



DHBW

Heidenheim

Cross-Site-Request-Forgery (CSRF)

Paul Brilmayer, Max Löhr & Stefan Moser

www.heidenheim.dhbw.de



Agenda

- Woher kommt CSRF?
 - Was ist CSRF?
 - Wie funktioniert CSRF?
 - Varianten von CSRF
 - Wie kann CSRF verhindert werden?
 - Code Beispiel
-

Woher kommt CSRF?

- 1988 – Paper von Norm Hardy
- 2001 – Definiert von Peter Watkins





Beispiele aus der echten Welt

Samy-Wurm – MySpace 2005:

- Kombination aus XSS und CSRF
- User besuchten Profil von Account 0
- Schädlicher JS-Code in “Über mich” Feld
 - -> wurde beim Laden des Profils ausgeführt
- Code konnte Aktionen im Namen des eigenen Profils ausführen
 - Darüber konnten Freunde zum eigenen Profil hinzugefügt werden
 - Aktualisierung des eigenen Profiles
 - -> Automatische weiterverbreitung

=> War kein bösartiger Angriff



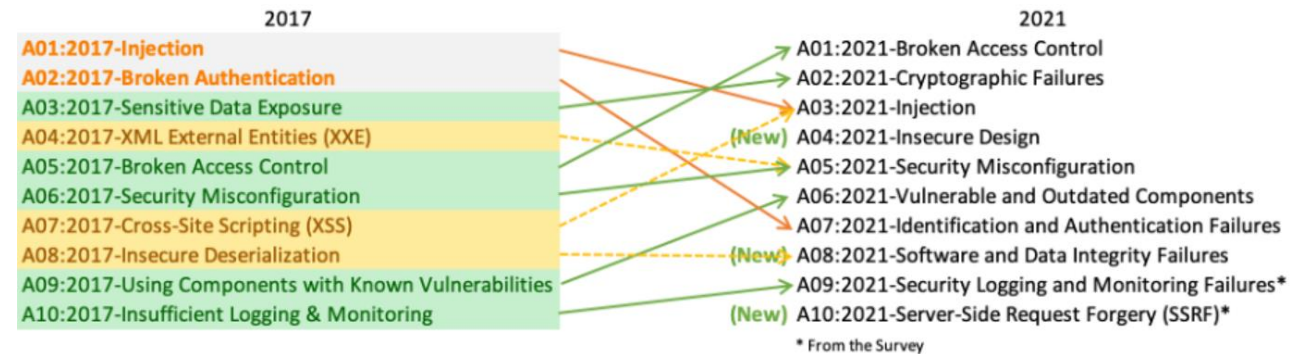
Beispiele aus der echten Welt

Facebook-Like-Betrug 2010:

- Angreifer erstellten gefälschte Websites / Angebote
 - Gratis Prämien, wenn man auf bestimmte Aktionen macht -> Link
 - Nach Klick auf Link -> CSRF Angriff
 - Automatisch “Gefällt mir”-Aktion auf Angreifer-Facebookseite
 - -> Popularität der Seiten künstlich steigern
 - Gefälschte Likes für legitimes Image
 - -> Scamseiten sehen vertrauenswürdig aus
 - => Mehr Leute klicken auf Links
 - -> Scamseiten konnten Werbung verkaufen
-

Was ist CSRF?

- Top 10 OWASP Schwachstelle



Factors

CWEs Mapped	Max Incidence Rate	Avg Incidence Rate	Avg Weighted Exploit	Avg Weighted Impact	Max Coverage	Avg Coverage	Total Occurrences
33	19.09%	3.37%	7.25	7.15	94.04%	47.90%	274,228

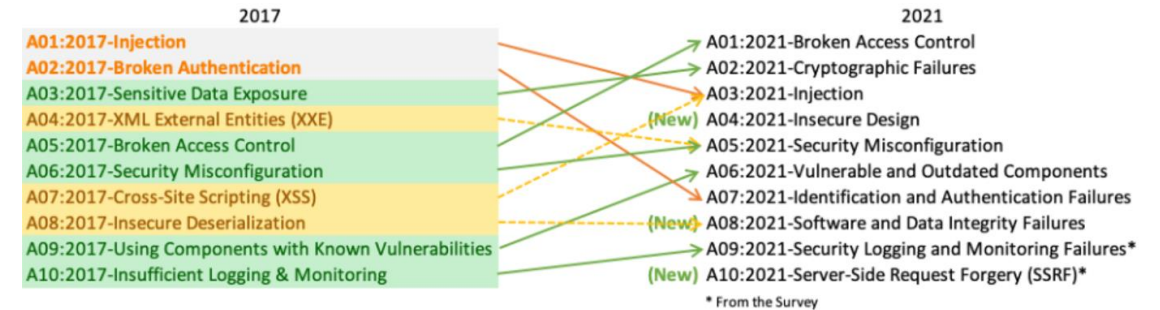


Was ist OWASP?

