

IT Security

Secure Software Engineering

Introduction + Mis-/Abuse Cases

Dr. Christian Tiefenau

whoami



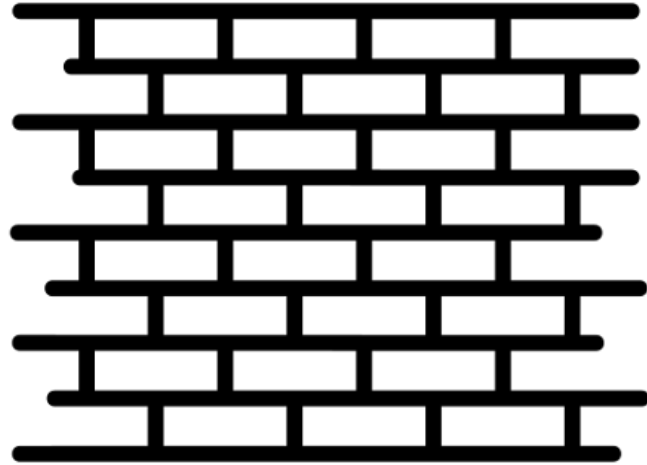
Dr. Christian Tiefenau
BESEC Group

<https://www.christiantiefenau.de>

Secure Software Engineering

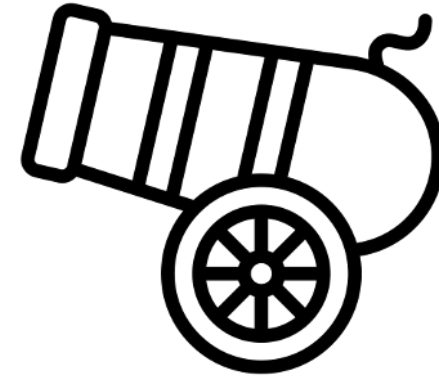
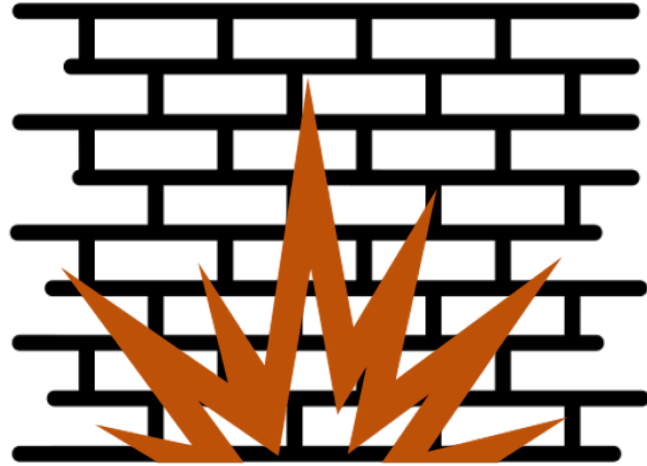
Why is it needed and what do I learn here?

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?

Software Security is...



- the process of designing, building, and testing software for security [McGraw 2004]
- following the „Security by Design“ principle

Recent security incidents



🔥 Sophos Firewall: PDF-Passwortschutz der SPX-Funktion umgebar

Sophos verteilt aktualisierte Firmware für die Firewalls. Im Secure PDF eXchange können Angreifer den Schutz umgehen und unbefugt PDF-Dateien entschlüsseln.

heute, 08:42 Uhr 5

🔥 Cisco: Schwere Sicherheitslücke in IOS XE ermöglicht Netzwerk-Übernahme

Geräte mit IOS XE und Web-UI können von Angreifern ohne Weiteres aus der Ferne übernommen werden. Cisco hat keine Patches, aber Empfehlungen für Betroffene.

17. Oktober 2023, 08:54 Uhr 16 UPDATE

🔥 Wordpress: Übernahme durch Lücke in Royal Elementor Addons and Template

Im Wordpress-Plug-in Royal Elementor Addons and Template missbrauchen Cyberkriminelle eine kritische Lücke. Sie nutzen sie zur Übernahme von Instanzen.

16. Oktober 2023, 13:18 Uhr 10

🔥 Sicherheitsupdates: Backdoor-Lücke bedroht Netzwerkgeräte von Juniper

Schwachstellen im Netzwerkbetriebssystem Junos OS bedrohen Routing-, Switching- und Sicherheitsgeräte von Juniper.

12. Oktober 2023, 10:09 Uhr 6

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

Also...



Startseite Aktuelle Sicherheitshinweise CSAF Über CERT-Bund Fragen & Antworten 🔍

[WID-SEC-2023-2739] ILIAS: Mehrere Schwachstellen

| CVSS Base Score | CVSS Temporal Score | Remoteangriff | Datum | Stand | Mitigation |
|-----------------|---------------------|---------------|------------|-------------------|------------|
| 7.2 (hoch) | 6.3 (mittel) | ja | 25.10.2023 | UPDATE 27.10.2023 | ja |

Betroffene Systeme

Betriebssystem

- Linux
- MacOS X
- Windows

Produkte

25.10.2023

- Open Source ILIAS < 8.6
- Open Source ILIAS < 7.26

Produktbeschreibung

ILIAS ist eine Open Source e-Learning Lösung.

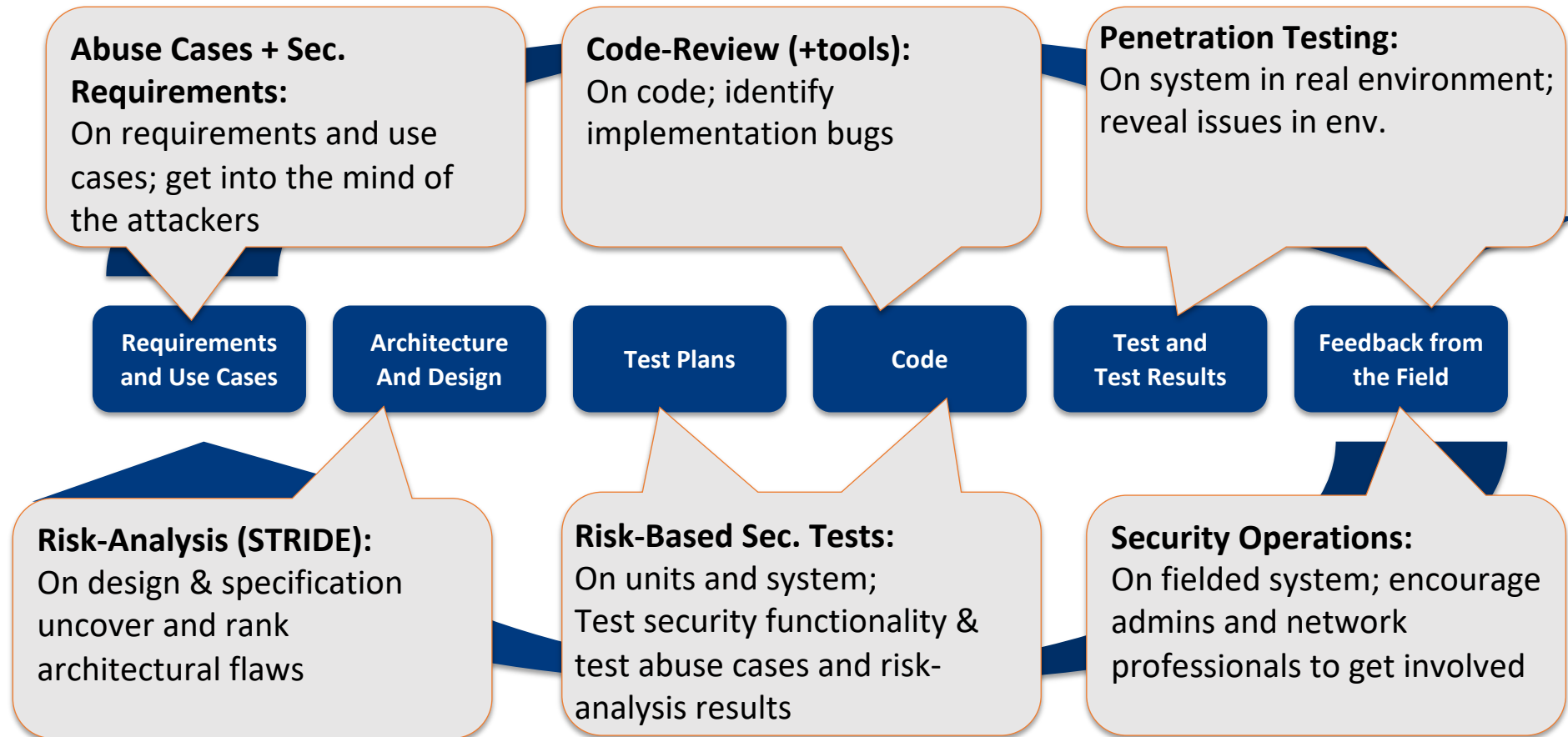
Angriff

Angriff

Ein authentisierter Angreifer kann mehrere Schwachstellen in ILIAS ausnutzen, um Dateien zu manipulieren, vertrauliche Informationen offenzulegen, Sicherheitsmaßnahmen zu umgehen und beliebigen Programmcode auszuführen.

