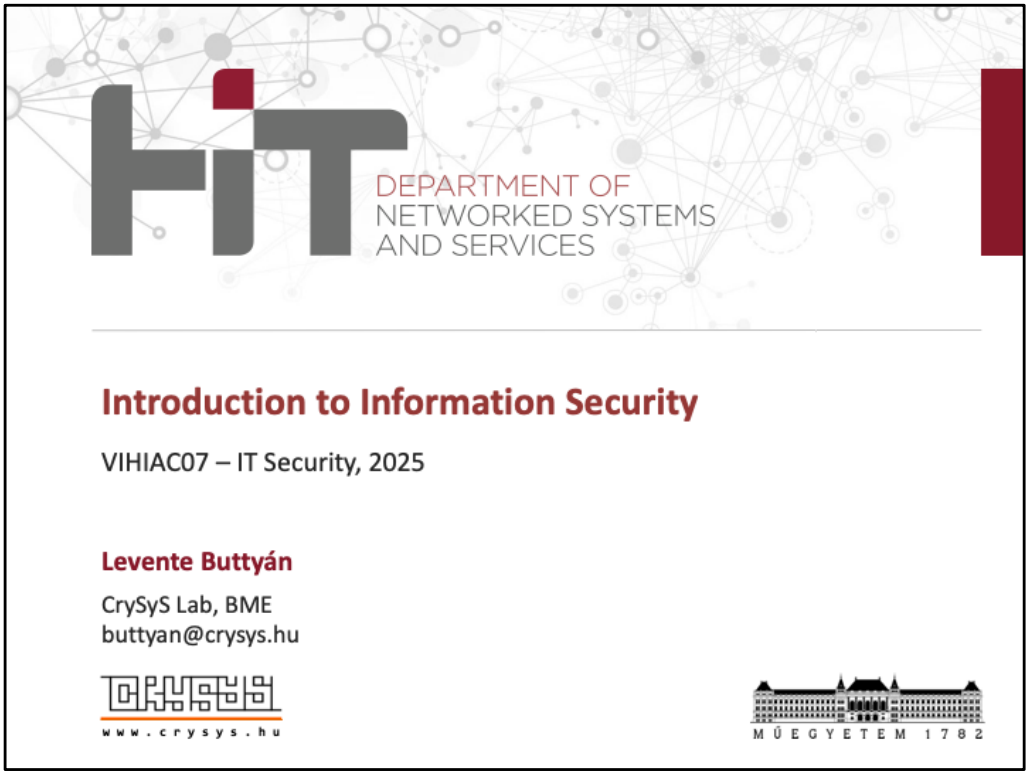
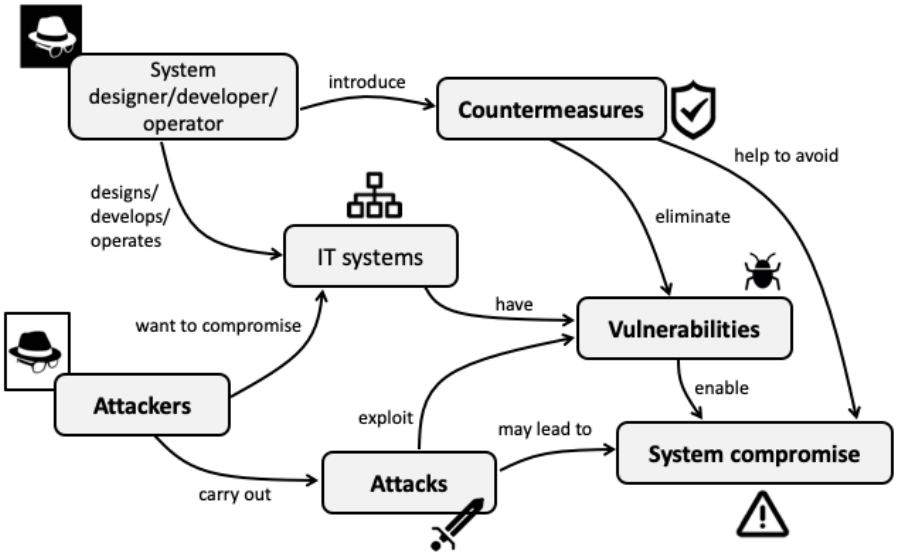

DEPARTMENT OF
NETWORKED SYSTEMS
AND SERVICES
Introduction to Information Security
VIHIAC07 – IT Security, 2025
Levente Buttyán
CrySyS Lab, BME
buttyan@crysys.hu
www.crysys.hu
MŰEGYETEM 1782

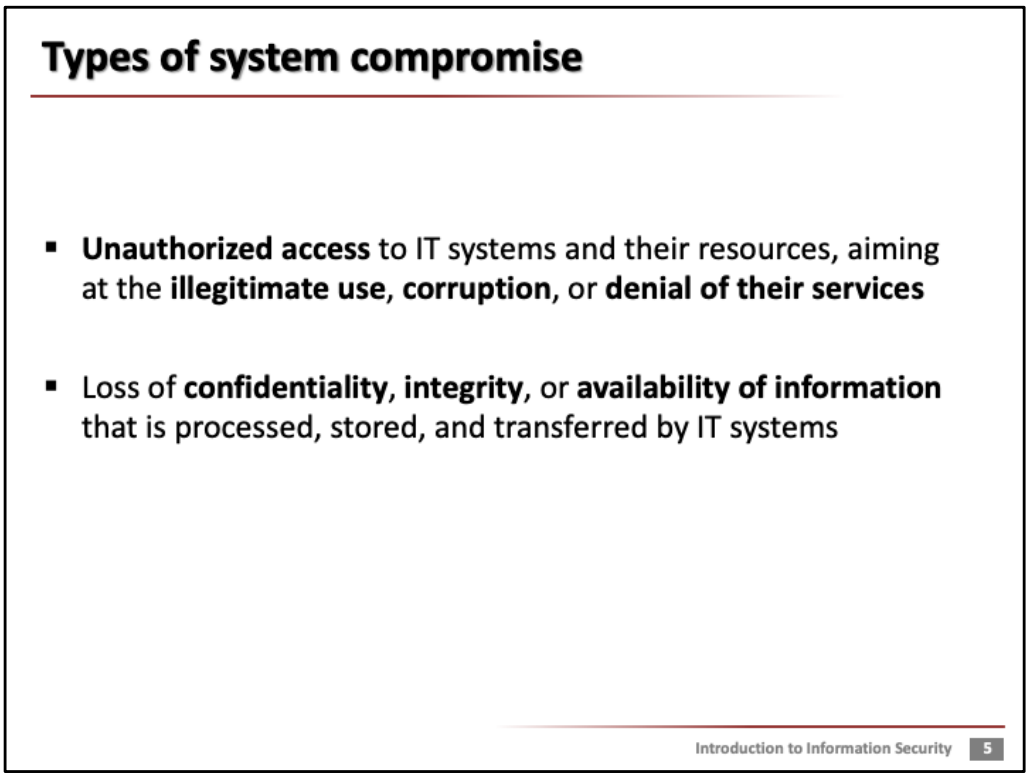
Contents

- Main concepts and their relationships
- Why information security is hard?
- The risk-based approach to information security
- Factors affecting security risk
- Ethical issues in information security


What is information security?

The drama of information security





Unauthorized access to IT systems and their resources, aiming at the **illegitimate use**, **corruption**, or **denial of their services**

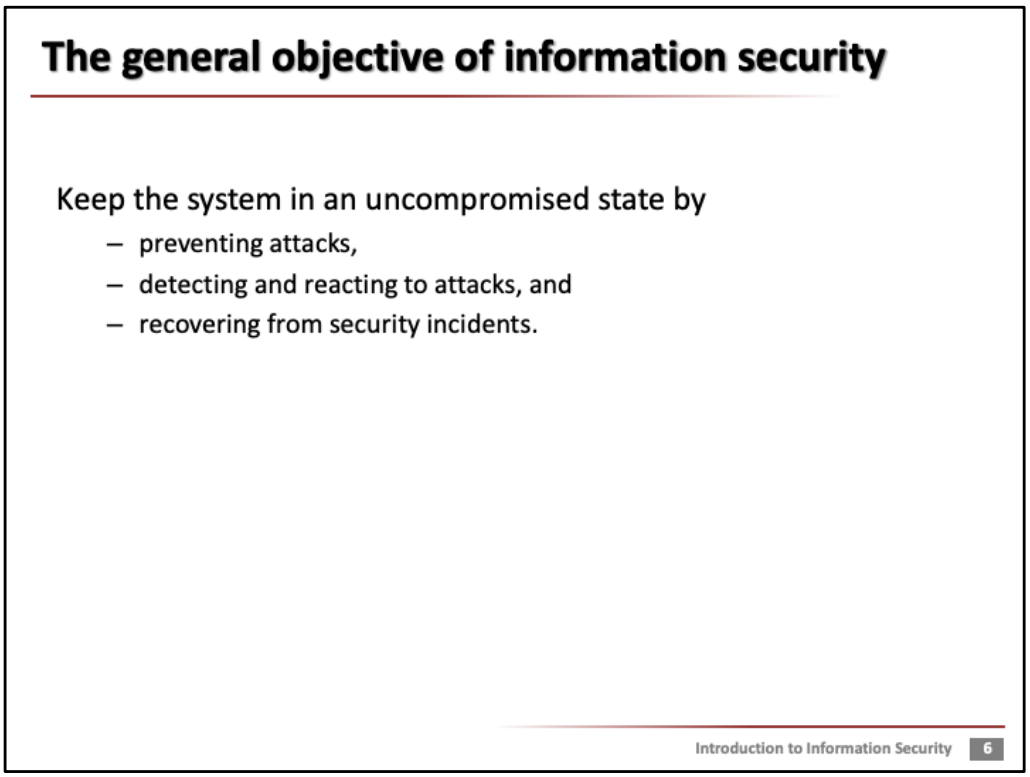
Examples:

- Illegitimate access to the account of a legitimate user
- Infecting a computer with a malware
- Flooding a server with a large amount of illegitimate requests, such that it can no longer serve legitimate requests (Denial-of-Service, DoS)

Loss of **confidentiality**, **integrity**, or **availability of information** that is processed, stored, and transferred by IT systems

Examples:

- Leakage of a password or some confidential business data
- Illegitimate modification of data stored in a database
- Encryption of data on a hard disk by a ransomware



- Preventing attacks

  We would like to prevent attacks being successful in the first place, but this may not always be possible or feasible (e.g., preventing a certain kind of attack may be too expensive or it would lead to an undesirable trade-off, such as extreme overhead or performance degradation).
- Detecting and reacting to attacks

  The next best thing that we can do is to detect attacks happening or being successful, and to quickly react to them. Detection usually aims at detecting the activity of the attacker before it has a serious consequence (i.e., the systems gets into a compromised state). Reaction usually aims at containing the attack (i.e., preventing escalation, minimizing impact), mitigating the attack (e.g., introducing a quick work around that immunize the system against the attack), and, eventually, hardening the system against the (given type of) attack.
- Recovering from security incidents

  Despite all prevention and detection efforts, attacks may sometimes be successful and our system gets compromised. In such cases, we need to efficiently recover from the incident. Recovery includes executing the following tasks: (1) we restore the system in an uncompromised state as soon as possible (i.e., we maximize business continuity), (2) we analyze the incident and understand how our system was compromised, and (3) we eliminate the vulnerabilities that allowed the attack to be successful.

