Behavioural Security Group

UNIVERSITÄT BONN

Secure Software Engineering

Winterterm 2025/26

Introduction (or “What is Secure?“)

Dr. Christian Tiefenau

Discussion

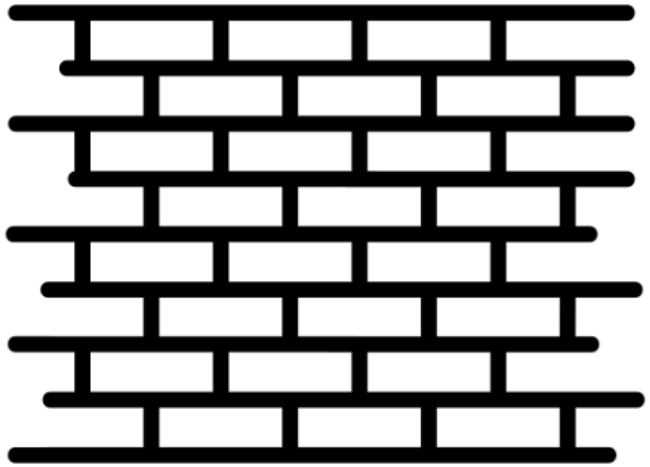
Are we safer because of these measures?

Security is not black & white, it‘s a scale

No security — realistic secure system — No system

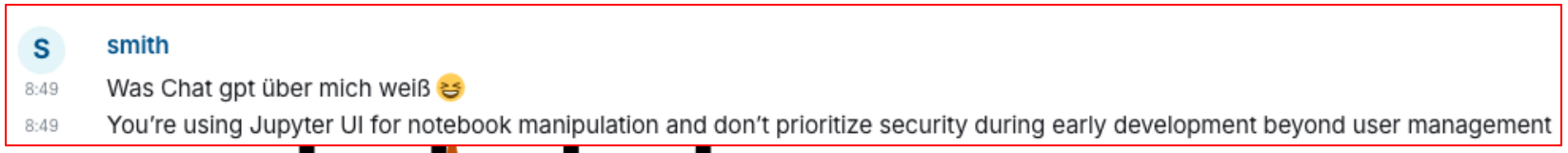
Security comes at a cost, usually privacy

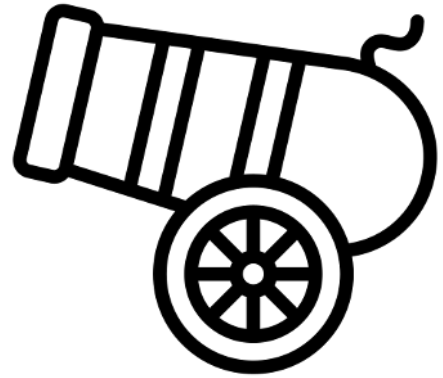
An Engineer‘s Concern



In software engineering we teach you how to ***build*** software

An Engineer‘s Concern





In software engineering we teach you how to ***build*** software

…but not as much ***breaking*** software

How do you know that you have built a system that cannot be broken into?

What evidence do you look for?

How do you know you’re done?

Outline today

- Important Terms & Relations
  - Asset, Threat, Adversary, Security, Security Policy, Safety, Compliance
  - Attack Vector, Vulnerability, Exploit, Attack
  - CIA-Triad
  - Security Properties (AAA)

- Vulnerability of the day
  - Cross-Site-Request-Forgery

Important Terms

Asset

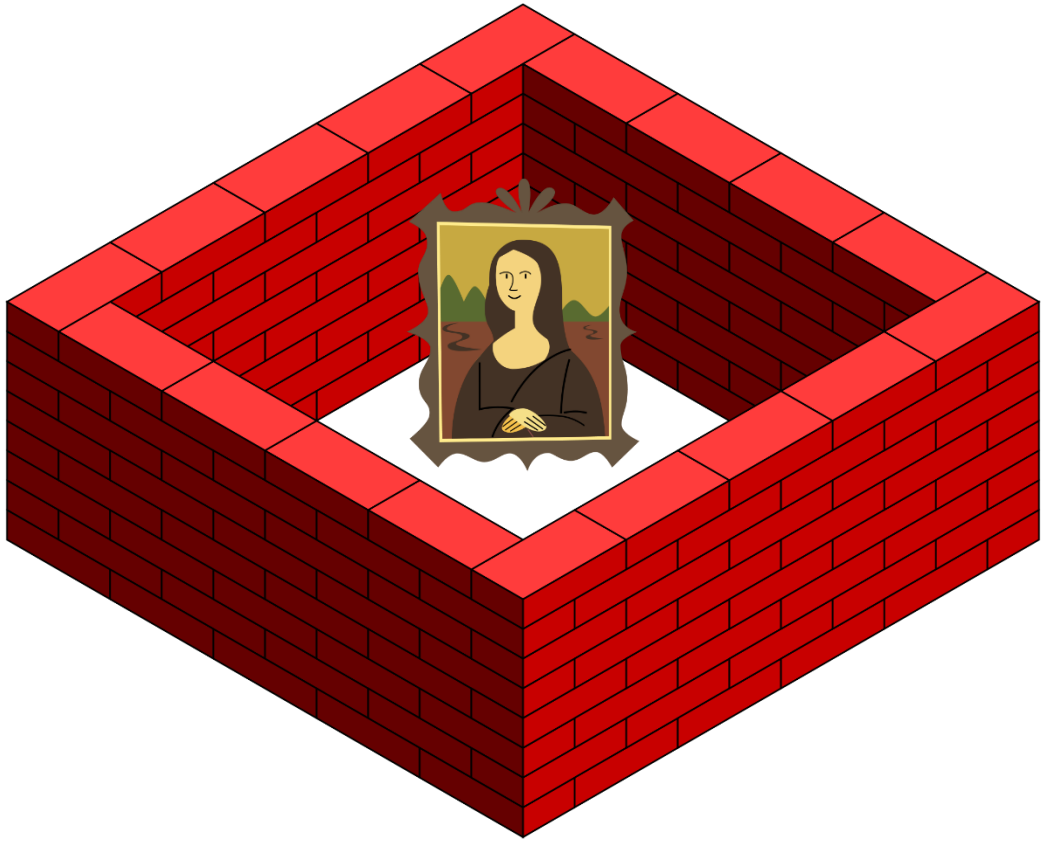
An **asset** is any tangible or intangible thing or characteristic that has value to an organization [ISO/IEC 27000:2014].

Threat & Adversary

A **threat** is a potential cause of an unwanted incident, which may result in harm to a system or organization [ISO/IEC 27000:2016].

An **adversary** is any person or a thing that acts (or has the power to act) to cause, carry, transmit, or support a threat [Younis and Malaiya 2015].

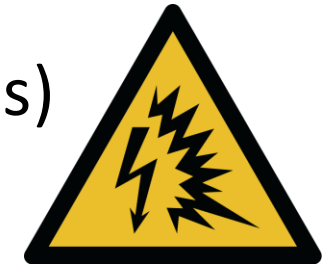
Security & Security Policy



Security provides a form of protection where a separation is created between the assets and the threat [OSSTMM 3].

A **security policy** is a definition of what it means to be secure for a system, organization or other entity.

Safety vs. Security



- **Safety** is the absence of catastrophic consequences on the user(s) and the environment [Avizienis et al. 2004].

- **Security** is concerned with the risks originating from the environment and potentially impacting the system, whereas safety deals with the risks arising from the system and potentially impacting the environment [Piètre-Cambacédèsa & Chaudet 2010].

- **Security** typically addresses malicious risks while safety addresses purely accidental risks [Piètre-Cambacédèsa & Chaudet 2010].

Challenges with term „Security“

- Exact definition depends on many factors and is specific to a given system
- Absolute statements such as “*XY is secure*” without providing the assumptions they rely on is shady
  - Typical security proofs work by reduction to assumptions that have not been refuted yet despite huge efforts
  - Often the underlying assumptions are too weak to draw such conclusions
  - Beware of marketing promises!
- Hence we *need to quantify security, but…*
- Security is very hard to quantify
  - “*We are twice as secure as we were one year ago*” does not make much sense
  - Yet we need to decide faithfully how much we invest into security

Compliance vs. Security

UNIVERSITÄT BONN

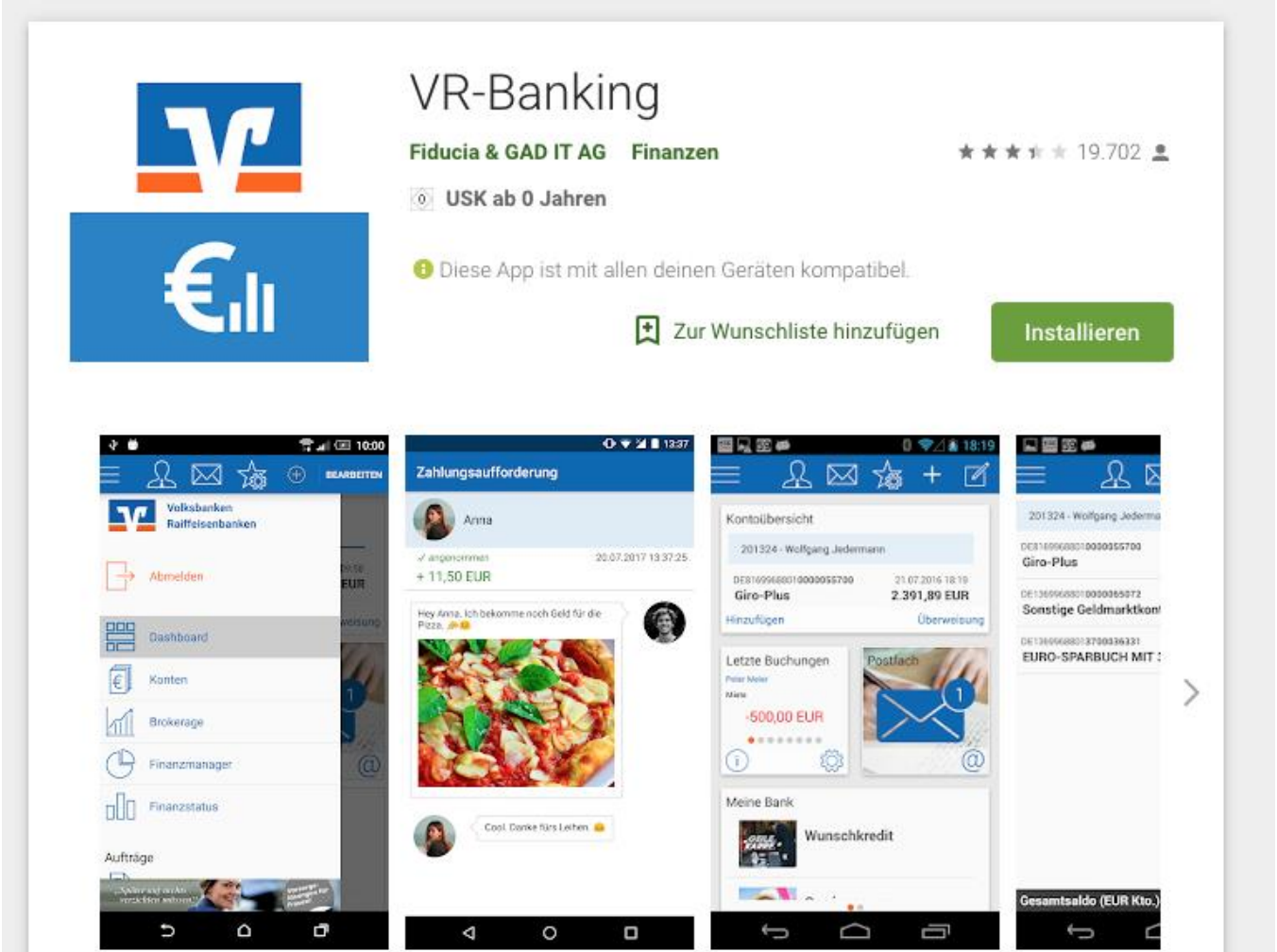
- **Compliance** means to comply with certain rules, usually laws or regulations.

