

# Secure Software Engineering

## Sommersemester 2023

# Über uns



# Max Stephan

- 23 Jahre alt
- Masterstudent an der THM
- Wissenschaftlicher Mitarbeiter an der THM
- Selbstständiger Softwareentwickler
- [max.stephan@mni.thm.de](mailto:max.stephan@mni.thm.de)

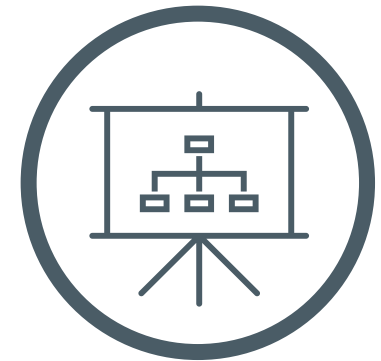


# Timon Pellekoorne

- 25 Jahre alt
- Masterstudent an der THM
- Wissenschaftlicher Mitarbeiter an der THM
- Selbstständiger Softwareentwickler
- [timon.pellekoorne@mni.thm.de](mailto:timon.pellekoorne@mni.thm.de)



# Organisatorisch



# Organisatorisch

## ■ Prüfungsleistung

- Projektarbeit
- Ausarbeitung und Präsentation

## ■ Prüfungsvorleistung

- Regelmäßige Mitarbeit
- Bereitschaft zur Präsentation von 50% der Aufgabenblätter.

# Prüfungsleistung

- Entwicklung einer sicheren Webanwendung
- Vorgegebenes Projekt:
  - Umfang
  - Dokumentation
  - Präsentation

## Abschlussprojekt - Secure Software Engineering

### Beschreibung

#### Projektbeschreibung

### Umfang

### Funktion

#### Funktionsbeschreibung

### Checkliste

- ☐
- ☐
- ☐ Funktionsanforderungen
- ☐

# Prüfungsvorleistung

- Bereitschaft zur Präsentation von 50% (12 Aufgaben) der Aufgabenblätter.
- Abgabe im Feedbacksystem
  - <https://feedback.mni.thm.de>

 <https://feedback.mni.thm.de/sse/login>



# Organisatorisch

- **Übungs- und Projektgruppe**
  - 2 – 3 Studierende
  - Bearbeitung der Übungen
  - Projektarbeit

# Wöchentlicher Ablauf

## Freitag

### 4. Block

#### Vorlesung



### 5. Block

#### Übung

- Vorstellung der Aufgabenblätter
- Hilfestellung



[https://moodle.thm.de/course/  
view.php?id=5747](https://moodle.thm.de/course/view.php?id=5747)





## Dokumente

- Vorlesungen
- Übungen



## Gruppenwahl

- 2 – 3 Studierende
- Bis zum 21.04.2023



id=5747

# Einführung



# Inhalt

- Einführung
- Passwörter
- Sichere Kommunikation
- Cyberangriffe
- Netzwerksicherheit
- Softwaresicherheit
- Sicherheitsnormen & Datenschutz
- Secure Deployment

# Hacking for fun?

STGB:

§ 118a (1): Widerrechtlicher Zugriff auf ein Computersystem<sup>1</sup>

§ 126b: Störung der Funktionsfähigkeit eines Computersystems<sup>1</sup>

§ 126c: Missbrauch von Computerprogrammen oder Zugangsdaten<sup>1</sup>

§ 122ff StGB: Verletzung Betriebsgeheimnis; Strafraumen: bis 3 Jahre

§ 246 StGB: Staatsfeindliche Verbindungen; Strafraumen: bis 5 Jahre

§ 252 StGB: Verrat von Staatsgeheimnissen; Strafraumen: bis 10 Jahre

§ 242 StGB: Hochverrat; Strafraumen: bis 20 Jahre

Weitere Strafen über BDSG, DSGVO, ... Definiert

<sup>1</sup> Geldstrafe oder bis zu 6 Monaten Haft



# Was bedeutet sichere Softwareentwicklung



# IT-Grundschutz Kompendium



- IT-Grundschutz Kompendium vom Bundesamt für Sicherheit in der Informationstechnik (BSI) entwickelte Vorgehensweise zum Identifizieren und Umsetzen von Sicherheitsmaßnahmen in Unternehmen
- In Version 2020 ist das Modul „Software-Entwicklung“ neu hinzugekommen.

# CON.8: Software-Entwicklung

## Anforderungen (Auszug)

- Auswahl eines Vorgehensmodells
- Auswahl einer Entwicklungsumgebung
- Einhaltung einer sicheren Vorgehensweise [Entwickler]
- Sicheres Systemdesign
- Verwendung von Bibliotheken aus vertrauenswürdigen Quellen
- Anwendung von Testverfahren [Tester]
- Bereitstellung von Patches, Updates und Änderungen [Entwickler]
- Versionsverwaltung des Quellcodes [Entwickler]



# Sicheres Systemdesign



- Grundsätzlich **MÜSSEN** alle Eingabedaten vor der Weiterverarbeitung geprüft und validiert werden.
- Bei Client-Server-Anwendungen **MÜSSEN** die Daten grundsätzlich auf dem Server validiert werden.
- Die Standardeinstellungen der Software **MÜSSEN** derart konfiguriert sein, dass ein sicherer Betrieb der Software ermöglicht wird.
- Bei Fehlern oder Ausfall von Komponenten des Systems **DÜRFEN KEINE** schützenswerten Informationen preisgegeben werden.
- Der Betrieb der Software **MUSS** mit möglichst geringen Benutzerprivilegien möglich sein.