- Source <https://wid.cert-bund.de/portal/wid/securityadvisory?name=WID-SEC-2023-2739>

SSE - Overview



Lecture - Components



- Vulnerability of the Day
 - Presentation of several code-level vulnerabilities
 - How to spot, prevent and mitigate them
 - From the Common Weakness Enumeration (<https://cwe.mitre.org>)
- Secure Software Development Lifecycle (SDLC)
 - How can we prevent vulnerabilities with secure software design?
 - Presentation of the SDLC-elements
 - Research findings

Vulnerability of the Day

SQL-Injection

Exploits of a Mom



xkcd.com/327

| OWASP Top Ten 2010 | |
|--------------------|-----------------------|
| 1 | Injection |
| 2 | Cross-Site Scripting |
| 3 | Broken Authentication |

| OWASP Top Ten 2013 | |
|--------------------|-----------------------|
| 1 | Injection |
| 2 | Broken Authentication |
| 3 | Cross-Site Scripting |

| OWASP Top Ten 2017 | |
|--------------------|-----------------------|
| 1 | Injection |
| 2 | Broken Authentication |
| 3 | Cross-Site Scripting |

SQL Injection - Setting

- Login to a system using an SQL database
- Ask for username and password
- Grant access only if authenticated correctly

```
//Get user input
System.out.print("Enter username: ");
Scanner scanner = new Scanner(System.in);
String user = scanner.nextLine();
System.out.print("Enter password: ");
String password = scanner.nextLine();

//Try to authenticate
System.out.println(auth(user, password, conn));
```

Goal:

```
>Enter username: bobby
>Enter password: table
```

Authenticated!!

```
>Enter username: sse
>Enter password: something
```

Not Authenticated!!

SQL Injection – Unsafe Implementation

```
private static String auth(String u, String pwd, Connection conn) throws
SQLException {
    ResultSet resultSet;
    //get results for user input
    resultSet = conn.createStatement().executeQuery(
        "SELECT * FROM Users WHERE Username='" + u + "' AND Password='" + pwd + "'");

    if (resultSet.next()) // any rows?
        return "Authenticated!!";
    else
        return "Not authenticated!!";
}
```

```
SELECT * FROM Users WHERE Username='bobby' AND Password='table'
```

SQL Injection – Unsafe Implementation

```
private static String auth(String u, String pwd, Connection conn) throws
SQLException {
    ResultSet resultSet;
    //get results for user input
    resultSet = conn.createStatement().executeQuery(
        "SELECT * FROM Users WHERE Username='" + u + "' AND Password='" + pwd + "'");

    if (resultSet.next()) // any rows?
        return "Authenticated!!";
    else
        return "Not authenticated!!";
}
```

```
SELECT * FROM Users WHERE Username='bobby' AND Password='table' OR '='
```

SQL Injection – Mitigation



- Escaping “bad” characters
 - Used character sets might change over time
 - Restrictions can degrade strength of passwords
- Use *Prepared Statements*
 - Separate the logic of the query from the input parameters
 - User input will not be part of the executable code

SQL Injection – Safe Implementation

```
private static String safe(String u, String pwd, Connection conn)
throws SQLException {
    PreparedStatement ps;
    //define prepared statement
    ps = conn.prepareStatement("SELECT * FROM Users WHERE Username=?
AND Password=?");
    ps.setString(1, u);
    ps.setString(2, pwd);
    //get results for user input executing the prepared statement
    ResultSet resultSet = ps.executeQuery();
    if (resultSet.next()) // any rows?
        return "Authenticated!!";
    else
        return "Not authenticated!!";
}
```

SQL Injection – Examples & Classification



| CVE-ID | |
|--|--|
| CVE-2018-11309 | Learn more at National Vulnerability Database (NVD) • CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information |
| Description | |
| Blind SQL injection in coupon_code in the MemberMouse plugin 2.2.8 and prior for WordPress allows an unauthenticated attacker to dump the WordPress MySQL database via an applyCoupon action in an admin-ajax.php request. | |

CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

| | |
|--|----------------------|
| Weakness ID: 89 Abstraction: Base Structure: Simple | Status: Draft |
|--|----------------------|

SQL Injection – Examples & Classification



Security › 7-Tage-News › 04/2020 › Warten auf Patches: Schwachstellen in Nagios XI gefährden Netzwerke

🔥 Alert!

Warten auf Patches: Schwachstellen in Nagios XI gefährden Netzwerke

Die Monitoring-Software für komplexe IT-Infrastrukturen Nagios XI ist verwundbar. Abhilfe gibt es noch nicht.

Lesezeit: 1 Min. In Pocket speichern

3



(Bild: Gorodenkoff/Shutterstock.com)

