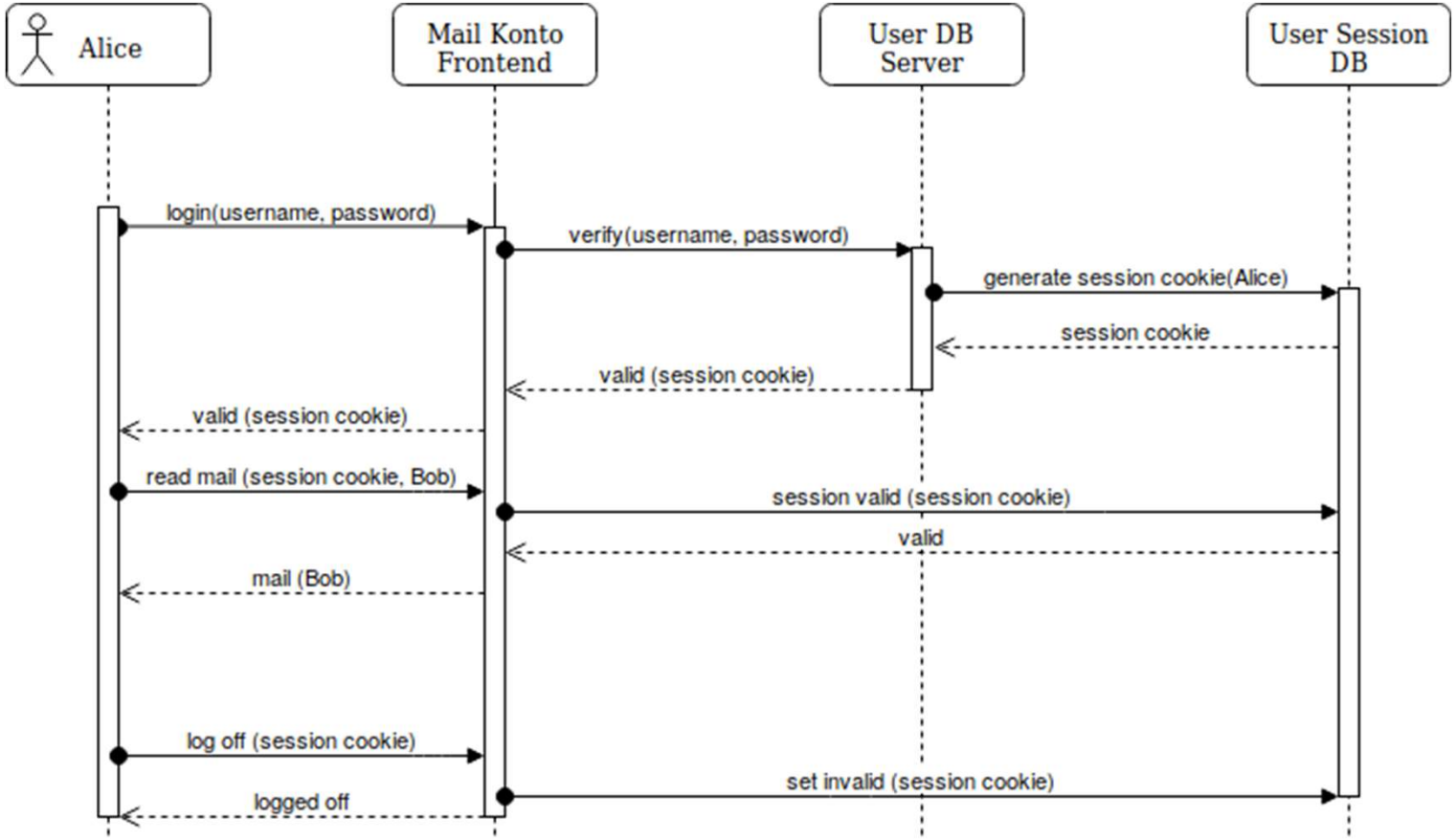
Inhalt heute

- 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



HTTP Sessions

E-Mails lesen:

1.Alice verwendet das Sitzungs-Cookie, um **read mail (session cookie, Bob)** vom **Mail Konto Frontend** abzurufen.
2.Das **Mail Konto Frontend** überprüft die Gültigkeit der Sitzung, indem es eine **session valid (session cookie)-**Anfrage an die **User Session DB** sendet.
3.Die **User Session DB** bestätigt die Gültigkeit der Sitzung und sendet eine **valid**-Antwort an das **Mail Konto Frontend**.
4.Das **Mail Konto Frontend** ruft die **mail (Bob)** ab und sendet sie an Alice.