- Security-compliance rules ***should*** **align** with security goals, but sometimes they don‘t or there‘s unclear evidence as to how strongly they support the security goals.

- Hence it is usually **advisable but insufficient** to be security-compliant

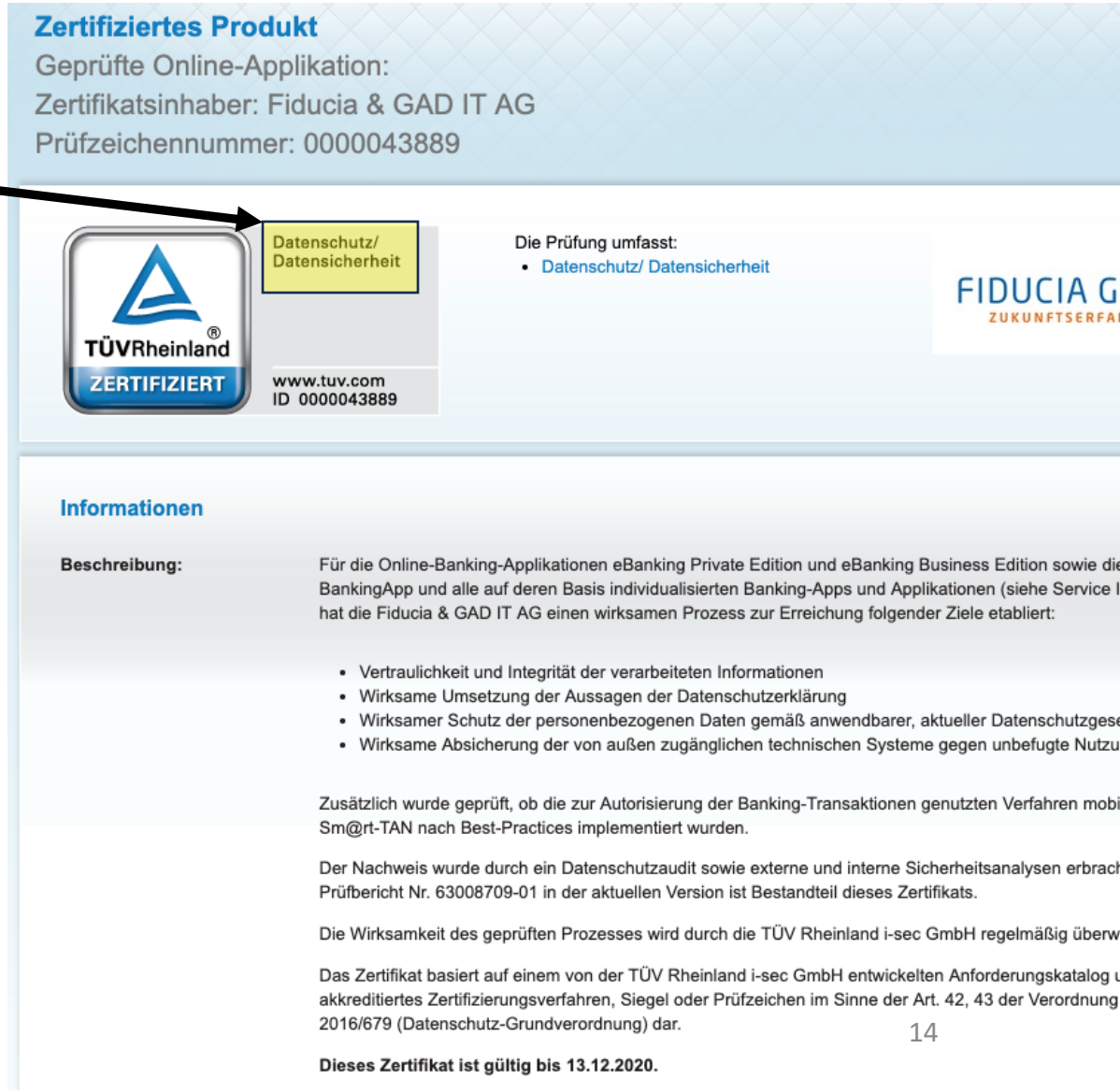
Example: Compliance vs. Security

- Banking app of a German bank (2018)



Example: Compliance vs. Security

- App was certified by TÜVRheinland
  - Privacy/Data security
- This is a good sign!



Zertifiziertes Produkt
Geprüfte Online-Applikation:
Zertifikatsinhaber: Fiducia & GAD IT AG
Prüfzeichennummer: 0000043889

TÜVRheinland® ZERTIFIZIERT
Datenschutz/ Datensicherheit
www.tuv.com
ID 0000043889

Die Prüfung umfasst:
- Datenschutz/ Datensicherheit

FIDUCIA G
ZUKUNFTSERFA

Informationen

Beschreibung:

Für die Online-Banking-Applikationen eBanking Private Edition und eBanking Business Edition sowie die BankingApp und alle auf deren Basis individualisierten Banking-Apps und Applikationen (siehe Service hat die Fiducia & GAD IT AG einen wirksamen Prozess zur Erreichung folgender Ziele etabliert:

- Vertraulichkeit und Integrität der verarbeiteten Informationen
- Wirksame Umsetzung der Aussagen der Datenschutzerklärung
- Wirksamer Schutz der personenbezogenen Daten gemäß anwendbarer, aktueller Datenschutzgese
- Wirksame Absicherung der von außen zugänglichen technischen Systeme gegen unbefugte Nutzu

Zusätzlich wurde geprüft, ob die zur Autorisierung der Banking-Transaktionen genutzten Verfahren mobi Sm@rt-TAN nach Best-Practices implementiert wurden.

Der Nachweis wurde durch ein Datenschutzaudit sowie externe und interne Sicherheitsanalysen erbrach Prüfbericht Nr. 63008709-01 in der aktuellen Version ist Bestandteil dieses Zertifikats.

Die Wirksamkeit des geprüften Prozesses wird durch die TÜV Rheinland i-sec GmbH regelmäßig überw

Das Zertifikat basiert auf einem von der TÜV Rheinland i-sec GmbH entwickelten Anforderungskatalog u akkreditiertes Zertifizierungsverfahren, Siegel oder Prüfzeichen im Sinne der Art. 42, 43 der Verordnung 2016/679 (Datenschutz-Grundverordnung) dar.

Dieses Zertifikat ist gültig bis 13.12.2020.

Example: Compliance vs. Security

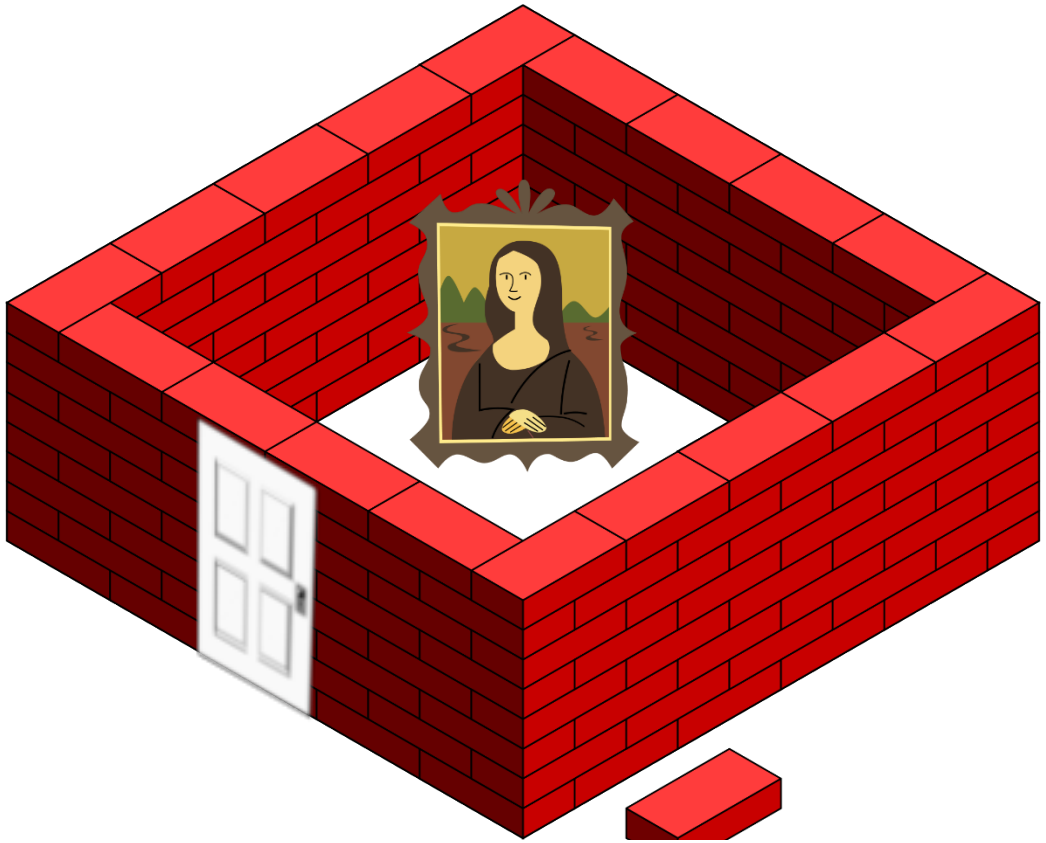
- Code of the app:

```
private final byte[] salt = { -4, 118, -128,
     -82, -3, -126, -66, -18 };
private SecretKey sk;
private SecretKeyFactory skf;

public String setupKey(String pass) {
     PBEKeySpec pbe = new PBEKeySpec(
          pass.toCharArray(), this.salt, 20);
     sk = this.skf.generateSecret(pbe);
     ...
}
```

- At least three issues:
  - Static Salt
  - Only 20 rounds of hashing
  - String used to store the password

Attack vector & Vulnerability



An **attack vector** is a path or means by which an attacker can gain access to a computer or network server in order to deliver a malicious outcome [ISO 27032:2012].

