Behavioural Security Group

UNIVERSITÄT BONN

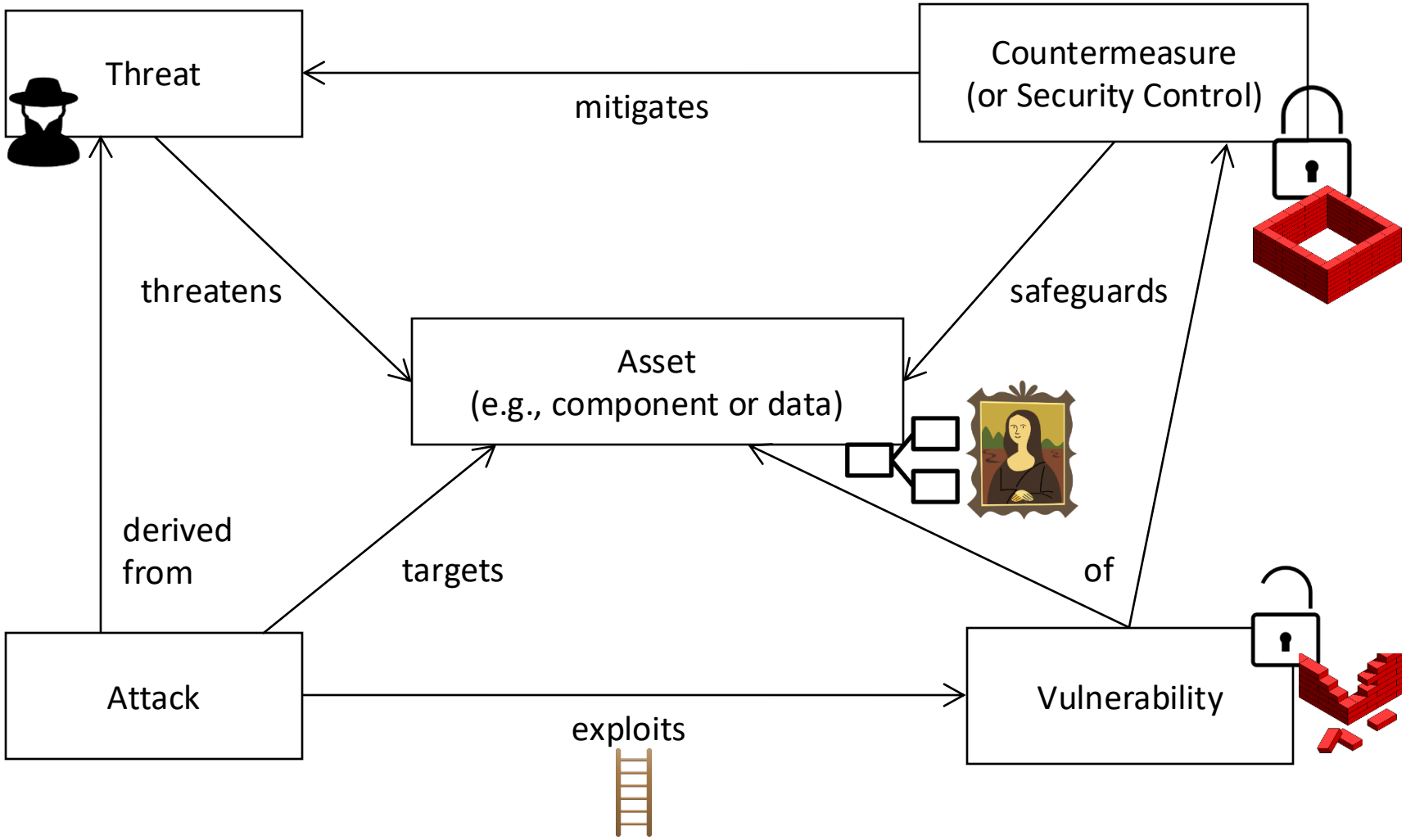
Secure Software Engineering

Winterterm 2025/26

Development Lifecycle & Security Requirements

Dr. Christian Tiefenau

Terms & Relations (1)



Security Properties (Security Objectives)

Behavioural Security Group

UNIVERSITÄT BONN

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

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

Core Security Properties – CIA Triad

- Last week: 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

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

Outline today

UNIVERSITÄT BONN

- Threats to Security Properties & How to Defend
  - STRIDE
  - Defense in depth, Least privilege, Fail securely, No security by obscurity
  - Detect and record, (Don’t) trust X, Keep it simple
- Vulnerability of the day
  - SQL-Injection
- Development Lifecycle & Security Requirements
  - Misuse and Abuse Cases

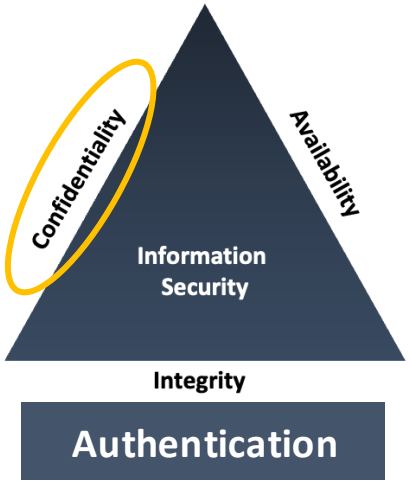
Threats to Your Security Properties

- **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.]

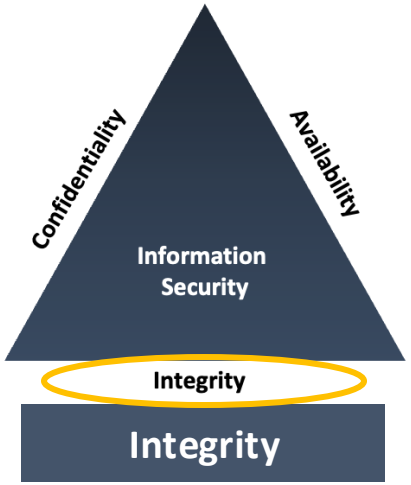


- IP Spoofing
  - Set source IP address to some other IP
- E-Mail Spoofing
  - Replace sender address
  - In SMTP, "From" is not checked
- Phishing Mails

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.]