- Open Web Application Security Project
 - OWASP Cheat Sheets
 - OWASP Tools
 - OWASP Trainings
 - OWASP Konferenzen
-

Was ist OWASP?

- Broken Access Control
- Cryptographic Failures
- Injection
- Insecure Design
- Security Misconfiguration
- Vulnerable and Outdated Components
- Identification and Authentication Failures
- Software and Data Integrity Failures
- Security Logging and Monitoring Failures
- Server-Side Request Forgery



Was ist CSRF?

“Cross-site Request Forgery is a vulnerability in a website that allows attackers to force victims to perform security-sensitive actions on that site without their knowledge.”

<https://owasp.org/www-community/attacks/csrf>

Was ist CSRF?

“Cross-site Request Forgery is a **vulnerability in a website** that allows attackers to force victims to perform security-sensitive actions on that site without their knowledge.”

<https://owasp.org/www-community/attacks/csrf>

Was ist CSRF?

“Cross-site Request Forgery is a vulnerability in a website that allows attackers to **force victims to perform security-sensitive actions** on that site without their knowledge.”

<https://owasp.org/www-community/attacks/csrf>

Was ist CSRF?

“Cross-site Request Forgery is a vulnerability in a website that allows attackers to force victims to perform security-sensitive actions on that site **without their knowledge.**”

<https://owasp.org/www-community/attacks/csrf>



Wie funktioniert CSRF?

- Website identifizieren, die anfällig für CSRF-Angriffe ist
- API Call herausfinden
- z.B. Website mit Bild, welches geladen werden soll

```
<Image src="{\"http://guteWebsite.com/transfer/sendTo=meinAccount&amount=400\"} />
```



Perpetrator

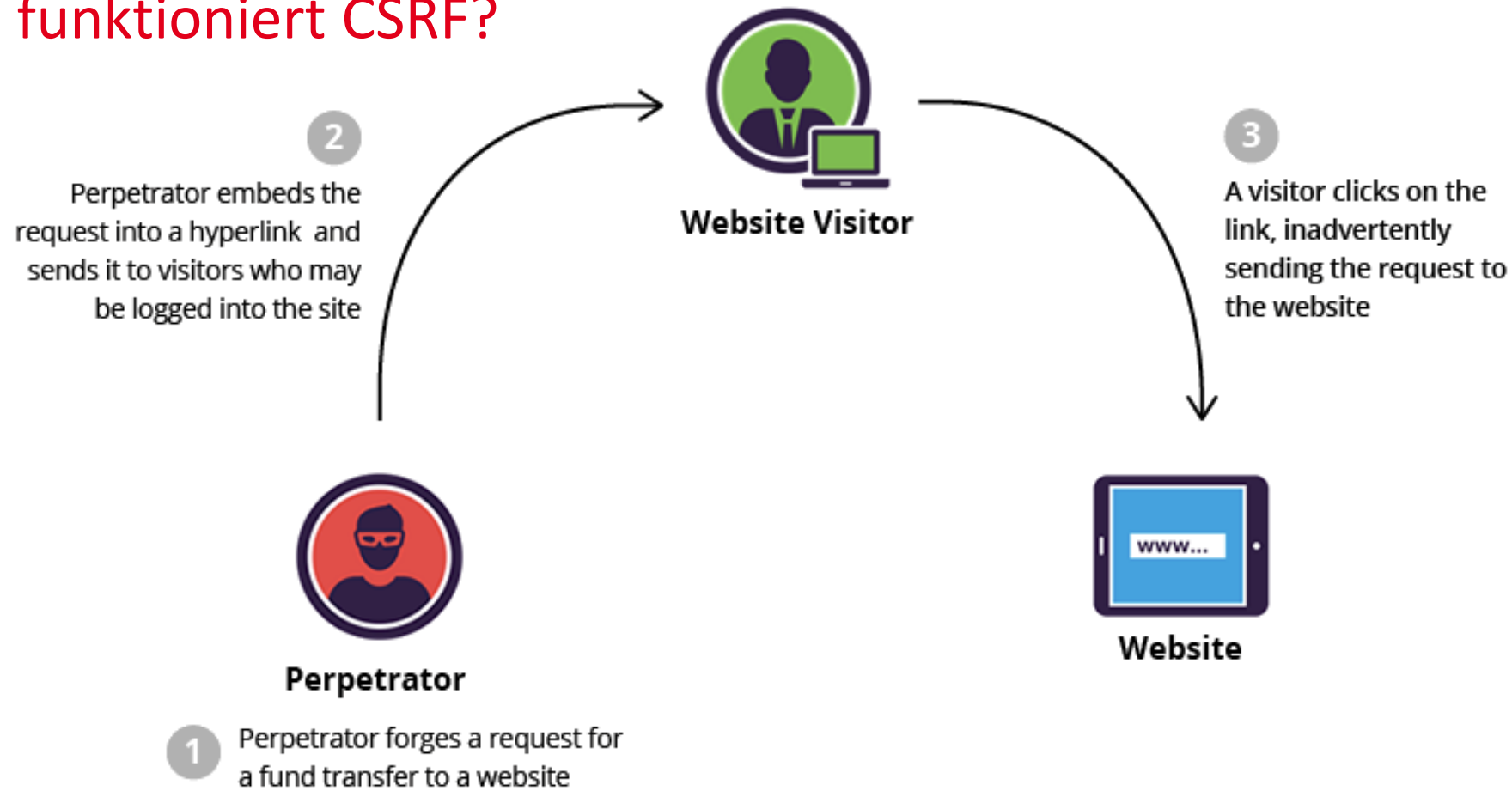
- 1 Perpetrator forges a request for a fund transfer to a website