A **vulnerability** is a weakness of an asset (or control) that can be exploited by one or more threats [ISO/IEC 27000:2016].

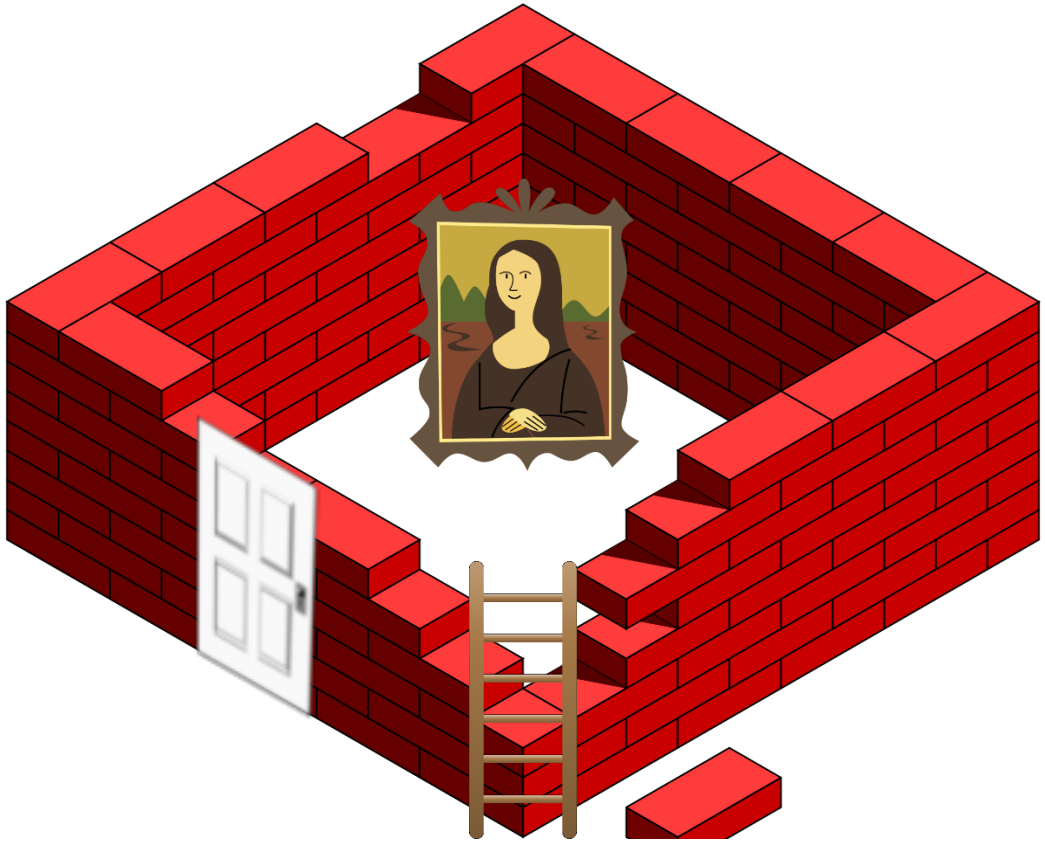
Software Vulnerability

- Informally, a bug with security consequences

- A design flaw or poor coding that may allow an attacker to exploit software for a malicious purpose
  - E.g., allowing easily-guessed passwords (poor coding)
  - E.g., complete lack of passwords when needed (design flaw)
  - McGraw: 50% are coding mistakes, 50% are design flaws

- Alternative definition: “an instance of a mistake in the specification, development, or configuration of software such that the execution can violate the explicit or implicit security policy” [Ozment 2007].

Exploit & Attack



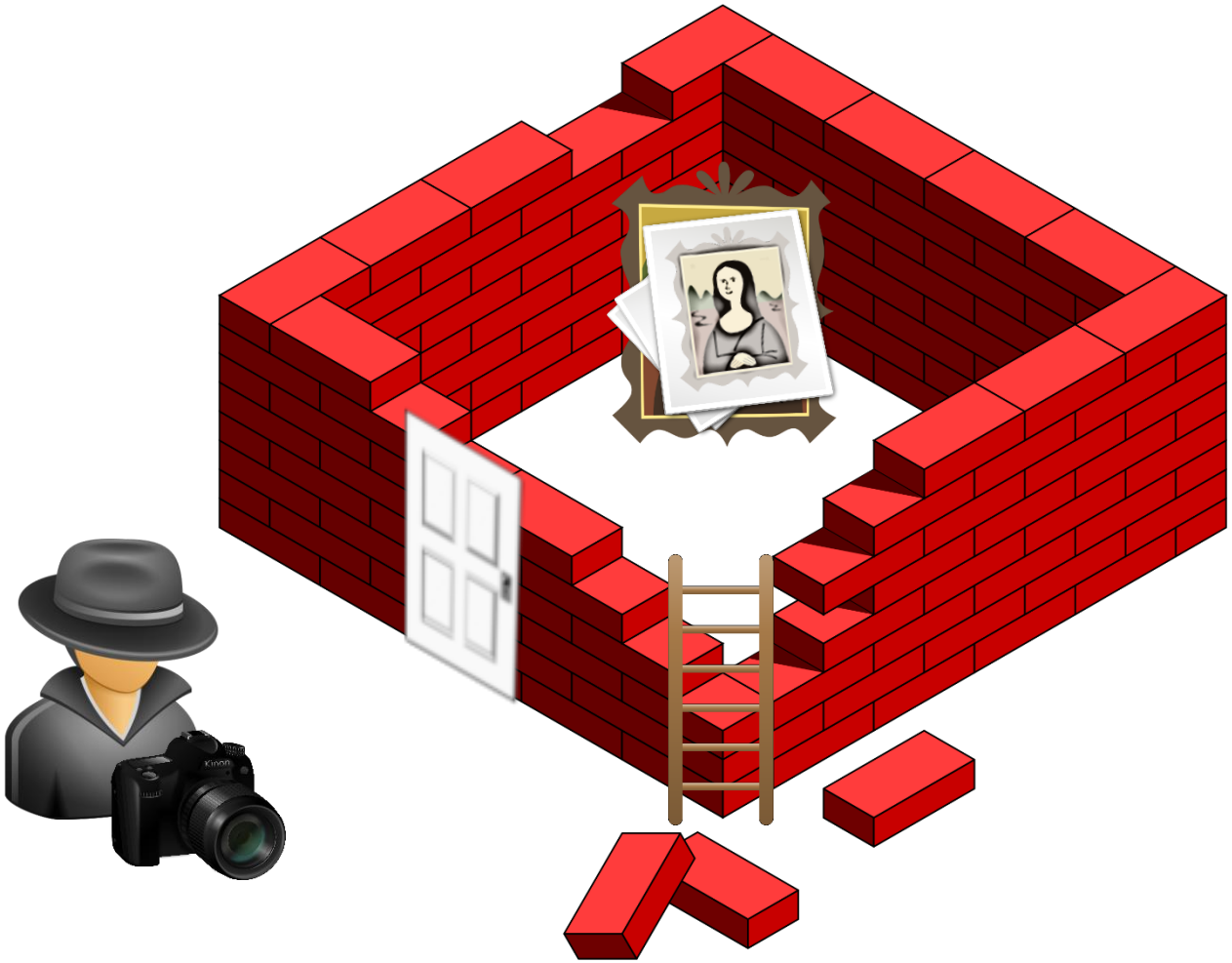
An **exploit** is a method that identifies and takes advantage of a vulnerability in an asset [Younis and Malaiya 2015].

An **attack** is an attempt to destroy, expose, alter, disable, steal or gain unauthorized access to or make unauthorized use of an asset [ISO 27000:2016].

Exploit

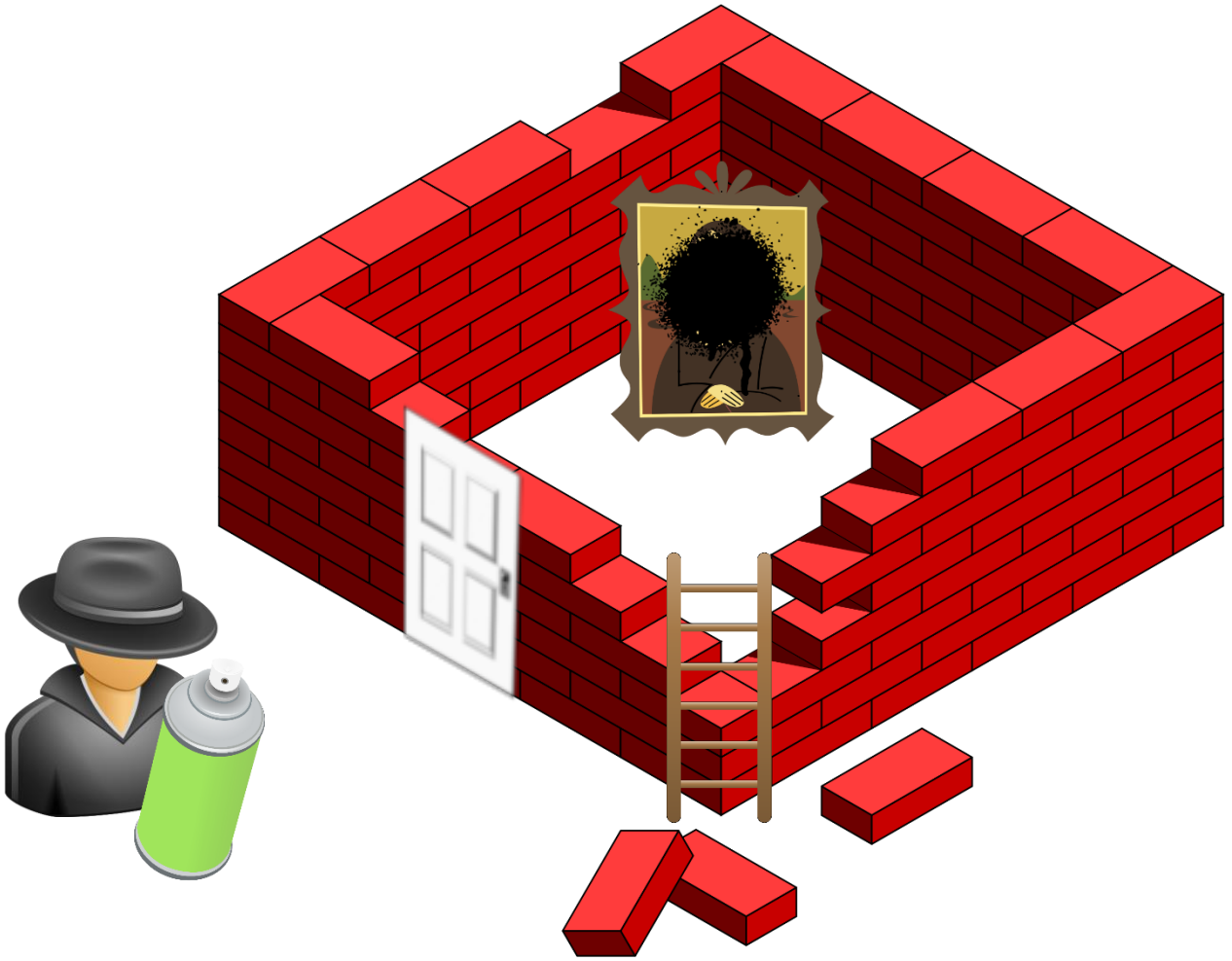
- **Exploit** – “a piece of software, a virus, a set of data, or sequence of commands that takes advantage of a vulnerability in order to cause unintended or unanticipated behavior to occur in software or an embedded device. i.e. maliciously using a vulnerability” [Frei et al. 2010].
  - Can be manual or automated
  - Malware may contain automated exploits
  - Exploits do not need to be malicious
  - Many different ways to exploit just one vulnerability

