# Softwaresicherheit von Web-Applikationen am Beispiel von SQL Injection und XSS

Christopher Hübner, Lukas Martens, Niko Kösters

# Use Case: Nutzer möchte Bestellung einsehen

1. `SELECT * FROM Bestellung WHERE Benutzer = $id`
2. Guter Input => 32 => Reguläre Nutzer ID
3. Schlechter Input => ABCDE => Keine Nutzer ID
4. Böser Input => `23'; UPDATE Benutzer SET ROLE = "ROLE_ADMIN"--`
5. Böses Resultat:

```
SELECT * FROM Bestellung WHERE Benutzer = '23'; UPDATE Benutzer SET
ROLE = "ROLE_ADMIN"--
```


# Ausgangssituation

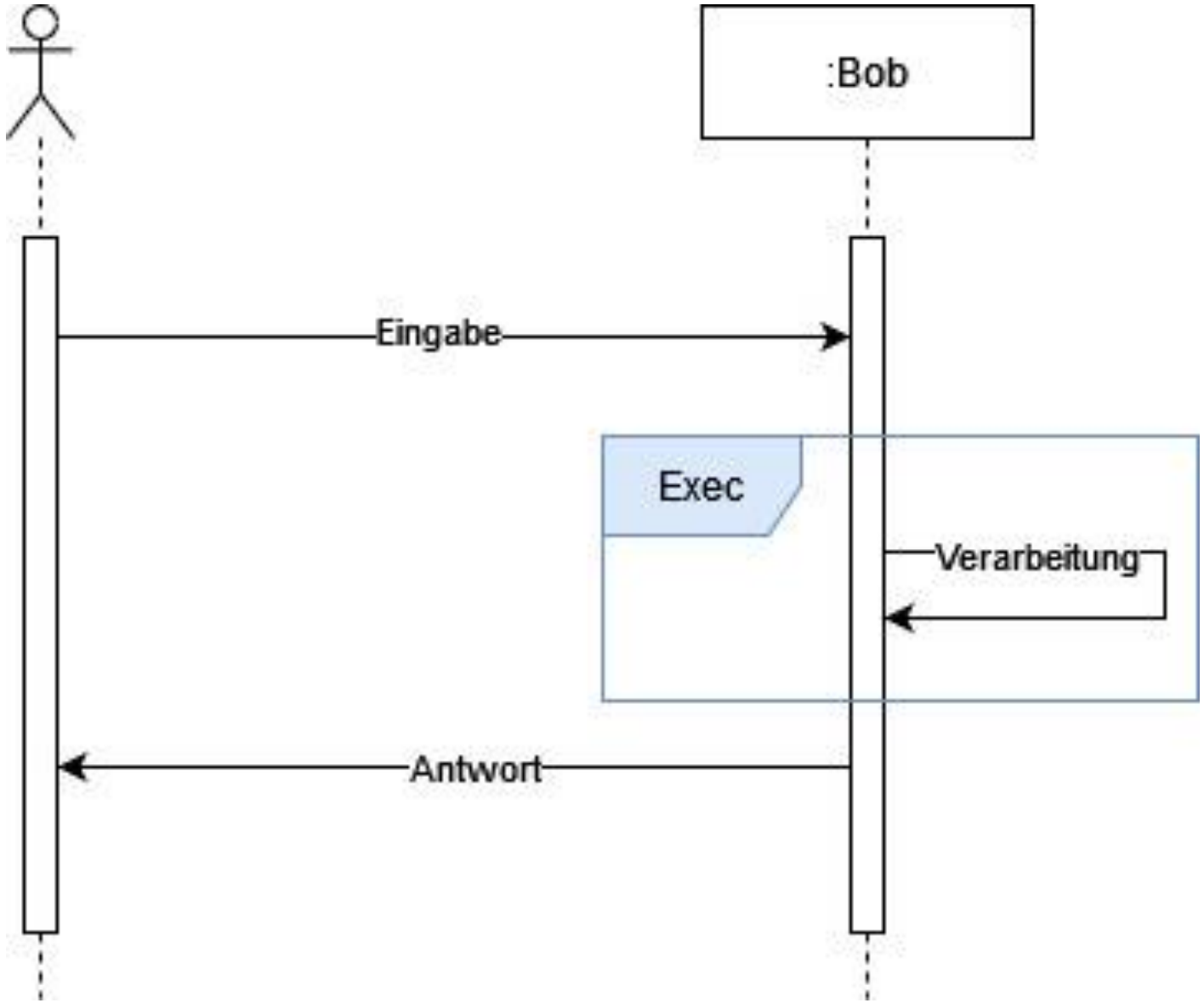
- Kunden anlegen.
- Kunden verwalten.
- Waren und Dienstleistungen anbieten.
- Käufe tätigen.
- Bestellungsprozessverwalten und kommunizieren.
- Und vieles mehr.