Wie funktioniert CSRF?

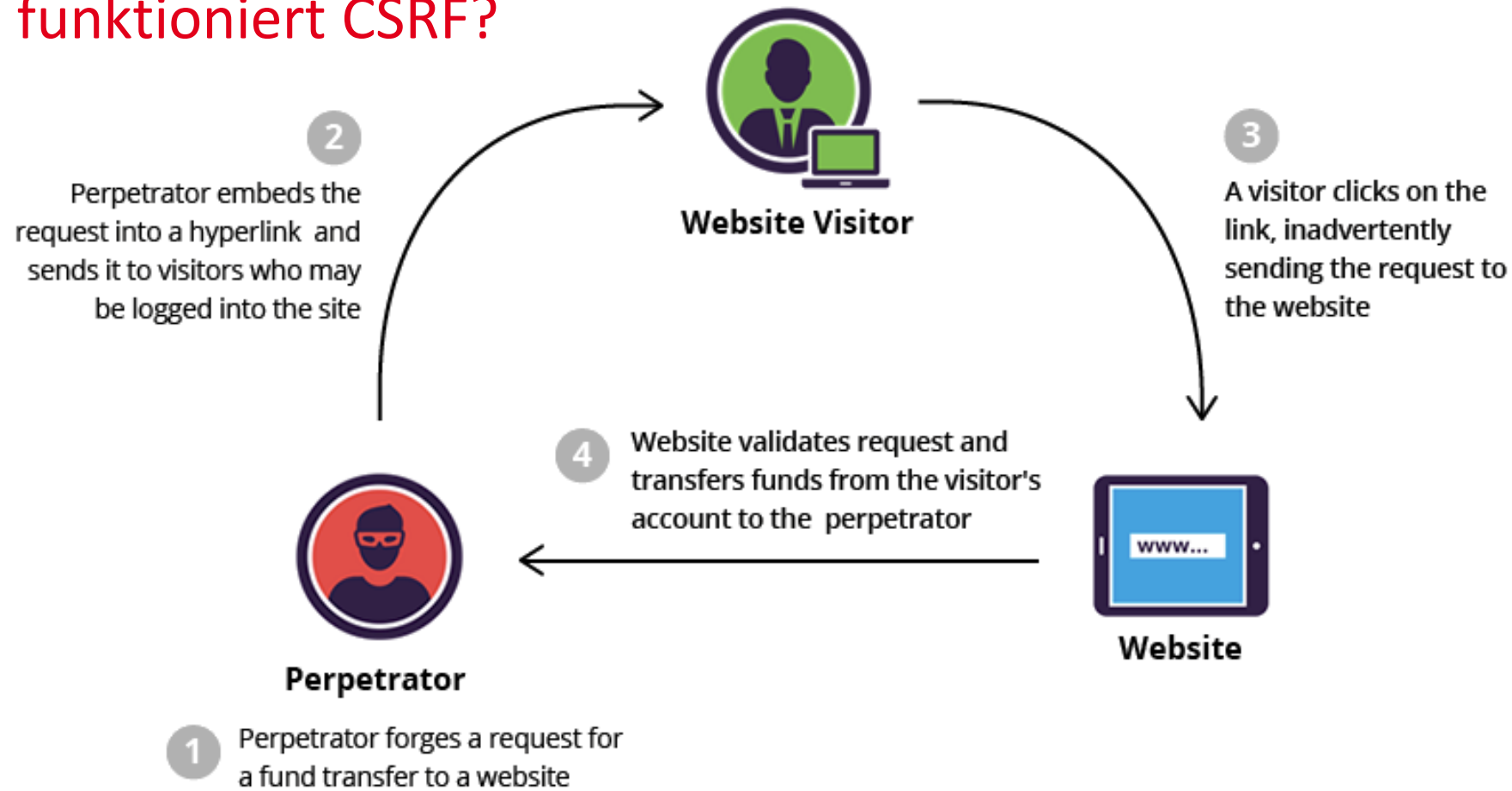


- Böswilligen Code / Link an Benutzer weitergeben
- Z.B. über Link in E-Mail / Social-Media-Nachricht / Kommentar in Forum / gefälschte Website

Wie funktioniert CSRF?



Wie funktioniert CSRF?



Varianten von CSRF-Attacken

- Reflected CSRF-Angriffe
 - Phishing-Mail
- Stored CSRF-Angriffe
 - Phishing-Webseite



Varianten von CSRF-Attacken

- CSRF in Verbindung mit Cross Site Scripting (XSS)
 - DOM-basierte CSRF-Angriffe
 - Skript auf Webseite
 - CSRF in Verbindung mit Session Hijacking-Angriffe
 - Session-ID klauen
 - CSRF in Verbindung mit Exploits
 - Browser Plugin
-



Nicht ausreichende Abwehrmaßnahmen

- URL-Filter
- IP-Adressenfilter
- CAPTCHAs
- Nur HTTP-Post akzeptieren
- HTTP-Referrer-Prüfung

```
8
9  <!--Malicious Form-->
10 <form id="malicious-form" action="http://localhost:3000/transfer" method="POST">
11   <input type="text" name="to" value="test2">
12   <input type="text" name="amount" value="9">
13   <input type="submit" value="transfer">
14 </form>
15 <script>
16   document.getElementById('malicious-form').submit();
17 </script>
```



Wie kann man sich dagegen Schützen? (Server-Side)

- Anti CSRF (Cross-Site Request Forgery Mitigation)
 - Anti CSRF Token
 - Hierbei werden Tokens auf der Website gespeichert diese werden für jede Session neu angelegt und im Request eingefügt.
 - Content Security Policy (CSP)
 - Implementieren einer CSP in Webanwendung
 - Verhindern von XSS / andere böswillige Skriptausführungen
 - => Gut konfiguriertes CSP kann dazu beitragen, Schutz gegen CSRF zu erhöhen.