More specific goals

- **Authentication**
  Verification of the claimed identity of an entity
- **Authorization and access control**
  Defining and enforcing an access control policy
- **Accounting**
  Making it possible to record, search, and prove retrospectively every access made to system resources and services

Authentication

- Verification of the (claimed) identity of an entity (user or process), typically to make an access control decision when the entity is accessing some service or resource of the system

Authorization and access control

- Defining access rights (authorization) and enforcing an access control policy (access control)
- An access control decision may depend on the verified identity of the entity making the access (see authentication), the nature of the access (i.e., the type of operation, e.g., read, write, …), the access rights associated with the entity, and other circumstances (e.g., time of access, place of origin of the access request, …)

Accounting

- Making it possible to record, search, and prove retrospectively every access to the system and every operation performed in it. This ensures accountability of users (i.e., making them responsible for their actions)

More specific goals

- **Confidentiality**

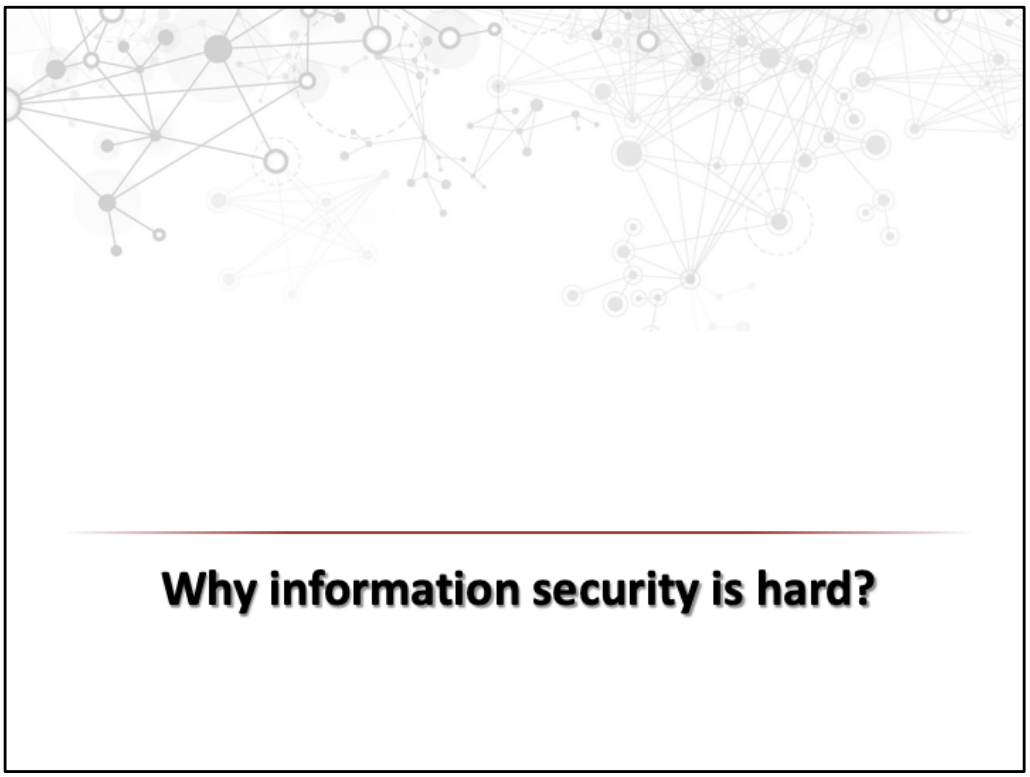
  Information can be obtained only by those who have permission to do so

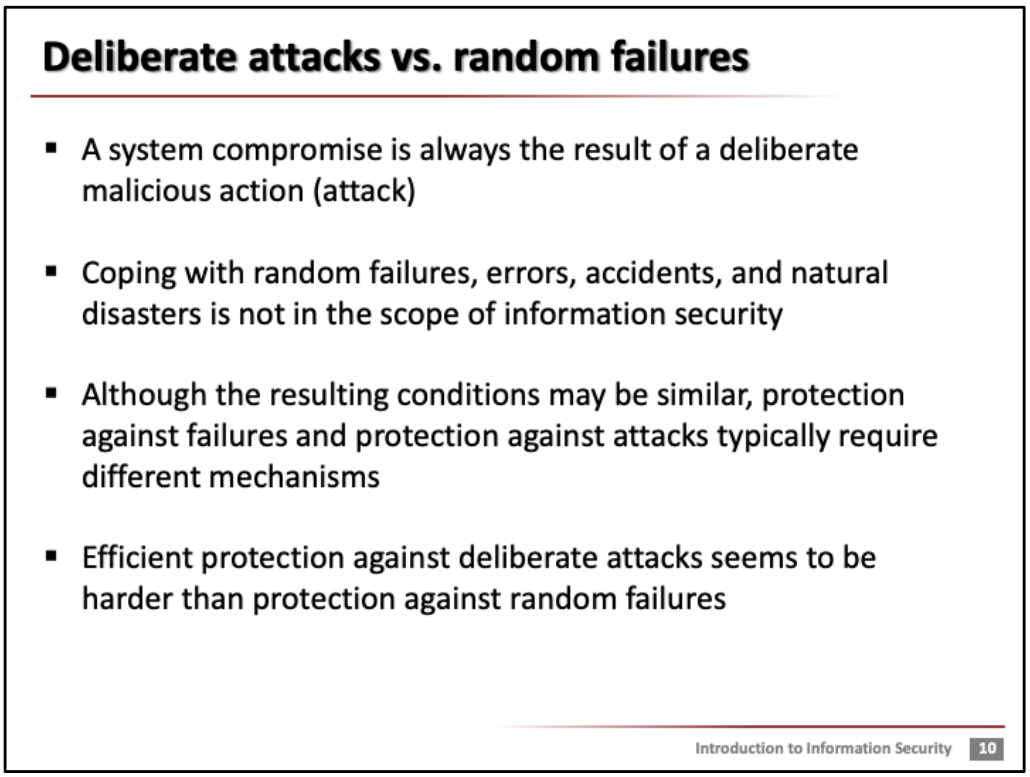
- **Integrity**

  Information can be modified only by those who have permission to do so

- **Availability**

  Information should be available whenever needed for legitimate users


Why information security is hard?



Undesirable conditions resulting from random failures, errors, accidents, and natural disasters are not in the scope of security

Examples:

- Data becomes unavailable due to a hardware failure in the hard disk
- A message is corrupted due to random communication errors
- A system is destroyed in an earthquake

Although the resulting conditions may be similar, protection against failures and protection against attacks typically require different mechanisms

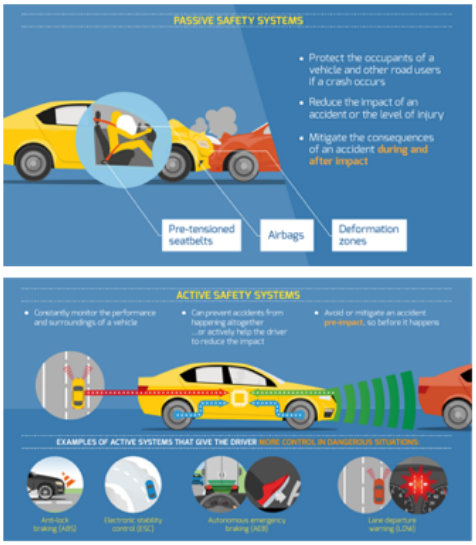
- Protection against deliberate attacks → security, trustworthiness
- Protection against random failures → fault tolerance, reliability, safety

Efficient protection against deliberate attacks seems to be much harder than efficient protection against random failures

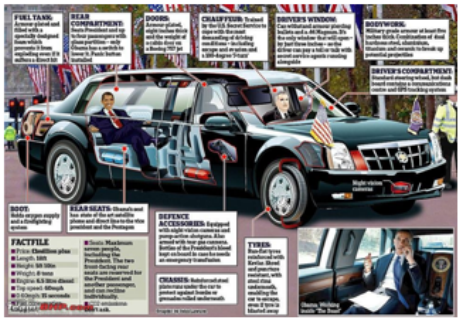
- We have better models for random failures (i.e., nature) than we have for attacks (attackers)
- Having better models means that we know what to expect, and we can prepare for that easier
- In case of attacks:
    - We don't always know what to expect
    - So we may prepare for the worst thing that can happen, but how about efficiently???

Deliberate attacks vs. random failures - illustration

Design for accidents



Design for attacks



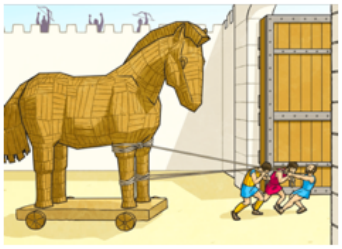
The weakest link property

- Security is like a chain: it breaks at its weakest link (no matter how strong the other links are)
- IT system are very complex and they have many components (those are the links through which the system can be broken)
- Security engineers have to make every component well-protected, but it is sufficient for attackers to find one weak component through which they can get in!
- Don’t forget that systems involve people, and they are often the “weakest link”

The weakest link property - illustration

The Greeks couldn't get into Troy through its walls...

so, they changed their strategy...