Security Properties - Confidentiality



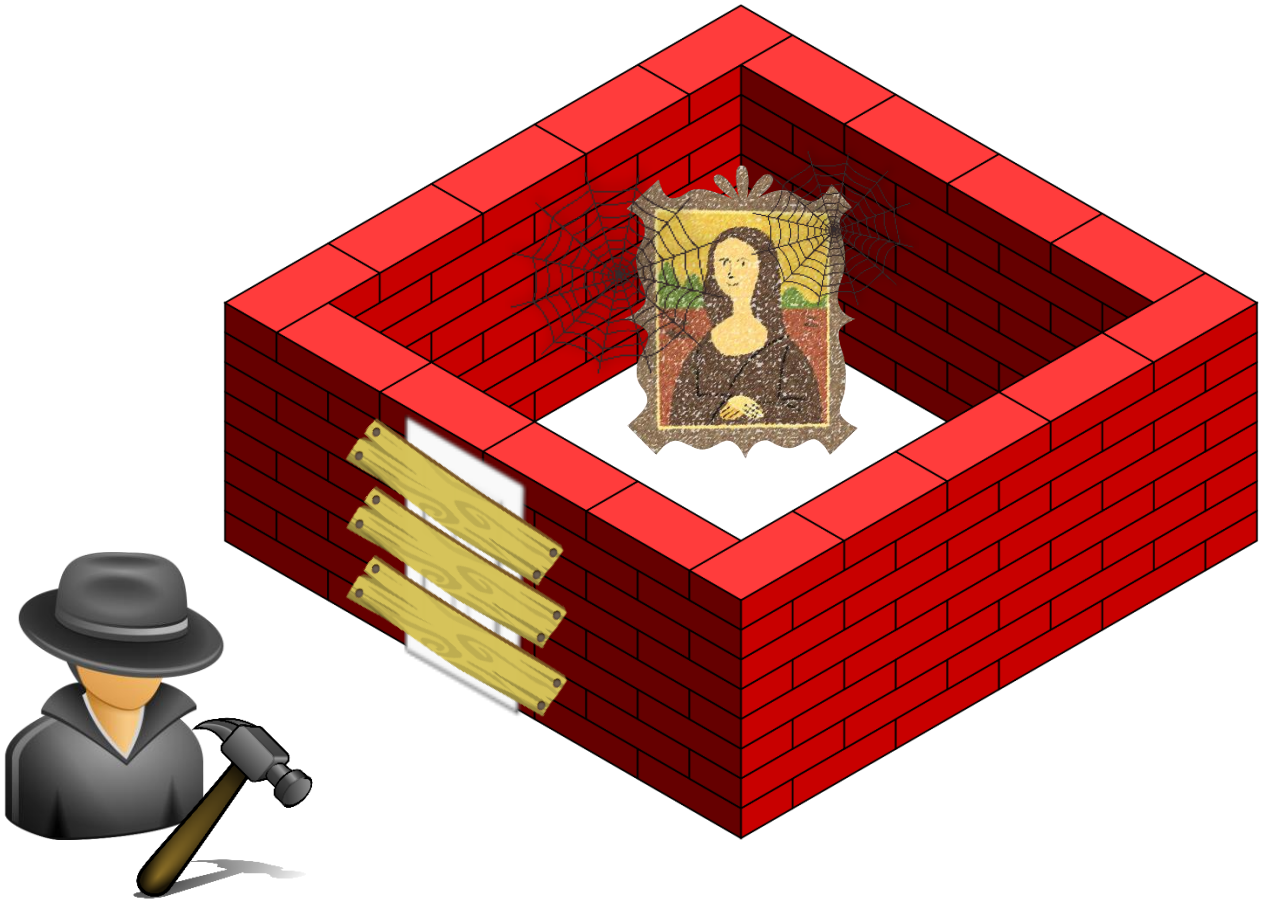
Confidentiality is the property that information is not made available or disclosed to unauthorized individuals, entities, or processes [ISO/IEC 27000:2016].

Security Properties - Integrity



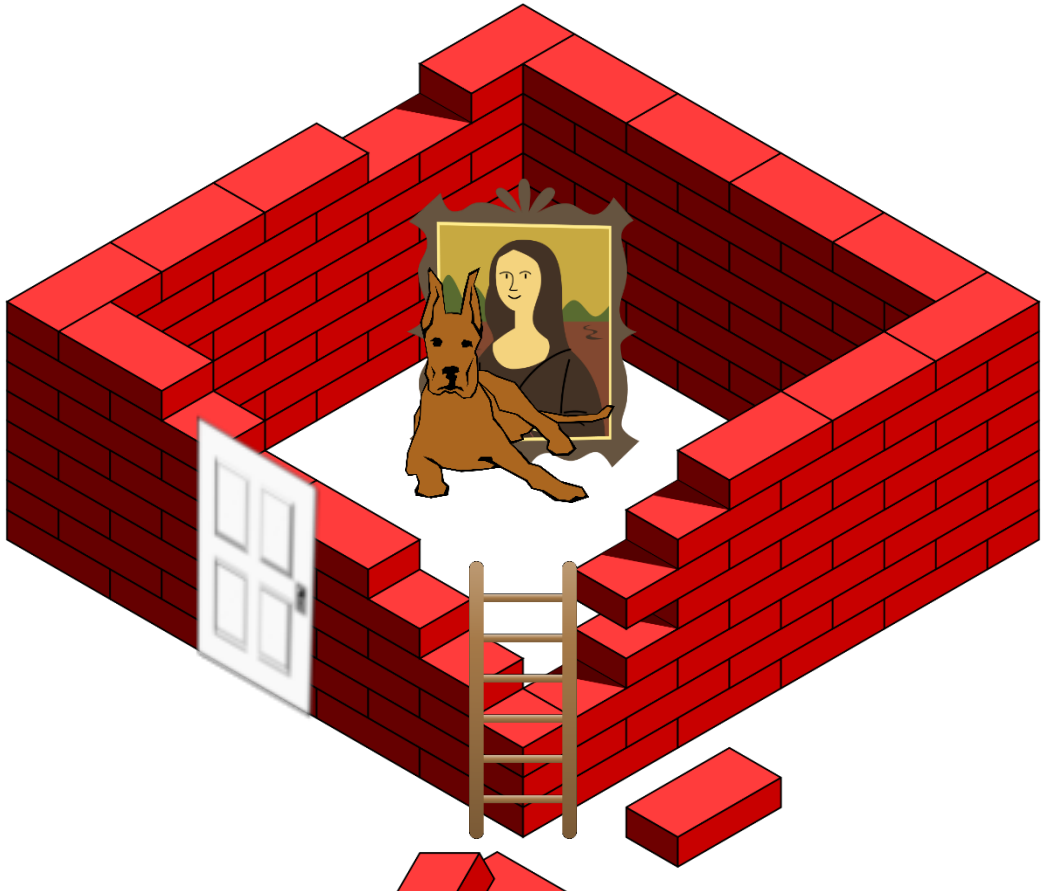
Integrity is the property of safeguarding the accuracy and completeness of assets [ISO/IEC 27000:2016].

Security Properties - Availability



Availability is the property of being accessible and usable upon demand by an authorized entity [ISO/IEC 27000:2016].