20.04.2020 11:52 Uhr | Security

Secure Software Engineering

Important Terms

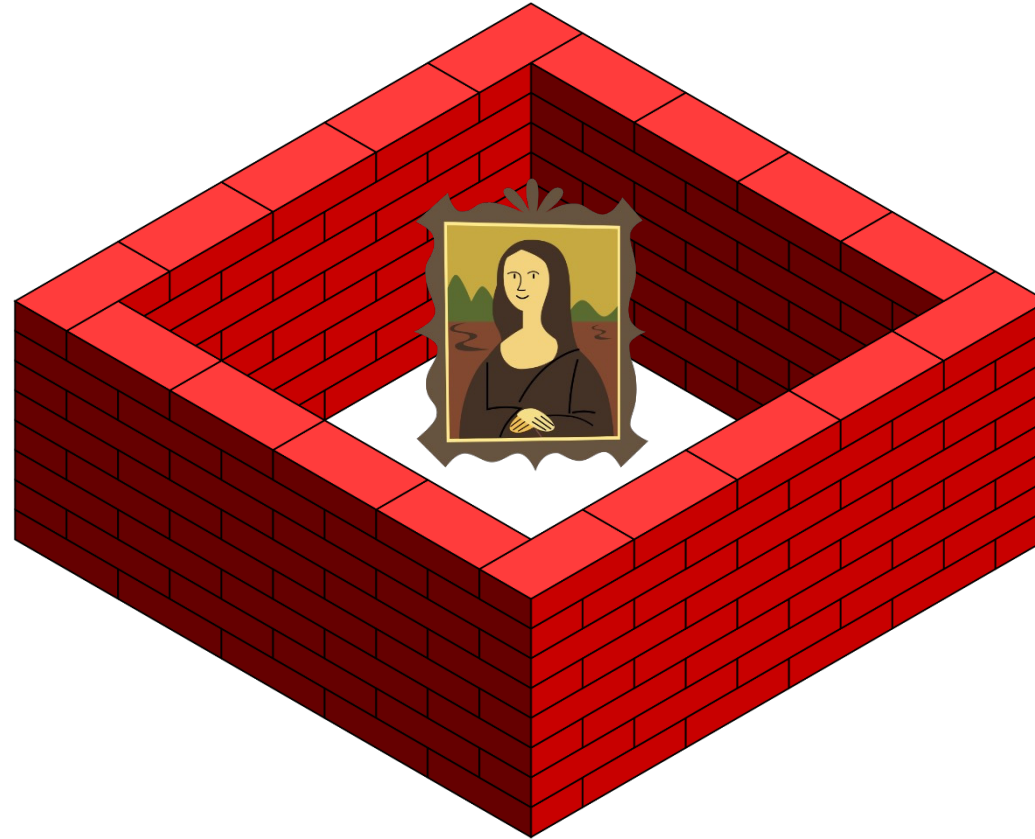
Asset



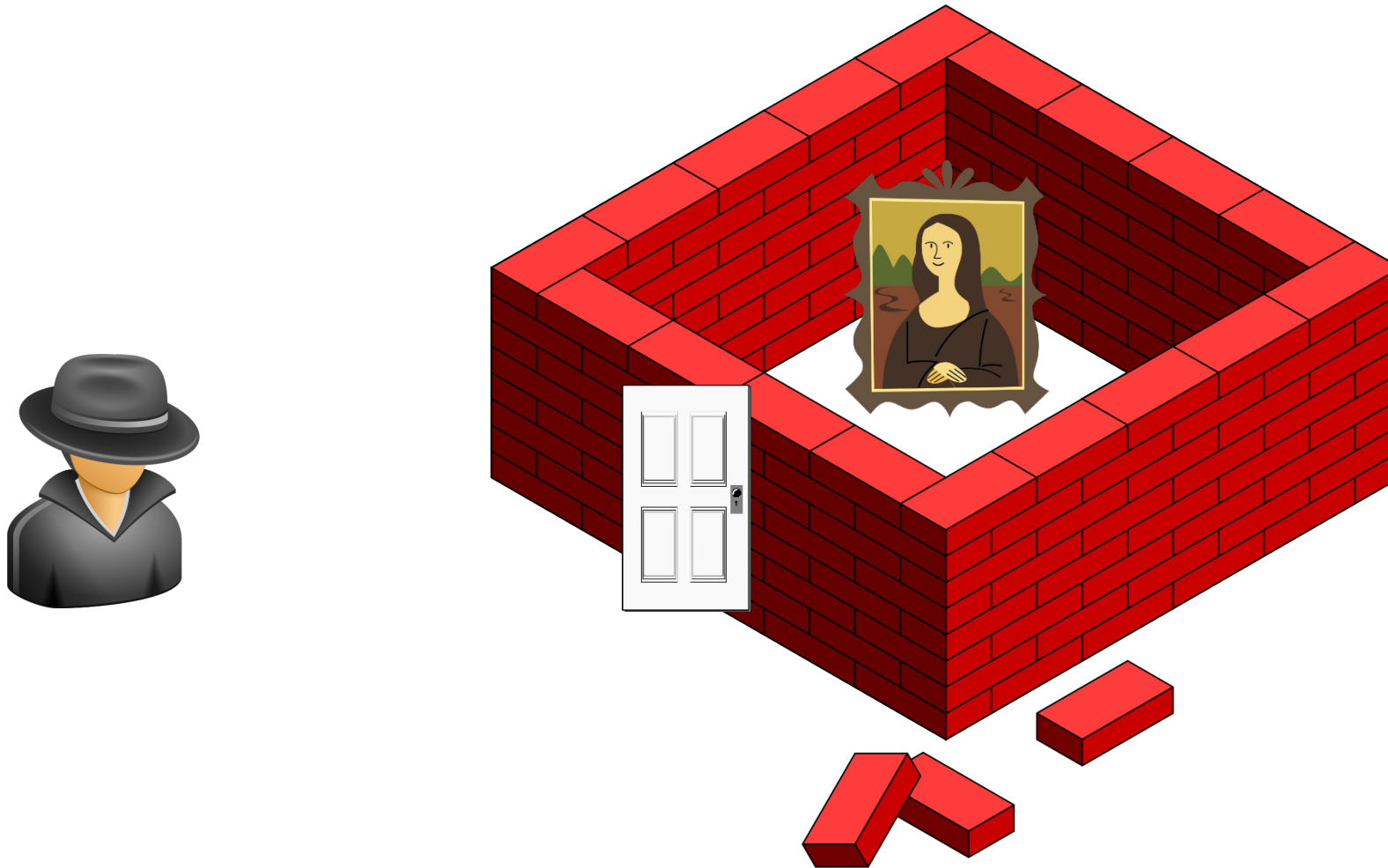
Threat & Adversary



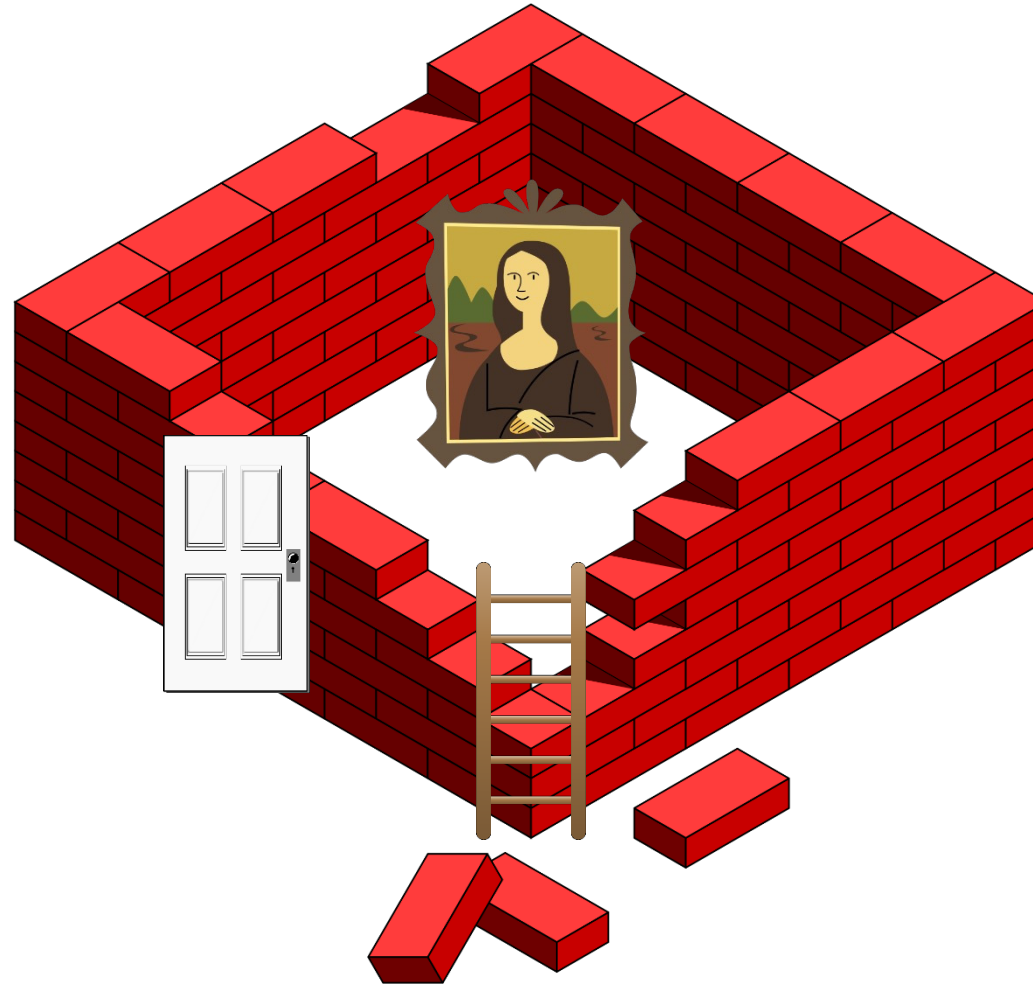