**Das Systemdesign MUSS dokumentiert werden. Es MUSS überprüft werden, ob alle Sicherheitsanforderungen an das Systemdesign erfüllt wurden.**

# CON.8: Software-Entwicklung

## Anforderungen (Auszug)

- Auswahl eines Vorgehensmodells
- Auswahl einer Entwicklungsumgebung
- Einhaltung einer sicheren Vorgehensweise [Entwickler]
- Sicheres Systemdesign
- Verwendung von Bibliotheken aus vertrauenswürdigen Quellen
- Anwendung von Testverfahren [Tester]
- Bereitstellung von Patches, Updates und Änderungen [Entwickler]
- Versionsverwaltung des Quellcodes [Entwickler]



# Twitch Leak



- 06.10.2021
- ca. 6000 Git-Repositories
  - Mobil-, Desktop- und Konsolen-Apps
  - geheime Projekte
  - Tochtergesellschaften
  - interne Tools
  - Mitarbeiter-Repositories



Auszahlungen von Twitch an ihre Streamer



3.000.000 Dokumente mit einer Gesamtgröße von 200 GB (ungezippt)

# Risikofaktoren

- Fehlende Verschlüsselung
- (ungeschützte) Schnittstellen
- Fehlkonfiguration / Keine Updates
- Keine Backups
- XSS / SQL-Injections / Malware
- Fehlendes Hashing / Salting von Passwörtern

# Open Web Application Security Project

*„Das Open Web Application Security Project®  
(OWASP) ist eine gemeinnützige Stiftung, die sich für  
die Verbesserung der Sicherheit von Software  
einsetzt“*

- [owasp.org](https://owasp.org)



## > OWASP Top Ten

# OWASP Top Ten

- Die OWASP Top Ten beinhaltet die wichtigsten Sicherheitsrisiken für Webanwendungen

*„Globally recognized by developers as the first step towards more secure coding.“*

- [owasp.org](https://owasp.org)