Abmeldevorgang:

1.Alice sendet eine **log off (session cookie)-**Anfrage an das **Mail Konto Frontend**.
2.Das **Mail Konto Frontend** setzt die Sitzung auf ungültig, indem es eine **set invalid (session cookie)-**Anfrage an die **User Session DB** sendet.
3.Das **Mail Konto Frontend** bestätigt die Abmeldung, indem es eine **logged off-**Antwort an Alice sendet.

#01 SOP

Same Origin Policy

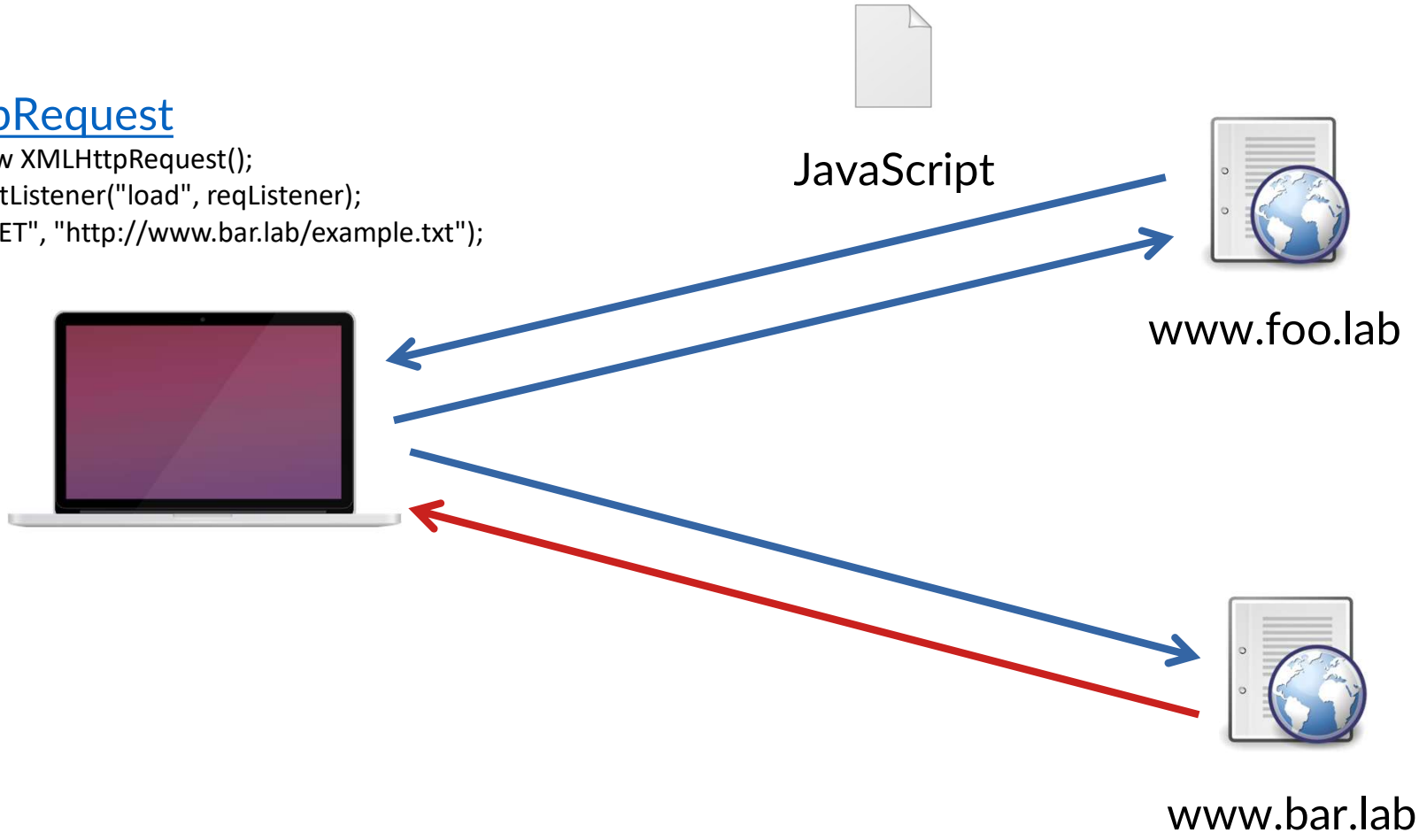
Same Origin Policy

- 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

Juventus
Technikerschule HF

HTTP Header

Rolle und Funktion von HTTP-Headern