Security & Security Policy



Attack vector & Vulnerability



Exploit & Attack

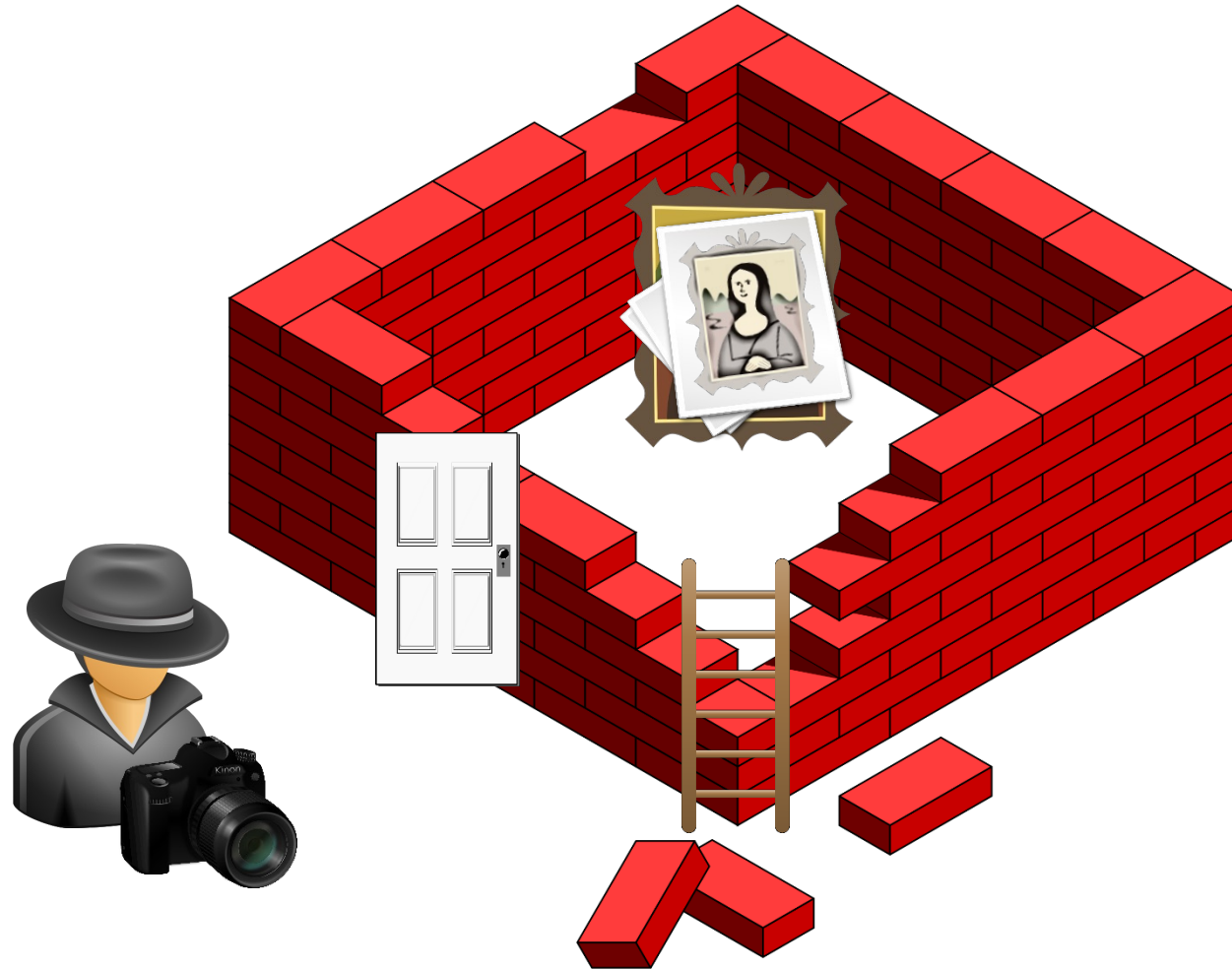


Security Properties

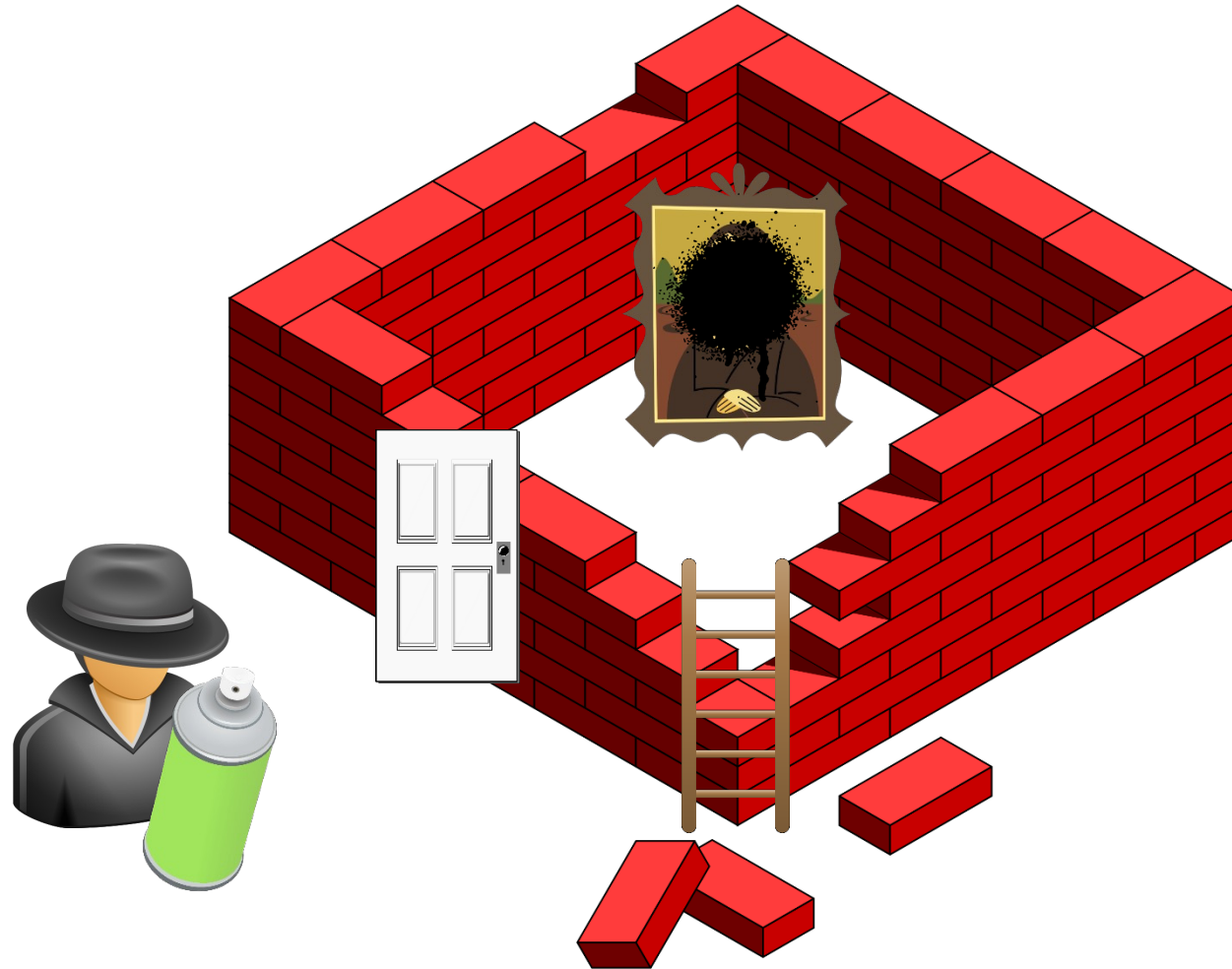
Confidentiality, Integrity, Availability : The CIA-Triad