# Sehr an den Haaren herbeigezogenes Beispiel, aber sehr Idealtypisch!

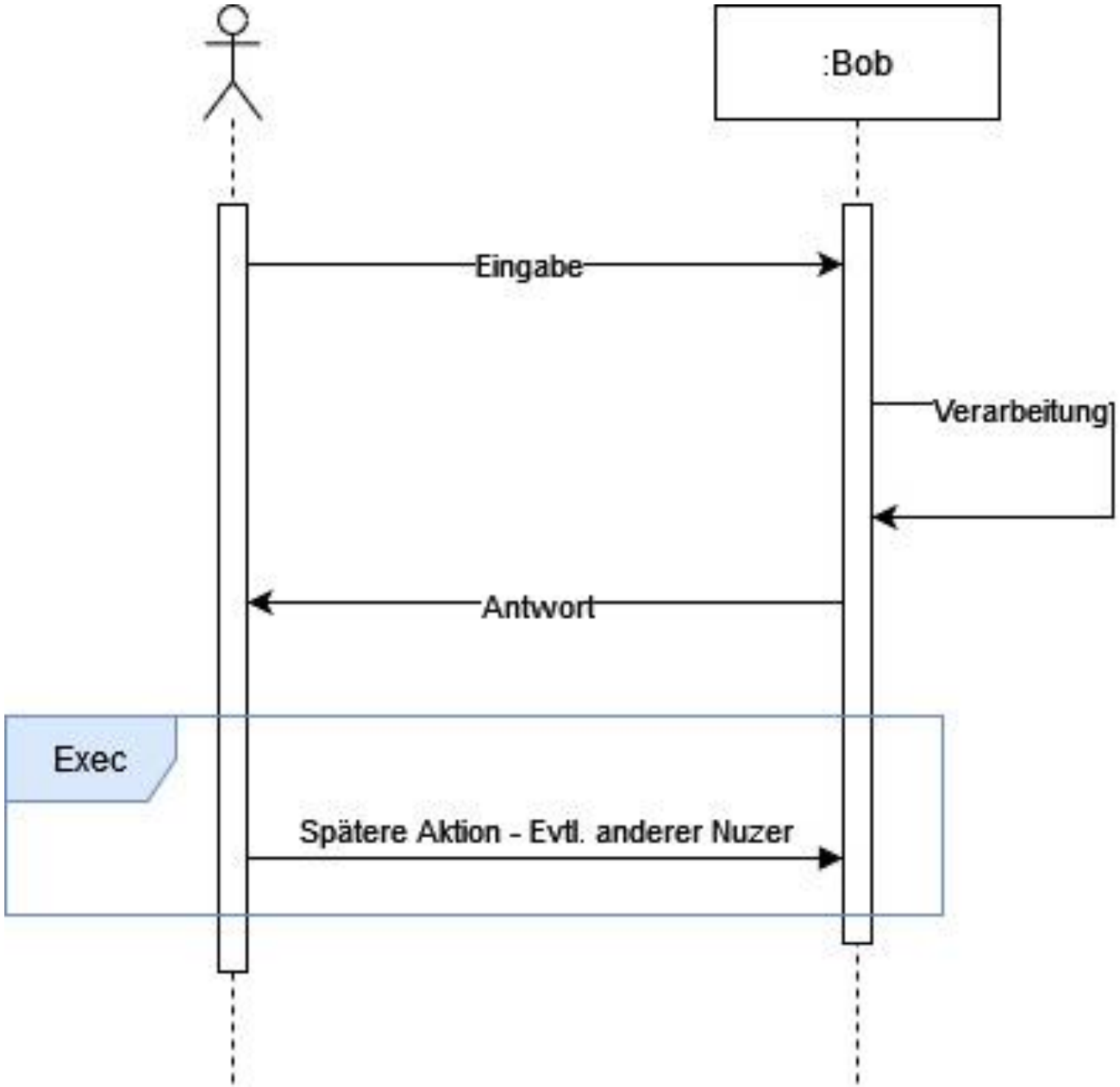
# Ziele

- Zuweisung von Nutzerrollen.
- Fälschung von Transaktionen.
- Umgehung der Authentifizierung und Authorisierungsmechanismen.
- Erlangen von Geschäftsgeheimnissen.
- Und vieles mehr.

