# Datensicherheit

## Vorlesung 3: 08.12.2025

### *Wintersemester 2025/2026 h_da*

***Heiko Weber, Lehrbeauftragter***

# Teil 2: Datensicherheit
# Themenübersicht der Vorlesung

1. Einführung / Grundlagen / Authentifizierung & Autorisierung
2. Kryptografie / Verschlüsselung & Signaturen / Vertrauen / Blockchain
3. Softwaresicherheit / Schadsoftware
4. Netzwerksicherheit / TLS / PGP & S/MIME / Firewalls & Netzwerksegmentierung
5. Hacking / Phishing / Einführung in den Datenschutz / Anonymität / Darknet
6. Datenschutzgesetze / Technische & Organisatorische Maßnahmen
7. Organisationssicherheit / Managementsysteme / Zusammenfassung

# Sicherheitsfunktionen vs. Nichtfunktionale Sicherheit

- **Sicherheitsfunktionen**
  - Funktionen, die direkt gewisse Schutzziele umsetzen
  - Beispiele: Authentisierung, Autorisierung, Verschlüsselung, Signaturen
- **Nichtfunktionale Sicherheit**
  - Eigenschaften, die indirekt gewisse Schutzziele umsetzen
  - letztendlich sind damit alle möglichen Aspekte einer Software oder eines sonstigen IT-Systems gemeint, die dafür sorgen, dass das Verhalten der Spezifikation entspricht und Fehler nicht zur Verletzung von Schutzzielen führen
- es gibt Bereiche, in denen nicht ganz klar zwischen Sicherheitsfunktionen und nichtfunktionaler Sicherheit unterschieden werden kann

# Terminologie

- **Schwachstelle** (Weakness)
  ein Software-Fehlertyp, der in gewissen Situationen zu einer Verwundbarkeit der Software führen kann

- **Verwundbarkeit** (Vulnerability)
  das Auftreten einer oder mehrerer Schwachstellen in einer Software, in der diese Schwachstelle genutzt werden kann, um ein Fehlverhalten hervorzurufen

- **Offenlegung** (Exposure)
  das Auftreten einer oder mehrerer Schwachstellen in einer Software, die Informationen oder Funktionen offenlegen, die einen Angriff auf ein System erleichtern

- **Auswirkung** (Impact)
  das Ergebnis, welches eine erfolgreich ausgenutzte Verwundbarkeit in einer Software haben kann

# Auswirkungen von Schwachstellen

(https://cwe.mitre.org/cwraf/enum_of_ti.html)

- **Daten verändern** (Modify data)
- **Daten lesen** (Read data)
- **DoS: unzuverlässiges Ausführen** (DoS: unreliable execution)
- **DoS: Ressourcenverbrauch** (DoS: resource consumption)
- **unerlaubtes Ausführen von Programmen oder Befehlen** (Execute unauthorized code or commands)
- **Rechte erreichen / Identität annehmen** (Gain privileges / assume identity)
- **Schutzmechanismus umgehen** (Bypass protection mechanism)
- **Aktivitäten verbergen** (Hide activities)

DoS = Denial of Service

**Auswirkungsebenen:**

- System – Die Entität hat Zugang zu oder Kontrolle über ein System oder einen Computer.
- Anwendung – Die Entität hat Zugang zu einer betroffenen Anwendung.
- Netzwerk – Die Entität hat Zugang zu oder von einem Netzwerk aus.
- Unternehmen – Die Entität hat Zugang zu einem kritischen Teil der Unternehmensinfrastuktur, z. B. dem Netzwerk-Router, DNS-Server, etc.

# Sicherheits-Standards und -Aufzählungen







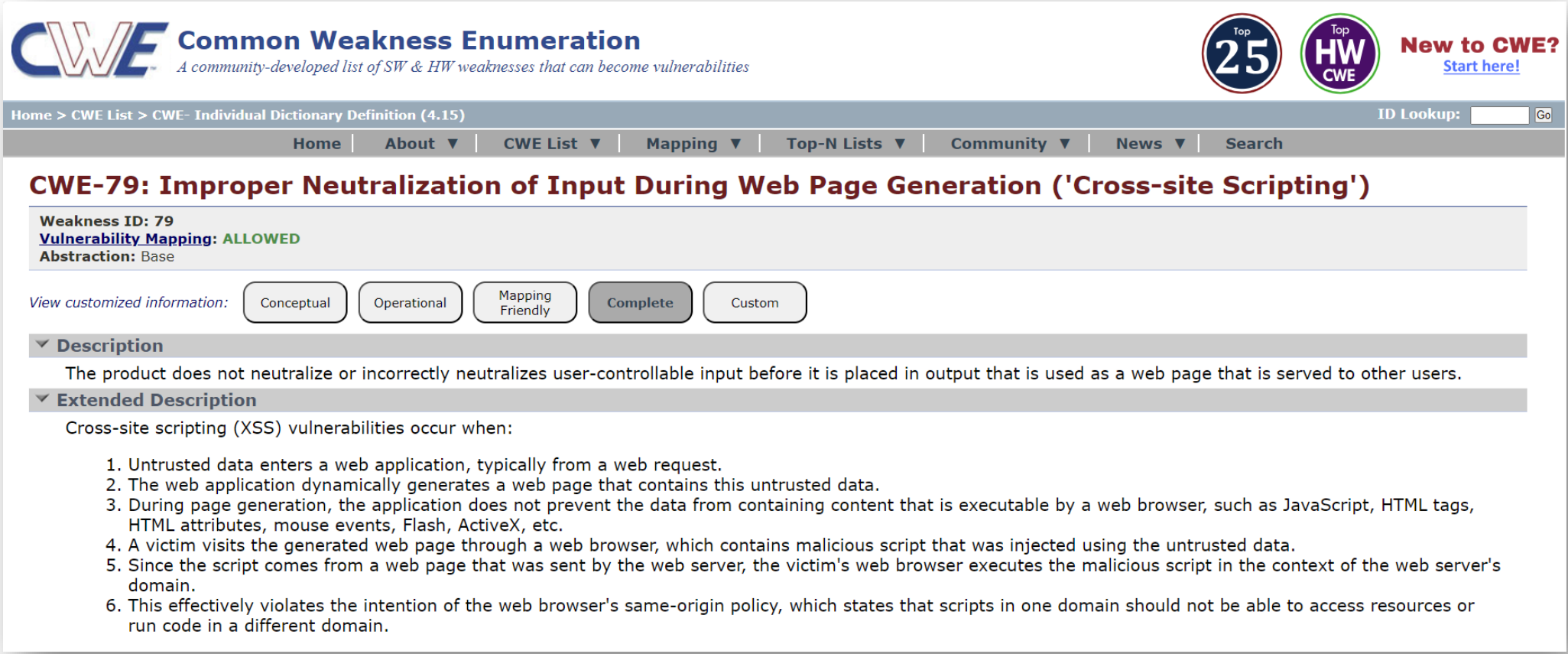




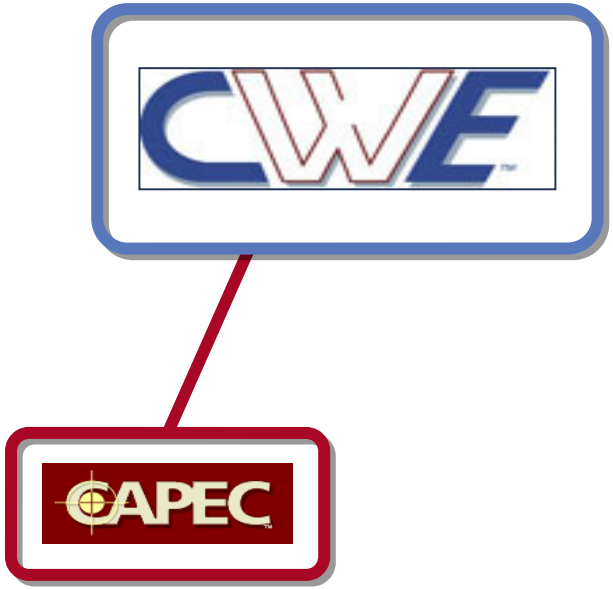
CWE

# Common Weakness Enumeration

- https://cwe.mitre.org/
- eine gemeinschaftlich entwickelte Aufzählung von Schwachstellentypen
- eine messbare Menge von Softwareschwachstellen, die es ermöglicht, effektiv über diese zu diskutieren und Werkzeuge zu erstellen, die diese erkennen und beheben können
- aktuell ca. 940 Schwachstellentypen in ca. 30 Kategorien aufgeführt
- CWE-ID = eindeutige Nummer, die einen Schwachstellentyp identifiziert