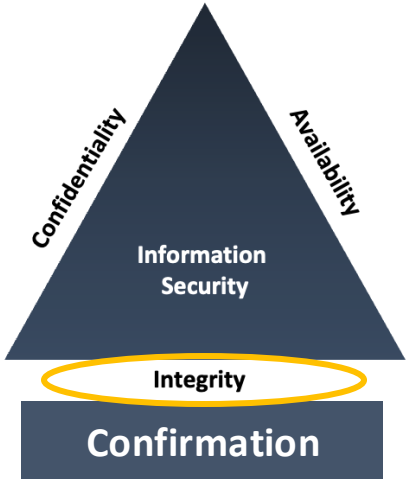
- "Web Tampering" [owasp.org]
  - See vulnerability of today

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.]



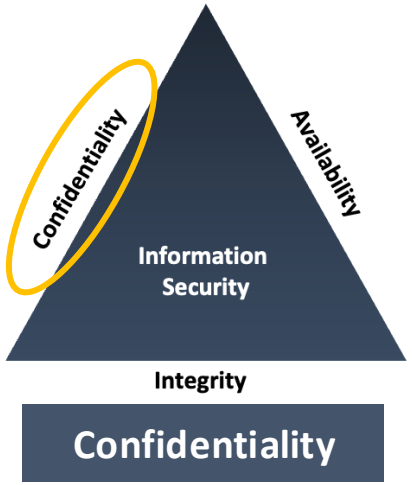
- Deleting Logs, Database transactions

„I did not order this product"

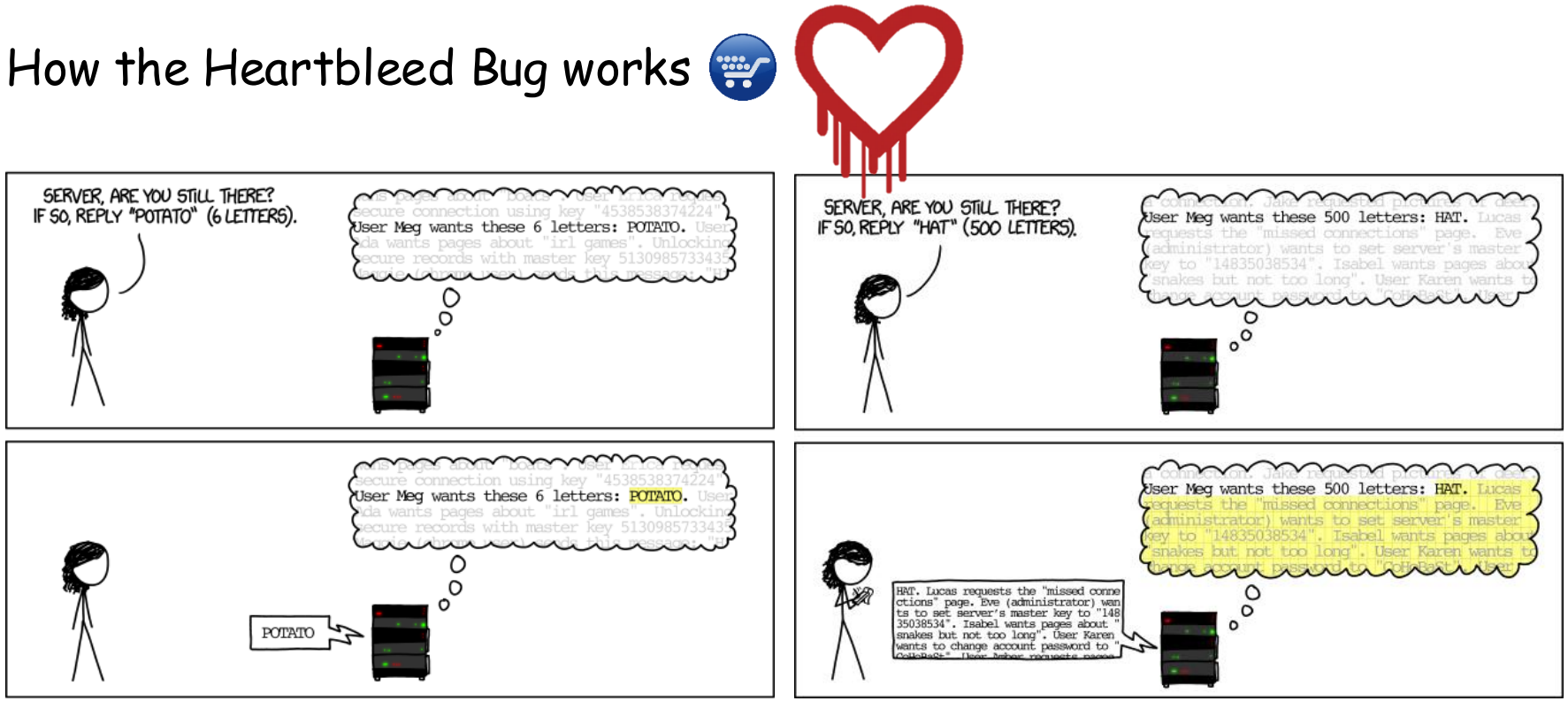
Threats to Your Security Properties – Information Disclosure

"Providing information to someone not authorized to see it."