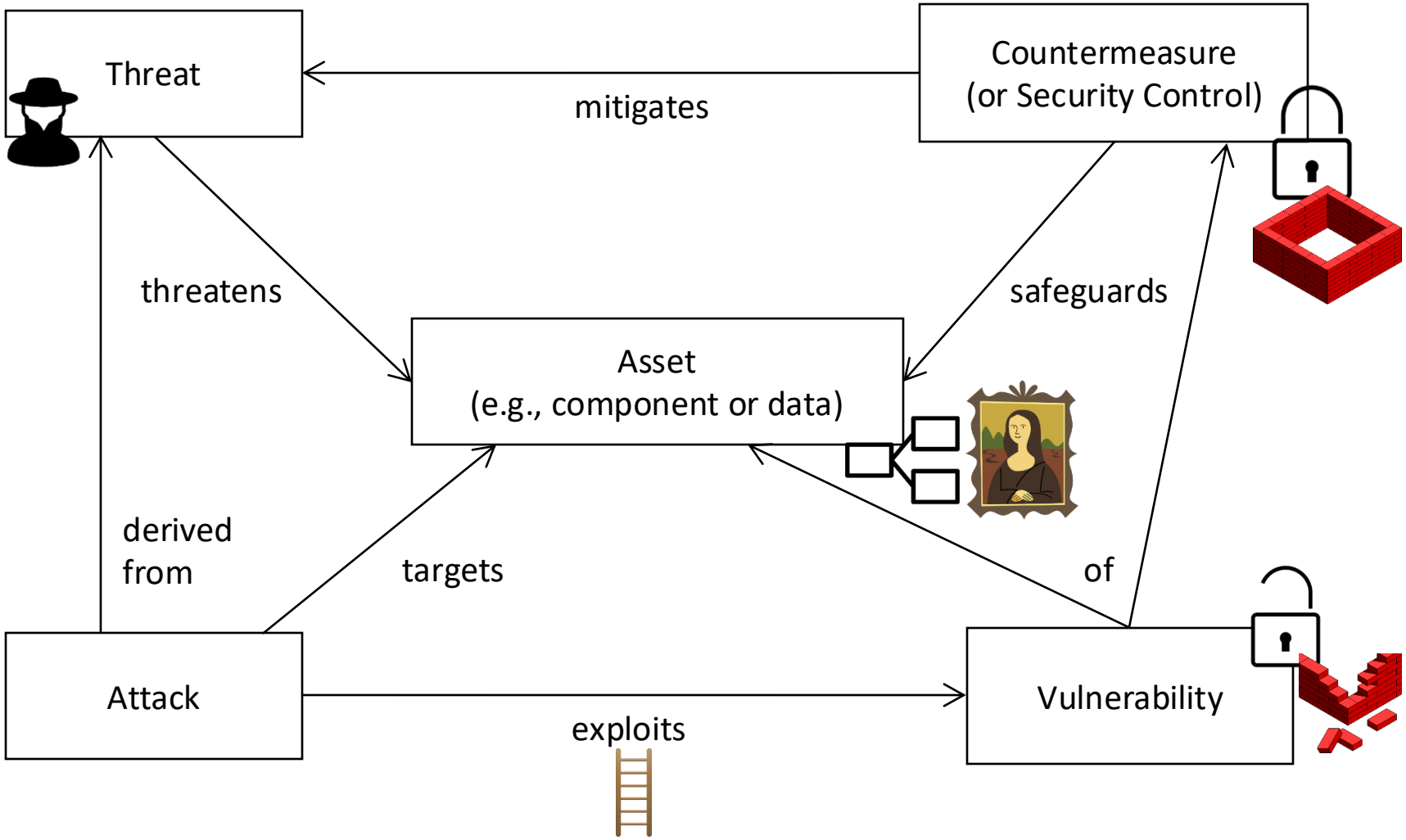
Countermeasure & Mitigation



A **countermeasure** (or **control** / **safeguard**) is used to minimize or eliminate the probability of a threat exploiting a vulnerability in an asset [Younis and Malaiya 2015].

Risk **mitigation** is the process of taking actions to eliminate or reduce the probability of compromising the confidentiality, integrity, and availability of valued information assets to acceptable levels [MS-ISAC].

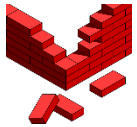
Terms & Relations (1)



[Exploit|Threat|Vulnerability] Protection

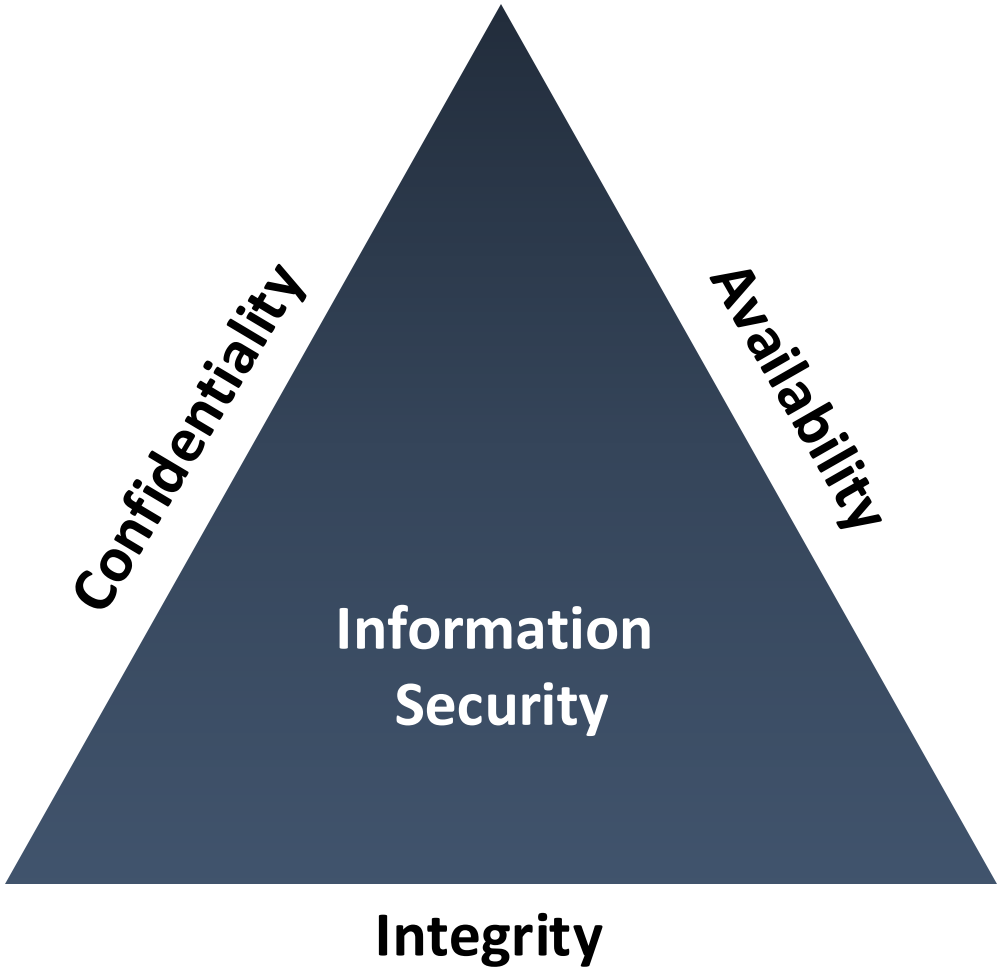
- Protect against exploits?
  - Requires protection against vulnerabilities
  - Attempts: anti-virus, intrusion detection, firewalls, etc.
    … cannot stop determined adversaries

- Protect against threats?
  - Engineer secure software!
  - Use forensics to find & eliminate
  - Mitigate by punishment, if possible
    … does not stop determined adversaries



- Protect against vulnerabilities?
  - Engineer secure software!
    … makes attacks more demanding!

CIA Triad

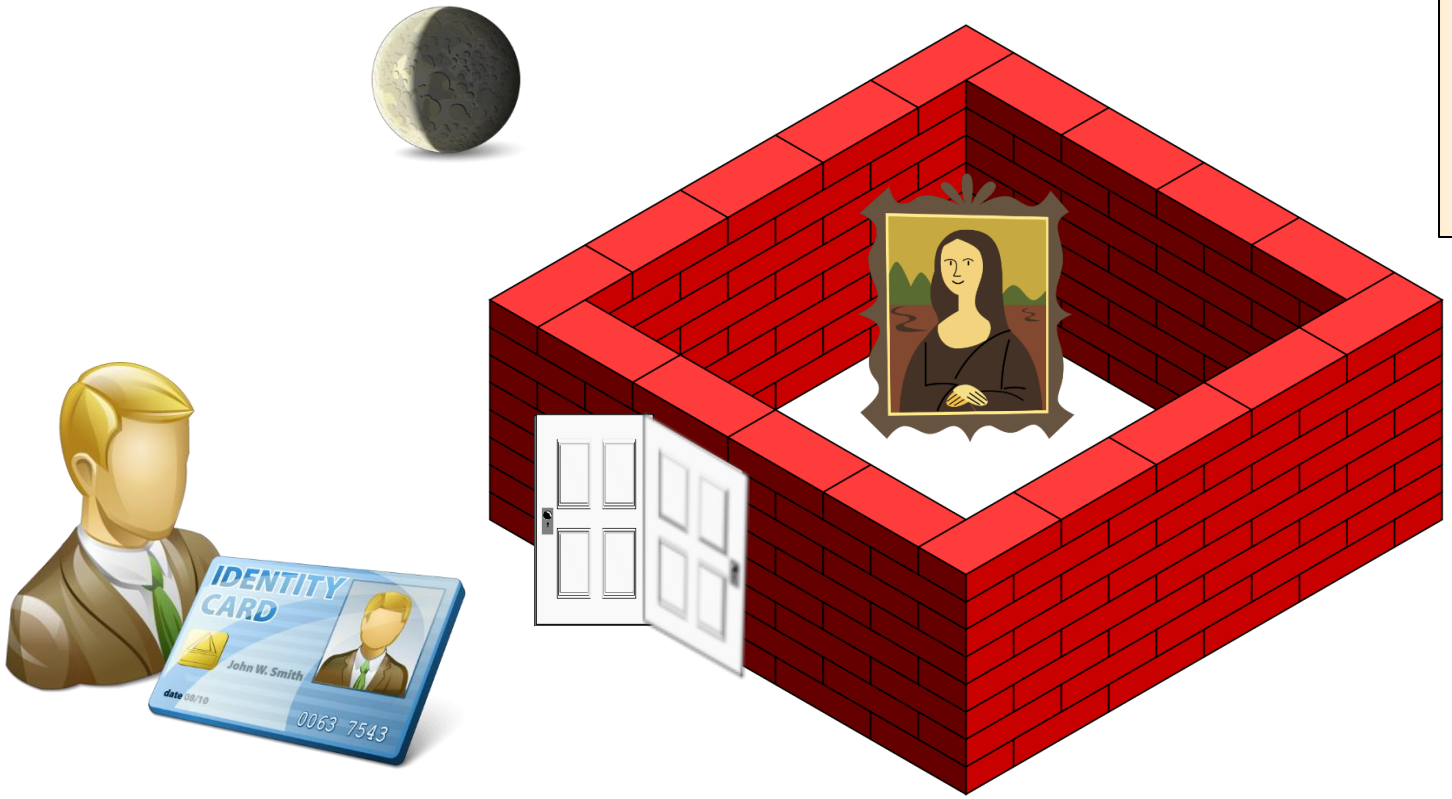

Confidentiality
Availability
Information Security
Integrity

Security Properties (Security Objectives)

- **Trusted Information (CIA triad)**
  -  Confidentiality
  -  Integrity
  -  Availability

- **Access Control (AAA principle)**
  -  Authentication
  -  Authorization
  -  Accountability / Auditability / Non-Repudiation