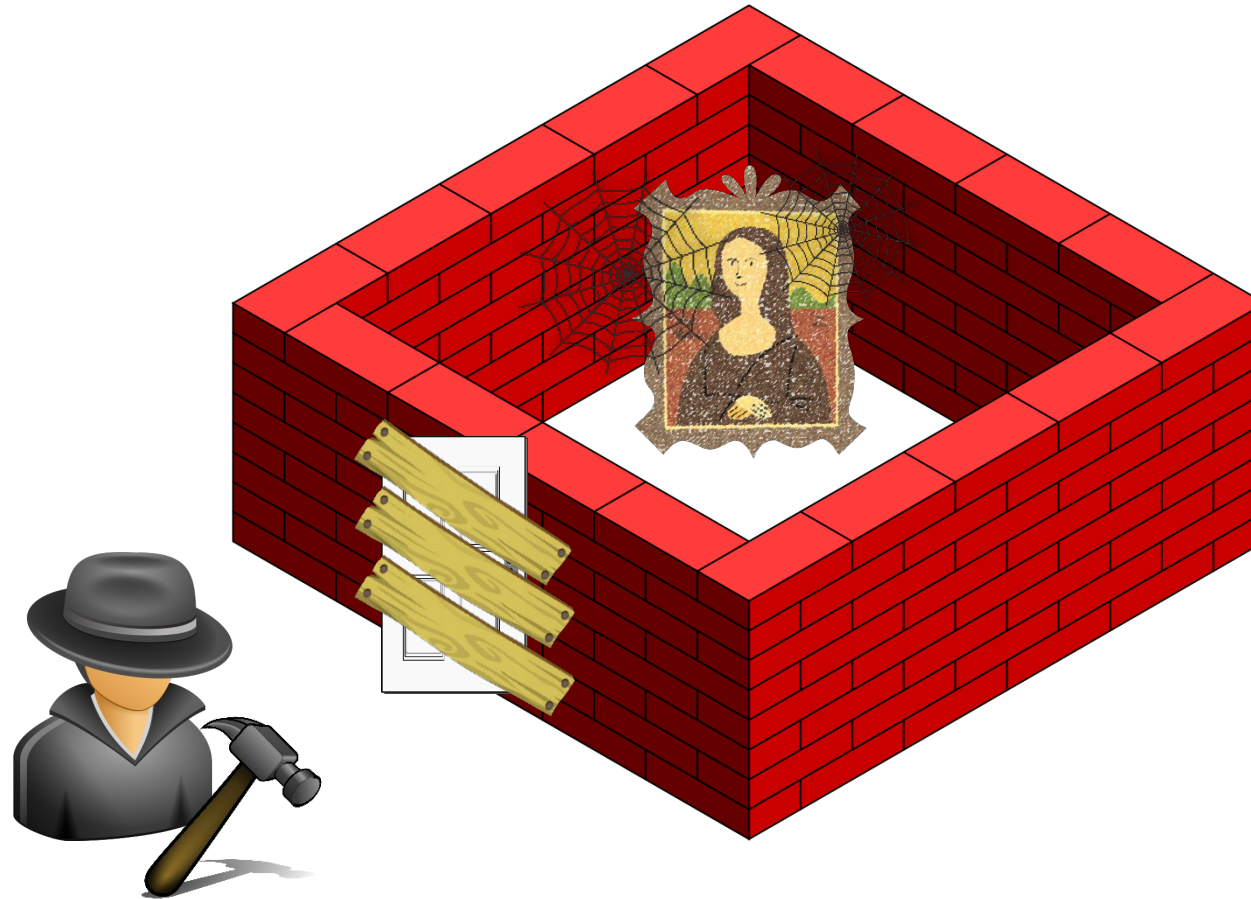
Security Properties - Confidentiality



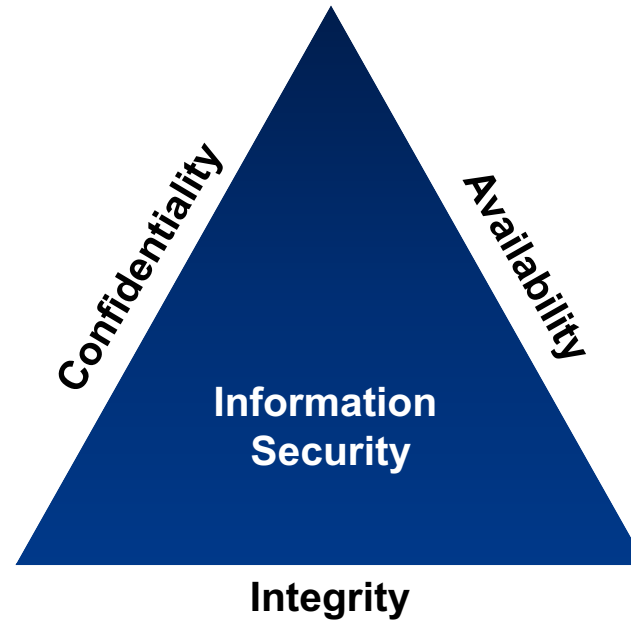
Security Properties - Integrity



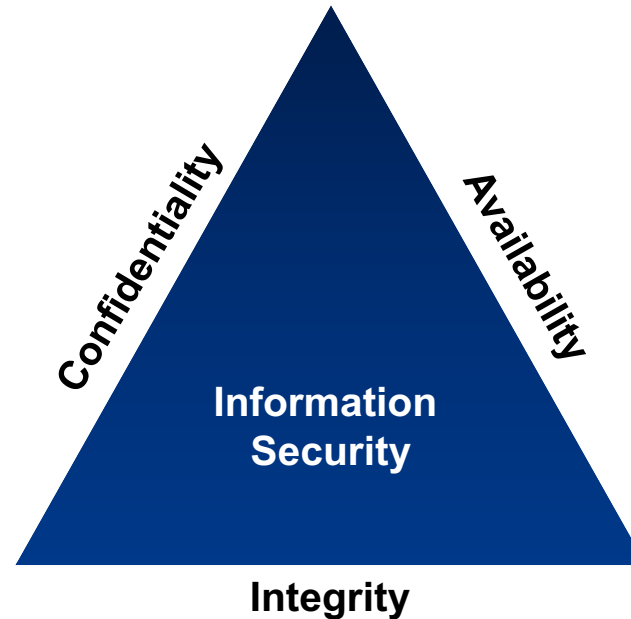
Security Properties - Availability



CIA Triad



Core Security Properties – CIA Triad



Now, let us put the CIA triad into context:

Assume you develop an online shop.

What are potential **CIA** properties for your online shop?



Core Security Properties – CIA Triad

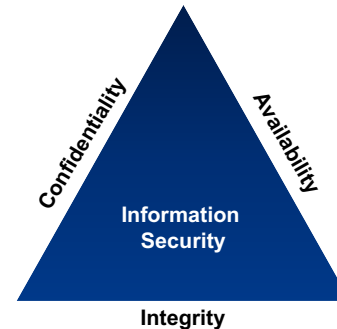
Confidentiality

The system must not disclose any information intended to be hidden.



your customers' credit card number

Note: open source software can still be confidential.



Integrity

The system must not allow assets to be subverted by unauthorized users.



changing prices or invoices

We must be able trust what is in the system

- *The data being stored*
- *The functionality being executed*

Availability

The system must be able to be available and operational to users.

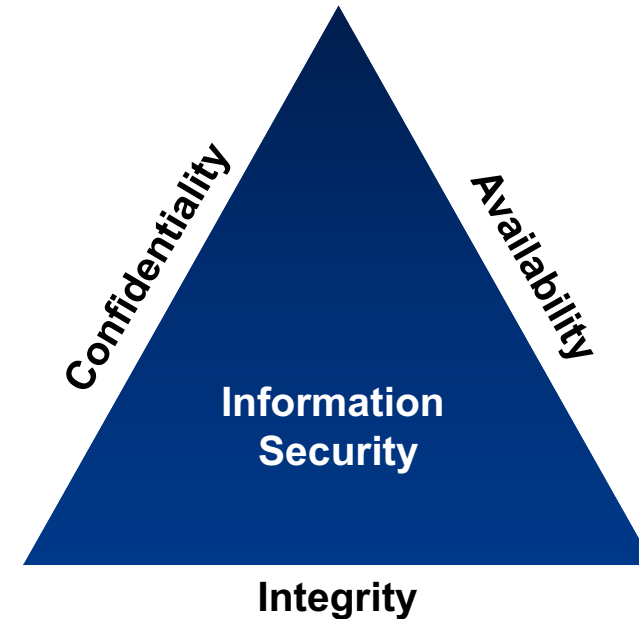


bringing down your online shop.

Any system performance degradation that can be triggered by a user can be used for denial of service attacks

Threats to Your Security Properties


- **S**poofing
- **T**ampering
- **R**epudiation
- **I**nformation Disclosure
- **D**enial of Service
- **E**levation of Privilege

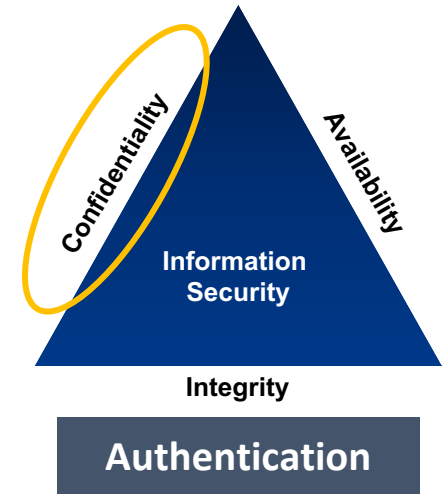


Threats to Your Security Properties - Spoofing

“Pretending to be something or someone other than yourself.”