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



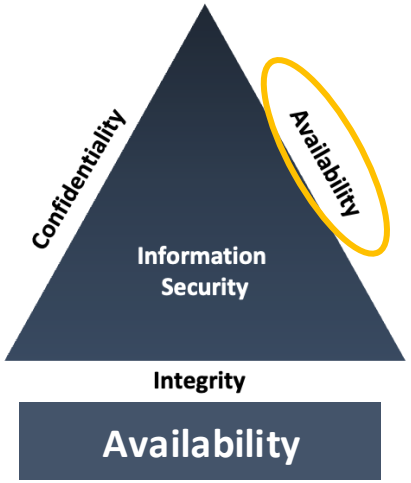
How the Heartbleed Bug works



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

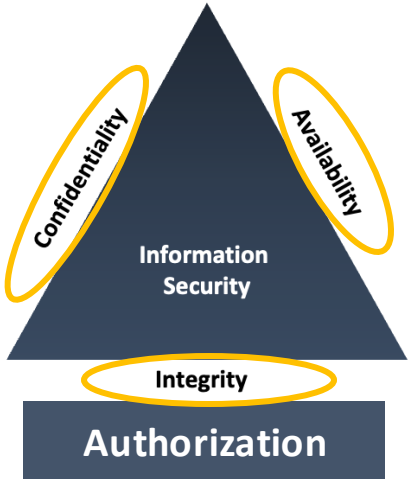
Threats to Your Security Properties – Elevation of Priviledge

Behavioural Security Group

UNIVERSITÄT BONN

"Allowing someone to do something they're not authorized to do."

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

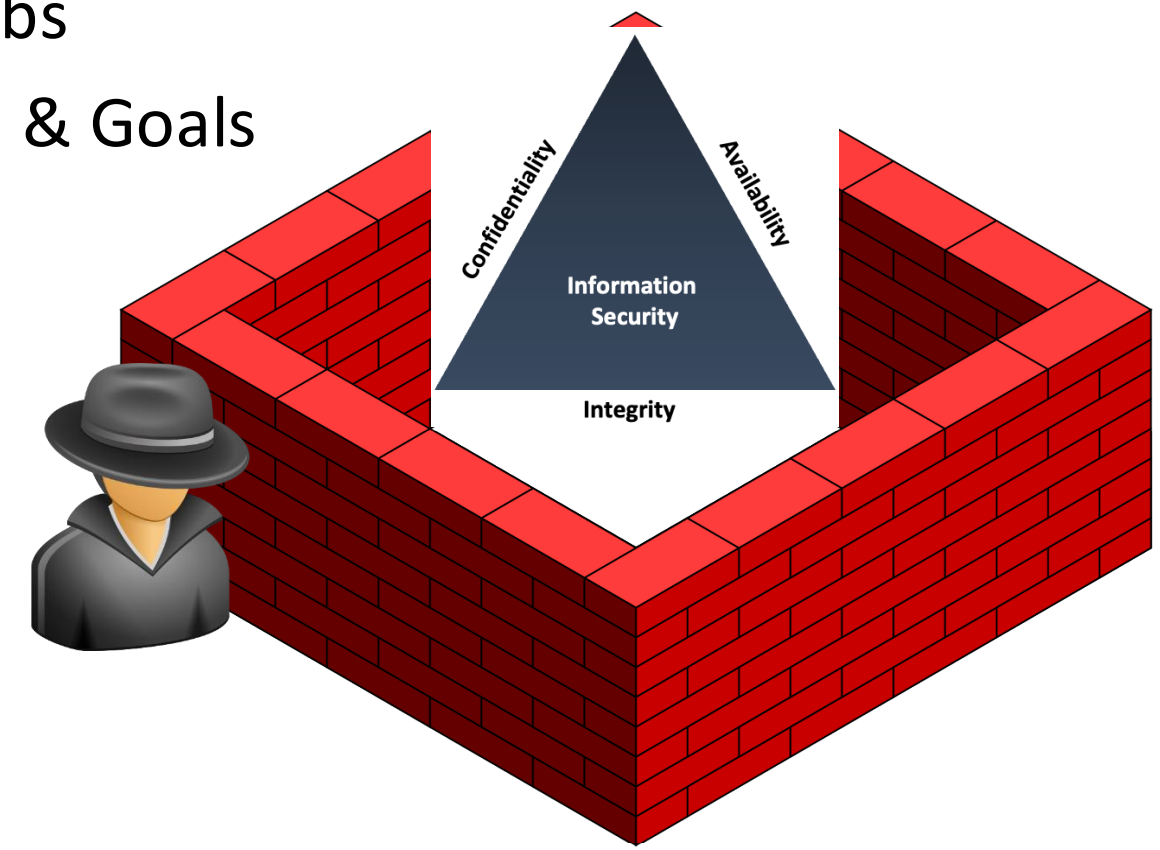


- Library for image processing
  - Used by many websites
    e.g., profile, product photos

- "Image Tragick"
  - Upload a compromised SVG
  - Execute code with the privileges of the calling server process, e.g., deleting all images

Defending your Security Properties

- How to defend against those threats?
- General philosophies and proverbs
- Must be adapted to your System & Goals



Misc. Philosophies and Proverbs

Behavioural Security Group

UNIVERSITÄT BONN

- Defense in depth
  - If they break into this, they can't get any farther
  - Think Middle-Age castles
  - Original meaning of "firewall", not today's firewall