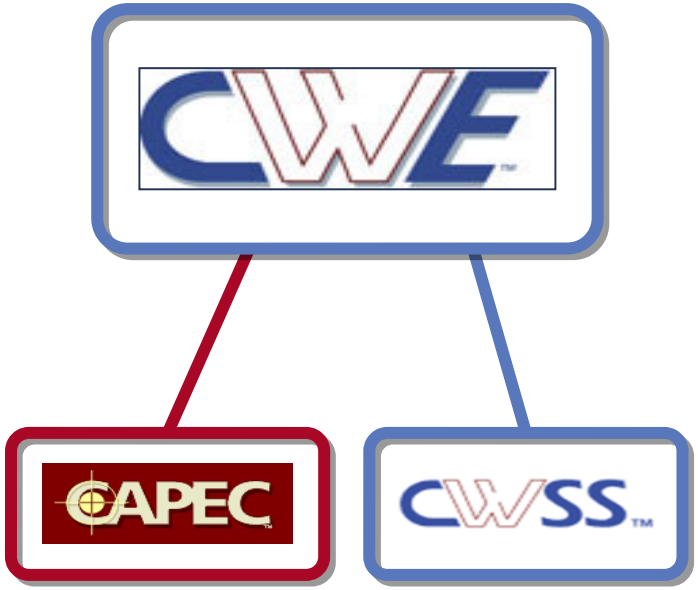


Common Weakness Enumeration
A community-developed list of SW & HW weaknesses that can become vulnerabilities
Top 25
Top HW CWE
New to CWE?
Start here!
Home > CWE List > CWE- Individual Dictionary Definition (4.15)
ID Lookup: Go
Home | About | CWE List | Mapping | Top-N Lists | Community | News | Search
CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
Weakness ID: 79
Vulnerability Mapping: ALLOWED
Abstraction: Base
View customized information: Conceptual Operational Mapping Friendly Complete Custom
Description
The product does not neutralize or incorrectly neutralizes user-controllable input before it is placed in output that is used as a web page that is served to other users.
Extended Description
Cross-site scripting (XSS) vulnerabilities occur when:
1. Untrusted data enters a web application, typically from a web request.
2. The web application dynamically generates a web page that contains this untrusted data.
3. During page generation, the application does not prevent the data from containing content that is executable by a web browser, such as JavaScript, HTML tags, HTML attributes, mouse events, Flash, ActiveX, etc.
4. A victim visits the generated web page through a web browser, which contains malicious script that was injected using the untrusted data.
5. Since the script comes from a web page that was sent by the web server, the victim's web browser executes the malicious script in the context of the web server's domain.
6. This effectively violates the intention of the web browser's same-origin policy, which states that scripts in one domain should not be able to access resources or run code in a different domain.


CWE
CAPEC

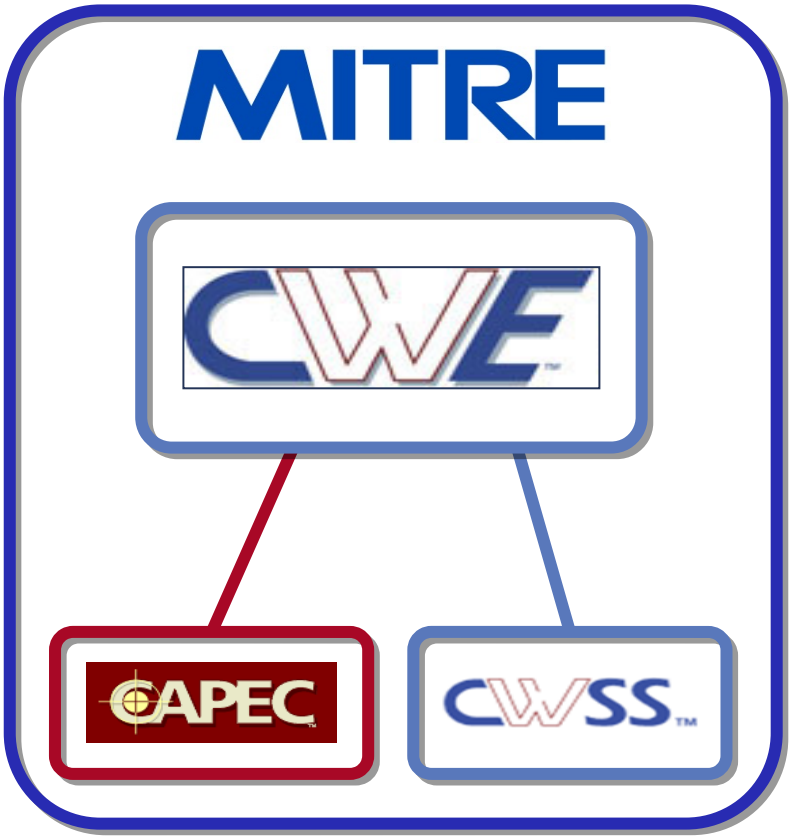
# Common Attack Pattern Enumeration and Classification

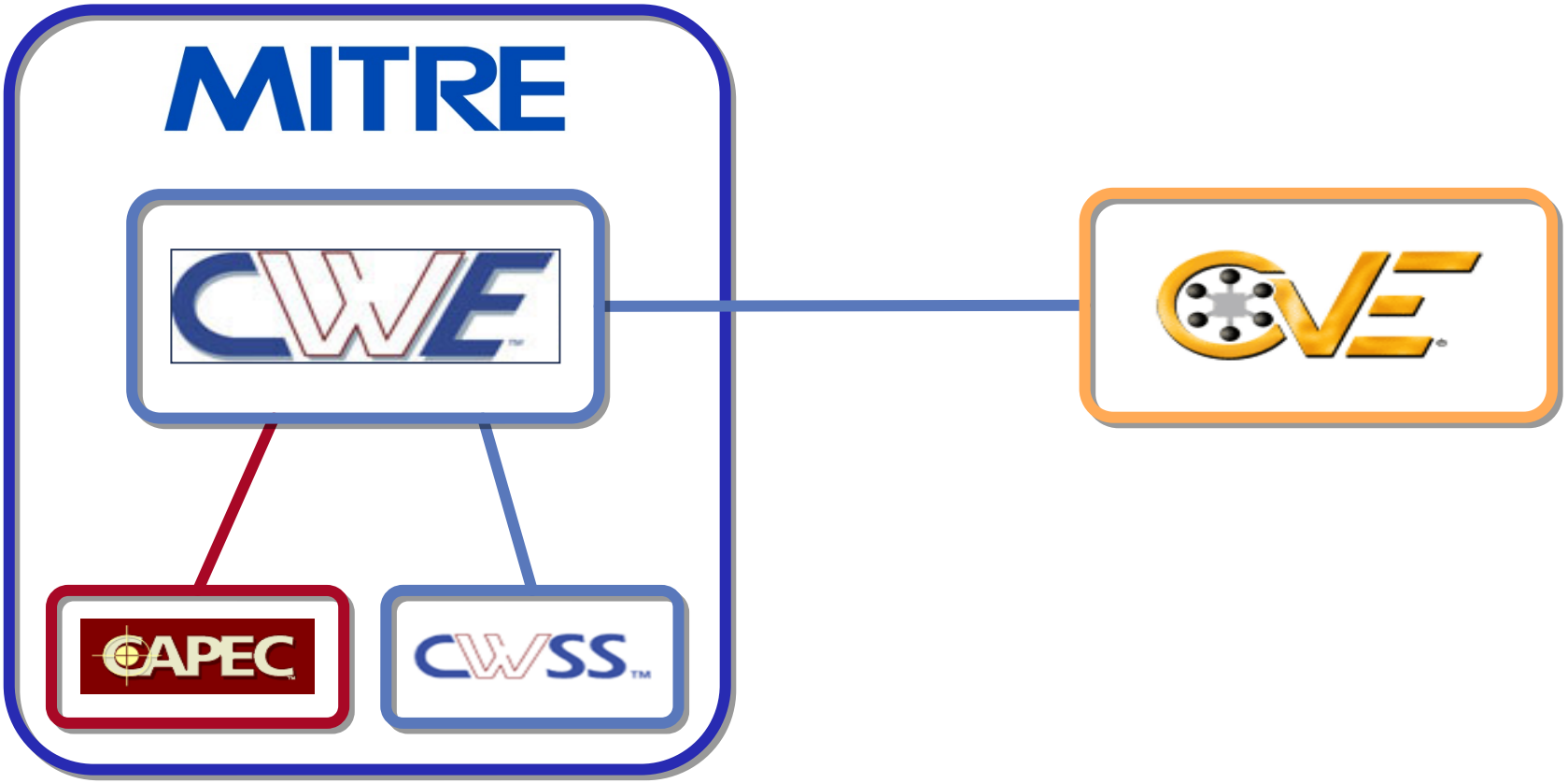

- https://capec.mitre.org/
- eine gemeinschaftlich entwickelte Aufzählung von Angriffsmustern für gängige Schwachstellen
- aktuell ca. 550 Angriffsmuster aufgeführt
- **Angriffsmuster:** Beschreibung einer gängigen Methode, um eine Verwundbarkeit in einem Softwaresystem auszunutzen


