

# Cross-Site Scripting

*Paul Sebastian Ziegler*

Cross-Site Scripting (XSS) ist die Schwachstelle in Webanwendungen schlechthin. Wie kaum eine andere Technik kombiniert diese Technik einfache Methoden und Ansätze zu letztendlich verheerenden Angriffen. Jedoch ist das Wissen um diese Schwachstelle und die damit verbundenen Angriffe derzeit lediglich Sicherheitsexperten vorbehalten. Es existieren zwar umfangreiche Berichte und Dokumentationen, aber diese können zumeist nur von Insidern verstanden werden. Der normale Programmierer oder Nutzer, der sich mit Cross-Site Scripting auseinandersetzen muss, bleibt in der Regel außen vor.

Dieses TecFeed ist bemüht, das zu ändern. In einfachen Schritten führt Sie der Autor in das komplexe Thema ein. Sie werden lernen, was Cross-Site Scripting ist und wie man mit seiner Hilfe Webanwendungen angreifen kann. Nach der Lektüre dieses TecFeeds werden Sie in der Lage sein, Schwachstellen zu erkennen und zu beheben.

**O'REILLY®**



## INHALT

Einleitung | 2

Aufbau eines XSS-Angriffs  
gegen eine ungesicherte  
Webanwendung | 2

Effekte, die ein Angreifer  
durch XSS hervorrufen kann | 8

Schutzmechanismen,  
die zu kurz greifen | 19

Der Aufbau starker Schutz-  
mechanismen – Escapen und  
listenbasiertes Filtern | 36

Das Gefahrenpotenzial von XSS  
heute und in naher Zukunft | 47

Zusammenfassung | 52

Anhang A – Liste verschiedener  
Angriffsvektoren | 53

Anhang B – safehtml | 54

Über den Autor | 72

Danksagung | 72

**TecFeeds**

[www.tecfeeds.de](http://www.tecfeeds.de)

## Einleitung

Cross-Site Scripting ist in den letzten Jahren zu einem Schlagwort der Computersicherheit avanciert und verkörpert für viele Menschen bereits *die* Schwachstelle in Webanwendungen schlechthin. Es wird häufig durch die Buchstabenkette XSS abgekürzt und gehört zu den meistbesprochenen Themen in allen gängigen Mailinglisten zum Thema Internetsicherheit. Wie kaum eine andere Technik kombiniert es einfache Methoden und Ansätze zu letztendlich verheerenden Angriffen. Jedoch ist das Wissen um diese Schwachstelle und die damit verbundenen Angriffe derzeit lediglich Sicherheitsexperten vorbehalten. Es existieren zwar umfangreiche Berichte und Dokumentationen, aber diese können zumeist nur von Insidern verstanden werden. Der normale Programmierer oder Nutzer, der sich mit Cross-Site Scripting auseinandersetzen möchte, bleibt in der Regel außen vor.

Das Ihnen vorliegende TecFeed versucht, das zu ändern. In einfachen Schritten führe ich Sie durch die elementaren Grundlagen. Sie werden lernen, was Cross-Site Scripting ist und wie man mit seiner Hilfe Webanwendungen angreifen kann. Weiterhin werden Sie nach der Lektüre in der Lage sein, Schwachstellen eigenständig zu erkennen und zu beheben.

Beim Cross-Site Scripting wird Skriptcode in variable Bereiche dynamisch erstellter Webseiten eingeschleust. Oft sitzen Programmierer und Systemadministratoren dem landläufigen Irrtum auf, dass dadurch lediglich störende Effekte (z.B. Nachrichtenboxen) hervorgerufen werden könnten. Das geht aber

an der Realität vorbei: Das Potenzial von Cross-Site Scripting umfasst den Diebstahl von Daten, die Verunstaltung von Nutzeroberflächen (»Defacements«), das Überlasten dritter Server (DDoS-Angriffe) und vieles mehr. Besonders gefährlich wird es durch den Umstand, dass praktisch jede dynamische Webseite, die nicht explizit abgesichert wurde, angreifbar ist. Ich werde Ihnen zeigen, wie Sie sich als Verteidiger gegen diese Angriffe behaupten können – sei es als Programmierer, als Systemadministrator oder schlicht als interessierter Nutzer.