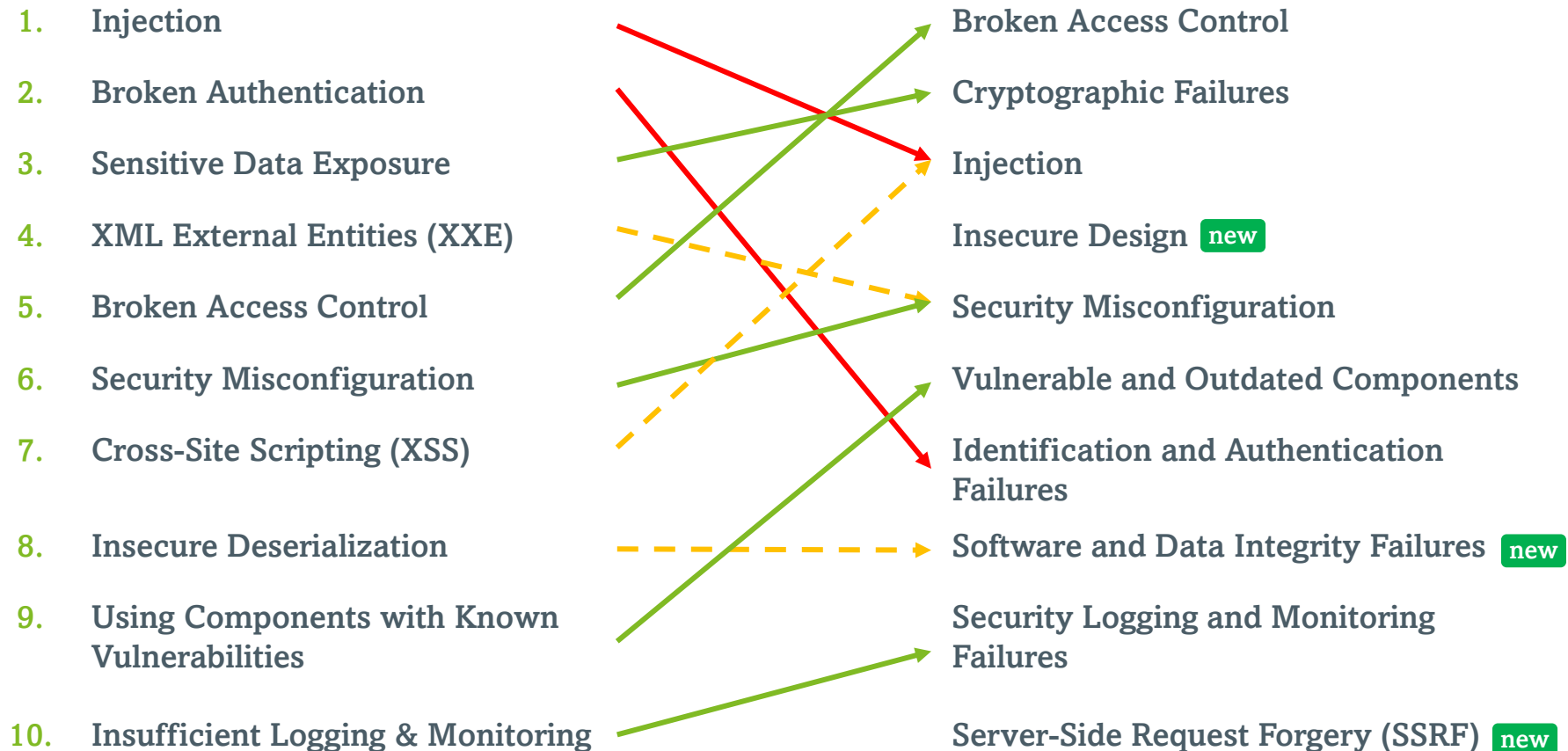
<https://owasp.org/Top10/>



# OWASP Top Ten

2017

2021



# OWASP-Projekte (Auszug)

- OWASP Application Security Verification Standard (ASVS)
- Zed Attack Proxy (ZAP)
- XSSer
- Juice Shop

# Risikofaktor Mitarbeiter



**amazon**

Passwort  
vergessen?  
**amazon**



Passwort-Reset  
per Telefon  
Sicherheitsfrage:  
Kreditkartennummer

# Risikofaktor Mitarbeiter



Neues  
Passwort:



Apple ID

Apple Keychain  
Access



Apple ID

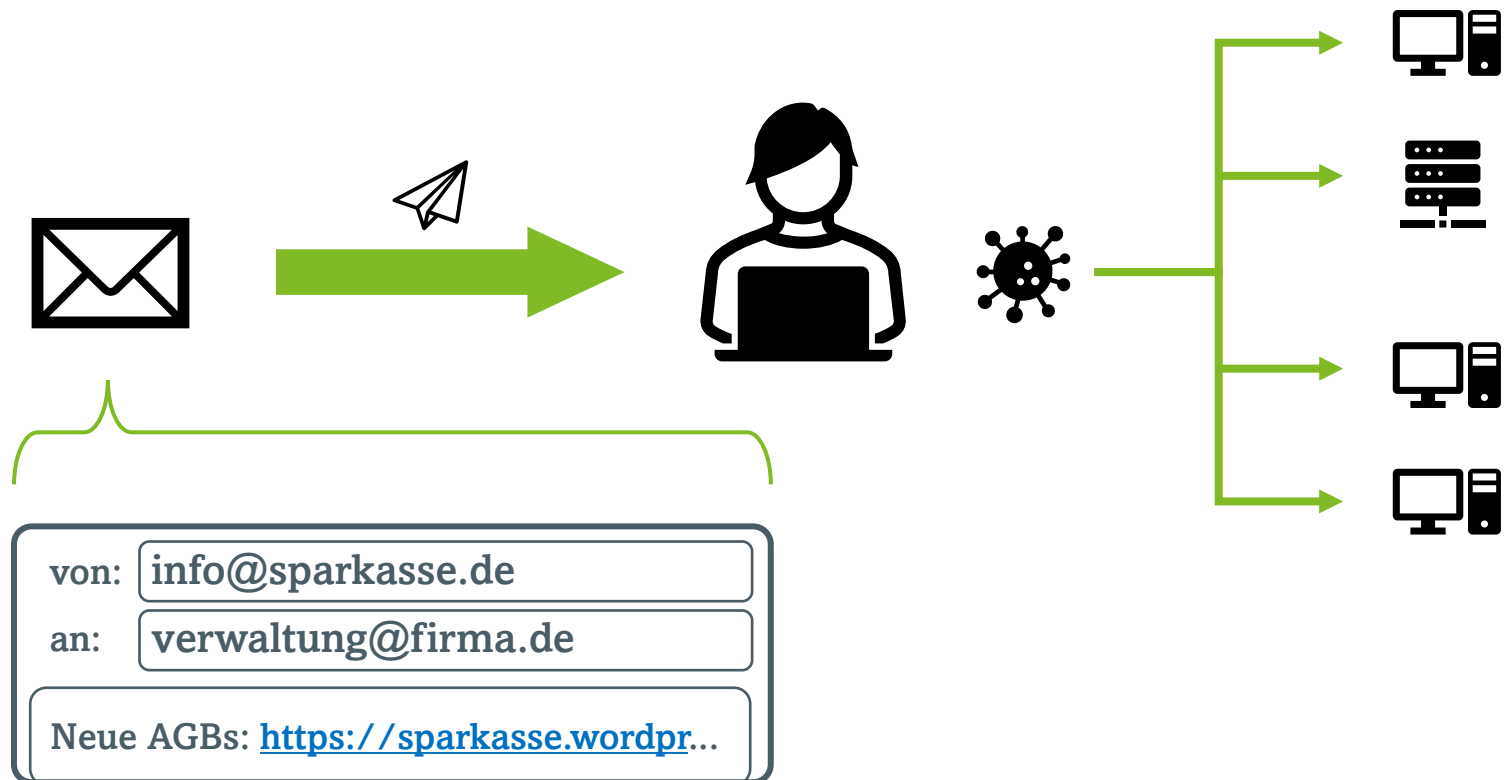


,



usw.

# Risikofaktor Mitarbeiter



# Mail-Spoofing

- **From:** Absender der E-Mail
- **To:** Empfänger der E-Mail
- **Cc:** Indirekter Empfänger (carbon copy, Durchschlagpapier)
- **Bcc:** weitere Empfänger (werden NICHT angezeigt)
- **Subject:** Betreff
- **Date:** Zeitpunkt der Erstellung

## Aufgabe:

Schreiben Sie eine E-Mail mit dem Absender „Günter Schabowski“, Absendedatum „09.11.89 18:00:00“.

Im Plain-Body soll ein anderer Inhalt als im HTML-Body stehen.

# Erkennung von Mail-Spoofing

- E-Mail Kopfzeile

Return-path: <studenten-giessen-bounces+timon.pellekooorne=mni.thm.de@lists.thm.de>

Envelope-to: timon.pellekooorne@mni.thm.de

Delivery-date: Fri, 24 Mar 2023 12:01:14 +0100

Received: from [192.168.186.80] (helo=mailserv.fh-giessen.de)

Received: from listserv.fh-giessen.de ([212.201.18.37])

Received: from localhost ([127.0.0.1] helo=listserv.fh-giessen.de)

Received: from mailgate-1.its.fh-giessen.de ([212.201.18.15])

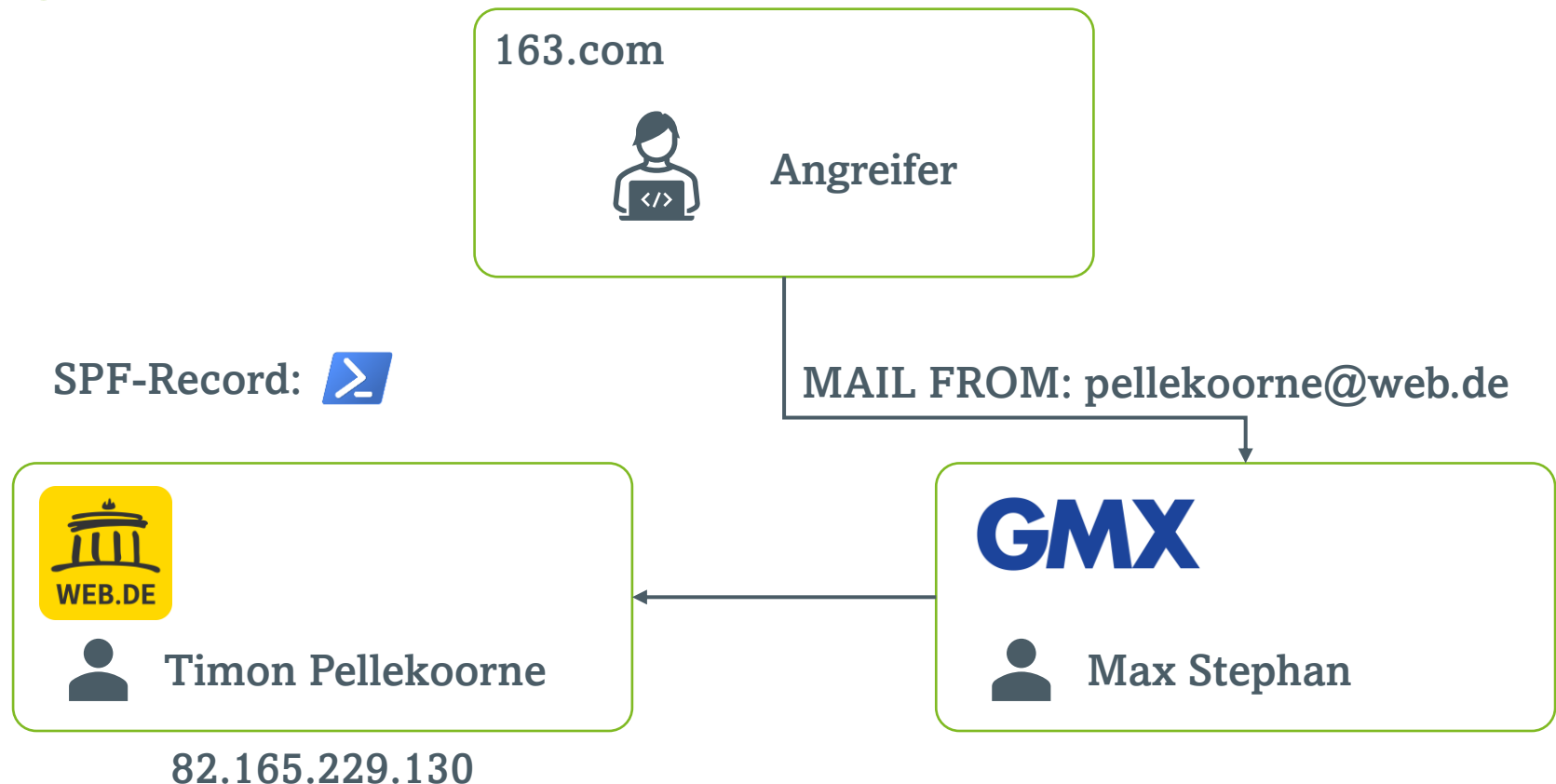
Received: from [10.193.246.81] by mailgate-1.its.fh-giessen.de with esmtpsa

