Juventus Technikerschule HF

Hacking Exposed

# #5 Websecurity I

Pascal Knecht

Video 0: Überblick

# Wichtiger Hinweis

- Dies ist eine Lehrveranstaltung.
- Die im Rahmen der Hacking-Exposed-Vorlesung vermittelten Kenntnisse sollen dazu beitragen, dass Sie Informationssicherheitsaspekte beachten und in Ihren Projekten berücksichtigen.
- Die HE-Vorlesung ist keineswegs als Anstiftung zum Hacken zu verstehen.

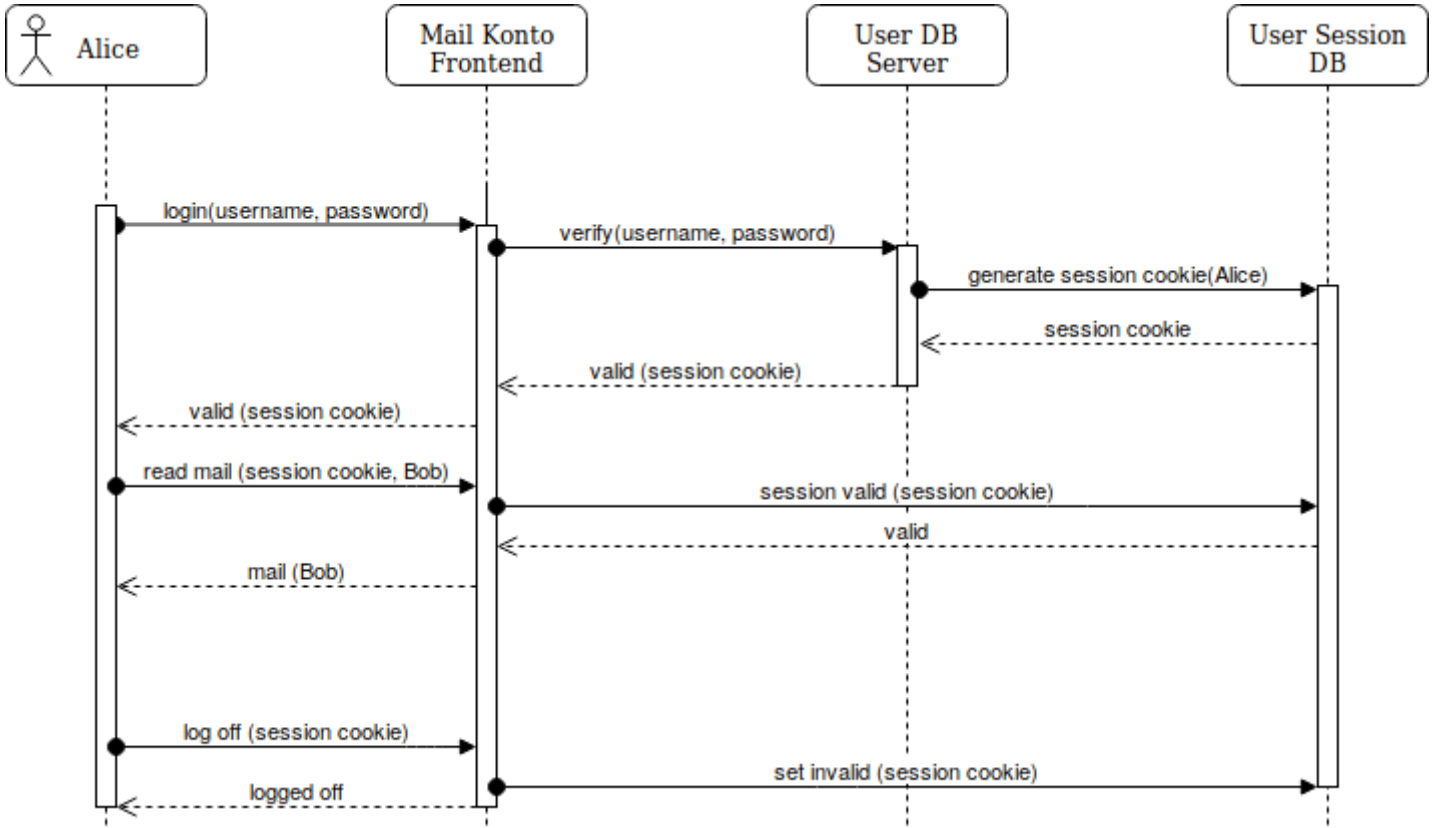
# Inhalt heute Abend

- Same Origin Policy (SOP)
- Content Security Policy (CSP)
- Cross Site Scripting (XSS)

# Ziele

- Sie wissen, was eine HTTP Session ist und wie sie aufgebaut ist.
- Sie kennen grundlegende HTTP Security Massnahmen wie Same Origin Policy (SOP) und Content Security Policy (CSP).
- Sie kennen die Funktionsweise und die Voraussetzungen für die Websecurity Attacken XSS .
- Sie sind in der Lage, einfache Attacken im Übungslabor auszuführen und verstehen ihre Funktionsweisen.
- Sie können mit dem Analyse Tool Burp Suite Webservices untersuchen.