## Aufbau eines XSS-Angriffs gegen eine ungesicherte Webanwendung

Aller Anfang ist schwer. Ganz gleich, welche fortgeschrittenen Techniken wir im Laufe dieses TecFeeds noch betrachten werden – Ausgangsbasis für erfolgreiches Angreifen und auch Verteidigen einer Webanwendung wird immer Ihr grundlegendes Verständnis all der Mechanismen sein, die Cross-Site Scripting ermöglichen. Leider sind neue Mechanismen nie einfach zu erfassen. Wir wenden uns daher zu Beginn einem einfachen Beispiel zu. Mit seiner Hilfe werden Sie in der Lage sein, sich einen ersten Überblick über Cross-Site Scripting zu verschaffen und zu verstehen, warum praktisch jede nicht explizit abgesicherte Webanwendung anfällig ist.

### Ein anfälliges PHP-Skript

Sehen Sie sich das folgende PHP-Skript an. Es implementiert – wenn auch in stark vereinfachter Weise – einen in heutigen Webanwendungen gängigen Mechanismus: Vom Nutzer wird

eine Eingabe erbeten, die daraufhin in leicht aufbereiteter Form wiedergegeben wird. Derartiges Verhalten findet sich in Blogs, Wikis, Gästebüchern, Nachrichtensystemen, Foren und vielen anderen Anwendungen.

### Beispiel 1 `xss1.php`<sup>1</sup>

```
<?php
$content = $_GET['inhalt'];
echo '
  <html>
  <head><title>Einfaches Beispiel</title></head>
  <body>';
if (!$inhalt) {
  Echo '
  <b>Bitte Nachricht eingeben:</b><br />
  <form action="xss1.php">
  <textarea name="inhalt" rows="16" cols="100">
  </textarea><br />
  <input type="submit" name="send" value="Abschicken">
  </form>
  ';
}
else {
  echo '
  <b>Die von Ihnen eingegebene Nachricht lautet:</b><br />
  '.$inhalt.'<br /><br /><a href="xss1.php">Zurück</a>';
}
```

```
echo '
  </body>
  </html>';
?>
```

Kopieren Sie den gezeigten Code in eine Datei und speichern Sie diese unter dem Namen `xss1.php` auf Ihrem Webserver ab.

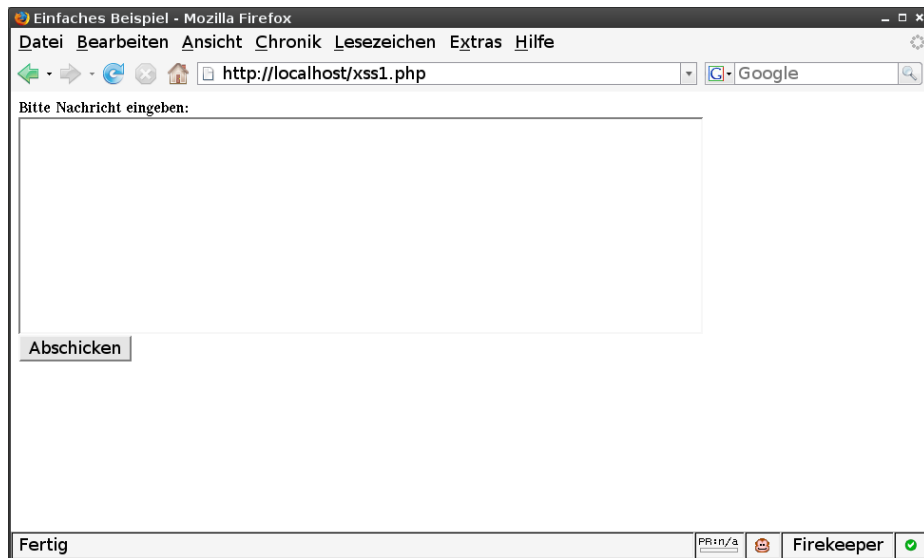


Es ist prinzipiell egal, welchen Webserver Sie für die hier gezeigten Beispiele verwenden. Da wir durchgängig mit in PHP verfassten Beispielen arbeiten werden, ist es jedoch notwendig, dass Ihr Webserver in der Lage ist, mit dieser Skriptsprache umzugehen. Weiterhin verfügt PHP über einige Mechanismen, die XSS-Angriffe erschweren oder sogar verhindern können. Um den hier gezeigten Beispielen durchgängig folgen zu können, müssen die Optionen `magic_quotes_gpc`, `magic_quotes_sybase` und `magic_quotes_runtime` in Ihrer `php.ini` deaktiviert sein.