- Protokolle:

- SPF
- DKIM
- DMARC

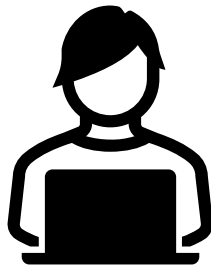
# Erkennung von Mail-Spoofing

## SPF





# Risikofaktoren



Risikofaktor Mitarbeiter



Risikofaktor Passwort

# Passwörter



# Passwörter

- **Beliebte Passwörter sind die unsichersten!**
  - Hasso-Plattner-Institut, 2020: 123456, 123456789, password, ...
- **Mehrfach genutzte Passwörter sind ebenfalls unsicher**
- **Brute-Force-Angriffe:**
  - Probieren aller Möglichkeiten (Wörterbücher, ...)
  - Im Mittel nötig: Hälfte aller möglichen Passwörter
  - Theoretisch kann jede Verschlüsselung gebrochen werden

# Brute-Force

Schlüsselgröße [Bit]	Anzahl Schlüssel	Benötigte Zeit bei 1 enc / $\mu s$	Benötigte Zeit bei $10^6$ enc / $\mu s$
32	$2^{32} = 4,3 * 10^9$	$2^{32} \mu s = 35,8 \text{ min}$	2,15 ms
56	$2^{56} = 7,2 * 10^{16}$	$2^{55} \mu s = 1142 \text{ y}$	10,01 h
128	$2^{128} = 3,4 * 10^{38}$	$2^{127} \mu s = 5,4 * 10^{24} \text{ y}$	$5,4 * 10^{18} \text{ y}$

```
#!/bin/sh
cat passwords.txt | \
while read i ; do
    echo "$i"
    # Abbruch bei Fehler
    if ["$?" != "0" ] ; then
        exit
    fi
done
```

# Sicherheitslücke: Passwörter

- **Testen von  $10^6$  Passwörtern**
  - Dauert wenige Sekunden
  - Deckt alle deutschen Wörter ab
  - Inklusive einer Zahl / eines Sonderzeichen
- **Daher: Sicheres Passwort verwenden**
  - Möglichst lang (mind. 8 Zeichen)
  - Buchstaben, Sonderzeichen, Zahlen
  - Enthalten kein Muster

→ Verwenden von Passphrasen

# Blockieren von Brute-Force-Angriffen



# Konten sperren

- Sperren von Konten nach einer bestimmte Anzahl von Versuchen

└→ Freigabe nach einer gewissen Zeit



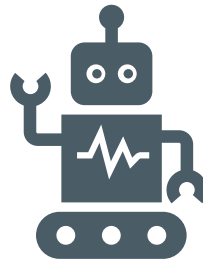
## Probleme

- Ein Angreifer kann einen Denial of Service (DoS) verursachen, indem er eine große Anzahl von Konten sperrt.
- Nur vorhandene Accounts können gesperrt werden
- Angreifer können Accounts sperren
- Ineffizient gegen das Bruteforcen eines Nutzernamen zu nur einem Passwort



# CAPTCHAS verwenden

- Für die Unterscheidung zwischen Mensch und Computer
- Mensch soll zu nahezu **100% korrekt** antworten
- Computer zu nahezu **100% falsch**





# Beispiel CAPTCHAS



14+36=?