- HTTP-Header sind wichtige Bestandteile von HTTP-Anfragen und -Antworten. Sie transportieren Metadaten über die HTTP-Transaktionen und können Anweisungen sowohl an den Server als auch an den Browser enthalten

- Wenn ein Browser eine HTTP-Antwort mit einem CSP-Header empfängt, interpretiert und befolgt er die darin enthaltenen Richtlinien. Dies hilft, die Sicherheit der geladenen Webseite zu erhöhen.

- HTTP/1.1 200 OK

- Content-Type: text/html

- Content-Security-Policy: default-src 'self'; img-src 'self' https://trusted.images.com; script-src 'self' https://trusted.scripts.com

Content Security Policy (CSP)

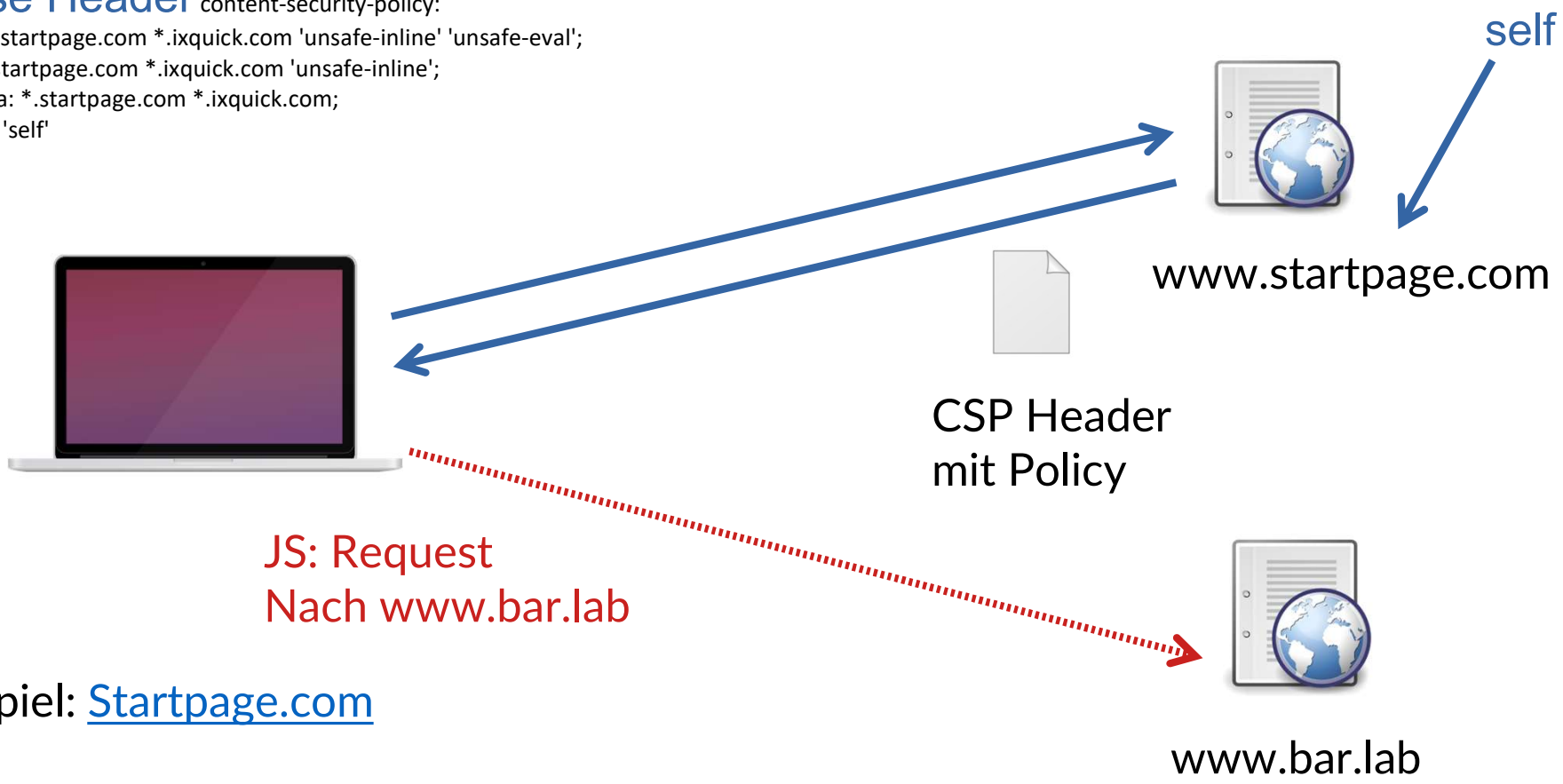
- Schutz vor XSS durch Einschränkung der Bezugsquellen → Browser Firewall.
- Anweisung an Browser: HTTP Header Response (nicht persistent)