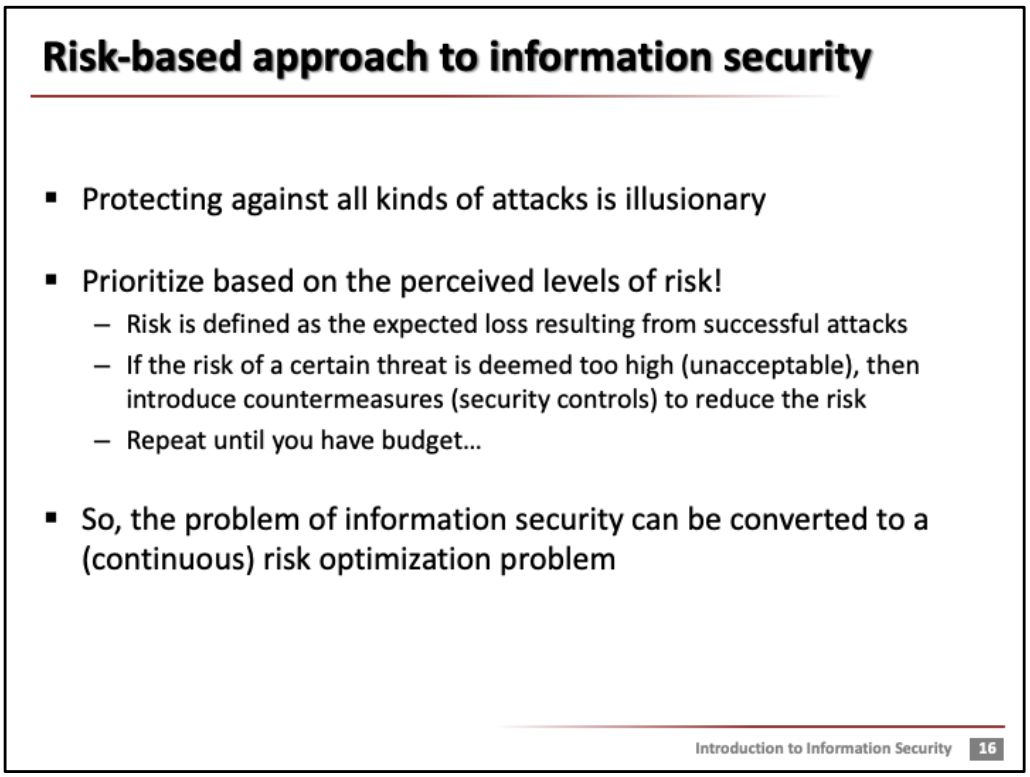


Security vs. other design criteria

- Security mechanisms almost always …
  - increase overhead
  - decrease performance
  - increase cost
  - decrease convenience
- Security vs. features
  - users want more features and options
  - features and options increase complexity
  - complexity is the worst enemy of security

features and options increase complexity, and complexity is the worst enemy of security:
e.g., if you have 20 options, then you need to consider 1 million possible configurations and make each of them secure


The risk-based approach to security



The general objective of information security is to keep the system in an uncompromised state by preventing attacks, detecting and reacting to attacks, and recovering from security incidents.

However, protecting the system against all kinds of attacks is illusionary. There are several reasons for that:

- there are too many ways to attack complex systems and our knowledge about the possibilities is typically incomplete
- it would be too expensive to introduce protection mechanisms against all kinds of attacks regardless their significance and likelihood of occurrence.

Prioritize based on the perceived risk of the threats you are facing!

- Risk is defined as the expected loss resulting from successful attacks
- If the risk of a certain threat is deemed too high (unacceptable), then introduce countermeasures (security controls) to reduce the risk
- Introducing countermeasures has some cost, and you typically have limited budget, so repeat until you have budget…

In this way, the problem of information security can be converted to a risk optimization problem

→ **Minimize risk under some budget constraints**

Risk affecting factors

Risk = Likelihood x Impact
(of attacks)

- Impact:
  - **potential loss** resulting from a successful attack
    - » direct loss (e.g., decreased revenue, cost of recovery)
    - » indirect loss (e.g., losing reputation, decreased trustworthiness)
- Likelihood:
  - likelihood of the attack being successful, which depends on the
    - » **attacker** (motivation, intent, opportunity, capabilities, resources)
    - » **vulnerabilities** (exploitable weaknesses of the system)
    - » **countermeasures** (security controls used to eliminate weaknesses)

When minimizing risk, it is useful to know what factors risk depends on…

Questions governing risk management

- What assets do we have in our system?
- What are the potential attacks on our assets?
- Who are the plausible attackers?

threat identification

- What is the expected loss when assets are attacked successfully?
- What are the known and potential vulnerabilities in the system?
- What is the likelihood of those vulnerabilities being exploited by the plausible attackers to realize attacks?

risk estimation

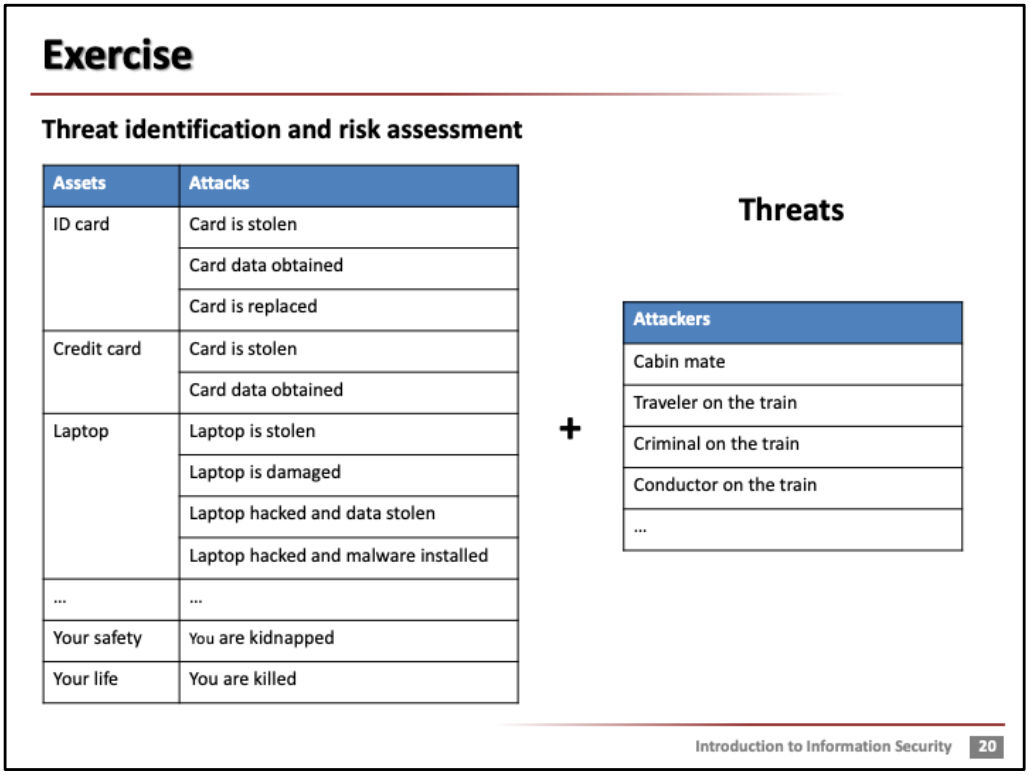
- What countermeasures can reduce the risk in a cost effective way?

risk mitigation

Exercise

Threat identification and risk assessment

Scenario
Finally, Friday afternoon, and your week-end in Paris is approaching!!! Your stuff (some clothes and sandwiches, your laptop, your wallet with your ID card, credit card, and some cash, and your return ticket) is packed in your suitcase, and you can leave for the train station to catch your night train. It's nice to imagine that you'll have your coffee tomorrow morning in one of the charming bistros of Paris! You reserved a place in a sleeping cabin for 4 people on the train. As you travel alone, you must share the cabin with strangers. A bit worrying, but you couldn't afford a private cabin, and Paris is worth the risk. Oh, btw, speaking about risk: what can go wrong during this night trip and how should you get prepared?



Every (asset, attack, attacker) triplet defines a threat

- e.g., your credit card being stolen by a criminal on the train is a threat
- e.g., your laptop being hacked by a cabin mate is a threat

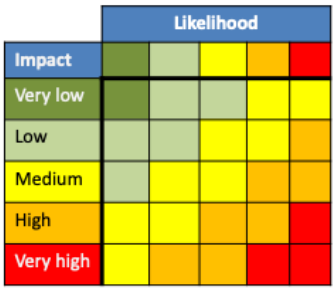
Exercise

Threat identification and risk assessment

Impact	Attack	Attacker	Likelihood	Risk
	Credit card is stolen	by cabin mate		
		by traveler		
		by criminal		
		by conductor		
	...	...		
	You are killed	by cabin mate		
		by traveler		
		by criminal		
		by conductor		

When estimating the likelihood of a threat being realized, consider the motivation of the attacker, as well as their intent and opportunity for the attack

Risk assessment



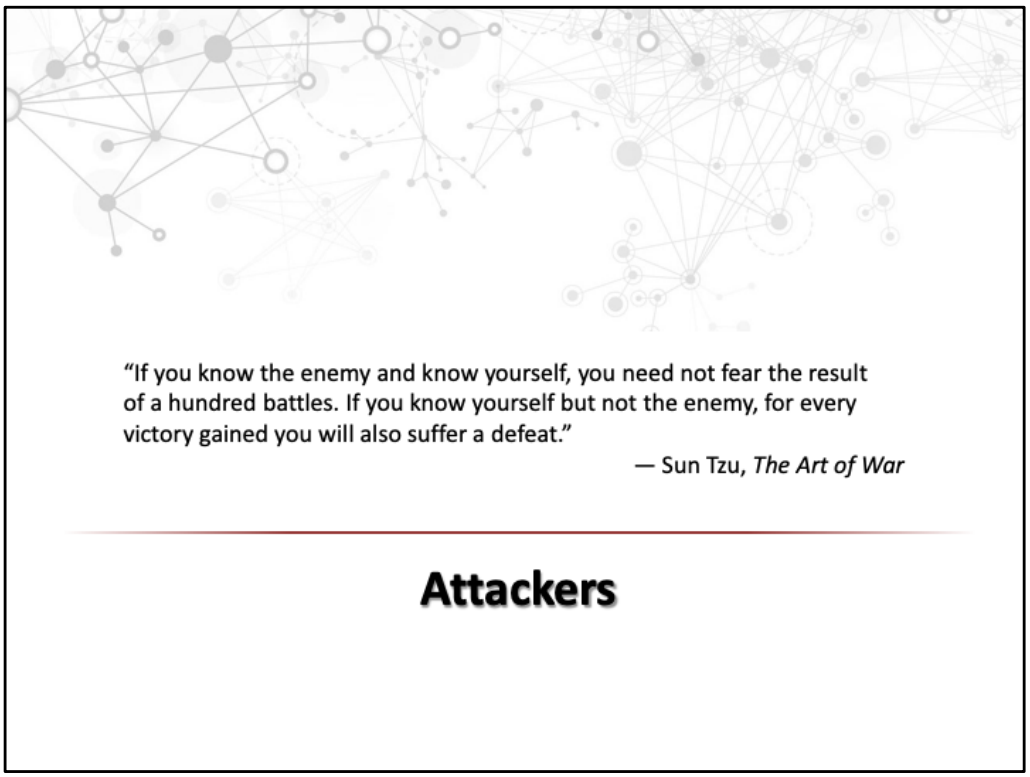
Risk calculation table (illustrative example)

- A given attack on an asset has some impact, often just ranked as Low, Medium, or High.
- A given attacker being able to carry out a given attack on an asset has a likelihood (i.e., the likelihood of a threat being realized). Often this is also ranked as just Low, Medium, or High.
- A matrix can define how to combine levels of impact and likelihood to obtain levels of risk.