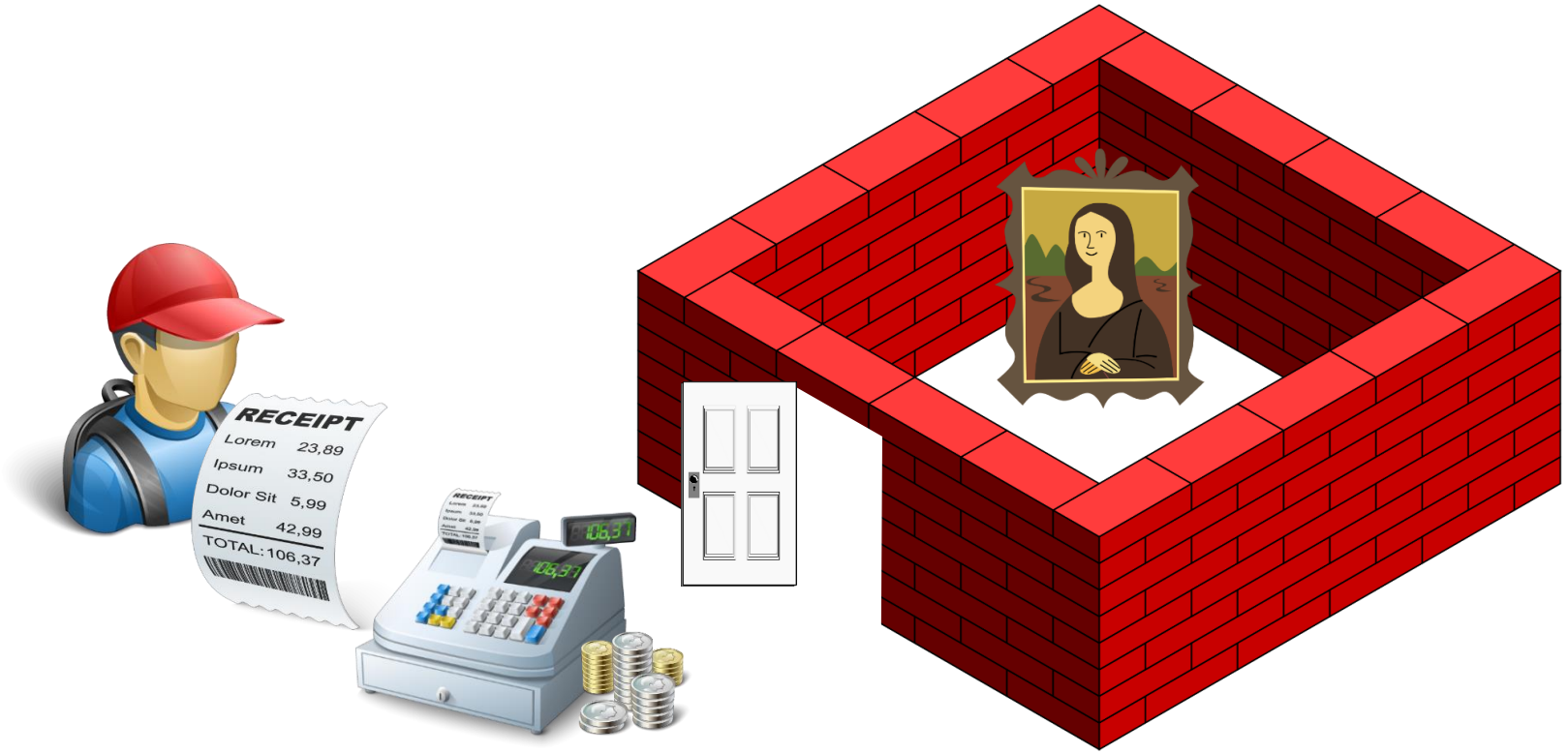
Security Properties - Authentication



Example:
„Outside working hours, persons shall authenticate themselves (e.g., the museum manager).“

- **Authentication** is the provision of assurance that a claimed characteristic of an entity is correct [ISO 27000:2016]

Security Properties – Authorization



Example:
„The painting shall only be accessible by authorized persons (that have paid).“

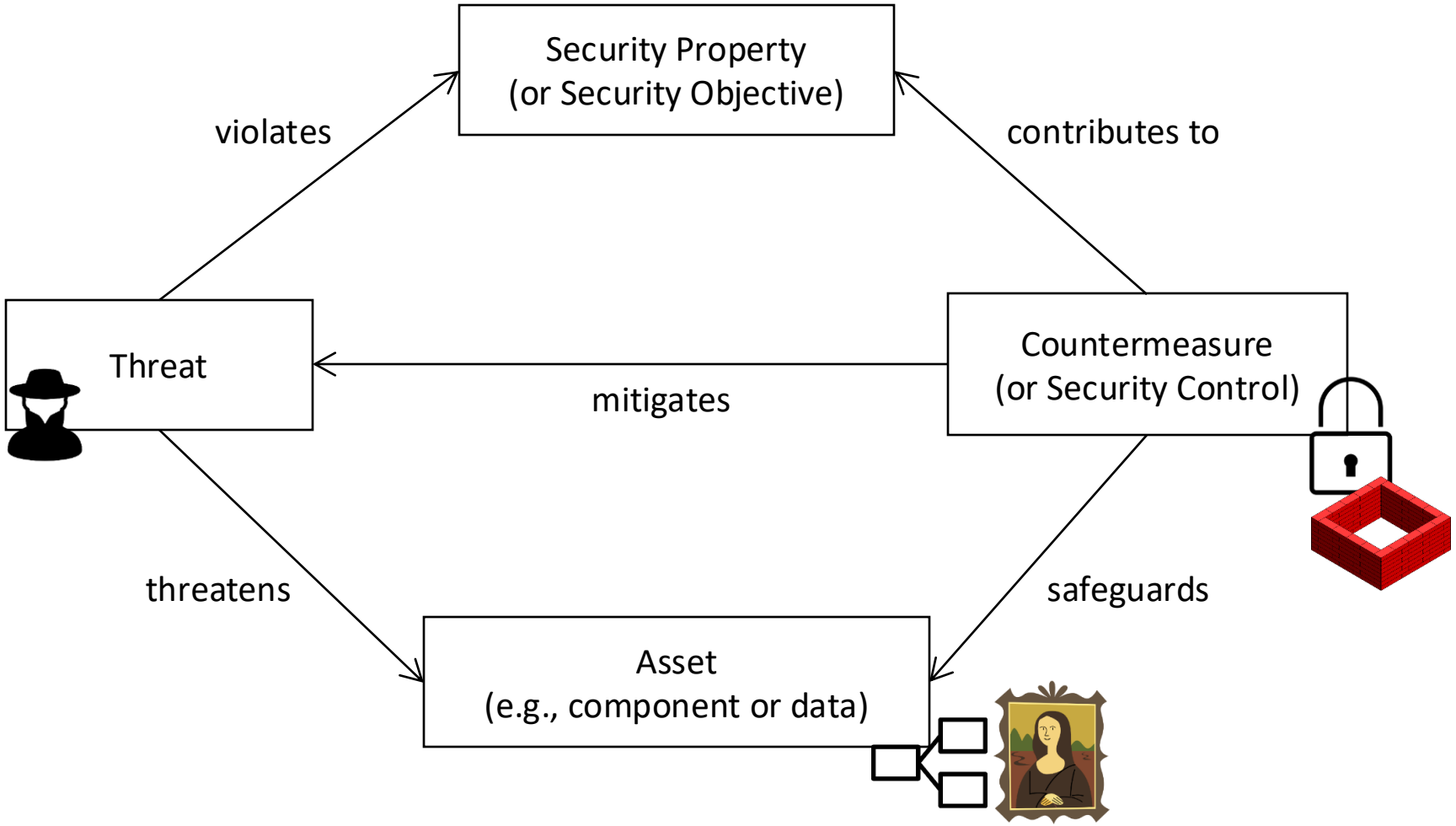
Authorization is a right or permission that is granted to a system entity to access a system resource [IEC 62443-1-1, D6E4]

Security Properties – Accountability/Auditability/Non-Repudiation

Example:
„It shall be provable, who had access to the painting.“

Non-Repudiation is the ability to prove the occurrence of a claimed event or action and its originating entities [ISO 27000:2016]

Terms & Relations (2)



Vulnerability of the Day https://votd.github.io/

Cross-Site Request Forgery

Authorization vs. Authentication vs. Encryption

UNIVERSITÄT BONN

- **Authentication:** ← **CSRF exploits inconsistent authentication**
  - Seeks to assure *who* is issuing a request
  - Typically involves checking the knowledge of a *secret* only the *right user/client/server* can know
    - Possible "secrets": password, secret cryptographic key, fingerprint
  - To not reveal the secret, the secret itself is never transmitted in plaintext, but instead simply a *proof that the secret is known*
    - More about this in Cryptography lectures!

- **Authorization**:
  - Seeks to determine which *permissions* an authenticated *user/client/server* has
  - Typically involves some kind of access-control lists

- **Encryption:**
  - Quite a different concept, really: seeks to keep data confidential. More on this later

Cross-Site Request Forgery

- **General Problem:**
  - Web application accepts state-modifying requests (typically GET) without any sort of user authentication
  - As a result, any web page running in the same browser can make those requests on the user's behalf

Cross-Site Request Forgery

- **General Problem:**
  - Web application accepts state-modifying requests (typically GET) without any sort of user authentication
  - As a result, any web page running in the same browser can make those requests on the user's behalf

evil.com

<html>
[....]
<img
src="https://facebook.com/delete_account.php?are_you_sure=yes">
</html>

Cross-Site Request Forgery

UNIVERSITÄT BONN

- **Mitigations:**
  - Modifying state only through POST requests is good practice but only makes attacks a little harder, not impossible
  - Better mitigations operate using tokens:
    - Embed pseudo-random token (nonce) into form or cookie
    - Check token when request is received at server
    - Problem: requires additional, often stateful logic on the server

