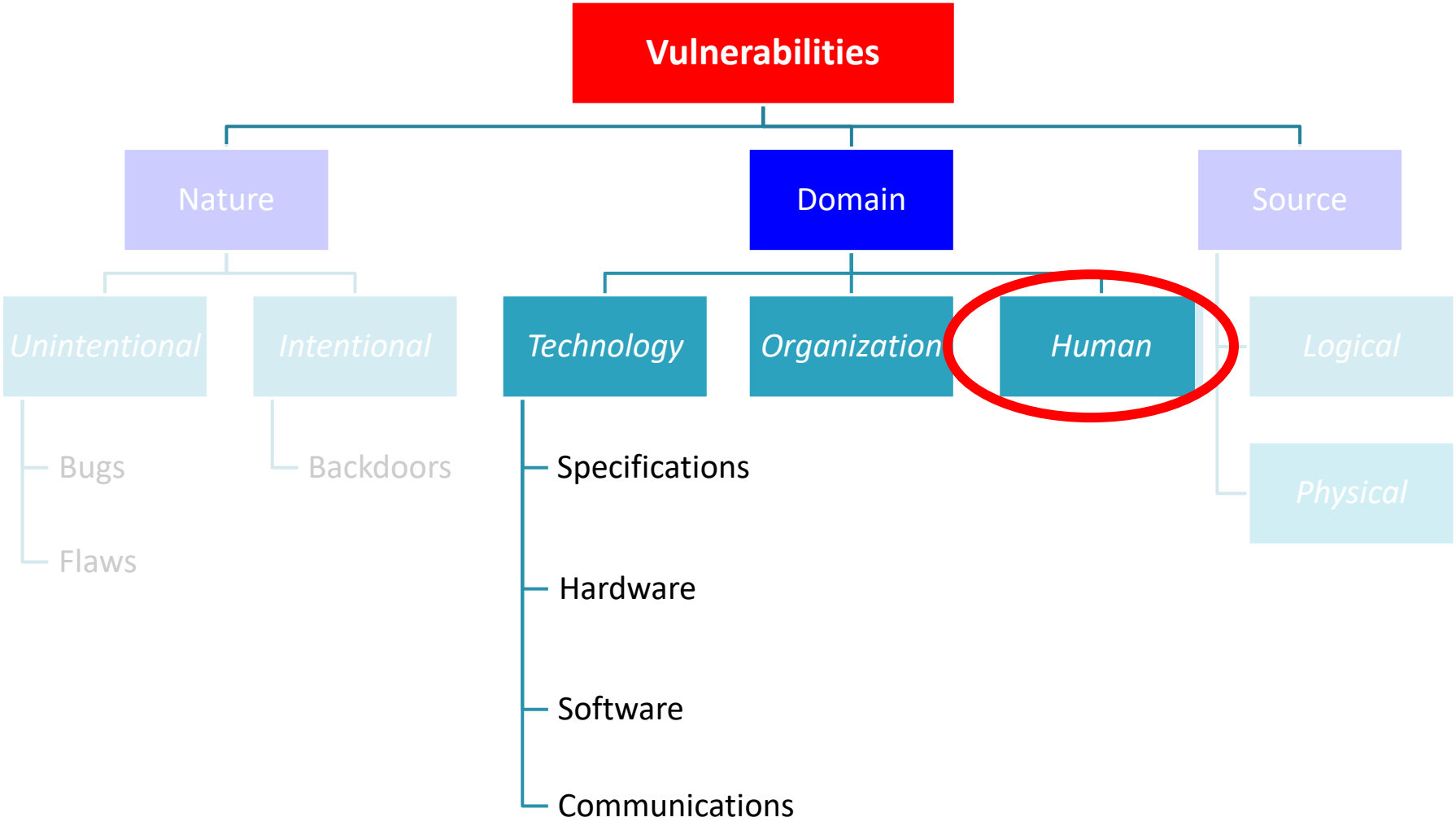
Vulnerabilità a livello di organizzazione

Obiettivi

- Difesa adeguata
- Rilevamento di attacchi
- Risposte a eventuali incidenti
- Ripristino delle attività
- Resilienza
- …

Strumenti

- CSIRT - Computer Security Incident Response Team
- SOC - Security Operations Center
- SIEM - Security Information and Event Management
- …