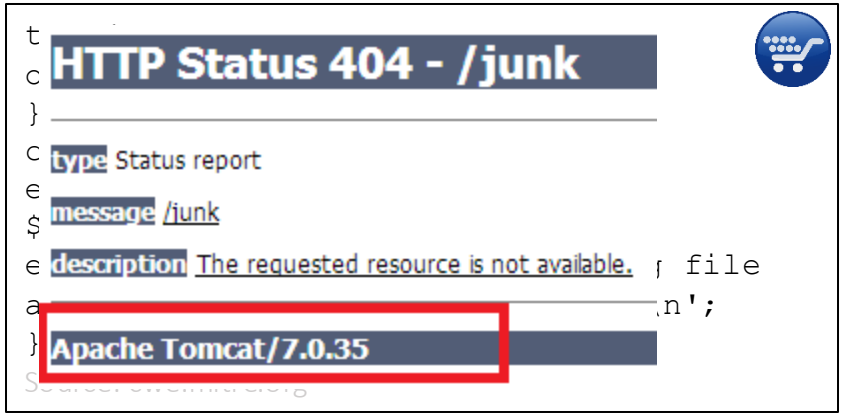
- Least privilege
  - Every user or module is given the least amount of privilege it needs
  - Evil: `sudo chmod -R a+rw /` `# give all users rw permissions to everything`
  - Bad: Use root user for database access

```
<?php $host = 'localhost'; $userID = 'root'; $password = 'password';
$db = mysql_connect($host, $userID, $password) or die ('Error connecting to mysql');
$name = 'testdatabase'; mysql_select_db($name);
$sql="SELECT * FROM theTable"; $result=mysql_query($sql);
?>
```

Source: OWASP.org

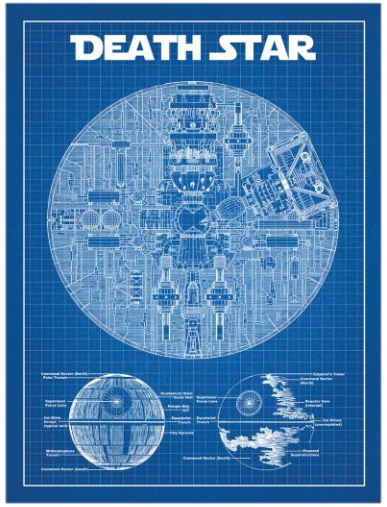
More Misc. Philosophies and Proverbs

- Fail securely
  - Exceptions put the system into weird states
  - Prevent error message information leak
  - Take care of those exceptions properly!



- **No** Security by obscurity
  - You can't rely upon being obscure to be secure
    - Crowds are good at guessing
    - Insiders are corruptible

"*If the Rebels have obtained a complete technical readout of this station it is possible, however unlikely, that they might find a weakness, and exploit it.*"
"*The plans you refer to will soon be back in our hands.*"
— General Cassio Tagge and Darth Vader (Star Wars, 1977)



Even More Misc. Philosophies and Proverbs

- Detect and record
  - Even if you can't always sift through that data ahead of time
  - Post-mortem analysis (Failure reconstruction, Root cause identification)

- Don't trust [input | environment | dependencies | *]
  - Know what to trust
  - Know whom to trust
  - Know how to trust

- Keep it simple
  - You ain't gonna need it (YAGNI)
  - Speculative generality can be risky
  - Minimize the attack surface

Discussion Exercise: Spam Bot Server

UNIVERSITÄT BONN

- Suppose we had a vulnerability in the university mail servers where you could send a special packet and it would bypass authentication for outgoing email. This allowed attackers to send emails using any account.

- Which of CIA does this violate?
  - Immediately?
  - As a secondary consequence?

- Using the following philosophies, discuss how each of these can be applied here:
  - Defense in Depth
  - Security by obscurity
  - Detect and record
  - Don’t trust input

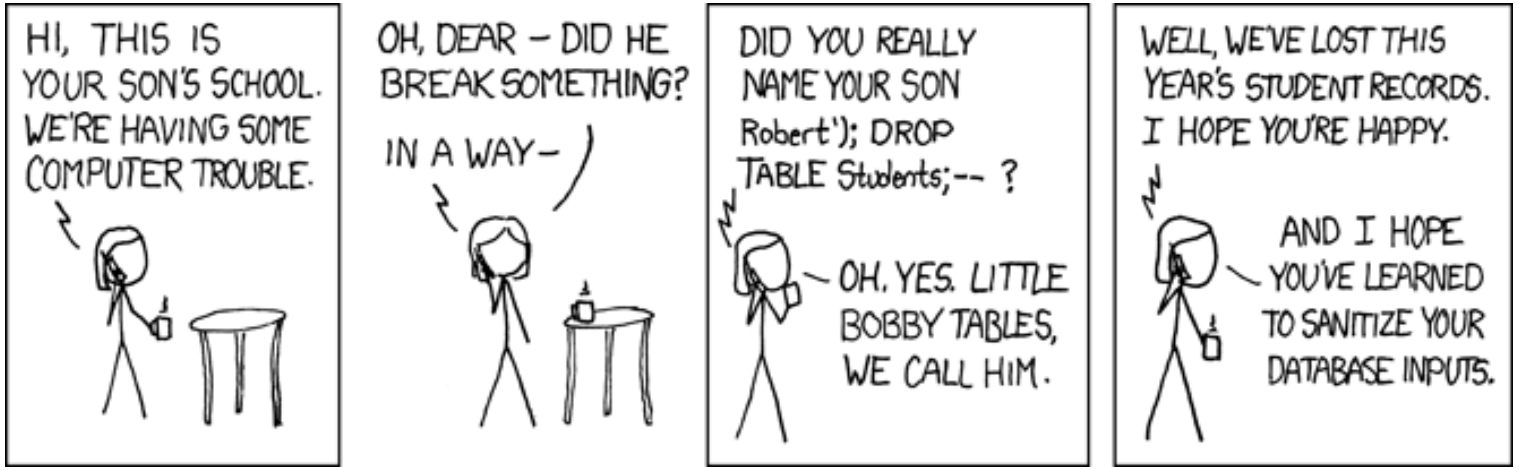
Behavioural Security Group

UNIVERSITÄT BONN

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

UNIVERSITÄT BONN

```
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

UNIVERSITÄT BONN

```
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

Waiting for patches: Vulnerabilities in Nagios XI endanger networks

Nagios XI, the monitoring software for complex IT infrastructures, is vulnerable. There is no fix yet.