# In Band SQL



# Out Band SQL


:Bob
Eingabe
Verarbeitung
Antwort
Exec
Spätere Aktion - Evtl. anderer Nuzer

# Vektoren und Gegenmaßnahmen

- SELECT, UPDATE, DELETE, INSERT, UNION
- Strings und Sonderzeichen
- Zahlen
- Filter

# Klassiker

- `SELECT * FROM Bestellung WHERE Benutzer = $id`
- `INSERT INTO Benutzer(name, role, hashedPW, ID) VALUES("Chris", "ROLE_ADMIN", hash, 99999)`
- `UPDATE Bestellung SET preis = 0 WHERE Benutzer.id = 9999 and Bestellung.id = 4`

# Union Select

```
SELECT * FROM Bestellung WHERE Benutzer = '123'
Union SELECT Benutzername, Passwort, ID FROM
Benutzer--
```

# Knackpunkte

- Das Union Select muss dieselbe Struktur liefern, wie das Original.
  - Datentypen müssen kompatibel sein.
  - Spaltenzahl muss identisch sein.
- Den das Resultat in der Struktur verändernde Unions, oder sonstige SQL-I sind nicht möglich.
  - Wenn das Resultat in der Struktur verfälscht ist, kann das Zielsystem diese nicht mehr verarbeiten.
  - Daher JOINs in der Regel kaum möglich.

# Datenbankstruktur ausspähen

1. Struktur der Datenbank ermitteln.
    1. Null kann in jeden DT umgewandelt werden.
    2. `= '123' Union SELECT NULL, NULL, NULL --`
    3. So kann die Anzahl der Spalten in einer Tabelle ermittelt werden.
2. Datentypen ermitteln.
    1. Suchen nach String.
    2. `= '123' Union SELECT "Hey", NULL, NULL --`
    3. Alle Datentypen und Spalten werden durchprobiert.

# Injection von Alice bis Bob

- Nachrichten müssen vom Client, auch beim Server und in der DB ankommen.
- Strings können beim HTTP-Transfer anders interpretiert werden.
- & und = müssen als %26, bzw. %3d kodiert werden.
- + muss als %2´b kodiert werden.
- ; muss als %3b kodiert werden.

# Verhindern von SQL Injection

- Validierung von Strings im Front und Backend
- Nur Zeichen erlauben, die nicht auf SQL Injection hindeuten.
- Prepared Statements

# Strategie: String in doppelte Anführungszeichen setzen

- Hintergrund: Escapte Anführungszeichen werden als Teil des Strings interpretiert.
- `''23'; UPDATE Benutzer SET ROLE = "ROLE_ADMIN"--''`

# XSS/Cross-Site-Scripting

- Das injizieren von Schadcode über die Nutzereingabe
- Ca. 90% der Webseiten verwenden JavaScript

Top 10 Web Application Security Risks (OWSP)

- 2017 stand XSS auf Platz 7 und Injektionen auf Platz 1
- 2021 wurden beide zusammengefasst und belegen Platz 3

# Mechanismen von XSS

- Wichtig für das Verständnis
  - HTML-Seitengenerierung
  - HTML Dynamisierung
  - Document Object Model (DOM)
  - Userinput Validierung?