Cross-Site Request Forgery

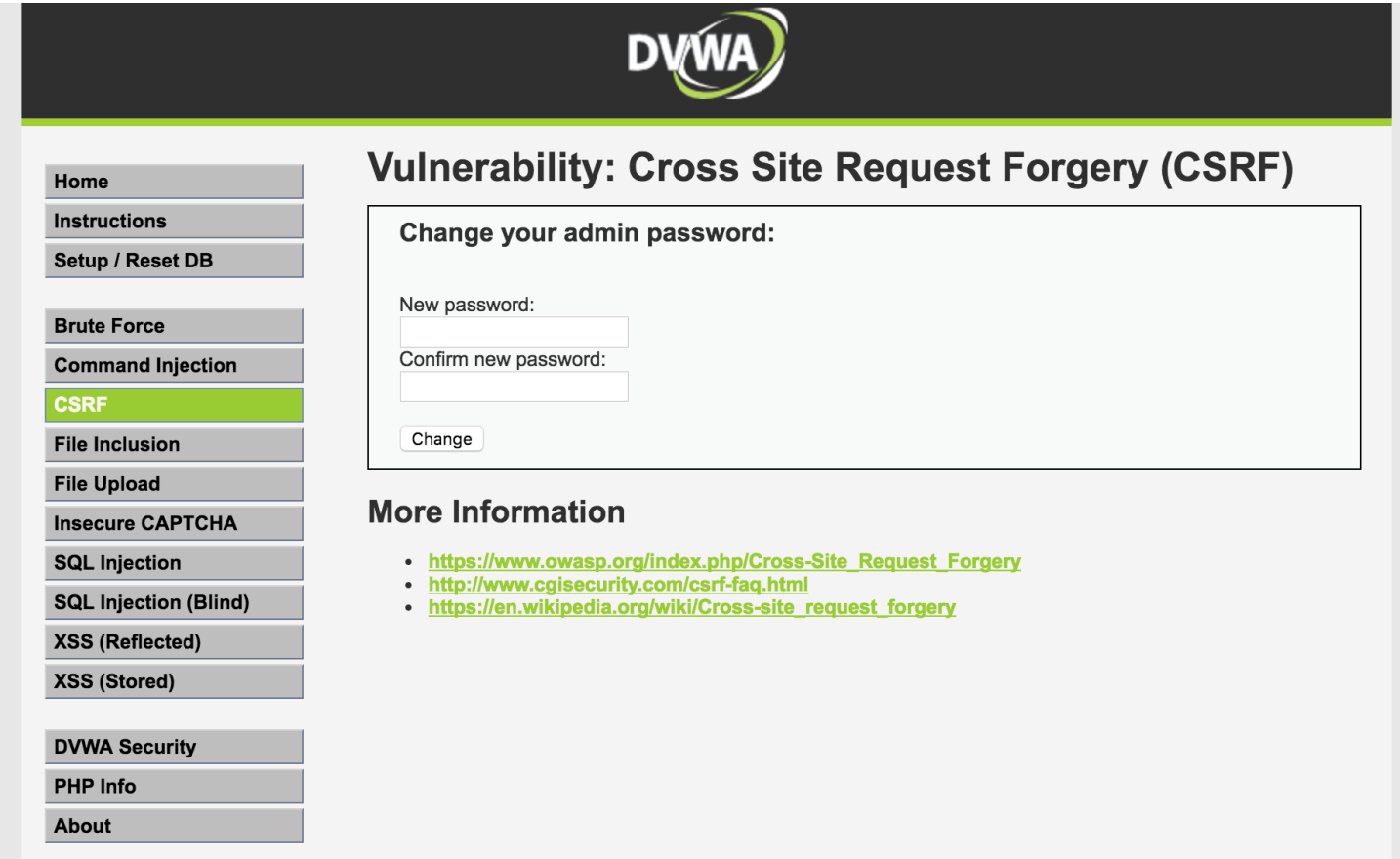
- **Real-world occurrence**

CVE-ID	
CVE-2012-0453	Learn more at National Vulnerability Database (NVD) • Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings
Description	
Cross-site request forgery (CSRF) vulnerability in xmlrpc.cgi in Bugzilla 4.0.2 through 4.0.4 and 4.1.1 through 4.2rc2, when mod_perl is used, allows remote attackers to hijack the authentication of arbitrary users for requests that modify the product's installation via the XML-RPC API.	

- XSRF is generally a threat to any web application
- https://cwe.mitre.org/data/definitions/352.html

VotD – Play around

- Site to test the workings of our VotD‘s: https://github.com/digininja/DVWA



Continuation

Software Security is…

- the process of designing, building, and testing software for security [McGraw 2004]

- following the „Security by Design“ principle

Recent security incidents

Alert! 13:27 Uhr | Security

Zero-Day-Lücken in Edge und Internet Explorer – Patches stehen noch aus

Ein Forscher hat Angriffspunkte für Universal-Cross-Site-Scripting-Attacken in Microsofts Browsern gefunden. Der Konzern scheint desinteressiert.

01.04.2019 17:27 Uhr | Security

Kritische Fehler: Schweizerische Post setzt E-Voting-System befristet aus

Nach Abschluss des öffentlichen Intrusionstests zieht die Schweizerische Post Konsequenzen für ihr E-Voting-System.

Alert! 26.03.2019 11:13 Uhr | Security

Sicherheitslücken: Einbrecher könnten Funkalarmanlage von Abus ausknipsen

Eine Funkalarmanlage von Abus ist über mehrere Sicherheitslücken angreifbar. Sicherheitsupdates gibt es bislang nicht.

Alert! 29.03.2019 10:05 Uhr | Security

Schwere Sicherheitslücke in SSL/TLS-Bibliothek axTLS

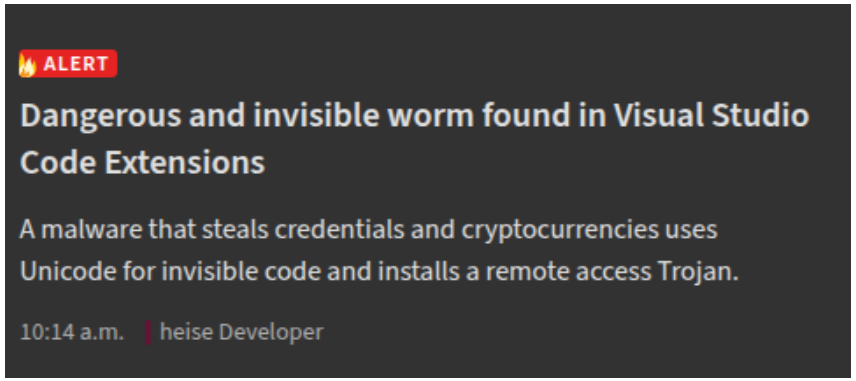
Webserver, die die Transportverschlüsselung über axTLS realisieren, sind für Angriffe empfänglich.

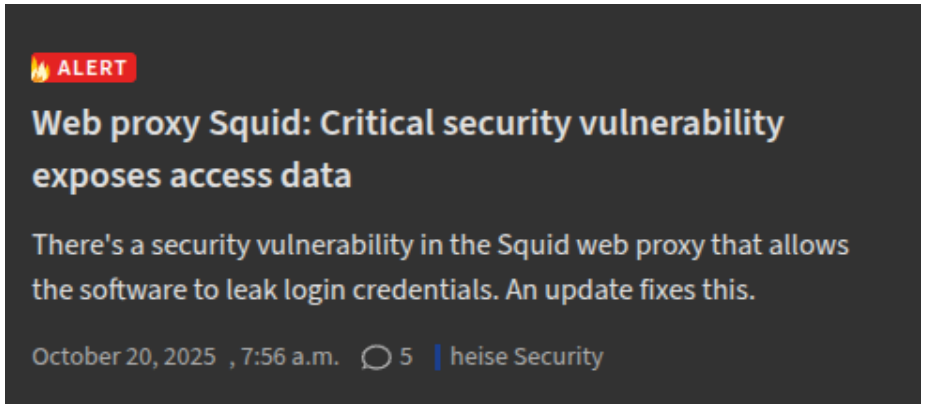
- Source: https://www.heise.de/security

Recent Security Incidents

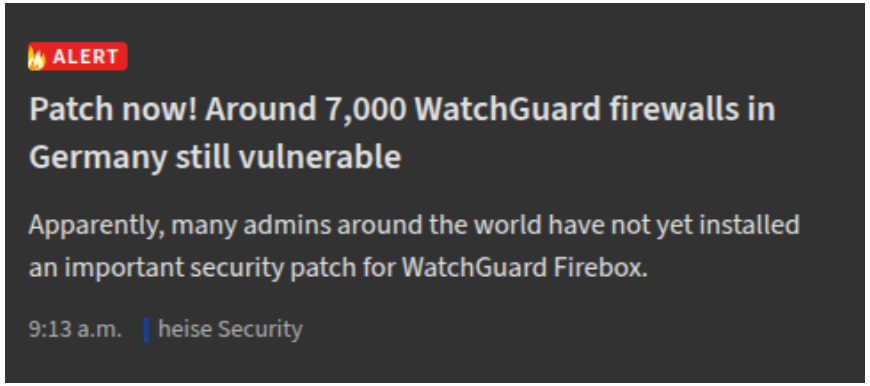
- Source: https://www.heise.de/security

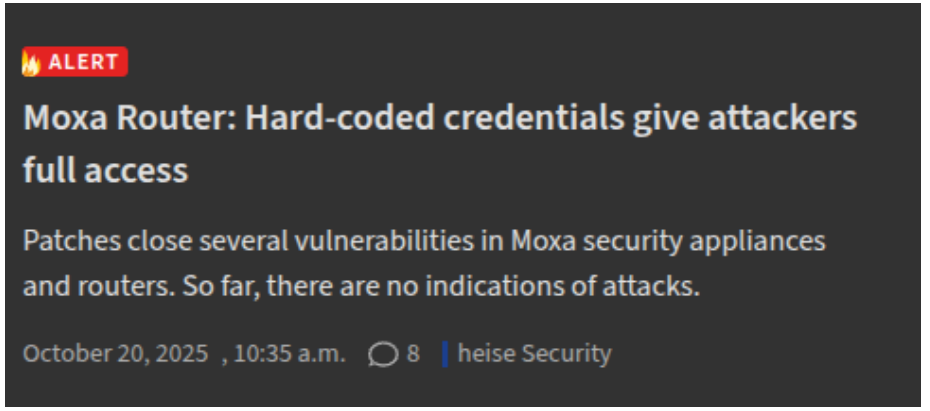
21.10.2025





21.10.2025





Software Security is…

- NOT an arcane black art
- Much of it seems arcane
  - Finding a severe vulnerability w/o source code
  - Crafting the exploit
  - Endless clever ways to break software
- But, initially you have much more knowledge about your software than the attackers do
- Don’t just leave it to the experts, take responsibility for knowing security

Software Security is…

UNIVERSITÄT BONN

- Gradual: need to be able to decide what is *probable* more than what is *possible*
- NOT a set of features
- Secure software > Security software
- Although tools and experts are helpful,
  - You can't just deploy a magical tool and expect all vulnerabilities to disappear
  - You can't outsource all of your security knowledge
- Even if you are using a security library, know *how* to use it properly

Software Security is…

- NOT a problem for just mathematicians

- Cryptography
  - Is important and needed
  - Cannot solve all of your security problems
  - Pick-proof lock vs. open window

- Proofs, access control rules, and verification are helpful, but inherently incomplete

Software Security is…

- NOT a problem for just networking and operating systems

- Software had security problems long before we had the internet

2015 DHS Study:

“90% of security incidents result from exploits against defects in software”

Software Security is…

- NOT only about internet-connected applications

- Think of “security in depth”: if the first line of defense is breached, what stops the attacker?
  - Once the attacker is in, hopping to the next more valuable asset/system is easier, typically

- Social Engineering and Phishing attacks
- “Lost” USB drives
- Insider threat (“Innentäter”)

→Company internal applications need software security as well!