Exercise

Threat identification and risk assessment

- Risk mitigation countermeasures
  - Keep your stuff in the suitcase
  - Lock the suitcase
  - Keep cards and money on you constantly
  - Use strong password on your laptop
  - Encrypt the hard disk of your laptop
  - Be vigilant
  - Check identify of conductor
  - …
- Risk transfer
  - Buy an insurance



http://en.wikiquote.org/wiki/Sun_Tzu

知彼知己，百戰不殆；不知彼而知己，一勝一負；不知彼，不知己，每戰必殆
It is said that if you know your enemies and know yourself, you will not be imperiled in a hundred battles; if you do not know your enemies but do know yourself, you will win one and lose one; if you do not know your enemies nor yourself, you will be imperiled in every single battle.
Ch. 3, the last sentence.
Variant translations
If you know others and know yourself, you will not be imperiled in a hundred battles; if you do not know others but know yourself, you win one and lose one; if you do not know others and do not know yourself, you will be imperiled in every single battle.
Know your enemy and know yourself, find naught in fear for 100 battles. Know yourself but not your enemy, find level of loss and victory. Know thy enemy but not yourself, wallow in defeat every time.
Literal translation: **Know [the] other, know [the] self, hundred battles without danger; not knowing [the] other but know [the] self, one win one loss; not knowing [the] other, not knowing [the] self, every battle must [be] lost.**

Characterizing attackers

- Motivation
- Capabilities
  - Information gathering
  - Technical expertise
  - Other (e.g., deception techniques, initial access, …)
- Resources
  - Money
  - Other (e.g., time, computing power, human, …)

Motivations

- Attacker groups or organizations typically have financial, social, or political motivations
- Individual attackers are often personally motivated
  - to achieve fame and status in some (hacker) community
  - revenge against employer or partner
- Besides motivation, intent and opportunity may also be important to consider

Information gathering capabilities

- Success of an attack heavily depends on the amount of information that the attacker has about the attacked system
- Information can be gathered before and during the attack
- Useful information include:
  - general system architecture, available services, used hardware and software components and their configuration settings, network topology and technology;
  - employed security mechanisms (firewall, IDS, anti-virus, ...);
  - known vulnerabilities of the used system elements and security solutions.

Levels of technical expertise

- Technical knowledge and skills are used to transform available information into a successful attack
- They can also be used to increase information gathering capabilities
- Levels of technical expertise:
  1. Understanding of the operation of computer systems and networks
  2. Being familiar with known vulnerabilities and exploit techniques, as well as with basics of IT security
  3. Ability to discover new vulnerabilities and construct exploit tools

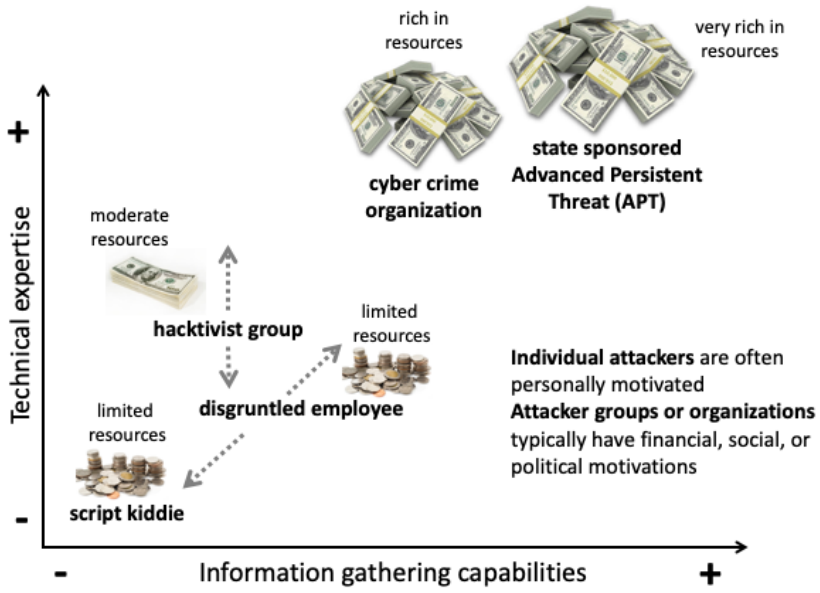
Resources (essentially money)

- Financial resources can be converted to information, knowledge, technical skills, human resources, ...

  Examples:

  - » Increase information gathering capabilities
    - – bribery, ransom
    - – purchase of technical documentations
    - – advanced social engineering
  - » Deepen technical expertise
    - – hiring of experts
    - – improving own competencies and capabilities
  - » Obtain advanced attack tools and methods
    - – zero-day exploits
    - – advanced cryptanalysis tools
    - – increased computing power

Common attacker models (profiles)



Individual attackers are often personally motivated
Attacker groups or organizations typically have financial, social, or political motivations

Script kiddie

- Motivations:
  - self-expression
  - achieving some status
- Technical expertise: **limited**
  - uses tools and methods developed by others
  - may minimally extend existing tools, or combine them in new ways
  - may improve in the long-term (education, self-study, practice)
- Information gathering capability: **limited**
  - mainly publicly available information
  - basic social engineering tricks
- Financial resources: **limited**
- No strategic planning, opportunistic target selection
  - chooses targets that seem to be easy to compromise
  - potential success due to negligence on the system owner's side

Disgruntled employee

- Motivations:
  - revenge (typically after having been fired, or still as an employee)
  - can be very determined, sometimes even irrational
  - well defined objectives, concious target selection
- Information gathering capabilities: **potentially advanced**
  - former employee or still empoyed → internal access to information
  - may have very detailed technical knowledge about the system
  - has personal connections to other employees (effective social engineering)
- Technical expertise: **potentially advanced**
  - depends on his (former) role in the company
- Financial resources: **limited**

Hacktivist group

- Loosely organized group mainly of amateurs
- Motivations:
  - spread or defense of some political or social ideology
  - objectives are often related to actual events (visible response to the event)
  - no long term strategy, ad hoc campaigns
- Information gathering capabilities: **moderate**
  - no resources to obtain internal information
  - may try to gather information by technical means (hacking)
- Technical expertise: **variable, potentially advanced**
  - few leaders who have potentially strong technical background and connections to cyber criminal circles
  - lot of followers who do what they are told to do
- Financial resources: **moderate**
- Examples: Anonymous, Syrian Electronic Army

Cybercrime organization

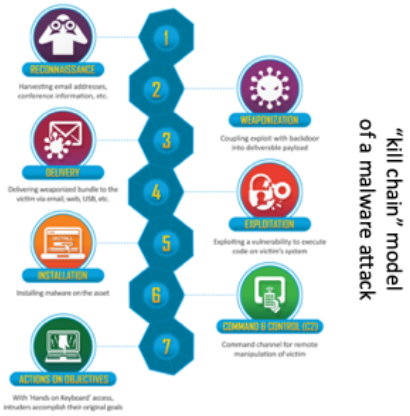
- One of the largest threat today for ordinary users and organizations
- Motivation: financial profit
- Information gathering capabilities: **potentially advanced**
  - technical approaches (spyware, hacking into servers and user accounts)
  - deception (phishing, social engineering)
- Technical expertise: **advanced**
- Financial resources: **potentially large**
  - can employ expert hackers, specialists
  - can buy exploits, malware, and other advanced attack tools
  - can operate a large background infrastructure
  - well-defined objectives and large scale attack campaigns in space and time
- Examples: many ...

State sponsored attacker (a.k.a. APT)

- One of the largest threat today for governments
- Motivations: political or economical goals
  - clear objectives (espionage or sabotage)
  - strategic planning, and long-term, targeted operations
- Information gathering capabilities: **advanced**
  - cyber espionage and surveillance tools
  - traditional intelligence gathering (e.g., SIGINT)
- Technical expertise: **advanced**
  - complex research, development, and training programs
- Financial resources: **large**
  - can employ or train expert hackers
  - can buy zero-day exploits, malware, and advanced attack tools legitimately
  - large background infrastructure
- Examples:
  - PLA Unit 61398 (China)
  - NSA TAO (USA)

Attacks

- An attack is a process or activity in which vulnerabilities are exploited by an attacker with malicious intent in order to compromise a system (i.e., subvert security goals)
- An attack may be a complex process…
- Understanding attack techniques is useful for designing effective countermeasures



"kill chain" model of a malware attack

CAPEC

Common Attack Pattern Enumeration and Classification

- Comprehensive collection of known attack patterns
- Freely available online: **https://capec.mitre.org/**
- 559 attack pattern categorized according to different aspects (e.g., attack mechanism, domains, etc.)