CWE
CAPEC
CWSS

# Common Weakness Scoring System

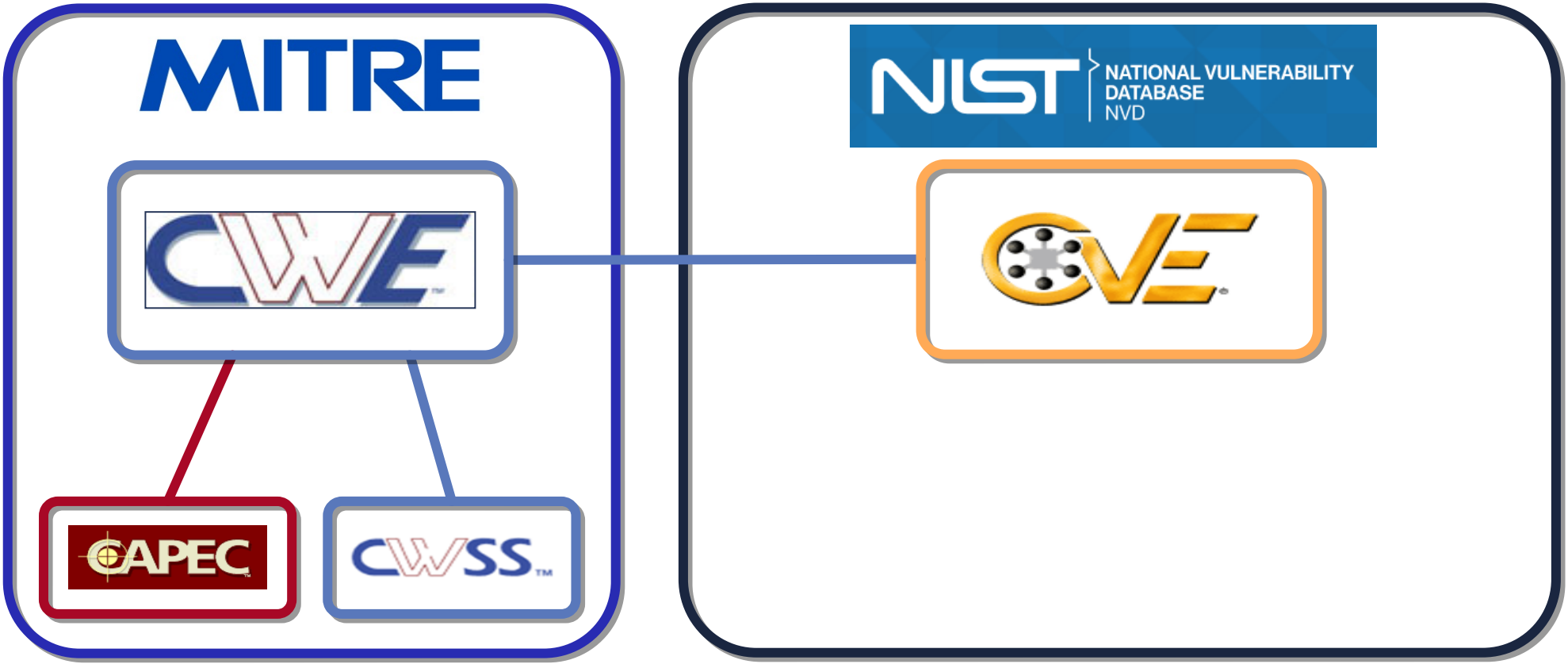

- https://cwe.mitre.org/cwss
- bietet ein einheitliches Vorgehen, um Schwachstellen, die in einem Softwaresystem gefunden wurden, zu priorisieren
- bietet ein quantitatives Maß der unbehobenen Schwachstellen, die in einer Software-Anwendung vorhanden sind
- kann von Software-Entwickler_innen genutzt werden, um die unbehobenen Schwachstellen zu priorisieren

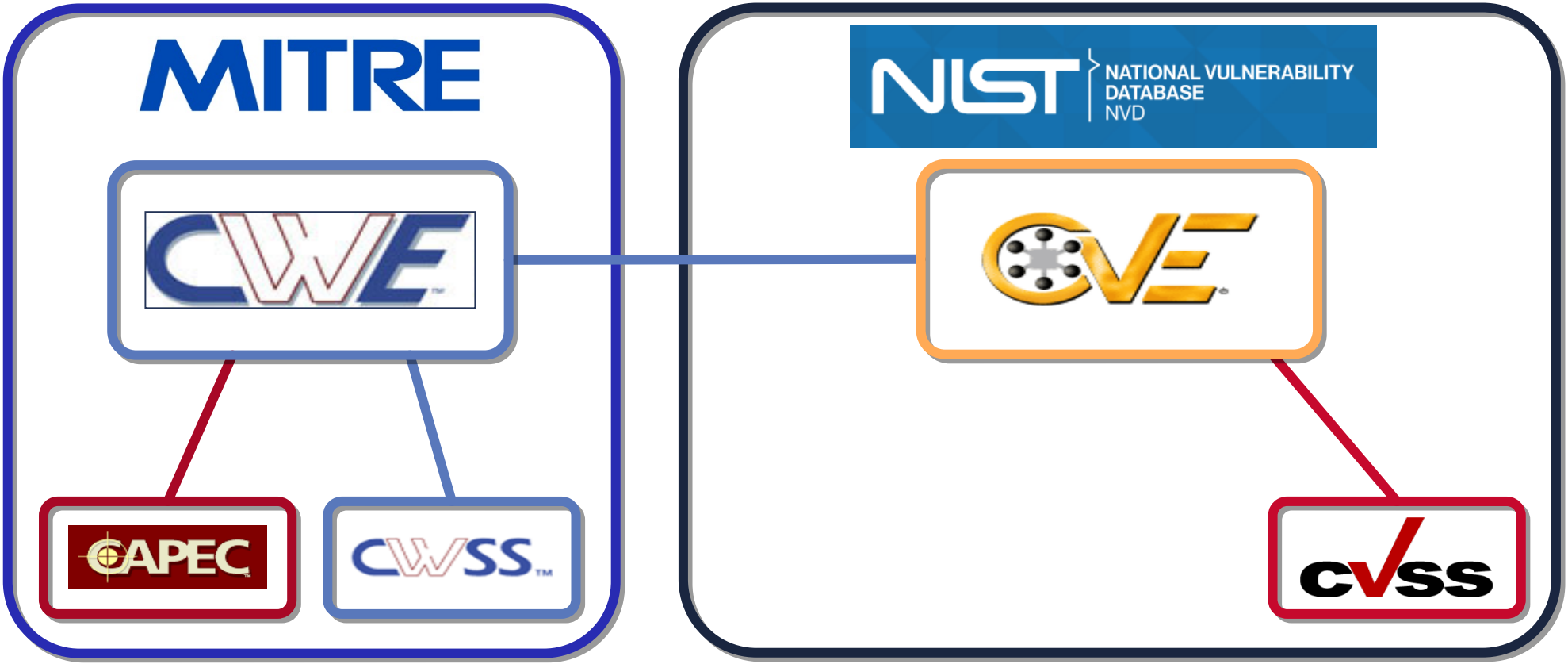

MITRE
CWE
CAPEC
CWSS


MITRE
CWE
CAPEC
CWSS
CVE

# Common Vulnerabilities and Exposures

- https://cve.mitre.org/
- eine Aufzählung von öffentlich bekannten Verwundbarkeiten und Offenlegungen, die in gängiger Software gefunden wurden

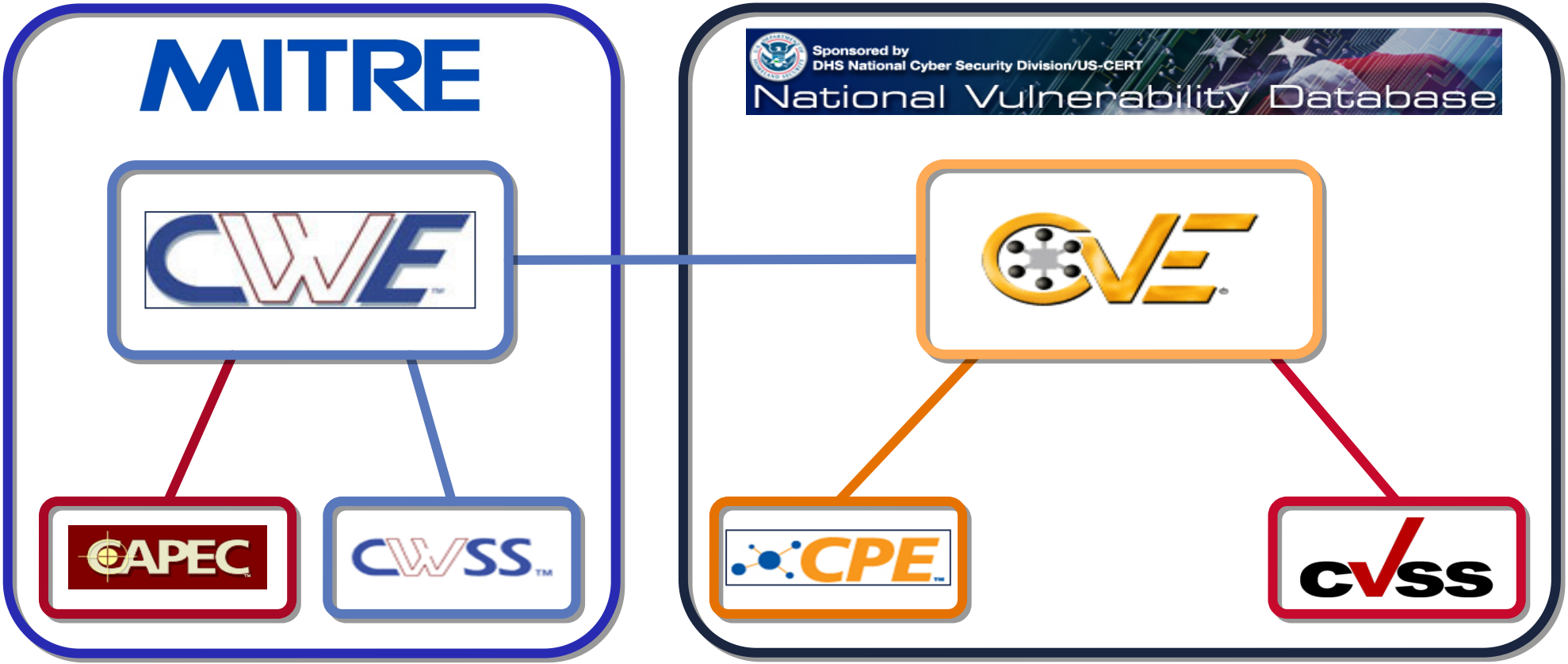

MITRE
CWE
CAPEC
CWSS
NIST
NATIONAL VULNERABILITY
DATABASE
NVD
CVE


MITRE
CWE
CAPEC
CWSS
NIST
NATIONAL VULNERABILITY DATABASE
NVD
CVE
CVSS

# Common Vulnerabilities Scoring System

- https://nvd.nist.gov/vuln-metrics/cvss
- bietet ein einheitliches Vorgehen, um Verwundbarkeiten, die in einem Softwaresystem gefunden wurden, zu priorisieren und deren Auswirkung zu beschreiben

| Rating | CVSS Score |
|---|---|
| None | 0.0 |
| Low | 0.1 - 3.9 |
| Medium | 4.0 - 6.9 |
| High | 7.0 - 8.9 |
| Critical | 9.0 - 10.0 |