[Shostack, A. (2014). Threat modeling : designing for security., Indianapolis, Ind. : Wiley.]


- IP Spoofing
 - Set source IP address to some other IP
- E-Mail Spoofing
 - Replace sender address
 - In SMTP, “From” is not checked
-  Phishing Mails

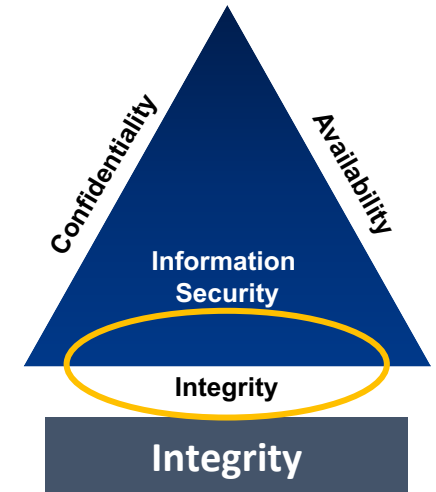


Threats to Your Security Properties - Tampering

“Modifying something on disk, on a network, or in memory.”

[Shostack, A. (2014). Threat modeling : designing for security., Indianapolis, Ind. : Wiley.]

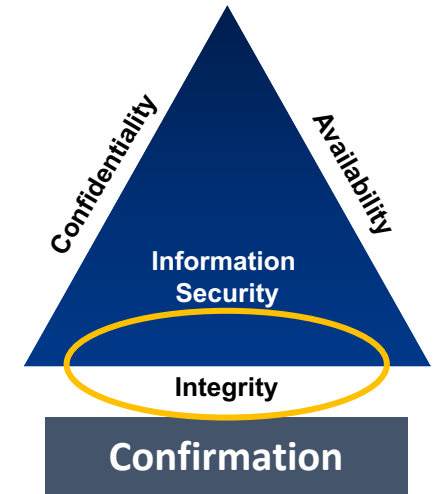
- “Web Tampering” [owasp.org]
 - See vulnerability of today
 -  Changing prices, offers, ..



Threats to Your Security Properties - Repudiation

“Claiming that you didn’t do something, or were not responsible. Repudiation can be honest or false, and the key question for system designers is, what evidence do you have?”

[Shostack, A. (2014). Threat modeling : designing for security., Indianapolis, Ind. : Wiley.]



- Deleting Logs, Database transactions

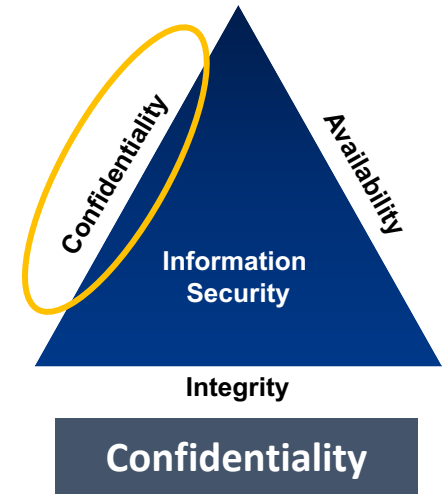
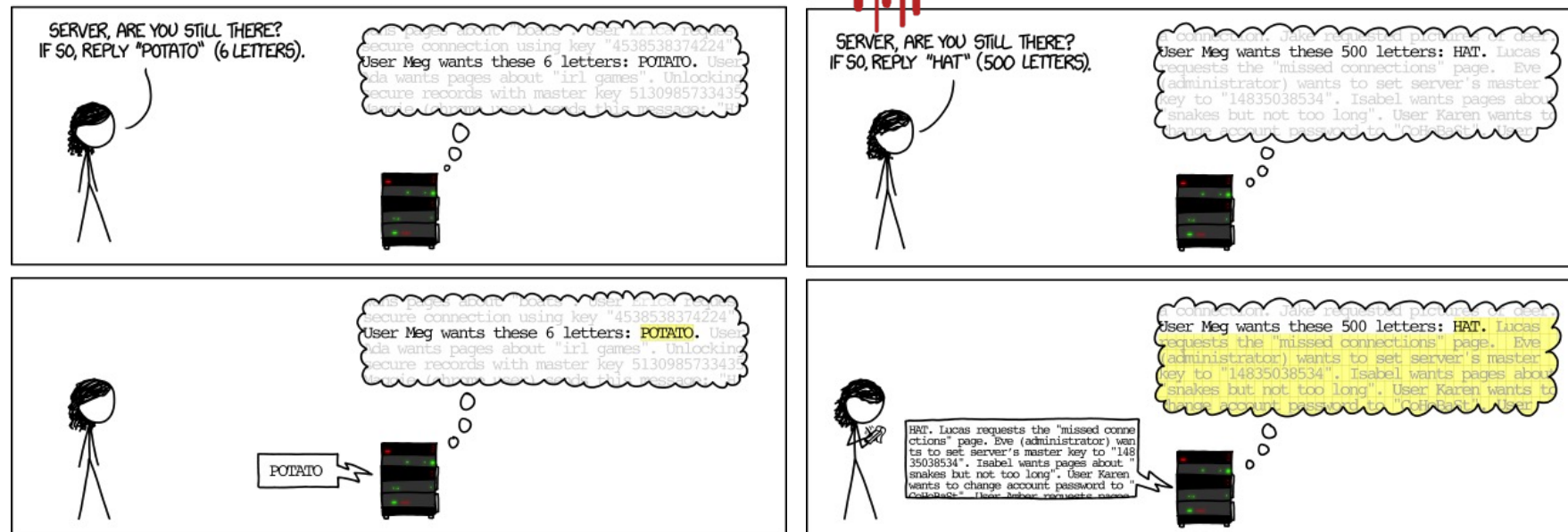
 „I did not order this product“

Threats to Your Security Properties – Information Disclosure

“Providing information to someone not authorized to see it.”

[Shostack, A. (2014). *Threat modeling : designing for security.*, Indianapolis, Ind. : Wiley.]

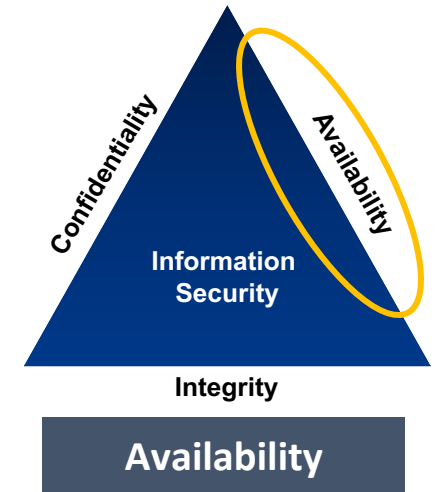
How the Heartbleed Bug works




Threats to Your Security Properties – Denial of Service

“Absorbing resources needed to provide service.”

[Shostack, A. (2014). Threat modeling : designing for security., Indianapolis, Ind. : Wiley.]




-  Performance degradation that can be triggered by an user, e.g., too many server requests (DDoS)
 - Blog of security blogger Brian Krebs was taken down in September 2016, after blogging about the illegal vDos provider
 - DNS services of Dyn were attacked in October 2016
 - Many prominent websites were not reachable
 - Botnet of thousands of IoT devices, e.g., IP-cameras

Threats to Your Security Properties – Elevation of Privilege

“Allowing someone to do something they’re not authorized to do.”