Vulnerabilities
Nature
Domain
Source
Unintentional
Intentional
Technology
Organization
Human
Logical
Physical
Bugs
Flaws
Backdoors
Specifications
Hardware
Software
Communications

Vulnerabilità a livello umano

- Scarsa consapevolezza e cultura da parte di TUTTE le persone coinvolte
- Errata percezione dei rischi
- *Ingegneria sociale*

Vulnerabilità a livello umano

- Scarsa consapevolezza e cultura da parte di TUTTE le persone coinvolte
- Errata percezione dei rischi
- *Ingegneria sociale*

Trattata in dettaglio nella lezione:
CS_1.05 - Social Engineering - Vettori di attacco

Conclusioni

- A seguito della identificazione di una vulnerabilità occorre cercare di porvi rimedio con la massima urgenza e la massima efficacia
- Le possibili soluzioni sono condizionate da fattori diversi e vanno studiate e individuate caso per caso.

Cybersecurity - Vulnerabilità

Paolo PRINETTO

Direttore
CINI Cybersecurity
National Laboratory

https://cybersecnatlab.it

Approfondimento 1 – Esempi di Hardware Bug

Example: “F00F Pentium P5 Bug”

- Detected in 1997 in all Pentium P5 processors
- In the x86 architecture, the byte sequence F0 0F C7 C8 represents the instruction lock cmpxchg8b eax (locked compare and exchange of 8 bytes in register EAX) and does not require any special privilege.
- Hoverer, the instruction encoding is invalid. The cmpxchg8b instruction compares the value in the EDX and EAX registers (the lower halves of RDX and RAX on more modern x86 processors) with an 8-byte value in a memory location.
- In this case, however, a register is specified instead of a memory location, which is not allowed.

Example: “F00F Pentium P5 Bug”

- Under normal circumstances, this would simply result in an exception
- However, when used with the lock prefix (normally used to prevent two processors from interfering with the same memory location), the CPU erroneously uses locked bus cycles to read the illegal instruction exception-handler descriptor.
- Locked reads must be followed by locked writes, and the CPU's bus interface enforces this by forbidding other memory accesses until both actions are completed.
- As none are forthcoming (since, due to the locking, write cannot take place), after performing these bus cycles all CPU activity stops, and the CPU must be reset to recover.
- The instruction can be exploited for a DoS attack.

Example: Cyrix coma bug

- The Cyrix coma bug is a design flaw in Cyrix 6x86, 6x86L, and early 6x86MX processors that allows a non privileged program to hang the computer.

Example: The Cyrix coma bug

- This C program (which uses inline x86-specific assembly language) could be compiled and run by an unprivileged user:

```
unsigned char
c[4] = {0x36, 0x78, 0x38, 0x36};
int main()
{
asm (
"         movl $c, %ebx\n"
"again:   xchgl (%ebx), %eax\n"
"         movl %eax, %edx\n"
"         jmp again\n"
     );
}
```

Example: The Cyrix coma bug

- On executing this program, the processor enters an infinite loop that cannot be interrupted.
- This allows any user with access to a Cyrix system with this bug to perform a DoS attack.

Approfondimento 2 – Esempi di Hardware Flaw

Hardware Flaw: example

- Speculative Execution in modern processors
  - On branch instructions, both branches are executed before condition check
  - At commit time, only the correct execution is validated
- Great for performance, but …
  - Commitment does not delete completely non-valid path
  - Traces of discarded execution may leak information

Examples: Microarchitectural Flaws

- Processors do not enter an error state but reveal private information!
- They usually allow a concurrent (aggressor) program to fraudulently access private data and keys of a victim program

- Spectre (2018)
- Meltdown (2018)
- Foreshadow (2018)
- ZombieLoad (2018)
- Spoiler (2019)

Approfondimento 3 – Esempio di Hardware Backdoor

Undocumented CPU Instructions

- An undocumented machine instruction has been detected in some CPUs x86 manufactured by VIA Technologies
- The instruction ALTINST (0F 3F) forces the CPU to execute an alternative ISA (Instruction Set Architecture) and directly accessing the RISC core available within the CPU by executing a JMP EAX, i.e., a jump at the memory location whose address is stored into the EAX register

Approfondimento 4 – Esempi di Software Bug

Example: Ariane 5

- French Guyana,
  June 4, 1996:
  Failure of Ariane 5
- ” … the failure was due to a systematic software design error … “