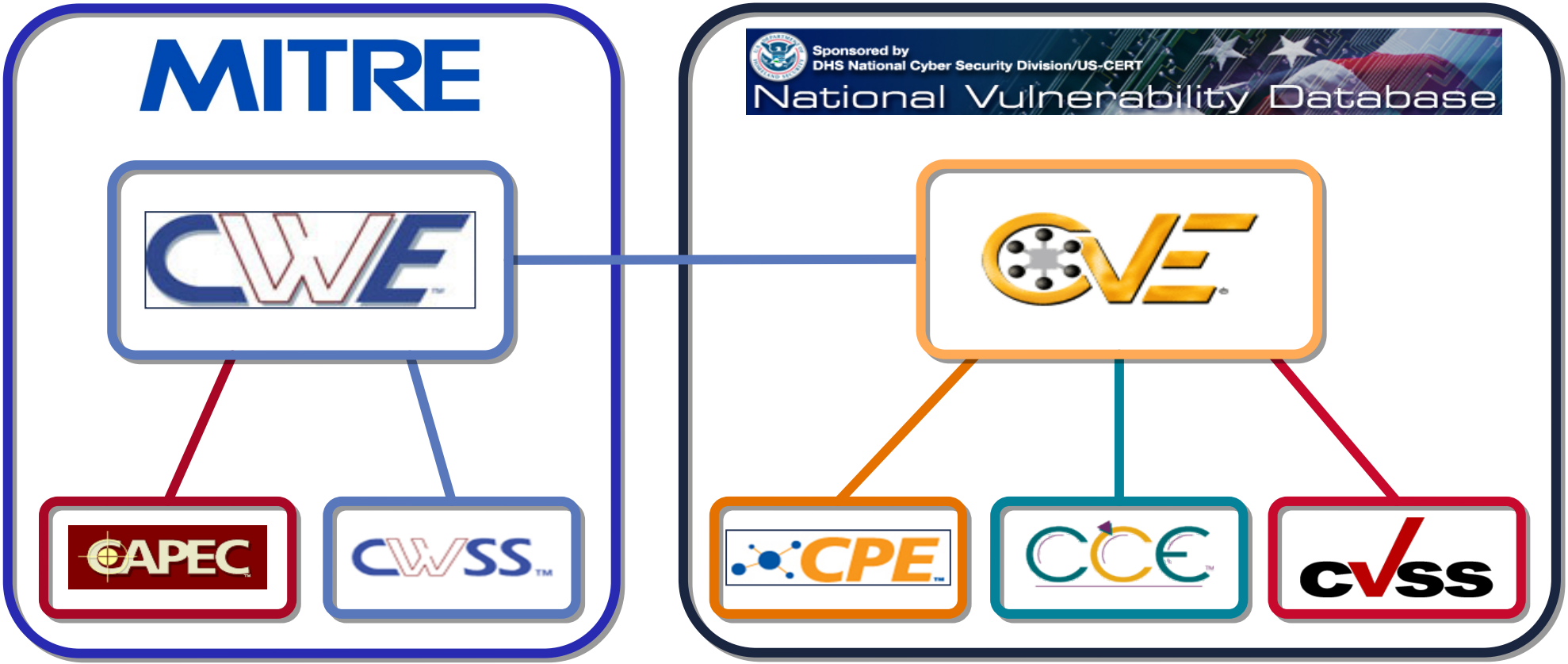

MITRE
CWE
CAPEC
CWSS
Sponsored by
DHS National Cyber Security Division/US-CERT
National Vulnerability Database
CVE
CPE
CVSS

# Common Platform Enumeration


- https://nvd.nist.gov/products/cpe
- eine Menge von NIST-Standards, mit denen Software-Anwendungen, Betriebssysteme und Hardware-Komponenten eindeutig identifiziert werden können
  - CPE Language
  - CPE Dictionary
  - CPE Matching
  - CPE Naming

z. B.: `cpe:/a:microsoft:powerpoint:2013`

- type: application (a)
- vendor: microsoft
- product: powerpoint
- version: 2013


MITRE
CWE
CAPEC
CWSS
Sponsored by
DHS National Cyber Security Division/US-CERT
National Vulnerability Database
CVE
CPE
CCE
CVSS

# Common Configuration Enumeration

- https://nvd.nist.gov/config/cce/index
- eine Aufzählung von Systemkonfigurationen, die bei der Beschreibung von Sicherheitsproblemen genutzt werden kann, um präzise die Situation zu beschreiben, in der ein Fehler auftritt

# CWE Top 25 Most Dangerous Software Errors

- https://cwe.mitre.org/top25/
- “CWE Top 25 Most Dangerous Software Errors” ist eine Liste der meistverbreitetsten und kritischsten Schwachstellen, die zu schwerwiegenden Verwundbarkeiten in Software führen können


- diese Schwachstellen sind üblicherweise einfach zu finden und leicht auszunutzen und sind gefährlich, weil sie den Angreifer/innen häufig ermöglichen ein System zu kapern, Daten zu klauen oder sie verhindern können, dass ein System normal funktioniert

# CWE Top 25 Most Dangerous Software Errors – Version 2024

| | | | | |
|---|---|---|---|---|
| **1. CWE-79**<br>Cross-site Scripting | **2. CWE-787**<br>Out-of-bounds Write | **3. CWE-89**<br>SQL Injection | **4. CWE-352**<br>Cross-Site Request Forgery | **5. CWE-22**<br>Path Traversal |
| **6. CWE-125**<br>Out-of-bounds Read | **7. CWE-78**<br>OS Command Injection | **8. CWE-416**<br>Use After Free | **9. CWE-862**<br>Missing Authorization | **10. CWE-434**<br>Unrestricted Upload of File with Dangerous Type |
| **11. CWE-94**<br>Code Injection | **12. CWE-20**<br>Improper Input Validation | **13. CWE-77**<br>Command Injection | **14. CWE-287**<br>Improper Authentication | **15. CWE-269**<br>Improper Privilege Management |
| **16. CWE-502**<br>Deserialization of Untrusted Data | **17. CWE-200**<br>Exposure of Sensitive Information to an Unauthorized Actor | **18. CWE-863**<br>Incorrect Authorization | **19. CWE-918**<br>Server-Side Request Forgery | **20. CWE-119**<br>Improper Restriction of Operations within the Bounds of a Memory Buffer |
| **21. CWE-476**<br>NULL Pointer Dereference | **22. CWE-798**<br>Use of Hard-coded Credentials | **23. CWE-190**<br>Integer Overflow or Wraparound | **24. CWE-400**<br>Uncontrolled Resource Consumption | **25. CWE-306**<br>Missing Authentication for Critical Function |

# CWE Top 25 Most Dangerous Software Errors – Version 2024

| | | | | |
|---|---|---|---|---|
| **1. CWE-79**<br>Cross-site Scripting | **2. CWE-**<br>Out-of-Bo | **CWE-89**<br>Injection | **4. CWE-352**<br>Cross-Site Request Forgery | **5. CWE-22**<br>Path Traversal |
| **6. CWE-125**<br>Out-of-bounds Read | **7. CWE-78**<br>OS Command Injection | **8. CWE-416**<br>Use After Free | **9. CWE-862**<br>Missing Authorization | **10. CWE-434**<br>Unrestricted Upload of File with Dangerous Type |
| **11. CWE-94**<br>Code Injection | **12. CWE-20**<br>Improper Input Validation | **13. CWE-77**<br>Command Injection | **14. CWE-287**<br>Improper Authentication | **15. CWE-269**<br>Improper Privilege Management |
| **16. CWE-502**<br>Deserialization of Untrusted Data | **17. CWE-200**<br>Exposure of Sensitive Information to an Unauthorized Actor | **18. CWE-863**<br>Incorrect Authorization | **19. CWE-918**<br>Server-Side Request Forgery | **20. CWE-119**<br>Improper Restriction of Operations within the Bounds of a Memory Buffer |
| **21. CWE-476**<br>NULL Pointer Dereference | **22. CWE-798**<br>Use of Hard-coded Credentials | **23. CWE-190**<br>Integer Overflow or Wraparound | **24. CWE-400**<br>Uncontrolled Resource Consumption | **25. CWE-306**<br>Missing Authentication for Critical Function |