3000 - Domains of Attack

- Software - *(513)*
- Hardware - *(515)*
- Communications - *(512)*
- Supply Chain - *(437)*
- Social Engineering - *(403)*
- Physical Security - *(514)*

1000 - Mechanisms of Attack

- Engage in Deceptive Interactions - *(156)*
- Abuse Existing Functionality - *(210)*
- Manipulate Data Structures - *(255)*
- Manipulate System Resources - *(262)*
- Inject Unexpected Items - *(152)*
- Employ Probabilistic Techniques - *(223)*
- Manipulate Timing and State - *(172)*
- Collect and Analyze Information - *(118)*
- Subvert Access Control - *(225)*

CAPEC – example

Software → Software Integrity Attacks →

CAPEC-185: Malicious Software Download

Attack Pattern ID: 185
Abstraction: Standard

View customized information: Conceptual | Operational | Mapping-Friendly | **Complete**

Description

An attacker uses deceptive methods to cause a user or an automated process to download and install dangerous code that originates from an attacker controlled source. There are several variations to this strategy of attack.

Typical Severity

Very High

Relationships

Nature	Type	ID	Name
ChildOf		184	Software Integrity Attack
CanFollow		159	Redirect Access to Libraries
CanPrecede		662	Adversary in the Browser (AITB)

View Name	Top Level Categories
Domains of Attack	Software
Mechanisms of Attack	Manipulate System Resources
Supply Chain Risks	Sustainment

Related Weaknesses

CWE-ID	Weakness Name
494	Download of Code Without Integrity Check

Targeted attacks

- "Targeted" means that the victim is not randomly chosen
- Such attacks use highly customized tools and intrusion techniques
- They are stealthy and persistent
- They are carried out by well-funded and well-staffed organizations (APTs)

Targeted attacks use highly customized tools and intrusion techniques

- malware delivery by spear phishing and social engineering
- using partners in the supply chain as stepping stones
- multiple different exploits (often 0-day or very fresh)

They are stealthy and persistent

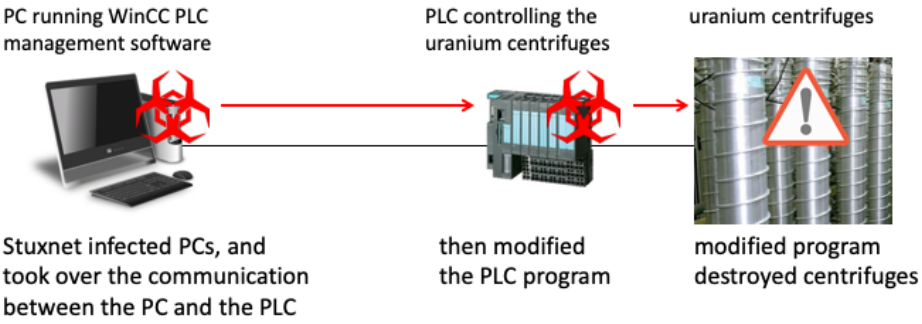
- bypassing mainstream AV and security products without detection
- careful design and intensive testing to avoid any anomalies

They are carried out by well-funded and well-staffed organizations (APTs)

- typically, military or state intelligence services

Stuxnet (June 2010)

- "The Most Menacing Malware in History" (Kim Zetter, Wired)
- Targeted the Natanz nuclear enrichment plant in Iran
- Used multiple zero-day exploits
- Possibly created by Western nation states



Duqu (October 2011)

Home / News & Blogs / Zero Day

Hungarian Lab found Stuxnet-like Duqu malware

By Ryan Naraine | October 21, 2011, 9:11am PDT

Summary: *The Laboratory of Cryptography and System Security (CrySyS) in Hungary confirmed its participation in the initial discovery of the Duqu cyber-surveillance Trojan.*

A security lab attached to the Budapest University of Technology and Economics in Hungary has come forward as the mystery outfit that found the Stuxnet-like "Duqu" cyber-surveillance Trojan.

According to Symantec's initial report on Duqu [PDF], the malware sample was passed along by an unnamed "research lab with strong international connections," a statement that led to speculation about the origins and intent of the threat.


Vulnerabilities

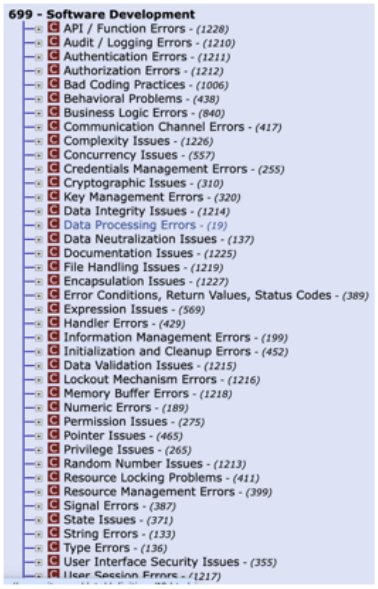
Vulnerability types

- **Technical –** design flaws and implementation errors in hardware, software, interfaces, and protocols
- **Physical –** weaknesses allowing for physical access (e.g., unlocked door)
- **Operational –** weaknesses in the policies and procedures used to operate the system
- **Personnel –** lack of security awareness, know-how, and trustworthiness of people (employees, operators, contractors)

CWE

Common Weakness Enumeration

- Comprehensive collection of known software and hardware weaknesses
- Freely available online: **https://cwe.mitre.org/**
- 933 weaknesses categorized according to various aspects (e.g., software, hardware, ...)



CWE – example

Research concepts → Protection Mechanism Failure → Insufficient Verification of Data Authenticity →

CWE-494: Download of Code Without Integrity Check

Weakness ID: 494
Abstraction: Base
Structure: Simple

View customized information: Conceptual | Operational | Mapping Friendly | **Complete** | Custom

Description

The product downloads source code or an executable from a remote location and executes the code without sufficiently verifying the origin and integrity of the code.

Extended Description

An attacker can execute malicious code by compromising the host server, performing DNS spoofing, or modifying the code in transit.

Relationships

Modes Of Introduction

Phase	Note
Architecture and Design Implementation	OMISSION: This weakness is caused by missing a security tactic during the architecture and design phase.

Applicable Platforms

Languages

Class: Not Language-Specific *(Undetermined Prevalence)*

Common Consequences

Scope	Impact	Likelihood
Integrity Availability Confidentiality Other	**Technical Impact:** *Execute Unauthorized Code or Commands; Alter Execution Logic; Other* Executing untrusted code could compromise the control flow of the program. The untrusted code could execute attacker-controlled commands, read or modify sensitive resources, or prevent the software from functioning correctly for legitimate users.	

Likelihood Of Exploit

Medium

CWE – example

Demonstrative Examples

Example 1

This example loads an external class from a local subdirectory.

Example Language: **Java** (bad code)

```
URL[] classURLs= new URL[]{
   new URL("file:subdir/")
};
URLClassLoader loader = new URLClassLoader(classURLs);
Class loadedClass = Class.forName("loadMe", true, loader);
```

This code does not ensure that the class loaded is the intended one, for example by verifying the class's checksum. An attacker may be able to modify the class file to execute malicious code.

Example 2

This code includes an external script to get database credentials, then authenticates a user against the database, allowing access to the application.

Example Language: **PHP** (bad code)

```
//assume the password is already encrypted, avoiding CWE-312

function authenticate($username,$password){

   include("http://external.example.com/dbInfo.php");

   //dbInfo.php makes $dbhost, $dbuser, $dbpass, $dbname available
   mysql_connect($dbhost, $dbuser, $dbpass) or die ('Error connecting to mysql');
   mysql_select_db($dbname);
   $query = 'Select * from users where username='.$username.' And password='.$password;
   $result = mysql_query($query);

   if(mysql_numrows($result) == 1){
      mysql_close();
      return true;
   }
   else{
      mysql_close();
      return false;
   }

}
```

CWE – example

Potential Mitigations

Phase: Implementation

Perform proper forward and reverse DNS lookups to detect DNS spoofing.
Note: This is only a partial solution since it will not prevent your code from being modified on the hosting site or in transit.

Phases: Architecture and Design; Operation

Encrypt the code with a reliable encryption scheme before transmitting.
This will only be a partial solution, since it will not detect DNS spoofing and it will not prevent your code from being modified on the hosting site.

Phase: Architecture and Design

Strategy: Libraries or Frameworks

Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.
Speficially, it may be helpful to use tools or frameworks to perform integrity checking on the transmitted code.

- When providing the code that is to be downloaded, such as for automatic updates of the software, then use cryptographic signatures for the code and modify the download clients to verify the signatures. Ensure that the implementation does not contain CWE-295, CWE-320, CWE-347, and related weaknesses.
- Use code signing technologies such as Authenticode. See references [REF-454] [REF-455] [REF-456].

Phases: Architecture and Design; Operation

Strategy: Environment Hardening

Run your code using the lowest privileges that are required to accomplish the necessary tasks [REF-76]. If possible, create isolated accounts with limited privileges that are only used for a single task. That way, a successful attack will not immediately give the attacker access to the rest of the software or its environment. For example, database applications rarely need to run as the database administrator, especially in day-to-day operations.

Phases: Architecture and Design; Operation

Strategy: Sandbox or Jail

Run the code in a "jail" or similar sandbox environment that enforces strict boundaries between the process and the operating system. This may effectively restrict which files can be accessed in a particular directory or which commands can be executed by the software.
OS-level examples include the Unix chroot jail, AppArmor, and SELinux. In general, managed code may provide some protection. For example, java.io.FilePermission in the Java SecurityManager allows the software to specify restrictions on file operations.
This may not be a feasible solution, and it only limits the impact to the operating system; the rest of the application may still be subject to compromise.
Be careful to avoid CWE-243 and other weaknesses related to jails.
Effectiveness: Limited
Note: The effectiveness of this mitigation depends on the prevention capabilities of the specific sandbox or jail being used and might only help to reduce the scope of an attack, such as restricting the attacker to certain system calls or limiting the portion of the file system that can be accessed.

CWE – example

Detection Methods

Manual Analysis

This weakness can be detected using tools and techniques that require manual (human) analysis, such as penetration testing, threat modeling, and interactive tools that allow the tester to record and modify an active session.

Specifically, manual static analysis is typically required to find the behavior that triggers the download of code, and to determine whether integrity-checking methods are in use.

Note: These may be more effective than strictly automated techniques. This is especially the case with weaknesses that are related to design and business rules.

Black Box

Use monitoring tools that examine the software's process as it interacts with the operating system and the network. This technique is useful in cases when source code is unavailable, if the software was not developed by you, or if you want to verify that the build phase did not introduce any new weaknesses. Examples include debuggers that directly attach to the running process; system-call tracing utilities such as truss (Solaris) and strace (Linux); system activity monitors such as FileMon, RegMon, Process Monitor, and other Sysinternals utilities (Windows); and sniffers and protocol analyzers that monitor network traffic.