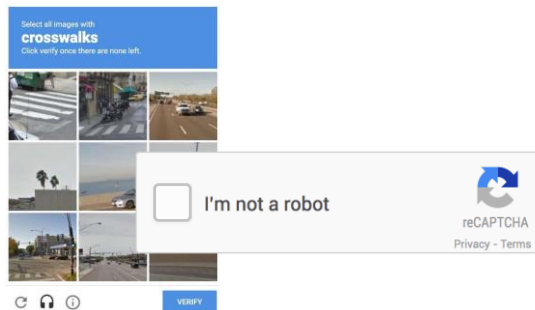
☐ 1  
 ☐ 2  
 ☒ 3



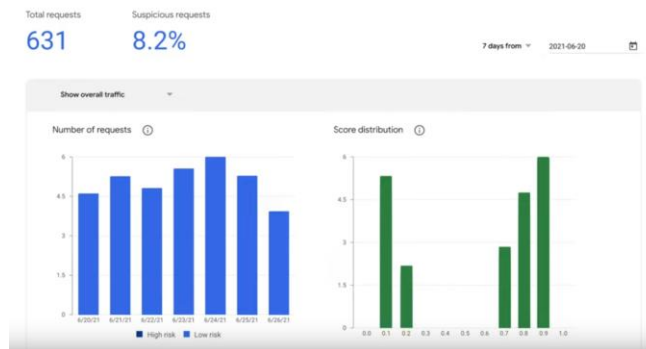
# reCAPTCHA



## > reCAPTCHA v2 (2014)



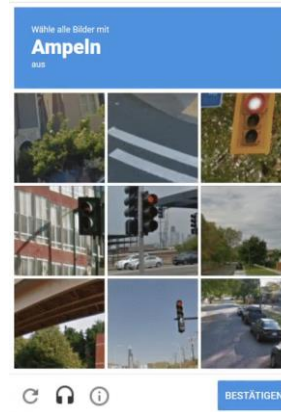
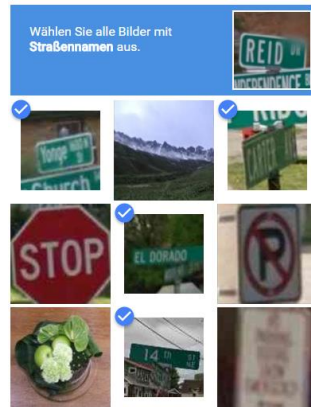
## > reCAPTCHA v3 (2018) / Enterprise (2020)



Analysiert das Verhalten eines Nutzer auf der Website, ob dieses menschlich ist

# Captchas und KI

- Google trainiert ihre KIs für Autos mit Hilfe der Nutzereingaben



- Friendly Captcha
  - Proof of work

# Geräte-Cookies

- Ausgabe eine Geräte-Cookies für den Client bei erfolgreicher Anmeldung
- Unterscheiden zwischen vertrauenswürdigen und unbekannten Clients
- Sperren von unbekannten Clients
- Vertrauenswürdige Clients individuell sperren

# weitere Gegenmaßnahmen

- Blockieren von bestimmten IP-Adressen
  - Großer Nutzergruppe komplett gesperrt
- „security by obscurity“ (Sicherheit durch Unklarheit)
  - Identische Rückmeldung
  - HTTP 401 Unauthorized ➡ HTTP 200 OK

- ```
<body>
  <input id="password">
    <!-- Bad username or password -->
</body>
```

# Mögliche Fehler

- E-Mail konnte nicht gefunden werden

E-Mail oder Telefonnummer

sse@test.de

! Das Google-Konto wurde nicht gefunden

- Passwort-Reset

Get Messages Write Chat Address Book Tag Quick Filter

From: Shodan <no-reply@mg.shodan.io> ☆  
Subject: Shodan Account Information  
To: Me ☆

Hi,

Somebody asked to reset your password on Shodan. If it wasn't you, you can safely ignore this email. Log in with this information and change your password:

**Account Information**

URL: <https://account.shodan.io/change-password>

Username: [REDACTED]  
Password: [REDACTED]

Thank you for using Shodan!

HELP CENTER // SUPPORT // PRIVACY POLICY // TERMS OF SERVICE

Shodan ©

# Geleakte Passwörter



# Datenbank

id	email	password
1	<u>max@mustermann.de</u>	password
2	<u>erika@musterfrau.de</u>	123456



id	email	password
1	<u>max@mustermann.de</u>	c0067d4af4e87f00dbac63b615682...
2	<u>erika@musterfrau.de</u>	d7190eb194ff9494625514b6d178c...



# SHA3-256-Hash

password



c0067d4af4e87f00dbac63b615682823705917  
2d1bbeac67427345d6a9fda484

c0067d4af4e87f00dbac63b615682823  
7059172d1bbeac67427345d6a9fda484



password



Der gleiche String ergibt immer den gleichen Hash!



Ein schlechtes Passwort bleibt ein schlechtes.

# SHA3-256-Hash

c0067d4af4e87f00dbac63b6156828237059172d1bbeac67427345d6a9fda484




 **Alle**

 Maps

 Videos

 Bilder

 Shopping

 Mehr

Suchfilter

Ungefähr 9 Ergebnisse (0,36 Sekunden)

Tipp: Begrenze die Suche auf **deutschsprachige** Ergebnisse. Du kannst deine Suchsprache in den **Einstellungen** ändern.