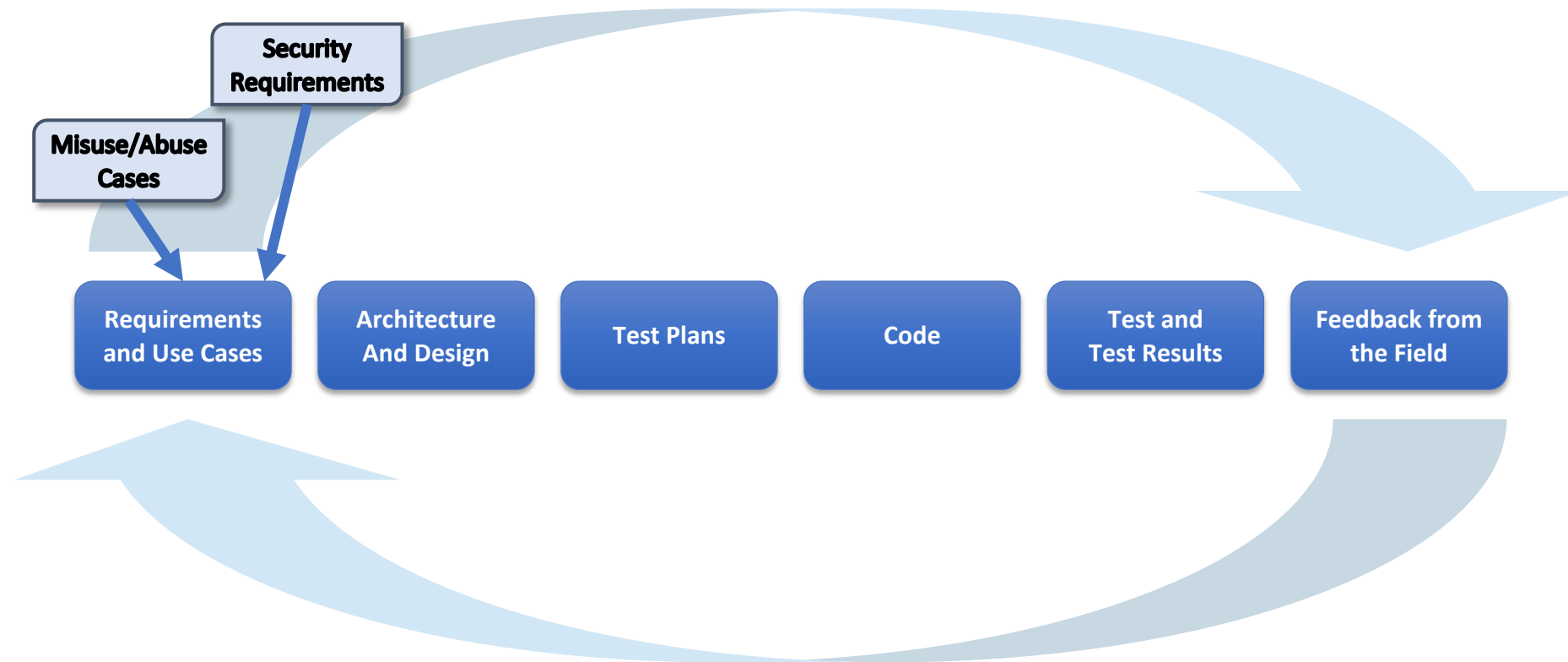
[Shostack, A. (2014). Threat modeling : designing for security., Indianapolis, Ind. : Wiley.]

- Library for image processing
 -  Used by many websites
e.g., profile, product photos
- “Image Tragick”
 - Upload a compromised SVG
 - Execute code with the privileges of the calling server process, e.g., deleting all images



Misuse and Abuse Cases

Security Touchpoints



Usual Viewpoint – Use Cases (as a Software Engineer)

- Software Development is about ...
 - making software do something
 - what the system should do
- Describe system in the form of...
 - Software requirements
 - Use Cases / Stories
- What a system will do when everything goes right
- Highly domain specific
- Describe how the surrounding environment has changed as a result of the system



Software Engineer

Use Case Contents

- Use cases include:
 - Actors
 - Preconditions
 - Main flow describes the primary scenario
 - Alternative scenarios describe how the system reacts to alternative cases



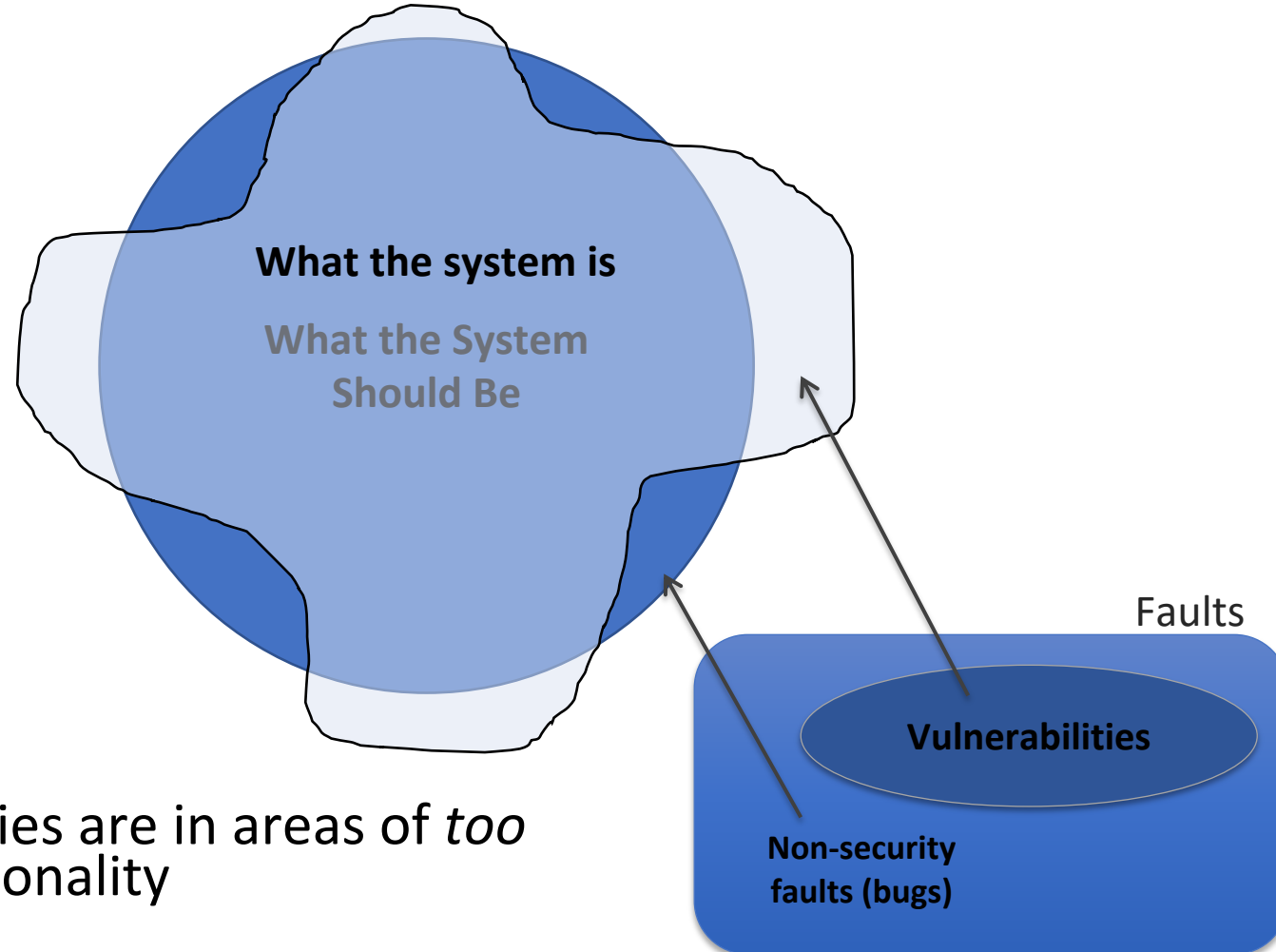
Example: Conclude a Contract

- Actor: Insurance Agent
- Precondition: Insurance agent is authenticated at management tool which is connected to the main customer database.
- Main Flow:
 1. Agent starts contract management tool
 2. Agent enters customer's name
 3. Selects and discusses insurance product with customer
 4. Fills out contract form in management tool
 5. Prints out contract document and consulting protocol
 6. Customer signs both documents
 7. Agent archives both documents
 8. Agent completes the contract in the management tool



Problem: Unintended Functionality & Abnormal Behavior

- Must be anticipated!