Attach the monitor to the process and also sniff the network connection. Trigger features related to product updates or plugin installation, which is likely to force a code download. Monitor when files are downloaded and separately executed, or if they are otherwise read back into the process. Look for evidence of cryptographic library calls that use integrity checking.

Automated Static Analysis

Automated static analysis, commonly referred to as Static Application Security Testing (SAST), can find some instances of this weakness by analyzing source code (or binary/compiled code) without having to execute it. Typically, this is done by building a model of data flow and control flow, then searching for potentially-vulnerable patterns that connect "sources" (origins of input) with "sinks" (destinations where the data interacts with external components, a lower layer such as the OS, etc.)

Effectiveness: High

Memberships

Nature	Type	ID	Name
MemberOf	C	752	2009 Top 25 - Risky Resource Management
MemberOf	C	802	2010 Top 25 - Risky Resource Management
MemberOf	C	859	The CERT Oracle Secure Coding Standard for Java (2011) Chapter 16 - Platform Security (SEC)
MemberOf	C	865	2011 Top 25 - Risky Resource Management
MemberOf	V	884	CWE Cross-section
MemberOf	C	991	SFP Secondary Cluster: Tainted Input to Environment
MemberOf	C	1354	OWASP Top Ten 2021 Category A08:2021 - Software and Data Integrity Failures
MemberOf	C	1364	ICS Communications: Zone Boundary Failures
MemberOf	C	1411	Comprehensive Categorization: Insufficient Verification of Data Authenticity

Publicly known vulnerabilities

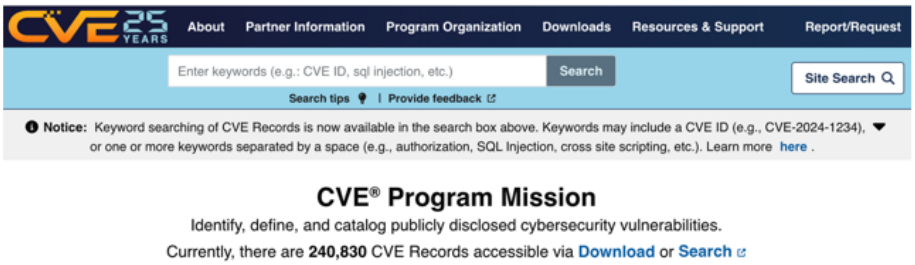
- Technical vulnerabilities that are discovered may be publicly disclosed through a *responsible disclosure procedure*
- Reported technical vulnerabilities get a globally recognized identifier (CVE ID)
- Information on reported technical vulnerabilities is stored in public vulnerability databases (e.g., US National Vulnerability Database)
- Public availability of vulnerability information helps keeping systems free from known vulnerabilities

Discuss the question: Is it good or bad to publicly disclose vulnerability information? Consider the following points:

- public availability of vulnerability information can help system operators to keep their systems free of at least the known vulnerabilities
- on the other hand, there may be systems where fixing known vulnerabilities is slow or even impossible

Common Vulnerabilities and Exposures (CVE)

https://www.cve.org/



NIST NVD

https://nvd.nist.gov/

NIST NVD – example

CVE-2021-22909 Detail

Description

A vulnerability found in EdgeMAX EdgeRouter V2.0.9 and earlier could allow a malicious actor to execute a man-in-the-middle (MitM) attack during a firmware update. This vulnerability is fixed in EdgeMAX EdgeRouter V2.0.9-hotfix.1 and later.

Severity

CVSS Version 3.x | CVSS Version 2.0

CVSS 3.x Severity and Metrics:

NIST: NVD **Base Score:** 7.5 HIGH **Vector:** CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:H/I:H/A:H

NVD Analysts use publicly available information to associate vector strings and CVSS scores. We also display any CVSS information provided within the CVE List from the CNA.

Note: NVD Analysts have published a CVSS score for this CVE based on publicly available information at the time of analysis. The CNA has not provided a score within the CVE List.

QUICK INFO

CVE Dictionary Entry:
CVE-2021-22909
NVD Published Date:
05/27/2021
NVD Last Modified:
06/08/2021
Source:
HackerOne

NIST NVD – example

References to Advisories, Solutions, and Tools

By selecting these links, you will be leaving NIST webspace. We have provided these links to other web sites because they may have information that would be of interest to you. No inferences should be drawn on account of other sites being referenced, or not, from this page. There may be other web sites that are more appropriate for your purpose. NIST does not necessarily endorse the views expressed, or concur with the facts presented on these sites. Further, NIST does not endorse any commercial products that may be mentioned on these sites. Please address comments about this page to nvd@nist.gov.

Hyperlink	Resource
https://community.ui.com/releases/Security-Advisory-Bulletin-018-018/cfa1566b-4bf8-427b-8cc7-8cffba3a93a4	Vendor Advisory

Weakness Enumeration

CWE-ID	CWE Name	Source
CWE-295	Improper Certificate Validation	NIST
CWE-300	Channel Accessible by Non-Endpoint	HackerOne

Known Affected Software Configurations Switch to CPE 2.2

Configuration 1 (hide)

cpe:2.3:o:ui:edgemax_edgerouter_firmware:*:*:*:*:*:*:*:*	Up to (including) 2.0.9
Show Matching CPE(s)	
Running on/with	
cpe:2.3:h:ui:edgemax_edgerouter:-:*:*:*:*:*:*:*	
Show Matching CPE(s)	

Zero-day vulnerabilities

- Vulnerabilities that are known only to attackers
- Zero-day vulnerabilities are dangerous, because potential victims are usually not prepared for them
- Fortunately, they are expensive, hence often used only in targeted attacks where
  - successfully compromising a particular target is important
  - risk of detection and exposure of the zero-day vulnerability is small

Note:
Some companies make their living out of finding and selling zero-day vulnerabilities (or exploits) to criminals and governments
See for more info: https://en.wikipedia.org/wiki/Market_for_zero-day_exploits

Why do vulnerabilities exist?

- Complexity of systems
- Lack or limitations of methods for designing and implementing error-free systems
- Limitation of resources
- Making wrong assumptions during design or operations
- Creating poor specifications for implementers

Complexity of systems

Lack or limitations of methods for designing and implementing error-free systems

- e.g., formal verification methods do not scale
- e.g., testing cannot be exhaustive in practice

Limitation of resources

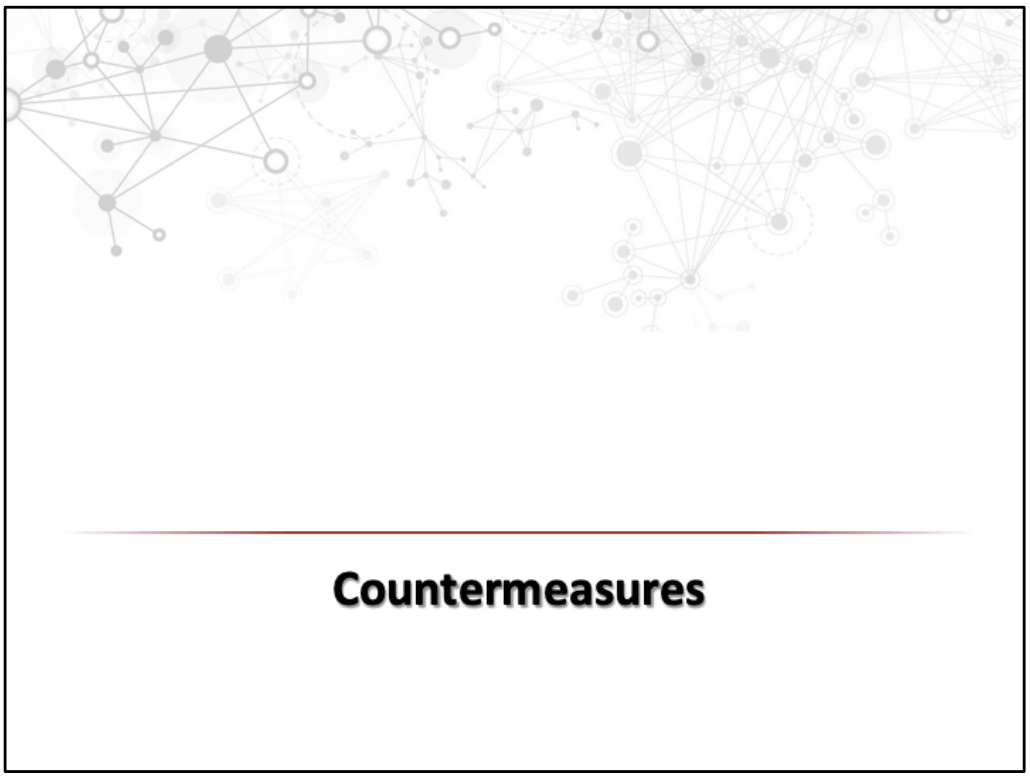
- money, time, knowledgeable work force

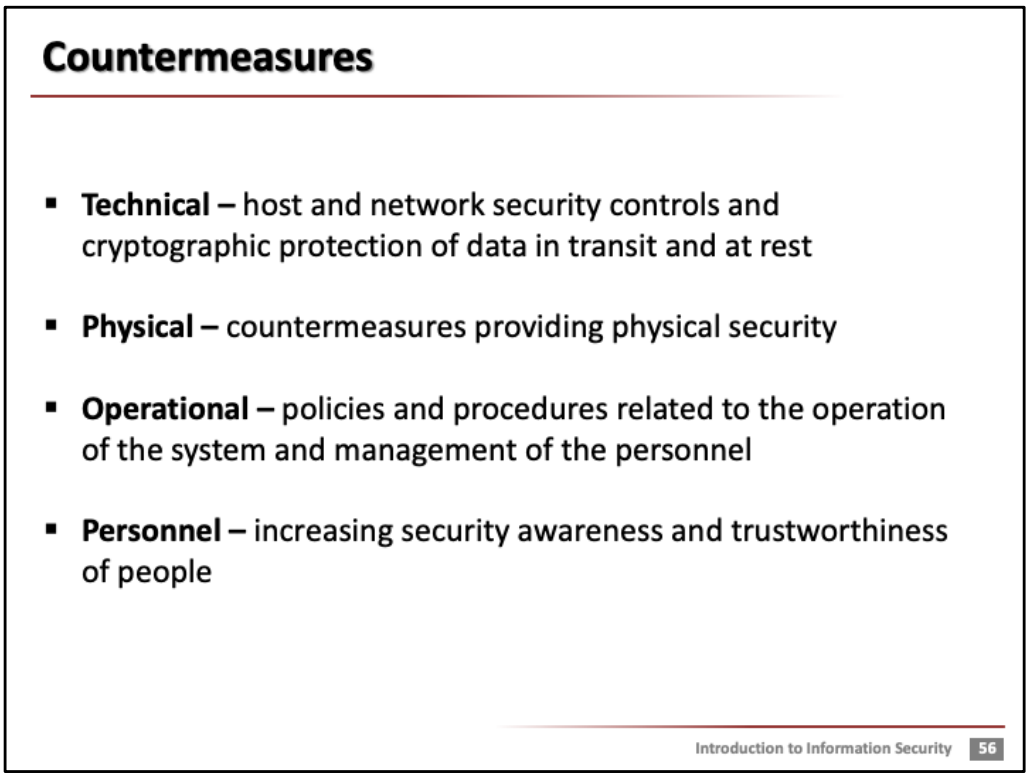
Making wrong assumptions during design or operations

