# Cross-Site-Scripting (XSS)

https://github.com/christopherchristensen

## XSS Session Hijacking

### XSS

Cross-Site-Scripting

- bösartige Skripts in gutartige und vertrauenswürdige Webseiten injizieren

### Wann treten XSS-Angriffe auf?

- wenn Angreifen Webanwendungen verwenden, um bösartigen Code an anderen Endbenutzer zu senden
- üblich in Form eines browserseitigen Skripts

### Fehler, die Angriffe erleichtern

- fehlende Validierung von Input-Daten vom Client
- fehlende Kodierung von Input-Daten vom Client

### Weshalb ist XSS gefährlich?

- Benutzer können nicht wissen, dass Skript nicht vertrauenswürdig ist, und führen ihn aus
- Angreifer kann somit auf folgende Daten zugreifen, die Browser speichert
  - Cookies
  - Sitzungstoken
  - andere sensible Informationen

## Wo treten XSS-Anfriffe häufig auf?

- Dort wo Daten über nicht vertrauenswürdige Quellen in eine Webanwendung (oft Webanfrage) gelangen
- Wenn, Daten in dynamischen Inhalten enthalten, die an Webbenutzer gesendet werden, ohne auf schädliche Inhalte zu überprüfen

## Form von bösartigen Inhalten

- Segmente von JS, HTML, FLash oder andere Art von Code, den Browser ausführen kann
- Grosse Vielfalt, aber normalerweise Übertragung von privater Daten (Cookies, etc.)

## 3 Arten von XSS-Attacken

- Reflektiertes XSS
  - schädliche Skript zum Webserver gesendet
  - dort nicht gespeichert
  - ungeprüft an Client zurückgesendet
- Persistentes XSS
  - Speicherung der schädlichen Skripte auf Webserver, bzw. DB
  - bei jedem Aufruf an Client gesandt
  - geeignet in Foren, Blogs, welche Benutzereingaben auf Server speichern und ungeprüft ausgeben
- DOM-basiertes DSS
  - auch “lokales XSS”
  - manipulierte URL
  - Schadcode clientseitig ungeprüft ausgeführt
  - Webserver nicht involviert
  - statische Webseiten gefährdet

## Session Hijacking

- Session-Cookie eines anderen Users erhacken
- Session gleich übernehmen

## Überprüfen, ob Chat Room auf XSS anfällig ist

- `Hello <script>alert("hi");</script>`

## Mitigation

- Input Validation
- Output Encoding
- HTTPOnly Flag
- Content Security Policy (CSP)
- WAF

## HTTPOnly Flag

- Session- und alle anderen Kunden-Cookies dürfen nicht von JS clientseitig manipuliert werden

## CSP

- browserseitiger Mechanismus
- definiert eine Whitelist für clientseitige Ressourcen der WebApp
- wird über speziellen HTTP-Header gesetzt
- weist Browser an, nur Ressourcen aus bestimmten Quellen auszuführen / rendern