# HTML-Seitengenerierung

- Struktur nach Tags
- Statisch

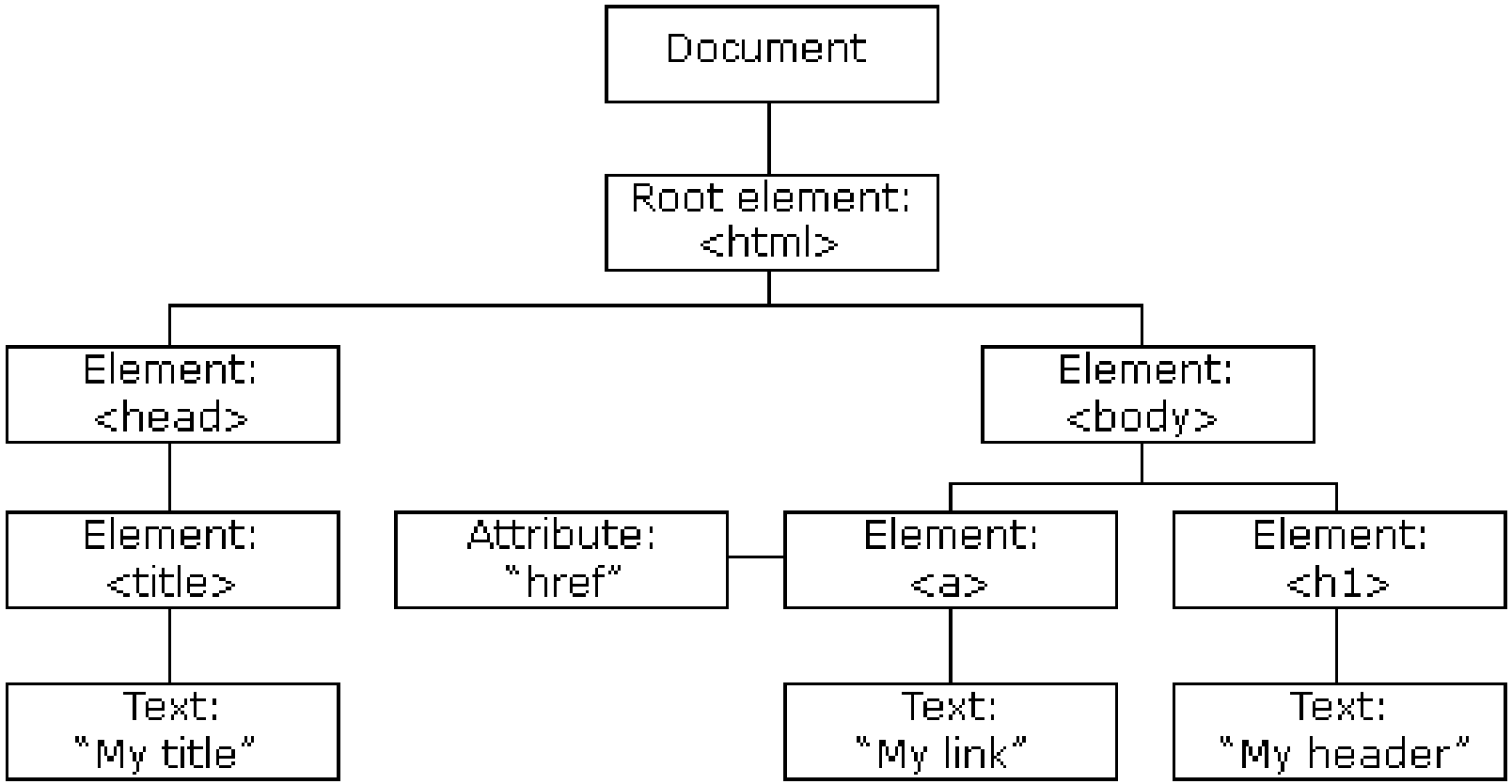
```
<html lang=„de“>
<head> … </head>
<body>
        <form name=„Beispiel“ method=„post“>
                <h1> Header </h>
                <input type=„text“ name=„Variable“ placeholder=„Default“>
        </form>
</body>
</html>
```

# HTML Dynamisierung

- Tools
  - JavaScript
  - Thymeleaf
  - Ajax

- Angezeigte Elemente aus Dateien/Inputs
- Üblich werden Inputs, welche Seitenübergreifend verwendet werden in der URL codiert

# Document Object Model




Quelle: https://www.w3schools.com/js/js_htmldom.asp

# Reflected XSS / Non-persistend XSS

- Schadcode/Payload wird für eine Anfrage verwendet
- Dem Opfer wird der Payload untergeschoben
- Der Payload wird vom Server auf das Opfer reflektiert
- Auf dem Server bleiben keine Rückstände des Payloads
- Der Payload ist in der Request und in der Response zu sehen

# Beispielwebseite

- Anzeigen einer Eingabe

```
<div>
    <h1>${eingabe}</h1>
</div>
```

- Eingabe: Test123
- http://beispiel.de/profil?eingabe=Test123

# Bösartige Eingabe

Eingabe: <script>alert(document.cookie)</script>

http://example.com/profile?eingabe=<script>alert(document.cookies)</script>

```
<div>
    <h1><script>alert(document.cookie)</script></h1>
</div>
```

- Ausgabe: Alertfenster mit Cookie des Users

# Stored XSS / Persistend XSS

- Der Server Speichert Nutzerinput auf irgendeine Art
- Schadcode/Payload bleibt über längere Zeit aktiv
- Der User besucht die Webseite und wird zum Opfer
- Der Payload wird auf dem Server abgelegt
- Der Payload ist in der in der Response zu sehen