- e.g., neglecting a given type of attack or a given type of attacker

Creating poor specifications for implementers

- as a result, implementers with little security knowledge and skills make decisions during implementation


Countermeasures



Technical – host and network security controls and cryptographic protection of data in transit and at rest

- e.g., firewalls, network IDS/IPS, anti-virus software, log collection and analysis
- e.g., authentication schemes, access control mechanisms
- e.g., memory protection in the OS, address space layout randomization, stack canaries, control flow integrity mechanisms, secure boot sequence, trsuted execution environments
- e.g., cryptographic primitives and complex cryptographic protocols for securing communications and storage of data

Physical – countermeasures providing physical security

- e.g., locks, fences, security guards, tamper resistant hardware, ...

Operational – policies and procedures related to the operation of the system and management of the personnel

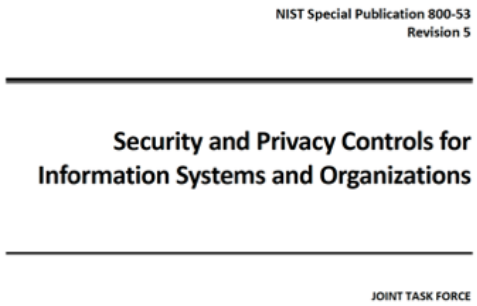
- e.g., password changing policies, regular security testing, ...
- e.g., hiring and firing procedures, vacation policies, ...

Personnel – increasing security awareness and trustworthiness of people

- e.g., security education, increasing employee satisfaction with good salaries

NIST Special Publication 800-53 (Rev 5)

- Catalogue of security and privacy mechanisms (controls) for information systems and organisations
- The list is meant to be exhaustive, covering requirements arising from mission and business needs, laws, executive orders, directives, regulations, standards and guidelines
- 20 family, 300+ mechanisms, 492 pages (!)

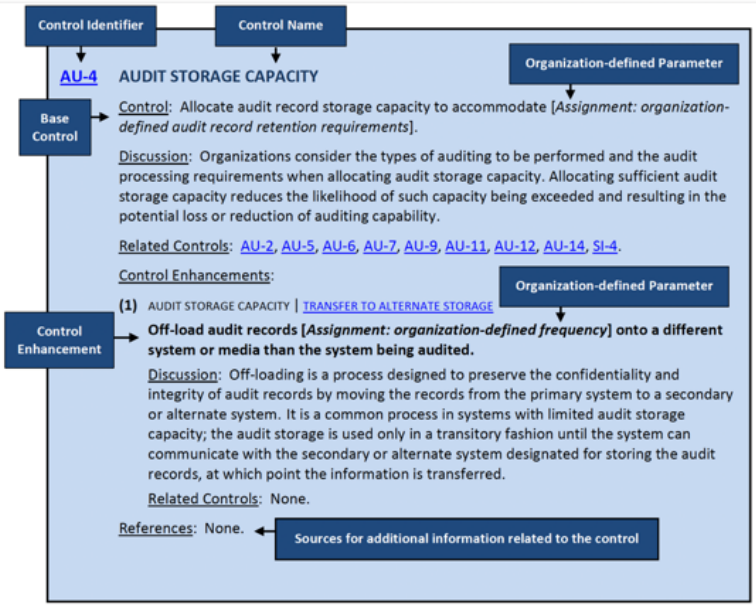


NIST SP800-53r5 – Families of mechanisms

TABLE 1: SECURITY AND PRIVACY CONTROL FAMILIES

ID	FAMILY	ID	FAMILY
AC	Access Control	PE	Physical and Environmental Protection
AT	Awareness and Training	PL	Planning
AU	Audit and Accountability	PM	Program Management
CA	Assessment, Authorization, and Monitoring	PS	Personnel Security
CM	Configuration Management	PT	PII Processing and Transparency
CP	Contingency Planning	RA	Risk Assessment
IA	Identification and Authentication	SA	System and Services Acquisition
IR	Incident Response	SC	System and Communications Protection
MA	Maintenance	SI	System and Information Integrity
MP	Media Protection	SR	Supply Chain Risk Management

NIST SP800-53r5 – Structure of an entry



NIST SP800-53r5 – example

SI-3 MALICIOUS CODE PROTECTION

Control:

a. Implement [*Selection (one or more): signature based; non-signature based*] malicious code protection mechanisms at system entry and exit points to detect and eradicate malicious code;

b. Automatically update malicious code protection mechanisms as new releases are available in accordance with organizational configuration management policy and procedures;

c. Configure malicious code protection mechanisms to:

1. Perform periodic scans of the system [*Assignment: organization-defined frequency*] and real-time scans of files from external sources at [*Selection (one or more): endpoint; network entry and exit points*] as the files are downloaded, opened, or executed in accordance with organizational policy; and

2. [*Selection (one or more): block malicious code; quarantine malicious code; take* [*Assignment: organization-defined action*]]; and send alert to [*Assignment: organization-defined personnel or roles*] in response to malicious code detection; and

d. Address the receipt of false positives during malicious code detection and eradication and the resulting potential impact on the availability of the system.

Discussion: System entry and exit points include firewalls, remote access servers, workstations, electronic mail servers, web servers, proxy servers, notebook computers, and mobile devices. Malicious code includes viruses, worms, Trojan horses, and spyware. Malicious code can also be encoded in various formats contained within compressed or hidden files or hidden in files using techniques such as steganography. Malicious code can be inserted into systems in a variety of ways, including by electronic mail, the world-wide web, and portable storage devices. Malicious code insertions occur through the exploitation of system vulnerabilities. A variety of technologies and methods exist to limit or eliminate the effects of malicious code.

Malicious code protection mechanisms include both signature- and nonsignature-based technologies. Nonsignature-based detection mechanisms include artificial intelligence techniques that use heuristics to detect, analyze, and describe the characteristics or behavior of malicious code and to provide controls against such code for which signatures do not yet exist or for which existing signatures may not be effective. Malicious code for which active signatures do not yet exist or may be ineffective includes polymorphic malicious code (i.e., code that changes signatures when it replicates). Nonsignature-based mechanisms also include reputation-based technologies. In addition to the above technologies, pervasive configuration management, comprehensive software integrity controls, and anti-exploitation software may be effective in preventing the execution of unauthorized code. Malicious code may be present in commercial off-the-shelf software as well as custom-built software and could include logic bombs, backdoors, and other types of attacks that could affect organizational mission and business functions.

Related Controls: AC-4, AC-19, CM-3, CM-8, IR-4, MA-3, MA-4, PL-9, RA-5, SC-7, SC-23, SC-26, SC-28, SC-44, SI-2, SI-4, SI-7, SI-8, SI-15.

Control Enhancements:

(1) MALICIOUS CODE PROTECTION | CENTRAL MANAGEMENT
[Withdrawn: Incorporated into PL-9.]

(2) MALICIOUS CODE PROTECTION | AUTOMATIC UPDATES
[Withdrawn: Incorporated into SI-3.]

(3) MALICIOUS CODE PROTECTION | NON-PRIVILEGED USERS
[Withdrawn: Incorporated into AC-6(10).]

(4) MALICIOUS CODE PROTECTION | UPDATES ONLY BY PRIVILEGED USERS

Update malicious code protection mechanisms only when directed by a privileged user.

Discussion: Protection mechanisms for malicious code are typically categorized as security-related software and, as such, are only updated by organizational personnel with appropriate access privileges.

Related Controls: CM-5.

(5) MALICIOUS CODE PROTECTION | PORTABLE STORAGE DEVICES

(6) MALICIOUS CODE PROTECTION | TESTING AND VERIFICATION

(a) Test malicious code protection mechanisms [*Assignment: organization-defined frequency*] by introducing known benign code into the system; and

(b) Verify that the detection of the code and the associated incident reporting occur.

Discussion: None.

Related Controls: CA-2, CA-7, RA-5.

(7) MALICIOUS CODE PROTECTION | NONSIGNATURE-BASED DETECTION
[Withdrawn: Incorporated into SI-3.]

(8) MALICIOUS CODE PROTECTION | DETECT UNAUTHORIZED COMMANDS

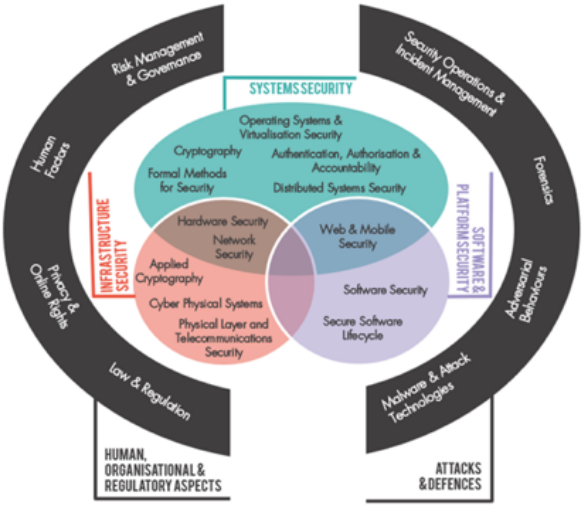
(a) Detect the following unauthorized operating system commands through the kernel application programming interface on [*Assignment: organization-defined system hardware components*]: [*Assignment: organization-defined unauthorized operating system commands*]; and

(b) [*Selection (one or more): Issue a warning; audit the command execution; prevent the execution of the command*].