# HTTP Sessions



# #01 SOP

## Same Origin Policy

Video 1: SOP

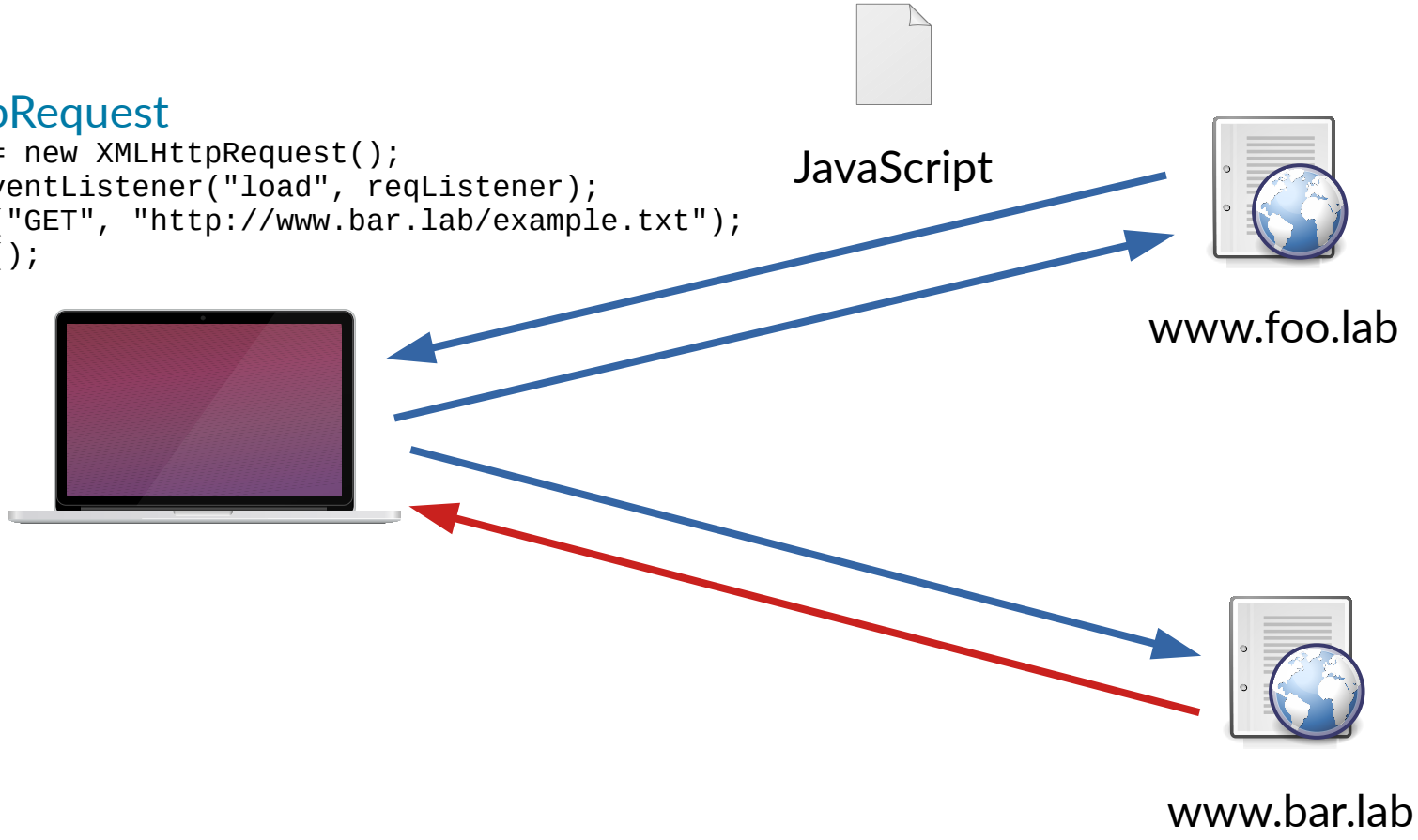
# Same Origin Policy

- Regelt die Interaktionsmöglichkeit von bspw. clientseitigen Skripten mit Webservern anderer «origin»
- Browser schränkt Netzwerk-Funktionalität für Skripte ein:
  - Ein XMLHttpRequest oder <img> kann nur Daten vom selben (origin) Server laden
  - Browser unterbindet Zugriff eines Skriptes auf Drittsysteme.
- Zwei URLs haben dieselbe Herkunft/Origin wenn:
  - Schema / Host / Port übereinstimmen
- Same Origin Policy im Detail bei MDN
- Wer das trotzdem nutzen möchte, verwendet CORS

# SOP Übersicht

## XMLHttpRequest

```
var oReq = new XMLHttpRequest();
oReq.addEventListener("load", reqListener);
oReq.open("GET", "http://www.bar.lab/example.txt");
oReq.send();
```



# SOP

- Aktive Komponente werden durch SOP eingeschränkt:
  - JavaScript
  - ActiveX
  - Flash
  - Java Applets

# SOP Übung

- Die Webseite https://www.juventus.ch/public/app.js wird vom Browser heruntergeladen und interpretiert. Welche der folgenden Requests würde die SOP zulassen?
  - https://www.juventus.ch/data/news.json
  - http://www.juventus.ch/static/overview.png
  - https://www.juventus.ch:8443/data/news.json
  - http://www.juventus.ch:8443/data/news.json
  - https://www.juventus.ch/static/logo.png
  - https://www2.juventus.ch/data/news.json

# #02 CSP

## Content Security Policy, die Browser Firewall

Video 2: CSP

# Content Security Policy (CSP)

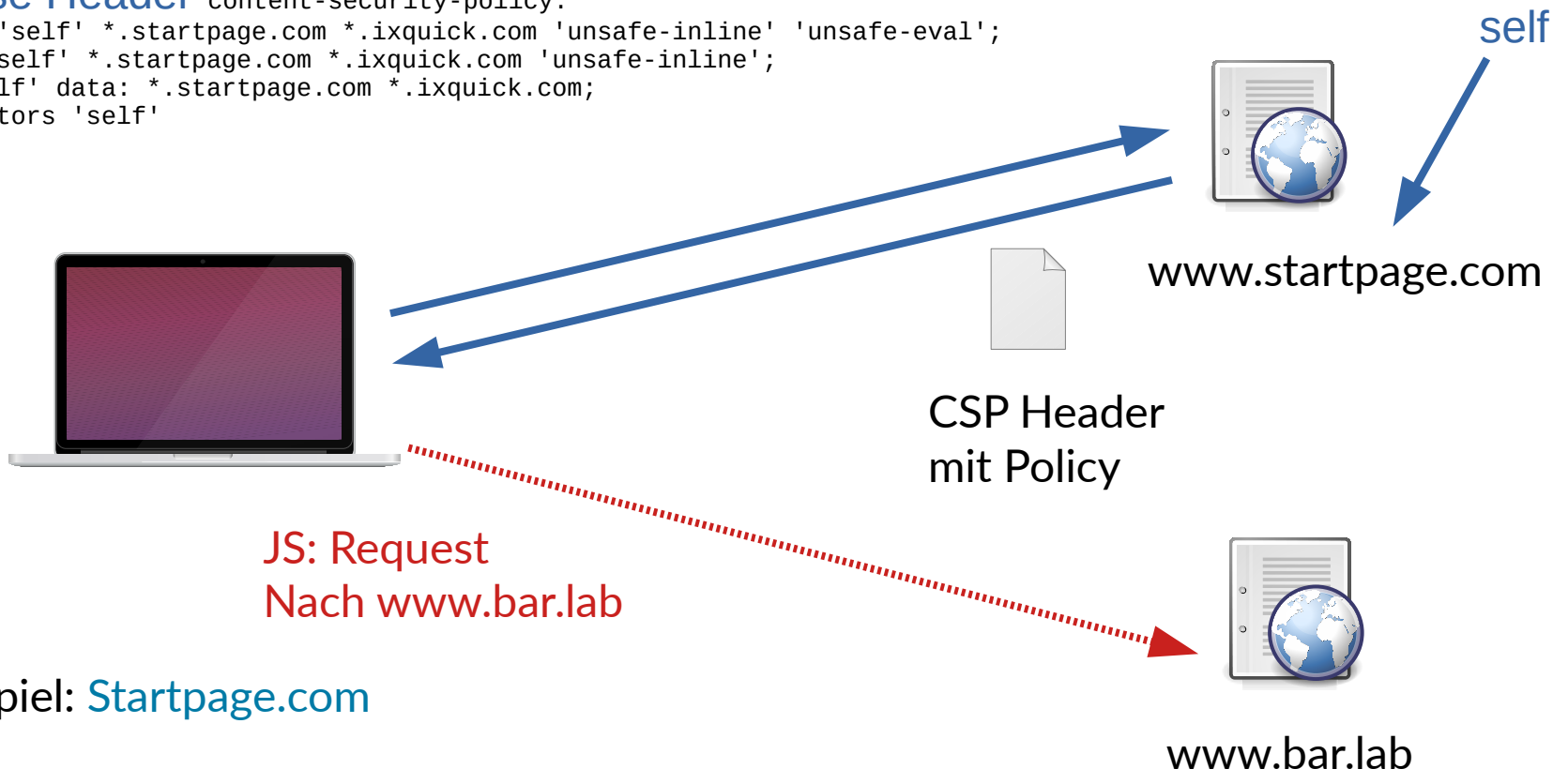
- Schutz vor XSS durch Einschränkung der Bezugsquellen → Browser Firewall.
- Anweisung an Browser: HTTP Header Response (nicht persistent)

```
Content-Security-Policy: default-src 'self'
```