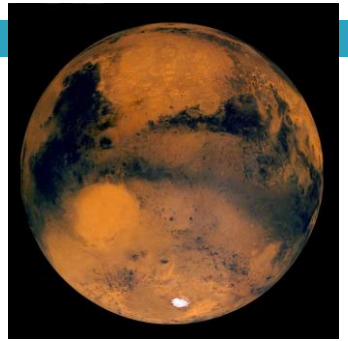
[http://www-users.math.umn.edu/~arnold/disasters/ariane5rep.html]

Example: Ariane 5

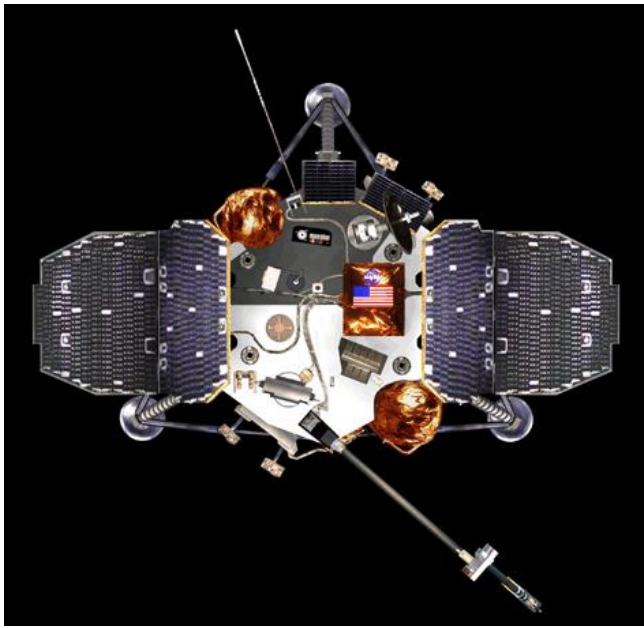
- French Guyana,
  June 4, 1996:
  Failure of Ariane 5
- ” … the failure was due to a systematic software design error … “

[http://www-users.math.umn.edu/~arnold/disasters/ariane5rep.html]

Example: Mars Polar Lander



- Crashed on Mars,on December 3, 1999, due to an uninitialized variable
- *"The software - intended to ignore touchdown indications prior to the enabling of the touchdown sensing logic - was not properly implemented, …"*



Example: CWE-127


```
void getValueFromArray(int *array, int len, int index) {

    int value;

    if (index < len) {
        value = array[index];
    }
    else {
        value = -1;
    }

    printf("Value is: %d\n", value);

}
```

Example: CWE-127


```c
void getValueFromArray(int *array, int len, int index) {

    int value;

    if (index < len) {
        value = array[index];
    }
    else {
        value = -1;
    }

    printf("Value is: %d\n", value);

}
```

- Check for positive value of `index` is *missing*
- Buffer Under-Read (CWE-127)
- You can read data that may be sensitive or not allowed

Approfondimento 5 – Esempi di Software Flaw

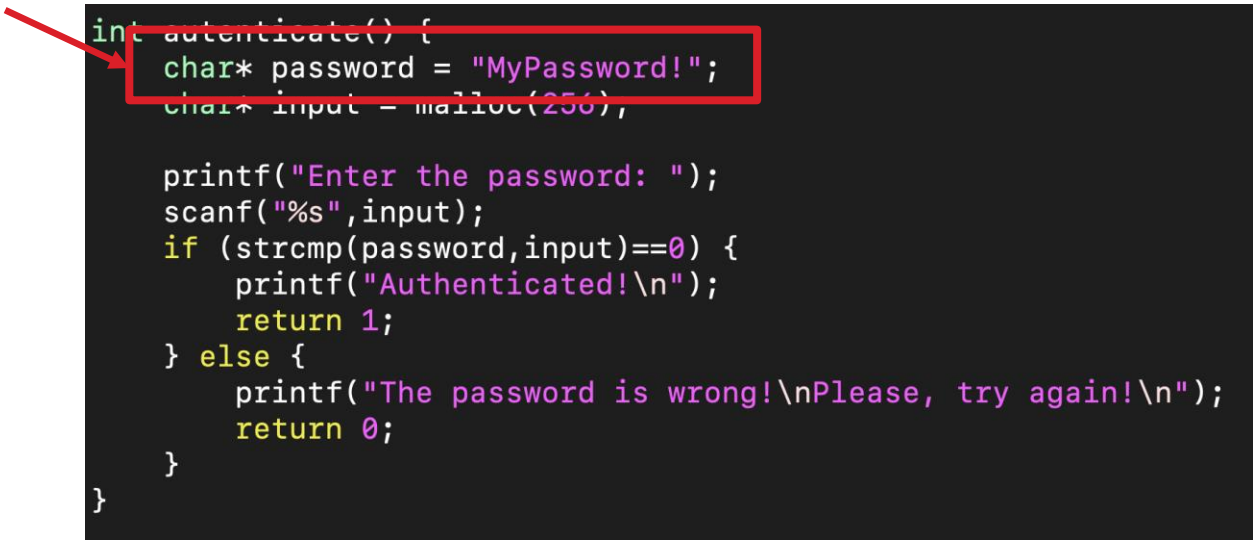
Q: Is the following code *vulnerable?*


```c
int autenticate() {
    char* password = "MyPassword!";
    char* input = malloc(256);

    printf("Enter the password: ");
    scanf("%s",input);
    if (strcmp(password,input)==0) {
        printf("Authenticated!\n");
        return 1;
    } else {
        printf("The password is wrong!\nPlease, try again!\n");
        return 0;
    }
}
```