**werden in Vorlesung 4 erklärt**

# CWE Top 25 Most Dangerous Software Errors – Version 2024

| | | | | |
|---|---|---|---|---|
| **1. CWE-79**<br>Cross-site Scripting | **2. CWE-787**<br>Out-of-bounds Write | **3. CWE-89**<br>SQL Injection | **4. CWE-352**<br>Cross-Site Request Forgery | **5. CWE-22**<br>Path Traversal |
| **6. CWE-125**<br>Out-of-bounds Read | **7. CWE-78**<br>OS Command Injection | **8. CWE-416**<br>Use After Free | **9. CWE-862**<br>Missing Authorization | **10. CWE-434**<br>Unrestricted Upload of File with Dangerous Type |
| **11. CWE-94**<br>Code Injection | **12. CWE-20**<br>Improper Input Validation | **13. CWE-77**<br>Command Injection | **14. CWE-287**<br>Improper Authentication | **15. CWE-269**<br>Improper Privilege Management |
| **16. CWE-502**<br>Deserialization of Untrusted Data | **17. CWE-200**<br>Exposure of Sensitive Information to an Unauthorized Actor | **18. CWE-863**<br>Incorrect Authorization | **19. CWE-918**<br>Server-Side Request Forgery | **20. CWE-119**<br>Improper Restriction of Operations within the Bounds of a Memory Buffer |
| **21. CWE-476**<br>NULL Pointer Dereference | **22. CWE-798**<br>Use of Hard-coded Credentials | **23. CWE-190**<br>Integer Overflow or Wraparound | **24. CWE-400**<br>Uncontrolled Resource Consumption | **25. CWE-306**<br>Missing Authentication for Critical Function |

# CWE-20: Improper Input Validation
## (unvollständige Eingabeüberprüfung)

- https://cwe.mitre.org/data/definitions/20.html
- Wenn Software die Eingabewerte nicht vollständig überprüft, können im Rahmen eines Angriffs Daten an die Anwendung geschickt werden, die so nicht erwartet wurden in den Bereichen, wo die Daten verarbeitet werden. Dies kann dazu führen, dass das System mit diesen Daten nicht umgehen kann und es zu Veränderungen im Kontrollfluss der Anwendung kommt, Kontrolle über beliebige Ressourcen erreicht oder beliebiger Code ausgeführt werden kann.
- kann sich auf folgende Schutzziele auswirken:
  - Vertraulichkeit
  - Integrität
  - Verfügbarkeit

# Schwachstellen, die zu CWE-20 gehören

- **aus den Top 25:**
  - CWE-77: Command Injection
  - CWE-79: Cross-site Scripting
  - CWE-89: SQL Injection
  - CWE-125: Out-of-bounds Read
  - CWE-190: Integer Overflow or Wraparound
  - CWE-502: Deserialization of Untrusted Data
- **weitere interessante:**
  - CWE-73: External Control of File Name or Path
  - CWE-99: Resource Injection
  - CWE-113: Improper Neutralization of CRLF Sequences in HTTP Headers
  - CWE-127: Out-of-bounds Write
  - CWE-158: Improper Neutralization of Null Byte or NUL Character
  - CWE-426: Untrusted Search Path

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

- Die Anwendung konstruiert einen Teil oder eine komplette SQL-Anweisung basierend auf Eingabewerten, ohne spezielle Teile der Eingabewerte zu neutralisieren, welche die beabsichtigte SQL-Anweisung verändern könnten, wenn sie an die Datenbank geschickt wird.
- Datenbankanfragen können manipuliert werden
- beliebige Daten können
  - aus der Datenbank gelesen werden
  - in die Datenbank geschrieben werden
  - aus der Datenbank gelöscht werden

| Mögliche Auswirkungen | Schutzziel verletzt |
|---|---|
| Daten lesen | Vertraulichkeit |
| Schutzmechanismus umgehen | Zugriffskontrolle |
| Daten verändern | Integrität |

# CWE-89: Beispiel

- lade die Adresse für User, die öffentlich sind (Wert public auf 1 steht)

PHP Beispiel:

```
…
$name = $_REQUEST["name"];
$query = "SELECT address FROM users WHERE public=1 AND name='".$name."'";
$result = mysql_query($query);
…
```

# CWE-89: Beispiel

- lade die Adresse für User, die öffentlich sind (Wert public auf 1 steht)

PHP Beispiel:

```
…
$name = $_REQUEST["name"];
$query = "SELECT address FROM users WHERE public=1 AND name='".$name."'";
$result = mysql_query($query);
…
```

**Eingabewert: Heiko**

**where: public=1 AND name='Heiko'**

**gibt nur die Infos zu Heiko aus, wenn sie öffentlich sind**

# CWE-89: Beispiel

- lade die Adresse für User, die öffentlich sind (Wert public auf 1 steht)

PHP Beispiel:

```
…
$name = $_REQUEST["name"];
$query = "SELECT address FROM users WHERE public=1 AND name='".$name."'";
$result = mysql_query($query);
…
```

**Eingabewert: x' OR public=0 OR name='Heiko**

**where: public=1 AND name='x' OR public=0 OR name='Heiko'**

**gibt auch die Infos zu Heiko aus, wenn sie privat sind**


Quelle: http://images.cryhavok.org/d/1775-1/License+Plate+SQL+Injection+Attack.jpg


Quelle: http://1.bp.blogspot.com/-URBVGKEBRss/U0GgQetIwEI/AAAAAAAAC20/b0DEdUTSCwo/s1600/i.img.jpe


Quelle: http://imgur.com/1EHtAqv

# Mögliche Mitigationen gegen CWE-20-Schwachstellen

- **Mitigation** = Entschärfung – eine Maßnahme, um potentielle Schwachstellen zu verhindern oder deren Auswirkung zu reduzieren

- **Eingabeüberprüfung**
  - einheitliche Eingabeüberprüfungssysteme einsetzen
  - nur gültige Werte zulassen (Allow Lists)
    - nur auf ungültige Werte überprüfen, wenn die Liste der gültigen nicht überprüfbar ist – es ist schwer alle ungültigen Fälle zu kennen
  - wenn Werte aus verschiedenen Quellen kombiniert werden, die Überprüfung erst nach dem Kombinieren der Werte anwenden

- **Angriffsoberfläche erkennen und verkleinern**
  - alle Stellen, in denen Eingaben in die Anwendung gelangen, müssen erkannt und analysiert werden

# CWE-116: Improper Encoding or Escaping of Output
## (falsche Codierung oder Formatierung von Ausgaben)

- http://cwe.mitre.org/data/definitions/116.html

- Die Anwendung generiert eine strukturierte Nachricht, um mit einer anderen Softwarekomponente zu kommunizieren, aber die Codierung oder Formatierung der Daten in der Nachricht fehlt oder wurde falsch durchgeführt. Dadurch kann die beabsichtigte Struktur der Nachricht verfälscht werden.

- kann sich auf folgende Schutzziele auswirken:
  - Verfügbarkeit
  - Vertraulichkeit
  - Integrität

# Beziehung zu CWE-20 (Improper Input Validation)

- je nach Beschaffenheit der strukturierten Nachricht, kann durch korrekt Eingabeüberprüfung verhindert werden, dass spezielle Zeichen und Zeichenfolgen die Struktur der Nachricht manipulieren können
- Eingabeüberprüfung ist aber nicht immer ausreichend, weil gewisse Zeichen in verschiedenen Kontexten erlaubt oder auch nicht erlaubt sein könnten

z. B. Sonderzeichen in Namen: `O'Reilly`

# Anwendungs-Eingabe und -Ausgabe

# Anwendungs-Eingabe und -Ausgabe

# Anwendungs-Eingabe und -Ausgabe

# OWASP

The Open Web Application Security Project

## https://www.owasp.org

*Auszug aus der deutschen Webseite:*

- OWASP ist eine unabhängige, weltweite Community [...]. Ziel des OWASP ist die Unterstützung von Unternehmen und Organisationen bei der Entwicklung und beim Betrieb sicherer Webanwendungen und das «Sichtbar-Machen» der Bedeutung der Sicherheit von Webanwendungen.
- Sämtliche OWASP-Instrumente, wie Dokumente, Foren oder die jeweiligen Länder-Chapters stehen kostenlos allen zur Verfügung, die daran interessiert sind, die Sicherheit von Webanwendungen zu erhöhen.
- Die Community ist frei und offen und heißt alle Interessierten sowie Wissens- und Erfahrungsträger herzlich willkommen. Zwanglos kann dies z. B. im Rahmen der OWASP Stammtische erfolgen, die regelmäßig in vielen deutschen Großstädten stattfinden.