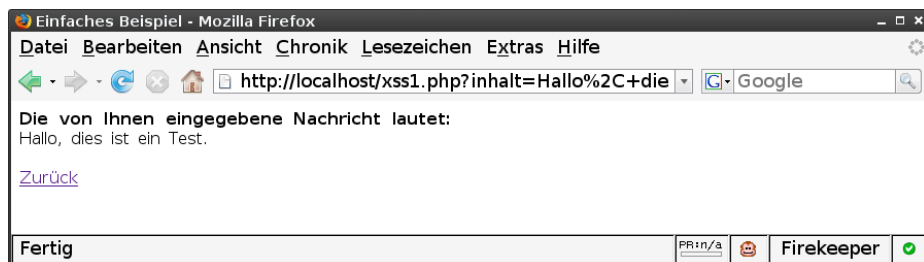
Öffnen Sie nun das Skript mit einem Browser Ihrer Wahl. Sollten Sie den Webserver auf Ihrem Arbeitsrechner laufen haben, so müssen Sie beispielsweise die URL `http://localhost/xss1.php` aufrufen. Ihnen wird daraufhin ein simples Formular präsentiert, wie es in Abbildung 1 zu sehen ist.

In das Textfeld können Sie beliebige Zeichenketten eintragen und dann abschicken. Ihre Eingabe wird Ihnen dann, wie in Abbildung 2 zu sehen ist, vom Skript präsentiert.

<sup>1</sup> Der in diesem TecFeed verwendete Code kann hier heruntergeladen werden: [http://www.oreilly.de/catalog/pdf\\_xssger/](http://www.oreilly.de/catalog/pdf_xssger/).



**Abb. 1** Anfälliges Skript im Ausgangszustand



**Abb. 2** Anfälliges Skript in normaler Nutzung

### Ausnutzen der Schwachstelle

Im letzten Abschnitt haben wir uns mit dem regulären Verhalten des anfälligen Skripts vertraut gemacht und die Nutzung erkundet. Wie in vielen Webanwendungen üblich, ermöglicht

uns das Skript, eine Eingabe verarbeiten und daraufhin in HTML-Code eingebunden darstellen zu lassen. Mitunter wird die Eingabe dazu in Datenbanken oder XML-Dokumenten gespeichert und für andere Nutzer zugänglich gemacht (z.B. kann der Nutzer eines Forums eine Nachricht übermitteln). Im Normalfall wird diese daraufhin in einer Datenbank abgespeichert und bei Bedarf einem anderen Nutzer, der den entsprechenden Teil des Forums liest, in hübsch aufbereiteter Form angezeigt.

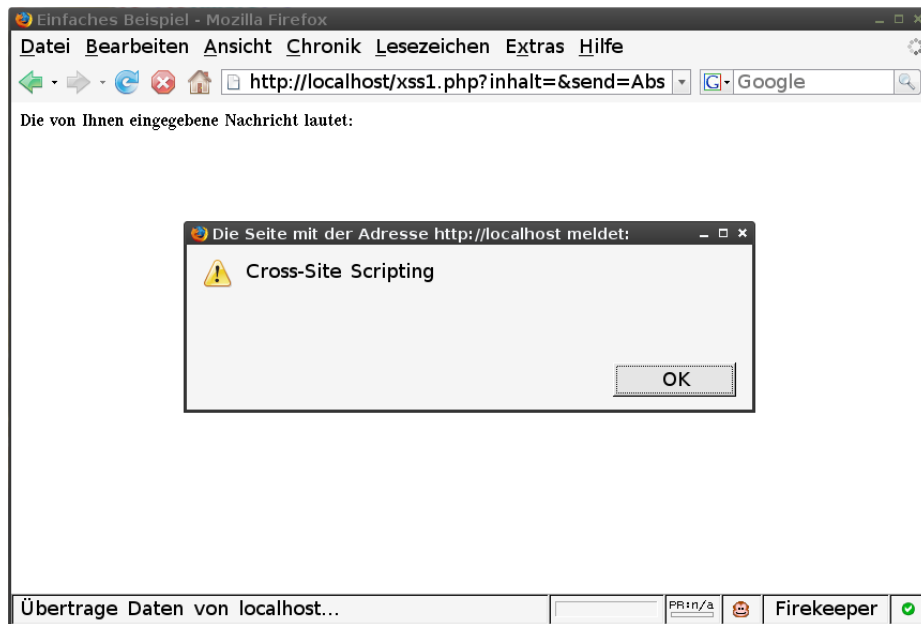
Was passiert nun aber, wenn ein böswilliger Nutzer keinen Text, sondern Skriptcode eingibt und absendet? Wie wird unser PHP-Skript reagieren, und wie wird sich der Browser verhalten? Um diese Frage zu klären, übergeben wir unserem Skript im Folgenden einen kurzen Block JavaScript und beobachten, was passiert.