<https://md5calc.com> › hash › pass... ▾ [Diese Seite übersetzen](#)

## SHA3-256 hash for "password" - Md5Calc.com

SHA3-256 hash for "password" is

"c0067d4af4e87f00dbac63b6156828237059172d1bbeac67427345d6a9fda484". Free online...

# Was können wir tun ?

- Salt/Pepper
- Sichere Hash-Algorithmen verwenden
- Benutzer zur Verwendung starker Passwörter anhalten (Passwort Überprüfung)

# Was können wir tun ?

- **Salt/Pepper**
- Sichere Hash-Algorithmen verwenden
- Benutzer zur Verwendung starker Passwörter anhalten (Passwort Überprüfung)

# Salt/Pepper

Was bringt ein Salt/Pepper?

## Rainbow Table

Password	Hash
password	f87d3cc032ff13dd
123456	0c88e054f285f1f9
555	f0585217ffa1fe36
...	...



Zufällig generierte Zeichenkette vor dem „hashen“ an das Passwort anhängen.

$123456 + \text{bdbd9b275e54c5be7a9116f37d2ea979}$ 
➡
 eba0dd3b4...

password
 salt

# Salt/Pepper

## Anforderungen

- Der Salt sollte einzigartig pro Nutzer und Passwort sein
- Der Salt sollte einem kryptografisch sicheren Zufallsgenerator entspringen



**Math.random() ist nicht sicher!**

- Der Salt sollte mindestens so lang wie der Hash sein
  - Ist der Salt zu kurz können Tabellen für alle möglichen Salt Konfigurationen berechnet werden.

# Salt/Pepper

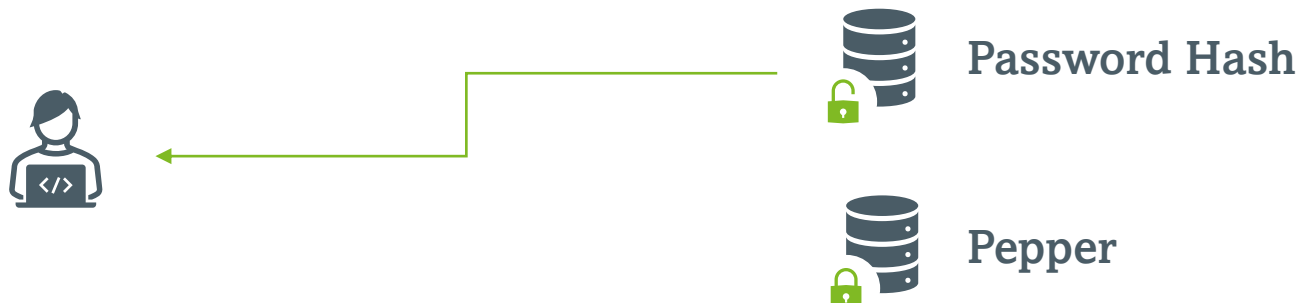
## Salt

- Speicherung bei den Login-Daten



## Pepper

- Speicherung unabhängig der Login-Daten



# Was können wir tun ?

- Salt/Pepper
- **Sichere Hash-Algorithmen verwenden**
- Benutzer zur Verwendung starker Passwörter anhalten (Passwort Überprüfung)



# Hash Algorithmen

Name	Geeignet für Passwörter	Verwendung
md5	✗	-
SHA-1	✗	-
SHA3-256	✗	-
Argon2id	✓	Wenn Möglich
scrypt	✓	Falls Argon2id nicht verfügbar
bcrypt	✓	bestehende Systeme
PBKDF2	✓	FIPS-140-Konformität



[https://cheatsheetseries.owasp.org/cheatsheets/Password\\_Storage\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html)

# Was können wir tun ?

- Salt/Pepper
- Sichere Hash-Algorithmen verwenden
- Benutzer zur Verwendung starker Passwörter anhalten (Passwort Überprüfung)

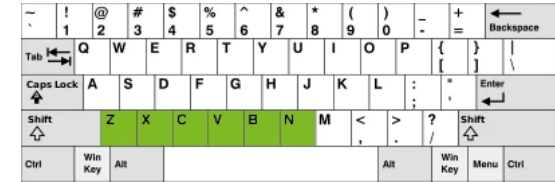
# Passwort Überprüfung

- Zeichenlänge prüfen
- Groß- /Kleinschreibung
- Sonderzeichen
- mind. 1 Buchstabe & 1 Zeichen
- Benutzereingaben überprüfen

**ALTERNATE**



# zxcvbn



- Low-Budget Password Strength-Estimation
- Entwickelt von Dropbox
- Berechnet:
  - Score zwischen 0 und 4
  - geschätzte Versuche, das Passwort zu knacken
  - Geschätzte Zeit zum knacken des Passworts