# OWASP Top 10
## Sicherheitsrisiken für Webanwendungen

**https://owasp.org/www-project-top-ten/**

- **A01:2021-Broken Access Control**
- **A02:2021-Cryptographic Failures**
- **A03:2021-Injection**
- **A04:2021-Insecure Design**
- **A05:2021-Security Misconfiguration**
- **A06:2021-Vulnerable and Outdated Components**
- **A07:2021-Identification and Authentication Failures**
- **A08:2021-Software and Data Integrity Failures**
- **A09:2021-Security Logging and Monitoring Failures**
- **A10:2021-Server-Side Request Forgery**

# Welche Software ist sicher und welche ist unsicher?

**Eigentlich gibt es nur Software, von der bekannt ist, dass sie unsicher ist und solche, von der noch nicht bekannt ist, ob sie unsicher ist.**

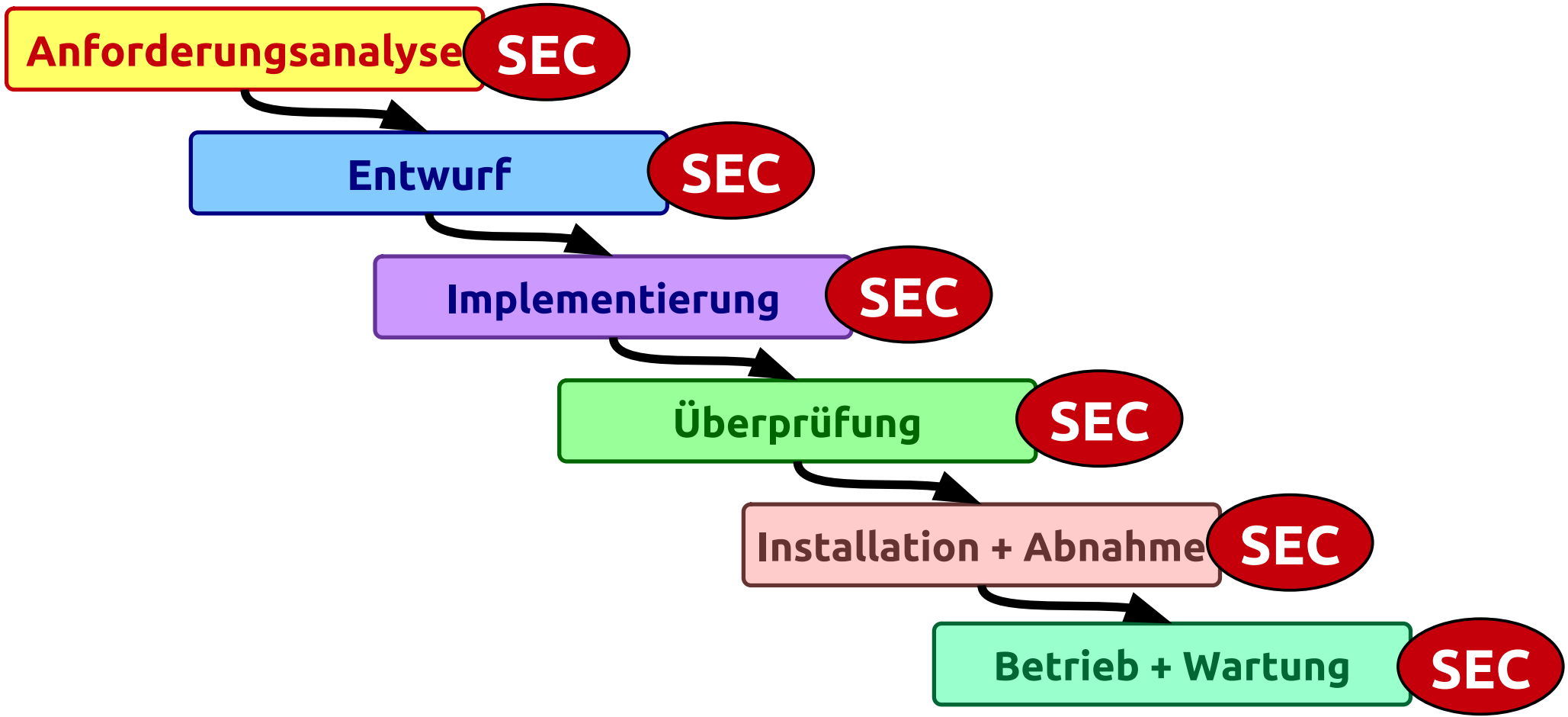
**D.h. es gibt Software, von der aus Erfahrung gesagt werden kann, dass sie laut den bisher bekannten Informationen relativ sicher ist. Absolut sichere Software gibt es nicht.**

# Was macht sichere Software aus?

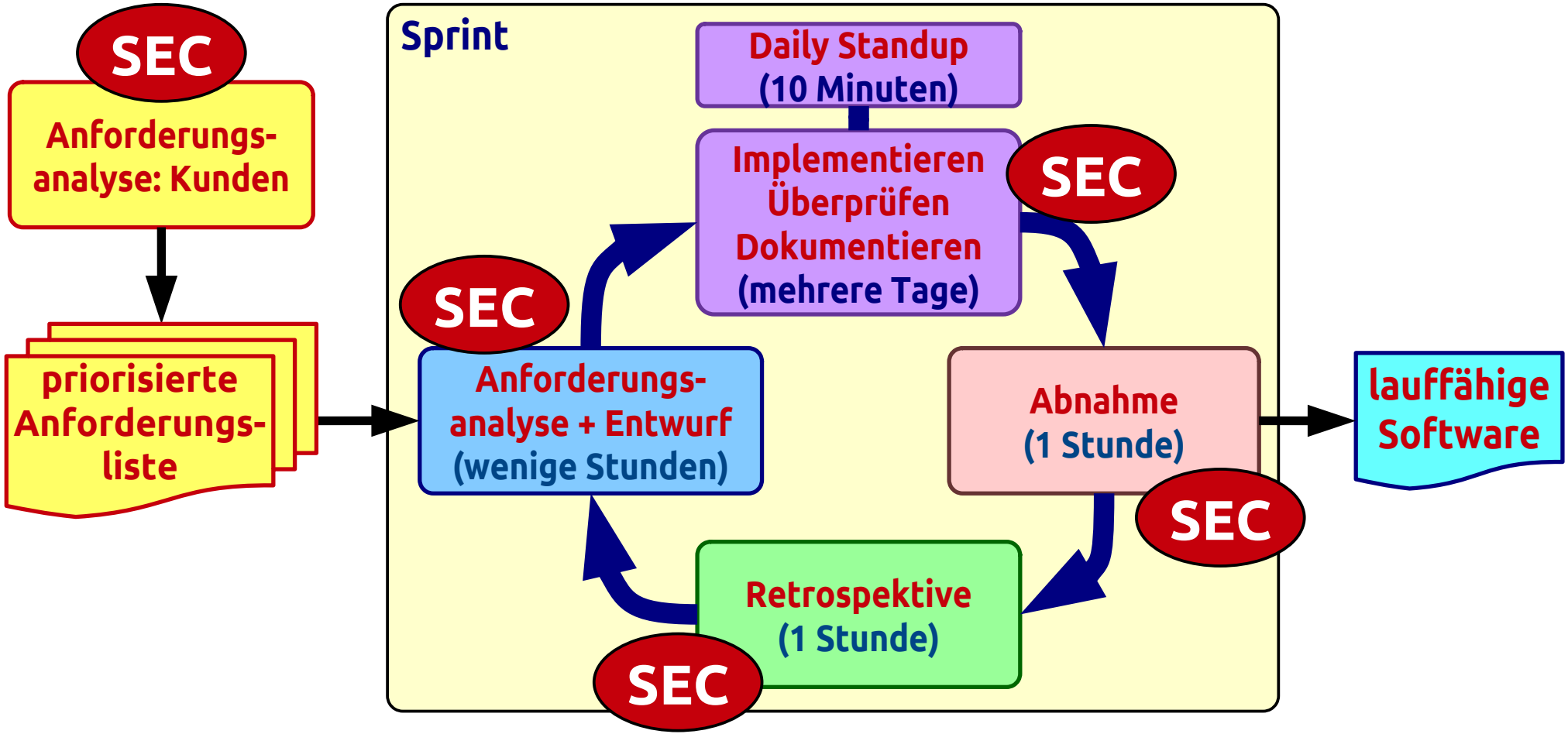
- Sicherheit wurde bei der Anforderungsanalyse berücksichtigt
- Sicherheit wurde beim Entwurf berücksichtigt
- Sicherheit wurde bei der Implementierung berücksichtigt
- Sicherheit wurde bei der Überprüfung berücksichtigt
- Sicherheit wurde bei der Installation und Abnahme berücksichtigt
- Sicherheit wurde bei Betrieb und Wartung berücksichtigt

- **Sicherheit wurde in allen Phasen der Software-Entwicklung berücksichtigt**
- **das „Security by Design“-Prinzip wurde berücksichtigt**

# Sicherheit in allen Phasen des Wasserfallmodells



# Sicherheit in allen Phasen des Scrum-Kreislaufs



# Wer stellt sichere Software her?

- die meiste Software wird heute so schnell und so günstig wie möglich hergestellt – da bleibt wenig Zeit, um Sicherheit ernst zu nehmen
- da in den letzten Jahren die Anzahl von Sicherheitsvorfällen ständig steigt, sind aber immer mehr Software-Firmen gezwungen, nun doch mehr Zeit in Sicherheit zu investieren
- nur selten wird aber das „Security by Design“-Prinzip befolgt, sondern der Hauptfokus liegt auf der Überprüfung