Discussion: Detecting unauthorized commands can be applied to critical interfaces other than kernel-based interfaces, including interfaces with virtual machines and privileged applications. Unauthorized operating system commands include commands for kernel functions from system processes that are not trusted to initiate such commands as well as commands for kernel functions that are suspicious even though commands of that type are reasonable for processes to initiate. Organizations can define the malicious commands to be

Knowledge areas of cybersecurity

https://www.cybok.org/knowledgebase1_1/



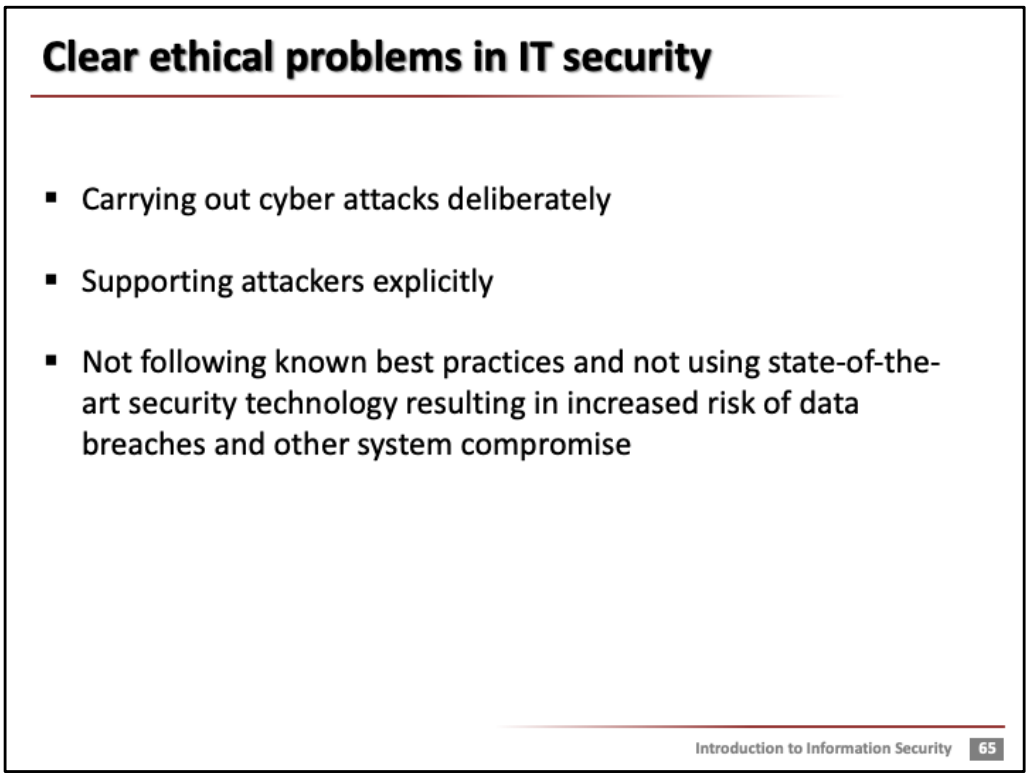
Summary

- IT security deals with intentional attacks on systems and information (prevention, detection and reaction, and recovery from incidents)
- For system operators, IT security is essentially risk management
- Risk (of a threat) is affected by many factors:
  - potential impact of the attack
  - characteristics of the **attacker** carrying out the attack
  - the **vulnerabilities** of the system that the attacker can exploit
  - the **countermeasures** used to make the attacker's task more difficult
- **Security is the result of the risk management process aiming at continuously maintaining a sufficiently protected state of the system**


Ethical issues in information security

Ethics, applied ethics, and engineering ethics

- Ethics
  - a.k.a. moral philosophy
  - A branch of philosophy that addresses questions of morality
    - » What is good and bad?
    - » What is right and wrong?
- Applied ethics
  - Attempts to apply ethical theory to real-life situations occurring in a particular domain of action (e.g., business, bio technology, engineering, IT security, …)
  - Answer to certain questions may not be clear yes or no
- Engineering ethics
  - Oligations of engineers to society, to their clients, and to the profession



- Carrying out cyber attacks deliberately
- Supporting attackers explicitly
    - creating and selling malware
    - hunting for vulnerabilities and selling on the black market
    - designing products with backdoors
    - ...
- Not following known best practices and not using state-of-the-art security technology resulting in increased risk of data breaches and other system compromise
    - using weak encryption (e.g., "home-made" ciphers)
    - using weak passwords
    - leaving unused ports open
    - not using firewalls and segregation of networks
    - no logging or not checking logs
    - designing products with weak protection (see e.g., IoT devices)
    - …

Ethical issues in IT security – the gray zone

- Ethical hacking
- Disclosure of vulnerabilities
- Reverse engineering
- Malware for the good
- C&C milking
- Uncovering cyber operations of state sponsored agencies
- Sharing malware samples for research purposes

These are examples for topics for discussion…

Ethical hacking

penetration testing

- the goal is to discover unknown vulnerabilities before attackers do so
- applies tools and techniques similar to those used by real attackers
- **always done under contract** which defines the scope and conditions
- examples: pentesting companies

hacking systems for a morally justifiable reason

examples:

» BKK hacker
https://techcrunch.com/2017/07/25/hungarian-hacker-arrested-for-pressing-f12

» Football leaks
https://www.zdnet.com/article/hacker-behind-football-leaks-arrested-in-hungary/

» hacking into a botnet C&C in order to seize its control
https://www.zdnet.com/article/avast-and-french-police-take-over-malware-botnet-and-disinfect-850000-computers/

» breaking encryption on the mobile phones of terrorists
https://en.wikipedia.org/wiki/FBI–Apple_encryption_dispute

Disclosure of vulnerabilities

"responsible disclosure"

- discovered vulnerabilities are usually published to help system operators to introduce countermeasures and to force vendors to develop a patch
- vulnerabilities are typically disclosed only after a period of time that allows for the vulnerability to be fixed or a work-around to be introduced

questions and issues

- Who should determine the grace period?
  - » e.g., Google Project Zero has a 90-day disclosure deadline which starts after notifying the vendor affected by the discovered vulnerability
- What if the vendor does not respond? What if the vulnerability cannot be fixed within the deadline?
  - » disclosing details would harm users of the vulnerable product or service
- even if a patch is released, not everyone will or can install it immediately
  - » e.g., in industrial systems patching should follow maintenance cycles
  - » disclosure may put critical systems at risk

Malware for the good

- Example: Hajime
  - malware that infects embedded IoT devices
  - does not have a malicious warhead
  - rather it fixes vulnerabilities and protects "victims" from other malware
  - appeared as a response to weak security of IoT devices and the emergence of IoT botnets such as Mirai
  - however, it is capable of receiving digitally signed commands from its creator…
- questions and issues
  - Shall we thank the creator of Hajime for "saving the IoT world"?
  - Would Hajime be morally justifiable if it had no remote control capabilities?
  - What if it had deleted itself after fixing and hardening vulnerable devices?

C&C milking

- breaking into the C&C servers of malware and extracting information
  - data collected by the attacker
  - database of victims
  - downloadable modules of the malware
  - logs that may give clues about location or identity of the attacker
- questions and issues
  - C&C servers are often ordinary servers on the Internet hacked and used by attackers for their own purposes --» by milking them we may get non-attacker data as well
  - How long should you observe the server?
    - » longer observations may reveal more useful data
    - » not stopping the server leaves victims under active attack

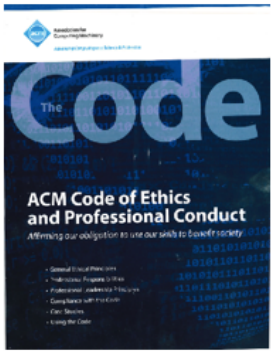
Uncovering cyber operations

- Duqu
  - imagine that you discover a malware in the wild which is not recognized by any anti-virus scanner and intrusion detection system
  - after some analysis, it turns out to be similar to an already known malware, and it is very likely that both were created by the same attacker
  - it is also widely believed that the known malware was created by a secret agency of a "friendly country"
- What should you do?
  - Abandon? – What if the new malware was discovered at a reputed company in your country? Could there be other victims? Why is this friendly secret agency attacking companies in your country?
  - Notify secret agency that you discovered their operations and ask them what you should do? – How would you do that? Call their help desk???
  - Publish your discovery? – Is it safe for you to do that? Would it prevent the secret agency achieving its goals? What would be the effect of the secret operation failing?

ACM Code of Ethics and Professional Conduct

- ACM = Association for Computing Machinery
- Motivations
  - advances in computing have intensified the depth and breadth of the field's impact on society
  - computing professionals should reflect upon the wider impacts of their work
- The Code is designed to inspire and guide the ethical conduct of all computing professionals, including current and aspiring practitioners, instructors, students, influencers, and anyone who uses computing technology in an impactful way

  https://www.acm.org/code-of-ethics




Control questions

Basic concepts, risk management

- What kind of problems does IT security deal with?
- What is an attacker, and why attacks can be successful against systems?
- What the system operator can do against attacks?
- What do we mean by the term system compromise?
- Define the following important concepts:
  - Confidentiality, integrity, availability (CIA)
  - Authentication, authorization, accountability (AAA)
- How do we define risk?
- What factors do affect risk?
- What are the main questions to consider during risk assessment?
- What does residual risk mean? Why can risk not be fully eliminated?

Attackers, attacks

- How can attackers be characterized?
- What information can be useful for a successful attack?
- What are the possible levels of the technical expertise of attackers?
- Why financial resources are important for attackers?
- What are the typical attacker profiles (models)? For each profile, summarize the motivation, the capabilities, and the resources available!
- What are the main features of targeted attacks?
- What is Stuxnet? Why was it important?

Vulnerabilities, countermeasures, ethics

- What type of vulnerabilities do exist in IT systems?
- What are the reasons for the existence of technical vulnerabilities?
- How do we handle discovered vulnerabilities? What does the term *responsible vulnerability disclosure* mean?
- What are zero-day vulnerabilities? Why are they dangerous?
- What type of countermeasures (security controls) do exist?
- Why ethics is important in IT security?
- List a few examples for potential ethical issues in IT security!

Levente Buttyán
buttyan@crysys.hu