- Weitere mögliche Anweisungen:
  - `img-src, media-src, object-src, script-src, style-src ...`
- Content Security Policy im Detail bei MDN

# CSP Überblick

Response Header
```
content-security-policy:
script-src 'self' *.startpage.com *.ixquick.com 'unsafe-inline' 'unsafe-eval';
style-src 'self' *.startpage.com *.ixquick.com 'unsafe-inline';
img-src 'self' data: *.startpage.com *.ixquick.com;
frame-ancestors 'self'
```




Beispiel: Startpage.com

# CSP Übung

- Von wo dürfen Inhalte geladen werden?
```
Content-Security-Policy: default-src 'self'
*.trusted.com
```

- Von wo dürfen Media-Dateien geladen werden?
```
Content-Security-Policy: default-src 'self'; img-src
*; media-src media1.com media2.com; script-src
userscripts.example.com
```

# Burp Suite

PORTSWIGGER WEB SECURITY

- Clientseitiger Forward Proxy zur Analyse von Webapplikationen.
- Wird per SOCKS im Browser konfiguriert.
- TLS-Scanning ist möglich, CA muss importiert werden.
- Intercept Modus ermöglicht jeden Request einzeln zu versenden und vorgängig zu editieren.

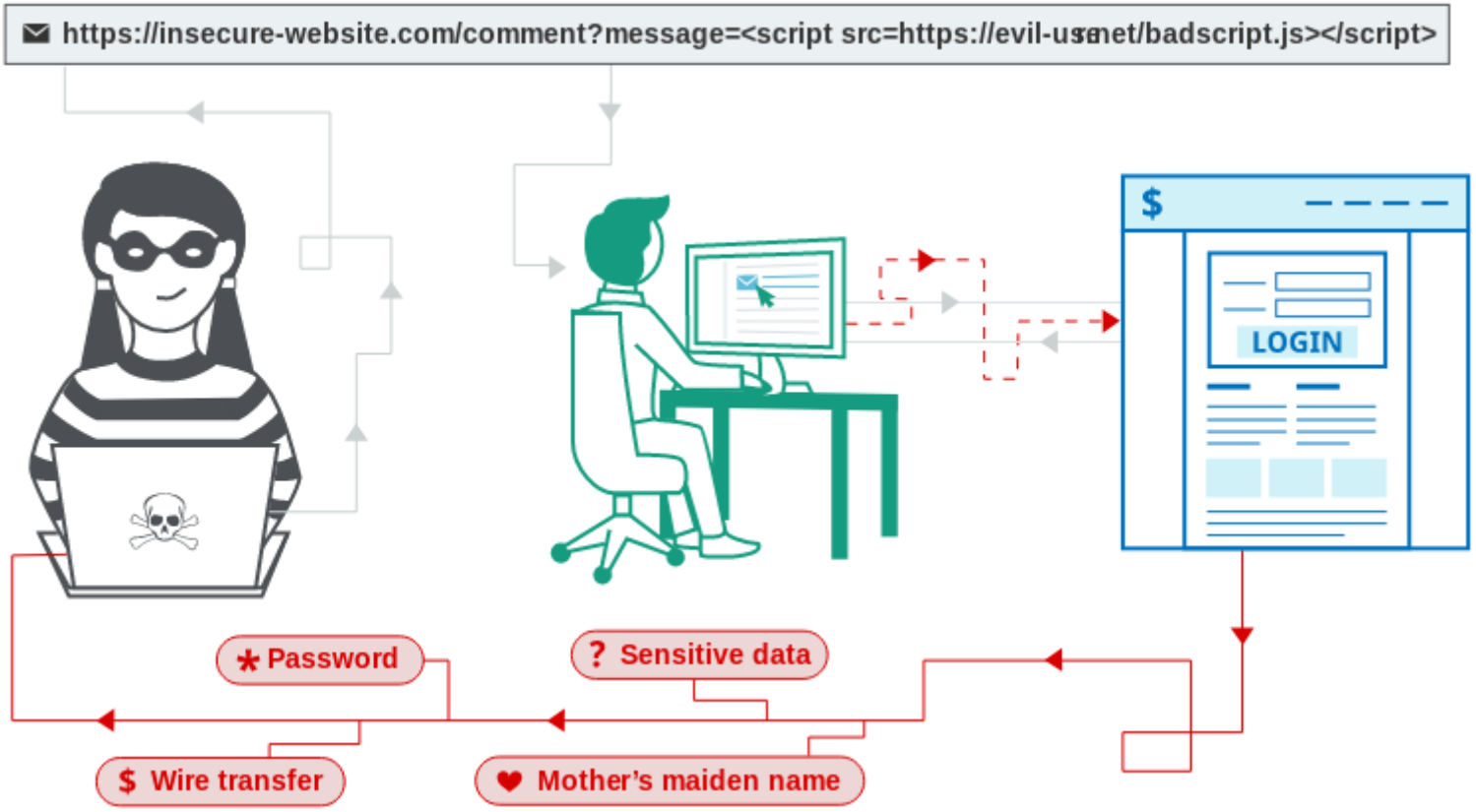
Video BurpSuite Einführung

# #03 XSS

## Cross Site Scripting

Video 3: XSS

# XSS im Überblick




https://portswigger.net/web-security/cross-site-scripting

# Variationen von XSS-Attacken

- Reflected XSS
  - Bösartiges Skript kommt vom aktuellen HTTP Request
- Stored XSS
  - Bösartiges Skript kommt von der Websiten DB
- DOM-based XSS
  - Schwachstelle liegt im Client-Side Code

# Reflected XSS

- Einfachste Variante aller XSS-Attacken:
  - Request: `https://insecure-website.com/status?message=All+is+well`