- Vulnerabilities are in areas of *too much* functionality

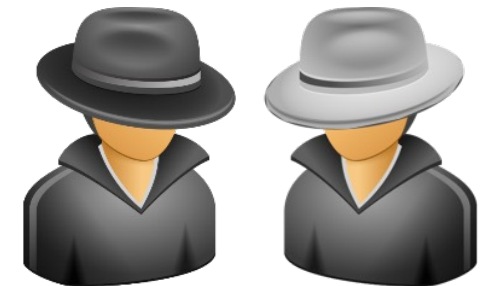
How to anticipate abnormal behavior

- Think like a Blackhat or Greyhat
- What the system should...
 - Do
 - Not do
 - What should not happen



Misuse and Abuse Cases

- A scenario within a use case in which an actor compromises the system
- Flow of events, but with malicious usage
- Define the harm done to the system



Misuse vs. Abuse



Misuse

- is unintentional
- still security-related (crime of opportunity)

Abuse

- is intentional
- imply the actor is actively seeking vulnerabilities



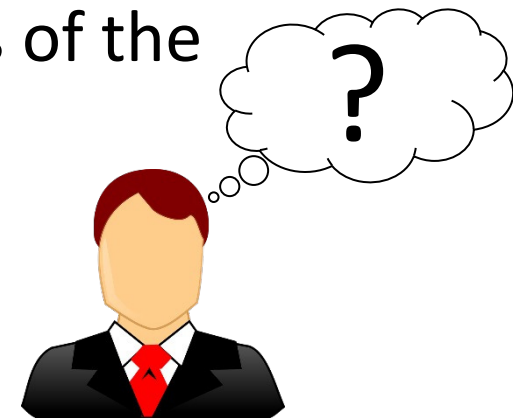
Example: Misusing „Conclude a Contract“



- Misuse case

1. Agent starts contract management tool
2. Agent enters customer's name
3. Agent misspells the customer's name
4. Agent is shown a set of personal information of "another" customer that is not associated to the agent

Harm done: Personal information and data protection rights of the "other" customer has been violated



Example: Abusing „Conclude a Contract“

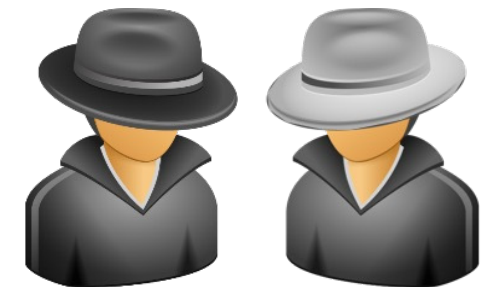


- Abuse case

Attacker: Someone who spoofed the insurance agent's credentials

- Repeat Main Flow steps 1-2 multiple times
 1. Agent starts contract management tool
 2. Agent enters customer's name
- To gather a lot of personal data from different customers
- Run script to scrap personal data with 10,000 requests per second...

Harm done: Personal data of a large set of customers is stolen

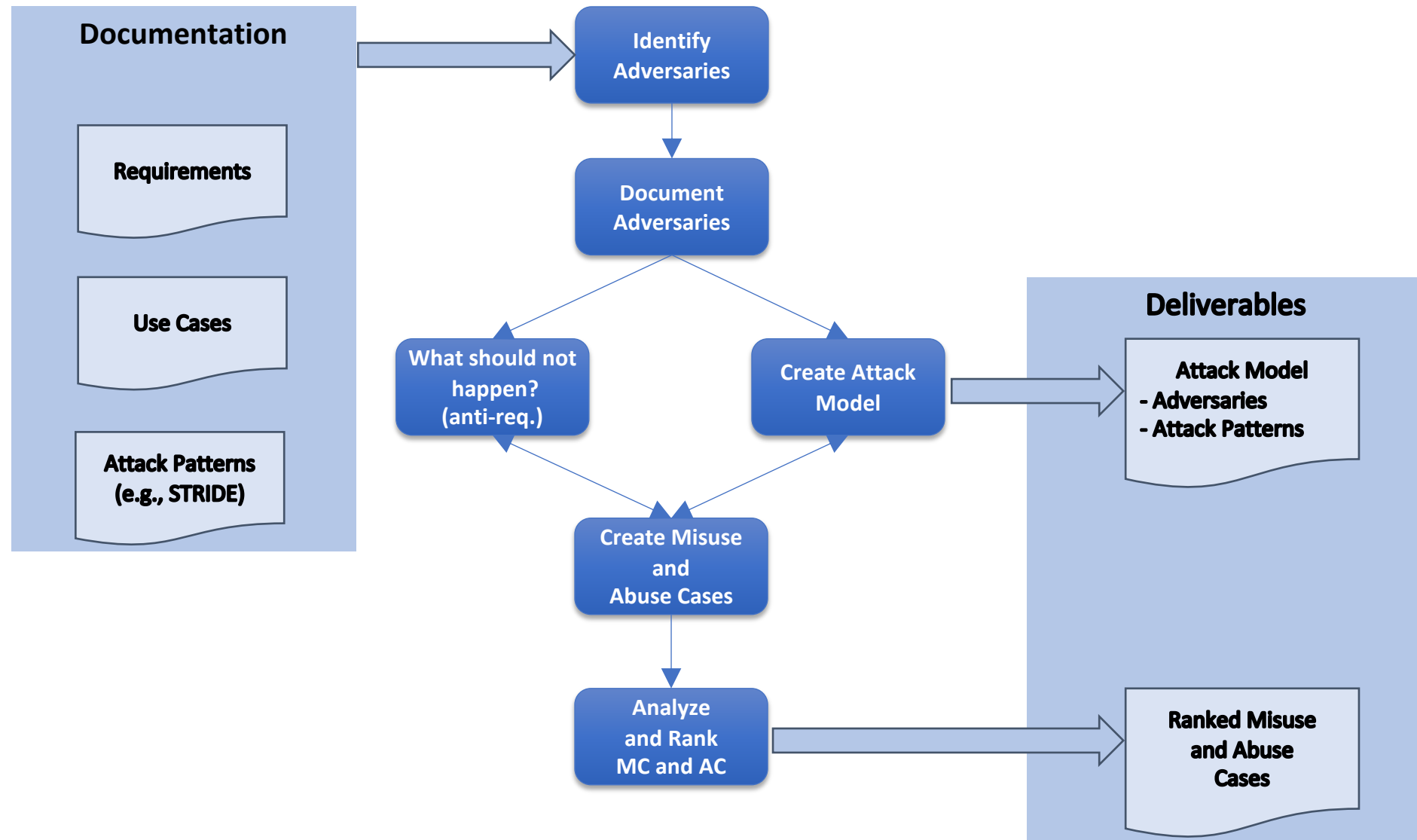


Be ready to dispute...

- System architects, project managers, and product owners may argue
 - *“But no one would do these things”*
 - *“This assumption is off-limits and unrealistic”*
- Only correct if we limit worldview to legitimate users
- Purpose of Abuse & Misuse Cases
 - Think out-of-the-box
 - Question assumptions, e.g., gravity → not if you work for NASA
 - Question privileges users have, e.g., can a secretary see all details of all calendars
 - Question use cases, e.g., to schedule a meeting do you need to see blocked times in a calendar of full-details



Misuse/Abuse Cases - Process



Lecture - Schedule



| # | Datum | Topic |
|----|--------|---|
| 1 | 16.10. | Introduction |
| 2 | 23.10. | Requirements, Misuse & Abuse Cases |
| 3 | 30.10. | Risk Analysis & Distrustful Decomposition |
| 4 | 06.11. | Risk Management & Test Planning |
| 5 | 13.11. | Defensive Coding / Pitfalls |
| 6 | 20.11. | Correct Usage of Security Mechanisms |
| 7 | 27.11. | Code Reviews / Vulnerability Assessment |
| 8 | 11.12. | Deployment |
| 9 | 18.12. | Insider Threats |
| 10 | 08.01. | Developer-Centered Research |
| 11 | 15.01. | Guest Lecture |
| 12 | 22.01. | Usability + Recap |



Lightning
Surveys 