3

(Image: Gorodenkoff/Shutterstock.com)

20.04.2020, 11:52 a.m. Reading time: 1 min. | Security

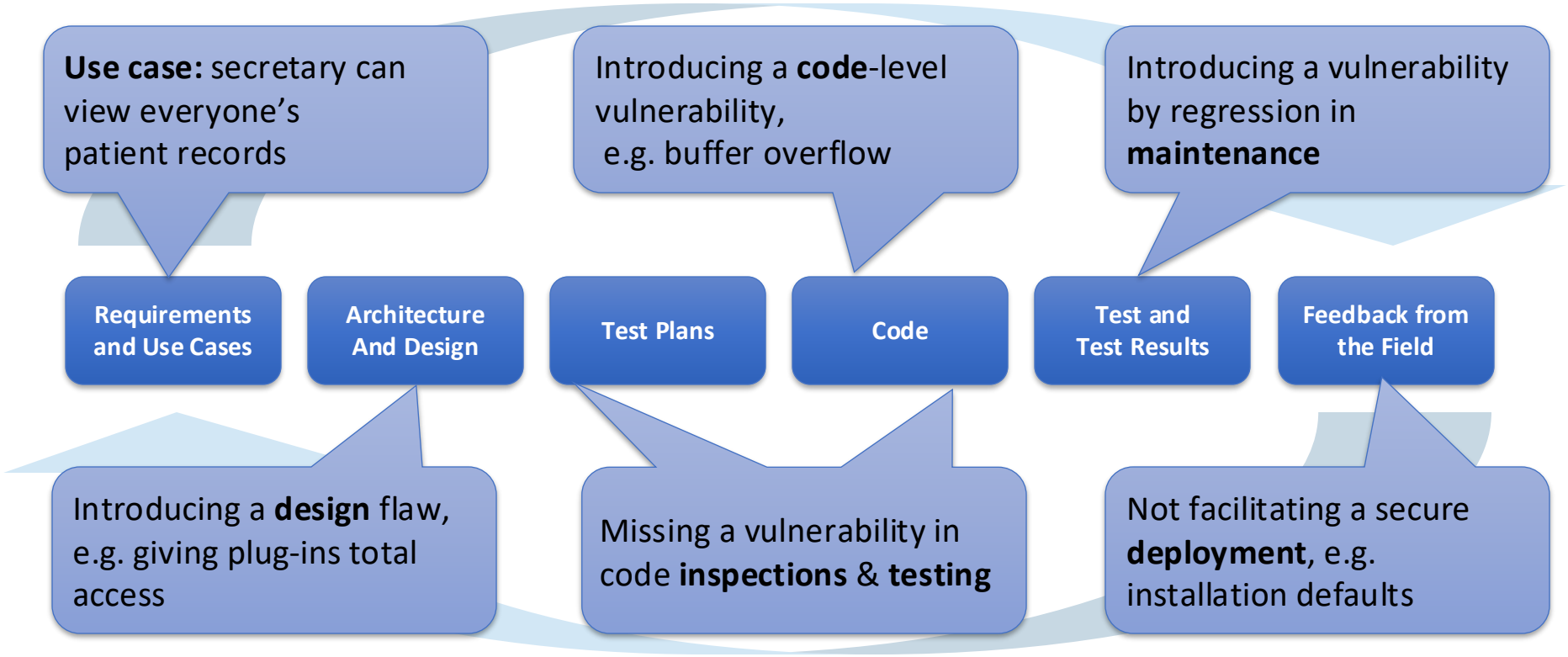
From Dennis Schirrmacher

https://www.heise.de/security/meldung/Warten-auf-Patches-Schwachstellen-in-Nagios-XI-gefaehrden-Netzwerke-4704444.html