 - Session Timeouts setzen
 - Abwägen zwischen Benutzerfreundlichkeit und Sicherheit
-



Wie kann man sich dagegen Schützen? (Server-Side)

- Anti CSRF Tokens
 - Generierung durch Server
 - Zufallszahlengenerator
 - Hash-Algorithmus
 - z.B. Durch Kombination aus User & Passwort zufälligen Token auf Website erstellen
 - CSRF-Tokens sicher und nicht vorhersehbar
 - In sicheren Speichermechanismen sichern
-



Wie kann man sich dagegen Schützen? (Client-Side)

- Nach der Nutzung ausloggen -> Session wird ungültig
 - Nicht auf alle Links drücken
 - Sessions beenden (ausloggen) bevor man andere Websites öffnet
 - Endgerät auf Malware prüfen
 - Zwei-Faktor-Authentifizierung
-



Aufruf des Direkten Links

- Angreifer trickst Nutzer aus auf einen Link zu drücken
- Dieser Link tätigt dann den API-Call

```
6 <!--Hyperlink-->  
7 <a href="http://localhost:3000/transfer?to=test2&amount=7">Click me!</a>
```

API-Call über Laden eines Bildes

- Angreifer trickst Nutzer aus um Bild in einer E-Mail zu laden
- Dieses tätigt dann den API-Call

```
3 <!--Image-->  
4 
```



Falsche Website

- Angreifer erstellt eine Kopie der Website
- Und sendet das Form an den Server im Name des Nutzers

```
9    <!--Malicious Form-->
10  <form id="malicious-form" action="http://localhost:3000/transfer" method="POST">
11    <input type="text" name="to" value="test2">
12    <input type="text" name="amount" value="9">
13    <input type="submit" value="transfer">
14  </form>
15  <script>
16    document.getElementById('malicious-form').submit();
17  </script>
```

Aufgabenblatt: GitHub