# DOM-based XSS

- Schadcode/Payload wird für eine Anfrage verwendet
- Dem Opfer wird der Payload untergeschoben
- Es werden URL fragments(#) verwendet
- Auf dem Server bleiben keine Rückstände des Payloads
- Der Payload ist in der Request und in der Response **NICHT** zu sehen

# Auswirkungen von XSS-Angriffen

- Datendiebstahl
- Schadcode-Verteilung
- Große Angriffskampagnen

# Auswirkungen - Datendiebstahl

- Direkte Übermittlung von Cookies oder Session Tokens
- Preisgabe privater Daten
- Eindringen in die Privatsphäre
- Identitätsdiebstahl

# Auswirkungen – Schadcode-Verteilung

- Tausch von legitimen Links gegen schädliche Links
- „Drive-by-Downloads“: Automatisches Herunterladen schädlicher Dateien.

# Auswirkungen – Große Angriffskampagnen

- Potenzielle Bedrohungen für kritische Infrastrukturen
- Lebensbedrohliche Auswirkungen bei großflächigen Angriffen

# Präventionstechniken

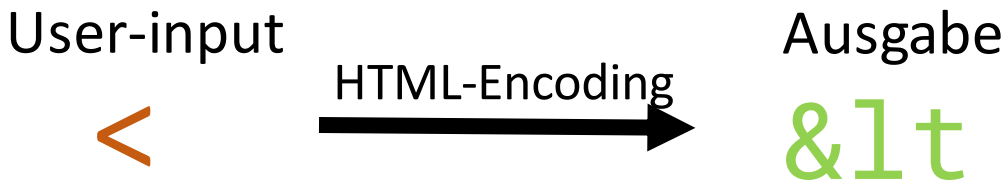
- Eingaben Validieren
- Ausgabe Encodierung
- Sicherheits-Header
- Sichere Kodierungspraktiken

# Präventionstechniken – Eingaben Validieren

- Sicherstellung der Überprüfung und Bereinigung von Nutzereingaben
- „Sanitizing“
- Unterschied zwischen Server- und Client-Validierung

# Präventionstechniken – Ausgabe Encodierung

- Umwandlung von Nutzereingaben in sichere Formate
- Verhindert die Interpretation spezieller Zeichen als HTML-Code oder Skript-Code



# Präventionstechniken – Sicherheits-Header

- Implementierung von HTTP-Sicherheitsheadern wie CSP
- Beschränkung der Quellen für das Laden von Inhalten

# Präventionstechniken – Sichere Kodierungspraktiken

- Vorsichtiger Umgang mit anfälligen HTML-Elementen und –Attributen
- Strenge Validierungs- und Bereinigungsmaßnahmen

# Tools und Testen

- Automatisierte Tools zur XSS-Detection
- Pen-Tests

# Tools und Testen – Burp Suite

- Umfassende Sicherheitstestsuite
- Benutzerfreundliches Interface für automatisierten und manuelle scans
- Auffinden von mehreren Schwachstellen
- Detaillierte Schwachstellen Berichte
- Verfügerbar in einer kostenlosen und einer professionellen Version

# Tools und Testen – OWASP ZAP

- Open-Source-Sicherheitstool
- automatisch Sicherheitsschwachstellen finden und testen
- Detaillierte Schwachstellen Berichte
- agiert als Man-in-the-Middle-Proxy


Quelle: https://www.zaproxy.org/getting-started/

# Tools und Testen – Pen-Tests

- Überprüfung durch IT-Security-Experten
- Kombination aus automatisierten und manuellen Tests

# Häufige Fehler

- Keine oder unzureichende Inputvalidierung
- Unveränderte Wiedergabe des Userinputs
- Eingabefehler unbeschränkt lassen
- Zero-Trust-Ansatz

# XSS Challenges

Hauptprobleme:

- Webseiten sind nicht einheitlich
- Libraries werden nicht aktuell gehalten
- Stetige Veränderung von Webseiten
- Abhängigkeiten sind weit verkettet

# XSS Trends

- Kaum große Änderungen über Zeit
- Sicherheitsregelungen einhalten
- KI zur Erkennung von Payloads

# Fazit und Ausblick - XSS

- XSS-Schutz erfordert fortwährende Aufmerksamkeit und Innovation.
- Einsatz von KI zur Verbesserung der XSS-Erkennung und -Abwehr.
- Es muss mit den neuesten Webtechnologien und Sicherheitsbedrohungen Schritt zu halten.
- Gemeinschaftliche Anstrengungen in Ausbildung und Zusammenarbeit für effektiveren Schutz.

# Vielen Dank für Ihre Aufmerksamkeit