Development Lifecycle & Principles

Misuse and Abuse Cases – „Opportunity Makes The Thief“

(Secure) Software Development Lifecycle

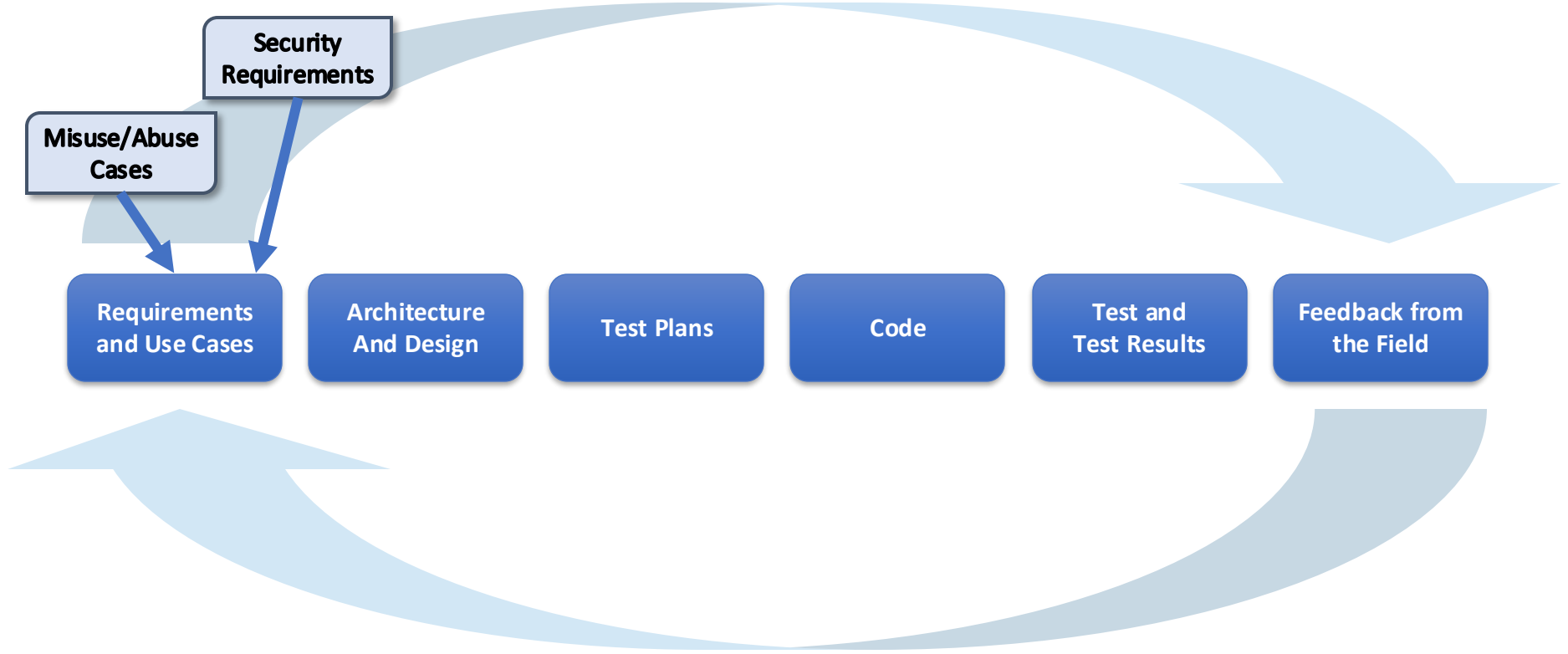


Reminder: Security is Not a Set of Features

UNIVERSITÄT BONN

- Secure is an emergent property of software
  - Simply adding one feature (e.g., encryption) to gain security is futile
  - Being secure is the result of many, many factors, not one feature
  - At which point in the development can we apply security?

Security Touchpoints



Usual Viewpoint – Use Cases (as a Software Engineer)

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

Use Case Contents

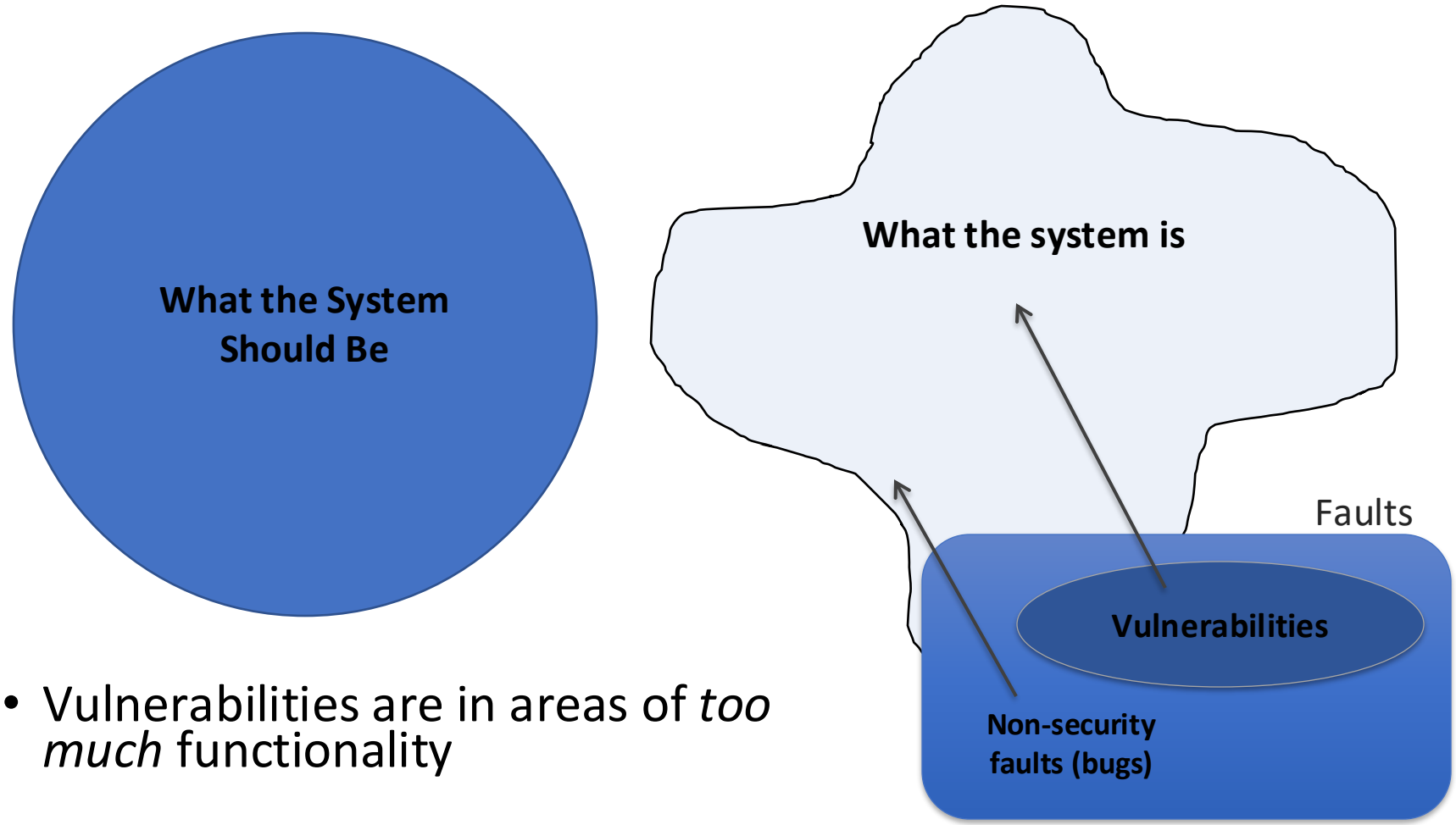
- Use cases include:
  - Actors
  - Preconditions
  - Main flow describes the primary scenario
  - Alternative scenarios describe how the system reacts to alternative cases

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

- They must be anticipated!