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 Überblick

Zusammenfassung

•**Content Security Policy (CSP)**: Ein Sicherheitsmechanismus, der einschränkt, welche Quellen für verschiedene Arten von Inhalten auf einer Webseite erlaubt sind.

•**Browser-Interpretation**: Der Browser beachtet die CSP-Richtlinien und blockiert Inhalte von nicht erlaubten Quellen, um die Sicherheit der Webseite zu gewährleisten.

•**Erlaubte und nicht erlaubte Anfragen**: Das Bild zeigt, wie der Browser erlaubte Anfragen (blaue Pfeile) zulässt und nicht erlaubte Anfragen (rote Pfeile) blockiert. CSP hilft dabei, die Sicherheit der Webseite zu erhöhen, indem es verhindert, dass schädliche Inhalte von nicht vertrauenswürdigen Quellen geladen werden.

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

- 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.

Burp Suite

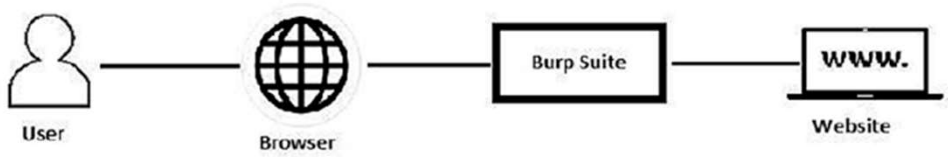


Figure 2-5. *A user accessing a website with Burp Suite*

Referring to the image above, at a very high level and in simple terms, the following sequence of events happens:

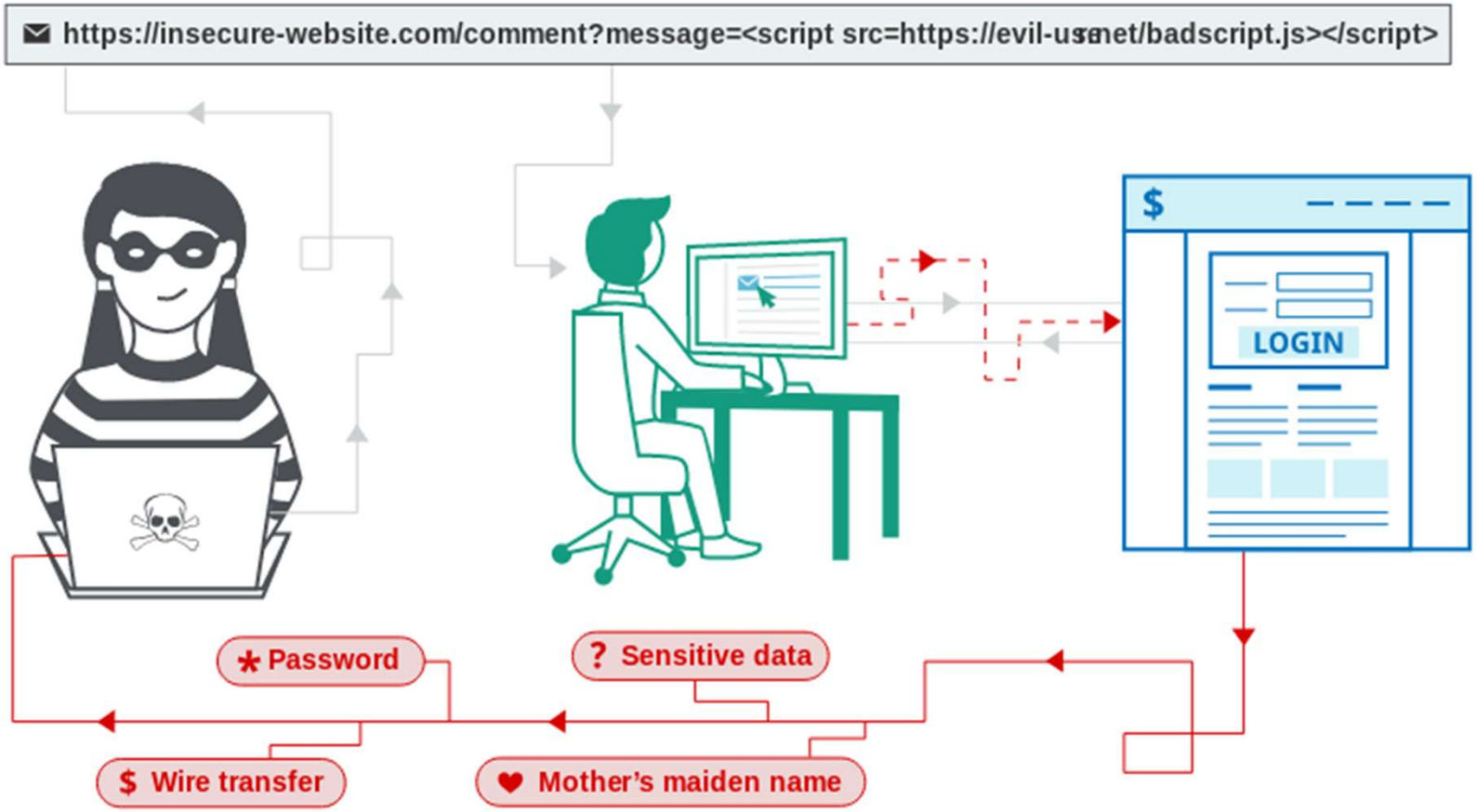
1. The end user opens up any browser of choice.
2. The user then enters the URL of the website he/she wishes to browse.
3. The browser redirects the request to Burp Suite, which then forwards the request to the target website.
4. The target website responds to the request and sends a response back to Burp Suite, which then passes on the response to be rendered in the browser.

#03 XSS

Cross Site Scripting

Juventus
Technikerschule HF

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: Status: All is well.
- Lässt sich mit JavaScript ausnutzen:
  - https://insecure-website.com/status?message=<script>/*+Bad+stuff+here...+*/</script>
  - Status: <script>/* Bad stuff here... */</script>

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

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: 5

Labor: https://github.com/hexposed/Lab

Empfehlungen fürs Selbststudium

- https://media.ccc.de/v/gpn17-8598-owasp_top_10
Edition 2017 – Latest & Greatest (46 Min)

- https://media.ccc.de/v/23C3-1560-en-csr
Causes, Attacks and Countermeasures (61 Min)

- https://media.ccc.de/v/gpn18-141-http-security-headers#t=1131