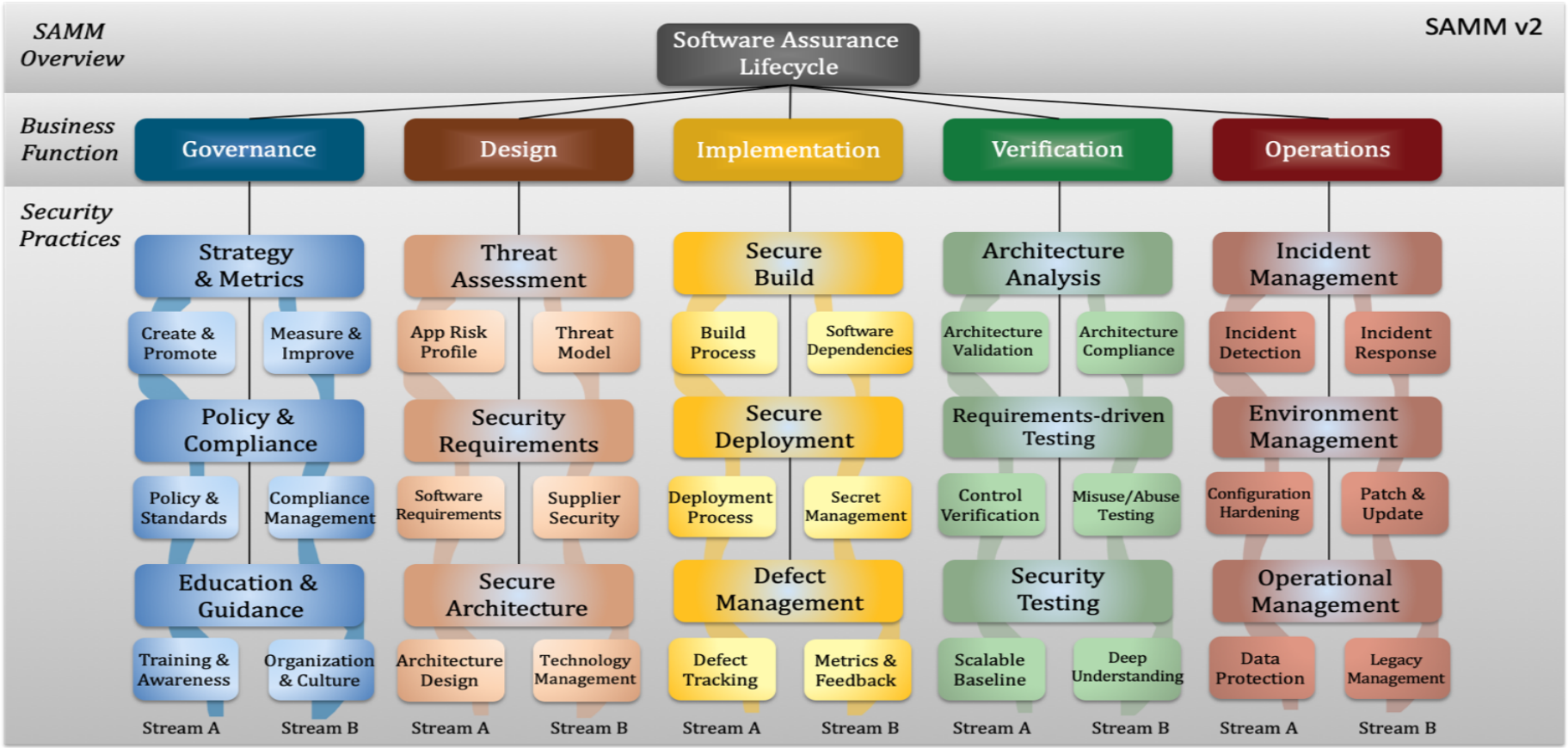
  >> umfangreiche Tests werden eingesetzt, um die schlimmsten Sicherheitsprobleme aufzudecken

# Entwicklung sicherer Software

- es existieren verschiedene Herangehensweisen, wie sichere(re) Software entwickelt werden kann
- einige fokussieren eher darauf, dass Sicherheit überhaupt in den verschiedenen Phasen der Software Entwicklung adressiert wird
  z. B. Software Assurance Maturity Model (SAMM)
- andere beschreiben konkrete Schritte und Verfahren, die bei der Software-Entwicklung angewandt werden sollen
  z. B. Security Development Lifecycle (SDL) von Microsoft

# Software Assurance Maturity Model (SAMM)

https://owaspsamm.org/

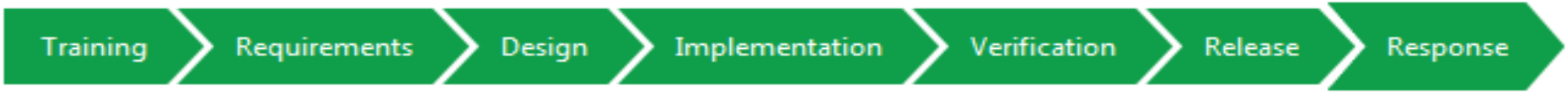


# Security Development Lifecycle (SDL)

https://www.microsoft.com/en-us/sdl/

## What is the Security Development Lifecycle ?

The Security Development Lifecycle (SDL) is a software development process that helps developers build more secure software and address security compliance requirements while reducing development cost



# Sicherheitsüberprüfung

- Softwarefirmen setzen verschiedene Sicherheitstests ein, um die schlimmsten Sicherheitsprobleme in der Software zu finden, bevor sie ein Black-Hat-Hacker findet
- die Sicherheitstests untersuchen die Software nach bekannten Mustern und Funktionalitäten, die auf eine Schwachstelle oder Verwundbarkeit hindeuten
- es wird grundsätzlich unterschieden zwischen **automatischen** und **manuellen** Tests und zwischen White-Box- und Black-Box-Tests
- ein **White-Box-Test** hat Zugriff auf die Interna einer Software, also auf den Quellcode und auf die Konfigurationen
- ein **Black-Box-Test** betrachtet die Software nur von außen und hat keine Informationen über die Interna einer Software

# Typen von Sicherheitsüberprüfungen

- **Statische Analyse**
  - automatisierte White-Box-Tests, die gegen den Quellcode oder die Binärdateien einer Software-Anwendung laufen
  - findet Schwachstellen
- **Dynamische Analyse**
  - automatisierte Black-Box-Tests gegen eine laufende Anwendung
  - findet Verwundbarkeiten
- **Fuzzing**
  - automatisierte Black-Box-Tests gegen eine laufende Anwendung
  - findet Verwundbarkeiten im Bereich Verfügbarkeit und Stabilität
- **Penetration Testing**
  - normalerweise eine Kombination aus den oben aufgeführten Tests in Kombination mit einer detaillierten manuellen Analyse

# Malware

- Malware = Malicious Software (bösartige Software / Schadsoftware)
- Software, die unerwünschte Funktionen ausführt
- verschiedene Arten – oftmals auch Mischformen:
  - Viren
  - Würmer
  - Trojanische Pferde

# Malware

- Malware = Malicious Software (bösartige Software / Schadsoftware)
- Software, die unerwünschte Funktionen ausführt
- verschiedene Arten – oftmals auch Mischformen:
  - Viren
  - Würmer
  - Trojanische Pferde
- je nach Art der bösartigen Funktion wird auch unterschieden nach:
  - Adware
  - Ransomware
  - Rootkits
  - Spyware
  - und viele weitere Unterkategorien...

# Viren

- selbstreproduzierend
- keine eigenständige Software – benötigen einen Wirt (beispielsweise eine übliche Anwendungssoftware)
- Wirtssoftware muss ausgeführt werden zum Aktivieren des Virus
- Funktionsweise:
  1. Start des infizierten Programms (Wirtssoftware): Aktivierung des Virus
  2. Virus infiziert selbständig andere Programme
  3. Virus aktiviert Schadensfunktion

# Viren

- Infizierbare Dateien:
  - ausführbare Dateien
    (Programmdateien, Programmbibliotheken, Skripte, …)
  - Dateien, die ausführbare Inhalte beinhalten
    (Makros – z. B. in PDF-Dateien, Office-Dateien, …)
  - Infektion von Bootsektoren
- Unterscheidung entsprechender Typen, z.B.
  - Dateiviren
  - Skriptviren
  - Makroviren
  - Bootsektorviren

# Infektionswege von Viren

- Verbreitung: passiv
  durch Kopieren einer infizierten Wirtsdatei auf ein noch nicht infiziertes System
- Verbreitungswege:
  - Versenden infizierter Dateien per E-Mail
  - Einsatz von Software aus fragwürdigen Quellen
    (z. B. über P2P-Filesharing-Netzwerke, Web-Downloads, …)
  - Kopieren fremder Software (am eigenen oder fremden Rechner)
  - Nutzung fremder oder gemeinsamer Datenträger
  - Arbeiten an Rechnern, deren Festplatte bereits infiziert ist
  - Verwendung infizierter Programme eines Fileservers

# Makroviren

- werden in den letzten Jahren besonders häufig per E-Mail verbreitet
- Verbreitung oft durch E-Mail-Anhänge mit Word- oder Excel-Dokumenten
- sehr effektiv, weil Office auf vielen Computern installiert ist und es nicht selten vorkommt, dass Office-Dokumente per E-Mail verschickt werden
- beim Öffnen des Office-Dokuments und dem Aktivieren der Makro-Funktion, wird der Virus aktiviert

# Würmer

- spezielle Art von Viren
- selbstreproduzierend
- selbstständige Software – kein Wirt benötigt
- Verbreitung: aktiv
  z. B. durch Versenden von E-Mails an Mitglieder der Adressliste oder automatischem Zugriff auf weitere System über Netzwerkverbindungen
- muss ausgeführt werden zum Aktivieren des Wurms