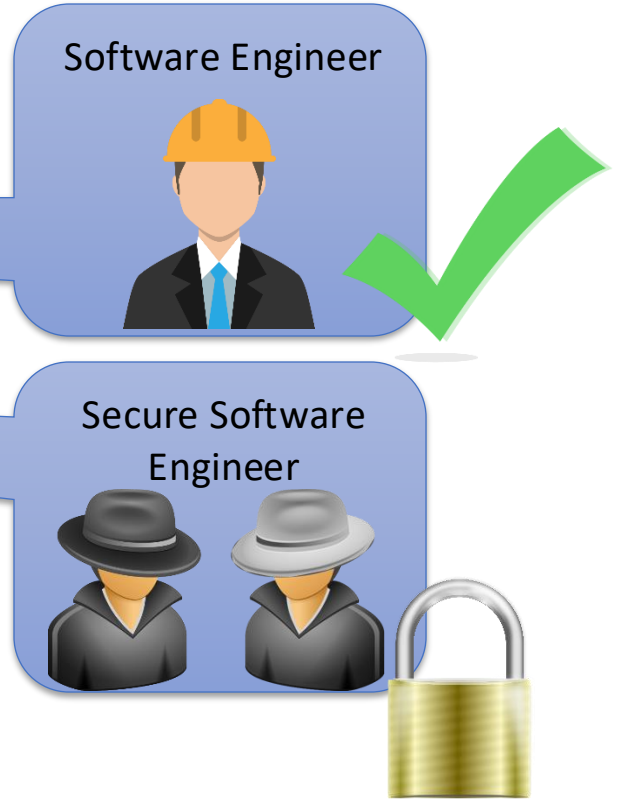


- Vulnerabilities are in areas of *too much* functionality

How to anticipate abnormal behavior

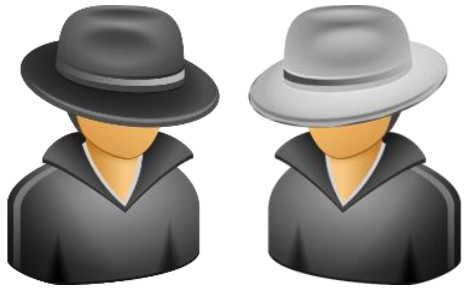
- Think like a Blackhat or Greyhat hacker!

- Who describes what the system should…
  - Do
  - Not do
  - What should not happen



Anticipate abnormal behavior with 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