Software Security is…

- NOT just about technology, but also about *people* and *processes*!
- People with low security awareness make mistakes that lead to security issues.
- Poor processes lead to random behavior of people, hindering controlled, effective and timely activities required for security.
- Poor technology leads to exploitable vulnerabilities.

Software Security is…

- A continuous challenge for everyone in an organization
  - Not just developers, all stakeholders
- A learnable mindset for software engineers
- The ability to prevent *unintended functionality*
  - At *all* layers of the software stack
  - In *all* parts of your system

Security Maturity

(1) Denial
- *I don't have to think about this. Let me just code.*
- *Leave it to the experts.*
- *I could never understand this anyway.*

(2) Irrational fear, superstition
- *EVERYTHING IS POSSIBLE NOW!!!*
- *EVERY MITIGATION IS NECESSARY!!!*
- *ENCRYPT EVERYTHING!!!*

(3) Bag of Tricks
- *Let's just try these tricks that worked in the past*
- *We've done these 10 things. That's a lot. Close enough, right?*

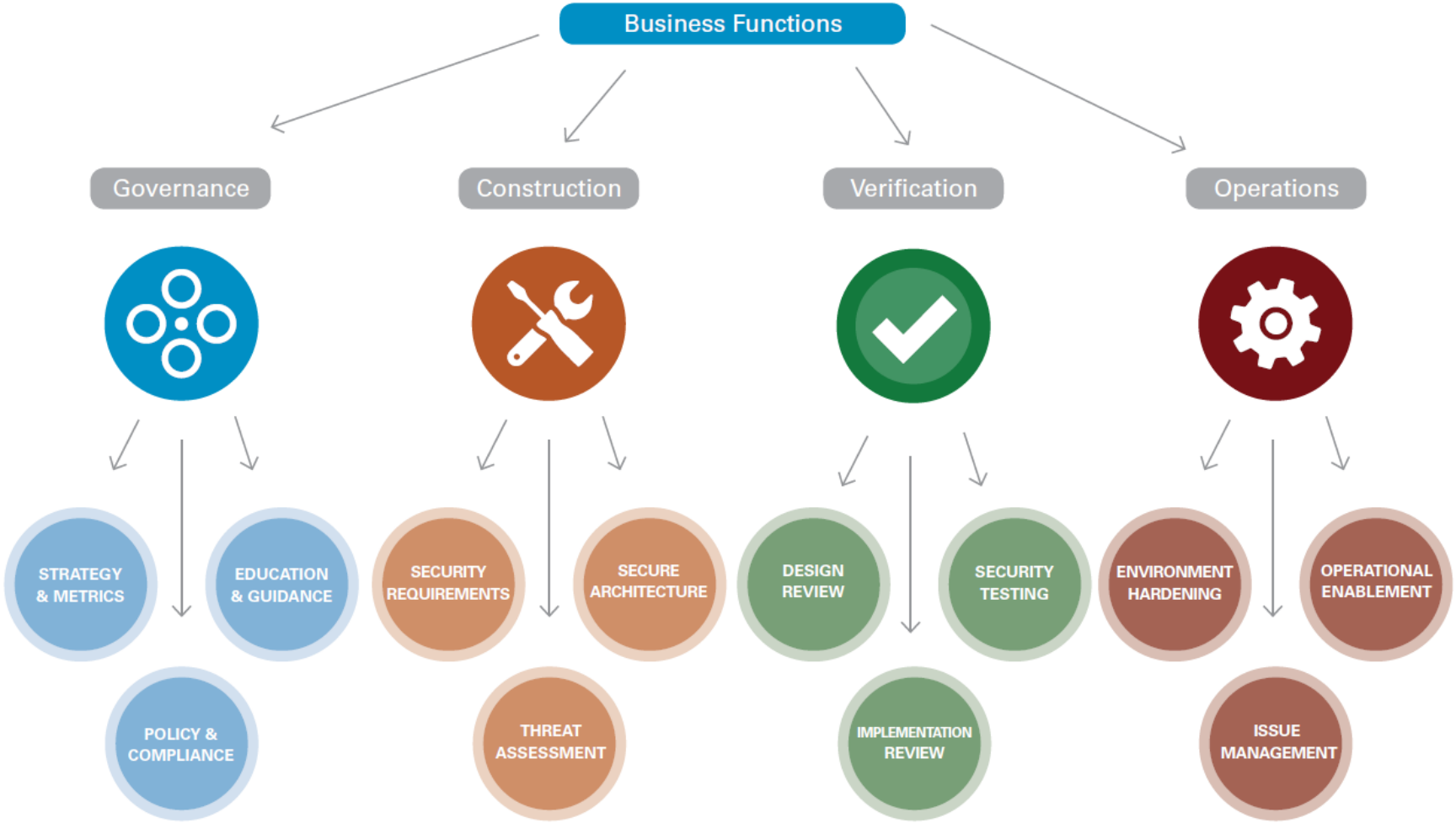
(4) Reasoned, Balanced, Defensive Mindset
- *If we do X, we mitigate Y, which is worthwhile because of Z.*

Security Maturity Models
OWASP Software Assurance Maturity Model (SAMM)

- Assessment of software development security practices maturity (like SPICE/CMMI for software development in general)

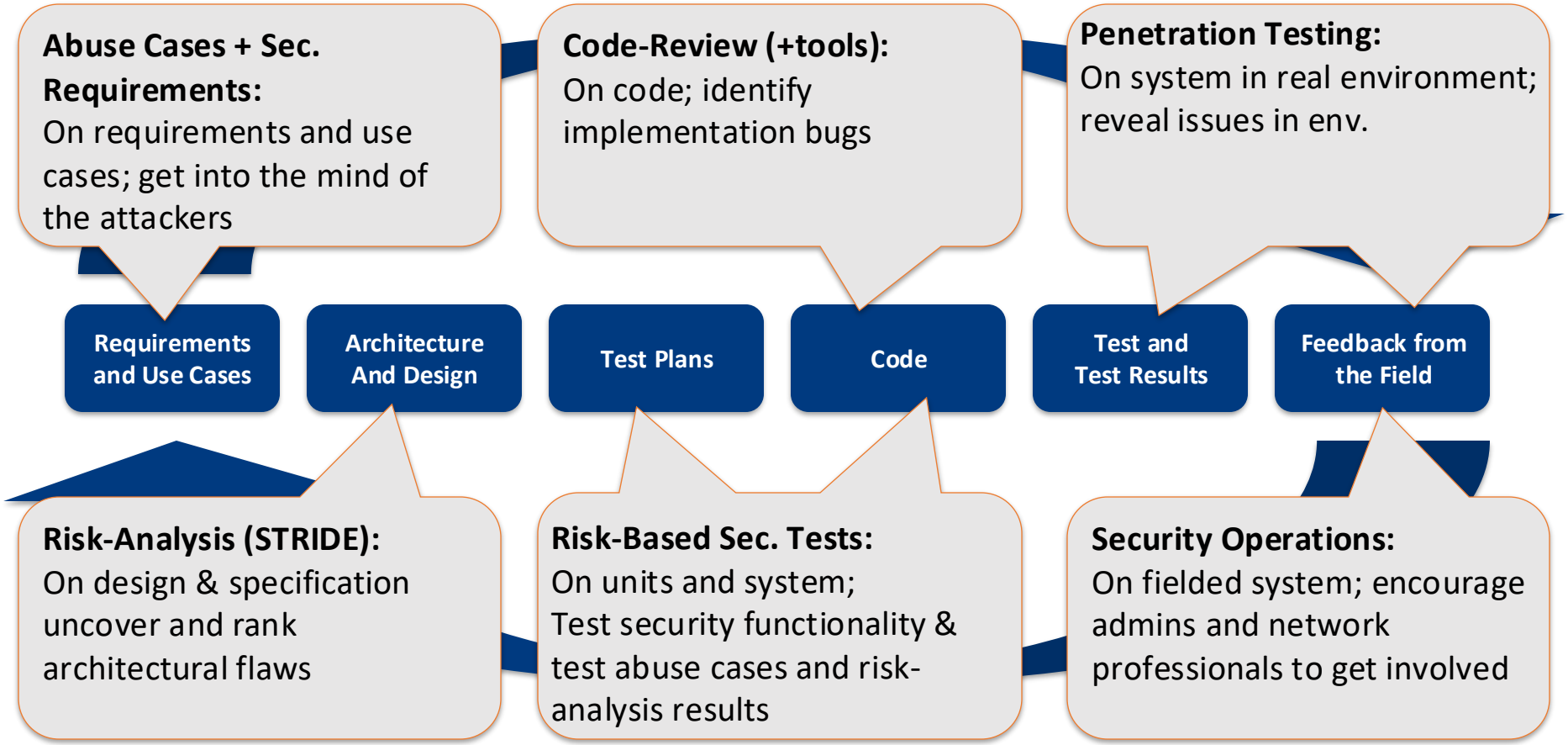
- Maturity Levels 0 to 3
  - 0: Implicit starting point (activities/practices being unfulfilled)
  - 1: Initial understanding and adhoc provision of security practices
  - 2: Increased efficiency and/or effectiveness of security practices
  - 3: Comprehensive mastery of the security practices at scale

Security Maturity Models
OWASP Software Assurance Maturity Model (SAMM)



[https://owaspsamm.org]

What's next?


Abuse Cases + Sec. Requirements:
On requirements and use cases; get into the mind of the attackers
Code-Review (+tools):
On code; identify implementation bugs
Penetration Testing:
On system in real environment; reveal issues in env.
Requirements and Use Cases
Architecture And Design
Test Plans
Code
Test and Test Results
Feedback from the Field
Risk-Analysis (STRIDE):
On design & specification uncover and rank architectural flaws
Risk-Based Sec. Tests:
On units and system;
Test security functionality & test abuse cases and risk-analysis results
Security Operations:
On fielded system; encourage admins and network professionals to get involved