# Trojanische Pferde

- auch teilweise als „Trojaner“ bezeichnet (wobei es ja eigentlich eher „Griechen“ heißen müsste – das Pferd enthielt ja Griechen, die die Trojaner angegriffen haben)
- Schadsoftware, die vom Benutzer unbemerkt Aktionen auf dessen Computer ausführt - zu diesen Aktionen gehören u. a.:
  - Löschen von Daten
  - Sperren von Daten
  - Modifizieren von Daten
  - Auslesen/Kopieren von Daten
  - Beeinträchtigen der Funktionalität von Computern oder Computernetzwerken

# Trojanische Pferde

- Schadsoftware, die vom Benutzer unbemerkt Aktionen auf dessen Computer ausführt - zu diesen Aktionen gehören u. a.:
  - **Löschen von Daten**
  - **Sperren von Daten**
  - Modifizieren von Daten
  - Auslesen/Kopieren von Daten
  - **Beeinträchtigen der Funktionalität von Computern oder Computernetzwerken**

**Verfügbarkeit**

# Trojanische Pferde

- Schadsoftware, die vom Benutzer unbemerkt Aktionen auf dessen Computer ausführt - zu diesen Aktionen gehören u. a.:
  - **Löschen von Daten**
  - **Sperren von Daten**
  - **Modifizieren von Daten**
  - Auslesen/Kopieren von Daten
  - **Beeinträchtigen der Funktionalität von Computern oder Computernetzwerken**

**Verfügbarkeit**

**Integrität**

# Trojanische Pferde

- Schadsoftware, die vom Benutzer unbemerkt Aktionen auf dessen Computer ausführt - zu diesen Aktionen gehören u. a.:
  - **Löschen von Daten**
  - **Sperren von Daten**
  - **Modifizieren von Daten**
  - **Auslesen/Kopieren von Daten**
  - **Beeinträchtigen der Funktionalität von Computern oder Computernetzwerken**

**Verfügbarkeit**

**Integrität**

**Vertraulichkeit**

# Trojanische Pferde

- im Gegensatz zu Computerviren und -würmern sind Trojanische Pferde nicht in der Lage, sich selbständig zu vervielfältigen
- sie werden ganz gezielt von einem Angreifer auf einem PC, Tablet, Smartphone oder sonstigem Computer installiert
- anhand ihrer Funktion, werden Trojanische Pferde in verschiedene Typen unterteilt:
  - Exploit, Backdoor, Rootkit, Banker, DDoS, Keylogger, Downloader, Fake Antivirus, Instant Messaging, Ransom, Spy, Mailfinder, Clicker, Proxy, Notifier, ...

# Funktionen Trojanischer Pferde

- **Exploit**
  Programme, die Daten oder Code enthalten, mit dem Schwachstellen auf dem lokalen Computer ausgenutzt werden, um weitere Rechte zu bekommen oder um unberechtigt Daten zu erlangen.
- **Backdoor**
  Eine Hintertür über die  ein anderer Benutzer die Kontrolle über den infizierten Computer erlangt. Backdoor-Trojaner werden häufig eingesetzt, um mehrere befallene Computer zu einem so genannten Botnet oder Zombie-Netzwerk zusammenzuschließen, welches dann zu kriminellen Zwecken verwendet wird, z. B. um einen DDoS-Angriff zu starten.
- **Keylogger** (Funktionalität von Spyware)
  Protokolliert unbemerkt die Tastatureingaben mit und übermittelt sie an den Angreifer. Somit können z.B. Passwörter abgefangen werden.

# Funktionen Trojanischer Pferde

- **Ransomware**
  Ein Programm, das in der Lage ist, Daten auf einem infizierten Computer so zu manipulieren, sodass es zu Störungen kommt oder bestimmte Daten nicht mehr genutzt werden können. Der Computer funktioniert erst wieder ordnungsgemäß, wenn ein gefordertes „Lösegeld“ bezahlt wurde.
- **Wiper**
  Funktioniert ähnlich Ransomware, was den Schaden angeht. Jedoch werden die Daten oder Funktionen unwiderruflich gelöscht oder gestört und können nicht mehr hergestellt werden.
- **Fake Antivirus**
  Trojan-FakeAV-Programme simulieren die Aktivität von Antiviren-Software. Sie dienen dazu, Geld zu erpressen – als Gegenleistung für den Schutz vor Bedrohungen, obwohl diese in Wirklichkeit überhaupt nicht existieren. Also eigentlich eine spezielle Art von Ransomware.

# „Staatstrojaner“

„Der Chaos Computer Club (CCC) hat eine eingehende Analyse staatlicher Spionagesoftware vorgenommen. Die untersuchten Trojaner können nicht nur höchst intime Daten ausleiten, sondern bieten auch eine Fernsteuerungsfunktion zum Nachladen und Ausführen beliebiger weiterer Schadsoftware. Aufgrund von groben Design- und Implementierungsfehlern entstehen außerdem eklatante Sicherheitslücken in den infiltrierten Rechnern, die auch Dritte ausnutzen können.

Nicht erst seit das Bundesverfassungsgericht die Pläne zum Einsatz des Bundestrojaners am 27. Februar 2008 durchkreuzte, ist von der unauffälligeren Neusprech-Variante der Spionagesoftware die Rede: von der "Quellen-TKÜ" ("Quellen-Telekommunikationsüberwachung"). Diese "Quellen-TKÜ" darf ausschließlich für das Abhören von Internettelefonie verwendet werden. Dies ist durch technische und rechtliche Maßnahmen sicherzustellen.“

*Quelle: http://www.ccc.de/updates/2011/staatstrojaner*

# Staatstrojaner für Quellen-TKÜ ab Herbst verfügbar

von Tomas Rudl am 27. April 2015, 14:57 in Kurzmeldungen / 10 Kommentare

Um gezielt Ende-zu-Ende-verschlüsselte Kommunikation abfangen zu können, entwickelt das Bundeskriminalamt (BKA) schon seit geraumer Zeit eine angepasste Version des Staatstrojaners. In einem Spiegel-Interview hat BKA-Chef Holger Münch nun verkündet, dass die Software ab dem Herbst 2015 einsatzbereit sein und auch den Ländern zur Verfügung stehen soll.

Die Software zur „Quellen-Telekommunikationsüberwachung" soll sich in einzelne Kommunikationsvorgänge einklinken, die über VoIP-Messenger wie Skype abgewickelt werden, und deren Inhalte an die Behörden weiterleiten, bevor die Verschlüsselung zu greifen beginnt. „Wir entwickeln ein Instrument, mit dem wir – nach richterlicher Genehmigung – an den Computer des mutmaßlichen Täters gehen, bevor er seine Kommunikation verschlüsselt", erklärte Münch. Für Quellen-TKÜ gelten deutlich niedrigere Hürden als für eine komplette Überwachung eines Rechners.

Die Neuentwicklung des Staatstrojaners war notwendig geworden, weil die zuvor eingesetzte DigiTask-Software zusätzliche Module nachladen konnte, die weit über Quellen-TKÜ hinausgingen. Nach einer vernichtenden CCC-Analyse sah sich das BKA gezwungen, das Vorhaben mit gesetzlichen Vorgaben in Einklang zu bringen, eine „Standardisierende Leistungsbeschreibung" zu erarbeiten und die Entwicklung kurzerhand selbst in die Hand zu nehmen. Ob dabei auch das Bundesamt für Sicherheit in der Informationstechnik (BSI) mitgeholfen hat, bleibt unklar.

*Quelle: https://netzpolitik.org/2015/staatstrojaner-fuer-quellen-tkue-ab-herbst-verfuegbar/*

# Schutz vor Malware

- kein Starten unbekannter Programme
- Prinzip der geringsten Berechtigung beachten
- Computer nicht unbeaufsichtigt lassen
- Anti-Malware-Software einsetzen
- Firewalls einsetzen, um Netzwerk-Angriffe einzuschränken
- immer die aktuellsten Sicherheitsupdates installieren
  (für das Betriebssystem und alle Anwendungen)

# Anti-Malware-Techniken

## Statische Techniken
die zu prüfende Software wird untersucht, ohne sie auszuführen

### Scanner
sucht nach bekannten Bitmustern in der Software (Signaturerkennung)

### Heuristik
sucht nach Virus-ähnlichen Programmbereichen

### Integritätsprüfungen
sucht nach unautorisierten Modifikationen in bekannter Software

## Dynamische Techniken
die zu prüfende Software wird ausgeführt und das Verhalten beobachtet

### Monitoring der Aktivitäten
sucht nach auffälligen Aktivitäten (Abweichungen von den „normalen“ Aktivitäten) – z. B. hohe Netzwerk-Kommunikation

### Emulation
Ausführen der Software in emulierter Umgebung oder in einer Sandbox und dabei Monitoring

# Prinzip der geringsten Berechtigung

- Berechtigungen so einschränkend wie möglich halten
- Software nur mit Administrator-Rechten ausführen, wenn unbedingt notwendig
- Software mit den Rechten ausführen, die genau das erlauben, was für die Funktionsweise der Software notwendig ist
- damit kann der Schaden, der bei Ausnutzen von Schwachstellen in der Software erreicht werden kann, eingeschränkt werden