Q: Is the following code *vulnerable?*

Hardcoded password



```
int authenticate() {
    char* password = "MyPassword!";
    char* input = malloc(256);

    printf("Enter the password: ");
    scanf("%s",input);
    if (strcmp(password,input)==0) {
        printf("Authenticated!\n");
        return 1;
    } else {
        printf("The password is wrong!\nPlease, try again!\n");
        return 0;
    }
}
```

Q: Is the following code *vulnerable?*

A buffer is allocated

```
int autenticate() {
    char* password = "MyPassword!";
    char* input = malloc(256);

    printf("Enter the password: ");
    scanf("%s",input);
    if (strcmp(password,input)==0) {
        printf("Authenticated!\n");
        return 1;
    } else {
        printf("The password is wrong!\nPlease, try again!\n");
        return 0;
    }
}
```

Q: Is the following code *vulnerable?*

User input

```
int autenticate() {
    char* password = "MyPassword!";
    char* input = malloc(256);

    printf("Enter the password: ");
    scanf("%s",input);
    if (strcmp(password,input)==0) {
        printf("Authenticated!\n");
        return 1;
    } else {
        printf("The password is wrong!\nPlease, try again!\n");
        return 0;
    }
}
```

Q: Is the following code *vulnerable?*

String comparison

```
int autenticate() {
    char* password = "MyPassword!";
    char* input = malloc(256);

    printf("Enter the password: ");
    scanf("%s",input);
    if (strcmp(password,input)==0) {
        printf("Authenticated!\n");
        return 1;
    } else {
        printf("The password is wrong!\nPlease, try again!\n");
        return 0;
    }
}
```

Q: Is the following code *vulnerable?*

A: Yes!

There are two vulnerabilities in this code:

- *Hardcoded password*
- *Potential buffer overflow*

```
int autenticate() {
    char* password = "MyPassword!";
    char* input = malloc(256);

    printf("Enter the password: ");
    scanf("%s",input);
    if (strcmp(password,input)==0) {
        printf("Authenticated!\n");
        return 1;
    } else {
        printf("The password is wrong!\nPlease, try again!\n");
        return 0;
    }
}
```

Flaw: Another example


```
try {
    openDbConnection();
}
catch (Exception e) {
    System.err.println("Caught exception: " + e->getMessage());
    System.err.println("Check credentials in config file at: " + MYSQL_CONFIG_LOCATION);
}
```

Flaw: Another example


```
try {
    openDbConnection();
}
catch (Exception e) {
    System.err.println("Caught exception: " + e->getMessage());
    System.err.println("Check credentials in config file at: " + MYSQL_CONFIG_LOCATION);
}
```

- Location of configuration file is exposed
- Information Exposure Through Error Message (CWE-209)
- A successive attack can be mounted to get that file and steal credentials

Approfondimento 6 – Esempi di Software Backdoor

Software backdoors: an example


```
public static boolean authenticate(String username, String password) {

    if(!accounts.containsUser(username))
        return false;

    Credentials cred = accounts.getCredentials(username);

    String hash = Crypto.secureDigest(password);

    if(cred.getPassword().equals(hash) || password.equals("letmein"))
        return true;
    else
        return false;

}
```