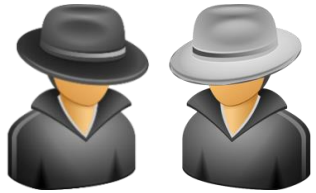
Misuse vs. Abuse

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

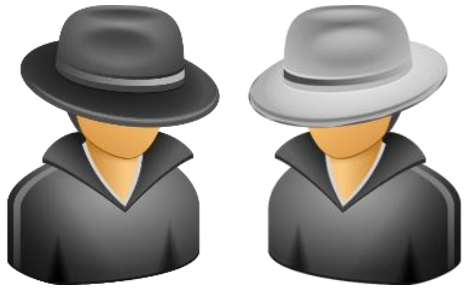
Example: Abusing „Conclude a Contract“

- 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

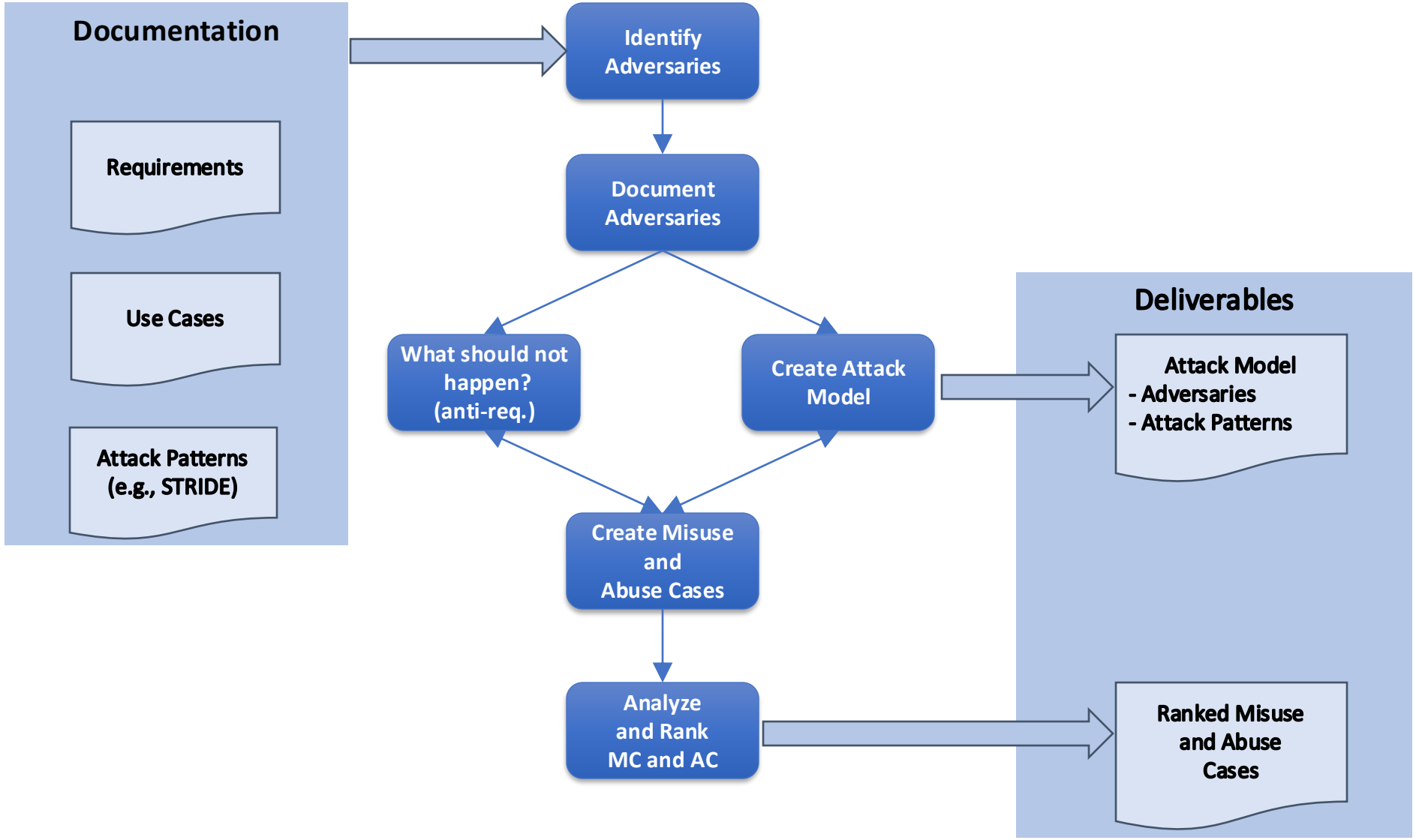


Be ready to dispute…

- 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

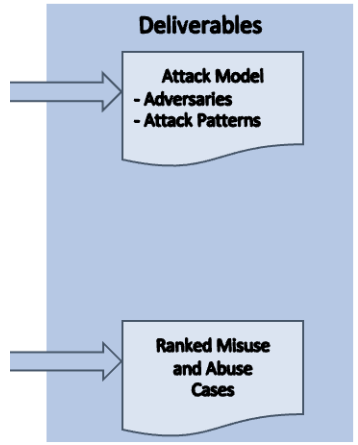
- Purpose of Abuse & Misuse Cases
  - Think out-of-the-box
  - Question assumptions, e.g., gravity not if you work for NASA
  - Question privileges users have, e.g., can a secretary see all details of all calendars
  - Question use cases, e.g., to schedule a meeting do you need to see blocked times in a calendar of full-details

Misuse/Abuse Cases - Process



Security Requirements

- Generalized forms of misuse and abuse cases



- Use cases trace to security requirements
  - Document these in the main flow

- E.g., from “Conduct a Contract”
  - Customers need to sign multiple documents or document lists all products clearly
  - Agents only have access to the personal data of their customers
  - All actors are limited to 10 server requests per minute

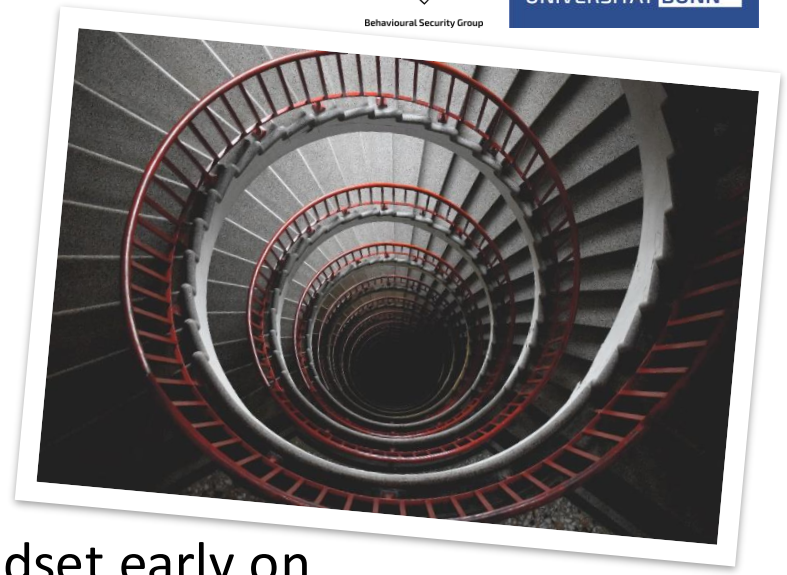
Initial Brainstorming or how to start

- Bring out your inner villain

- Best done with years of experience
  - Involve peers
  - Involve network security people

- At the beginning not perfect
  - … but that's fine!

Isn‘t this infinite?



- Yes
  - The only absolutely secure system is no system

- But even one good abuse case goes far
  - Easier to think beyond one scenario
  - Starts a discussion
  - Gets stakeholders and developers into a balanced mindset early on
  - Motivates good design decisions

- It is important to create abuse cases systematically
  - Structured Brainstorming
  - Security experts needed
  - Cooperation with domain experts
  - Reuse of existing knowledge

Further Tips – How to start

- **Keys**
  - Domain, domain, domain.
    Don't focus on coding and vulnerabilities
  - Malicious actors are creative
  - Question the assumptions of the system
  - Focus on what the actor *can* do over *will* do (prioritize later)

- **Actor Inspiration**
  - Think of the best engineers on your team
    - Fire them and humiliate them publicly
    - Now challenge them to break your system
    - What would they go after?
    - What knowledge could they leverage the most?

Further Tips – How to start

- **Assumptions Inspiration**
  - What are the other non-malicious users expecting in this domain?
  - What are the ramifications of violating access restrictions?

  - Where could an attacker “sit in the middle”
    - Sniff the network?
    - Load a plug-in?

Further Tips – How to start

- **Coding Guidelines & Security Standards**
  - Check your company's guidelines
  - What are the "default" Security Requirements you have to consider
  - What is the "baseline"
  - Use these standards as a starting point

Next Lecture