- Gängigen Passwörtern
- Gängige Nachnamen/Vornamen gemäß US-Volkszählungsdaten
- beliebte englische Wörter aus Wikipedia und US-Fernsehen und -Filmen
- gängige Muster
  - Datumsangaben
  - Wiederholungen (aaa)
  - Sequenzen (abcd)
  - Tastaturmuster (qwertyuiop)
  - l33t speak

# zxcvbn-ts

Realistic password strength estimation written in  
typescript

Get Started

Introduction

# Passwort Überprüfung

- Zeichenlänge prüfen
- Groß- /Kleinschreibung
- Sonderzeichen
- mind. 1 Buchstabe & 1 Zeichen
- Benutzereingaben überprüfen



# Zwei-Faktor-Authentifizierung



# Begriffserklärung

## Authentisierung

Benutzername

**kammer**

Passwort

\*\*\*\*\*



Nutzer erbringt  
Identitätsnachweis

## Authentifizierung



Prüfen des  
Identitätsnachweises

## Autorisierung



Schreibrecht



Leserecht



Ausführungsrecht



# Authentisierung

- Wie soll ein Nutzer seine Identität nachweisen ?

## Faktoren



Wissen



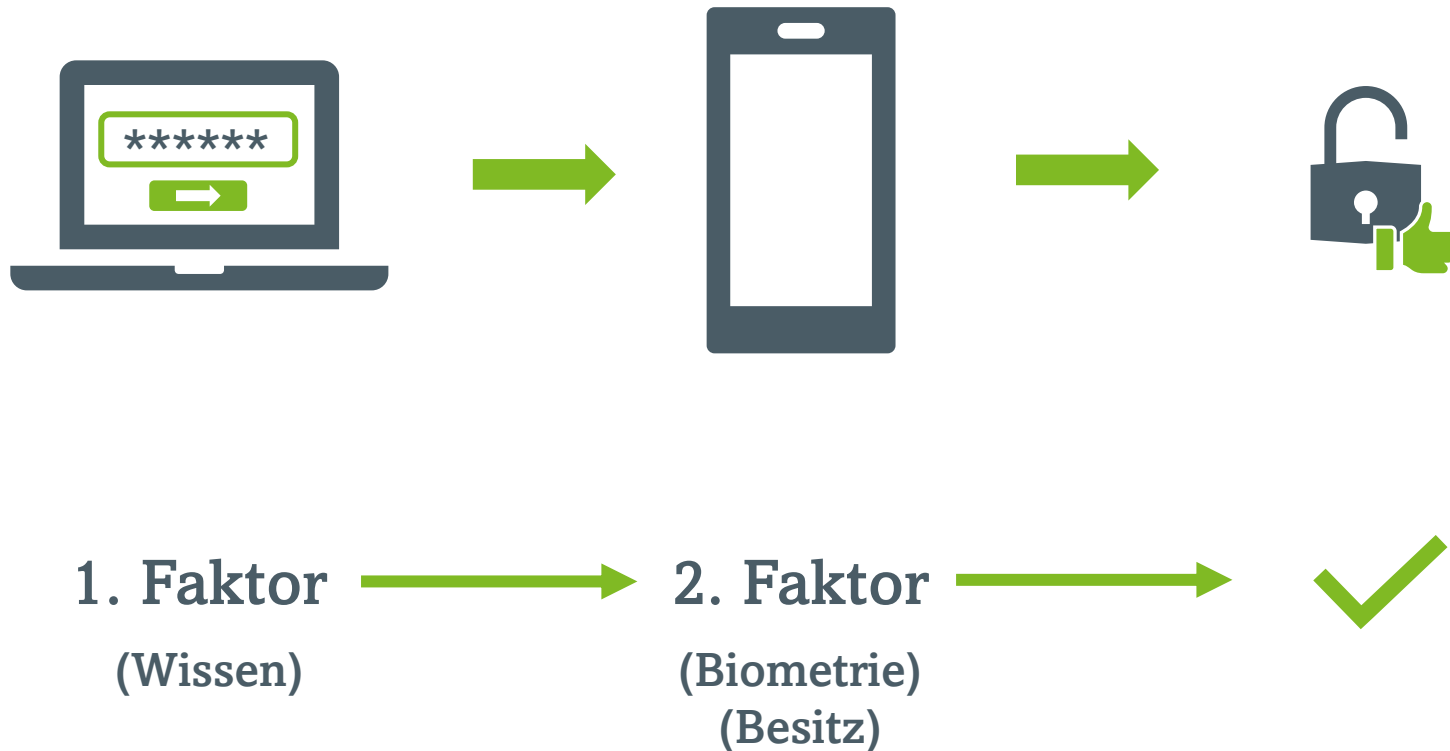
Biometrie



Besitz

➤ Zwei-Faktor-Authentisierung

# Funktionsweise



# Algorithmen

- **OTP - One Time Password**
- **HOTP - Hash-based One-Time Password**
- **TOTP - Time-based One-Time Password**
- **FIDO U2F - Fast Identity Online Universal Second Factor**

# TOTP - Time-based One-Time Password