  - Website rendert: `<p>Status: All is well.</p>`
- Lässt sich mit JavaScript ausnutzen:
  - `https://insecure-website.com/status?message=<script>/*+Bad+stuff+here...+*/</script>`
  - `<p>Status: <script>/* Bad stuff here... */</script></p>`

Video XSS Reflected Demo

# Stored XSS

- Persistenz einer XSS-Attacke bspw. in einem Foren-Post
  - Es findet keine Prüfung der Usereingaben statt
  - Usereingabe wird 1:1 in Webseite gerendert

Video XSS Stored Demo

# DOM-based XSS

- Im Selbststudium erarbeiten:
  https://portswigger.net/web-security/cross-site-scripting#dom-based-cross-site-scripting

# Für was kann XSS verwendet werden?

- Versetzt Angreifer in die Lage
  - Sich als Opfer auszugeben/tarnen
  - Jede Interaktion auszuführen wie es das Opfer kann
  - Zugriff auf alle Daten die das Opfer sehen kann
  - Login Credentials des Users auslesen
  - Darstellung der Webseite verändern (temporär)
  - Bösartigen Code in Webseite persistent einzufügen

# XSS finden und testen

- Reflected- & Stored-XSS
  - Jedes Eingabefeld individuell befüllen und prüfen ob Antwort als gerendertes HTML zurück kommt
  - Jeden einzelnen Eingabeort individuell prüfen
    - `<script>alert("XSS vuln");</script>`

# XSS verhindern

- Eingaben filtern
- Ausgabe codieren
- Korrekte Response Headers setzen
- Content Security Policy (CSP)

# Übungen & Labor

Übungen: HE5

Labor: github.com/ryru/HackingExposed

# Empfehlungen fürs Selbststudium

- OWASP Top 10 Edition 2017 – Latest & Greatest (46 Min)
- CSRF, the Intranet and You Causes, Attacks and Countermeasures (61 Min)
- HTTP Security & Headers (54 Min)

# CC BY-SA 4.0