Rufen Sie erneut `xss1.php` mit dem Browser Ihrer Wahl auf. Tragen Sie nun folgende Eingabe in das Textfeld ein und schicken Sie diese ab:


```
<script>alert("Cross-Site Scripting")</script>
```

Statt der erwarteten Webseite wird sich ein Mitteilungsfenster öffnen, das die Zeichenkette `Cross-Site Scripting` enthält.

Sie haben soeben erfolgreich Ihren ersten Cross-Site Scripting-Angriff durchgeführt.



**Abb. 3** Durch JavaScript verursachte Alert-Box

 Sollte sich kein Mitteilungsfenster öffnen, obwohl Sie den JavaScript-Block korrekt abgetippt haben, empfiehlt es sich zu überprüfen, ob der Codeblock zwar im Quellcode der Webseite enthalten, aber durch andere Zeichen verfälscht sein sollte. Ist das der Fall, dann ist wahrscheinlich eine der Optionen `magic_quotes_gpc`, `magic_quotes_sybase` oder `magic_quotes_runtime` in Ihrer `php.ini` aktiviert. Diese Optionen müssen deaktiviert werden, um die hier gezeigten Beispiele ausprobieren zu können. Außerdem muss der von Ihnen genutzte Browser JavaScript-Inhalte zulassen.

## Analyse des generierten HTML-Codes

Wodurch wurde der im vorigen Schritt beobachtete Effekt hervorgerufen? Betrachten wir einmal den vom Skript generierten HTML-Code genauer.

### Beispiel 2 Generierter HTML-Code

```
<html>
<head><title>Einfaches Beispiel</title></head>
<body>
<b>Die von Ihnen eingegebene Nachricht lautet:</b><br />
<script>alert("Cross-Site Scripting")</script><br /><br />
<a href="xss1.php">Zurück</a>
</body>
</html>
```

Wie Sie sehen können, wurde unsere Eingabe direkt in den HTML-Code eingebaut. Der Codeblock ist somit zu einem Bestandteil der Seite geworden und lässt sich von sonstigen Inhalten nicht mehr unterscheiden. Ein Browser, der diese Seite anzeigt, hat keinerlei Möglichkeit festzustellen, welche Bestandteile der Seite durch einen Nutzer übergeben und welche vom Programmierer vorbereitet wurden. Verarbeitet der Browser daraufhin den HTML-Code, um ihn darzustellen, so behandelt er den von uns eingebauten JavaScript-Block genau wie jeden anderen auch.

Die Schwachstelle im PHP-Skript lässt sich in folgender Zeile finden:

```
echo '<b>Die von Ihnen eingegebene Nachricht lautet:</b>
<br />'.$inhalt.'<br /><br /><a href="xss1.php">Zurück</a>';
```

Der Inhalt der Variablen `$inhalt`, die unsere Eingabe enthält, wird direkt in eine Zeichenkette eingefügt, die daraufhin in den HTML-Code übertragen wird. Bereits der `echo`-Befehl kann nicht mehr zwischen dem ursprünglichen String und unserer Eingabe unterscheiden. Dementsprechend finden wir unsere Eingabe unverändert in der generierten Webseite wieder.

Eine derartige Einbindung von Eingaben in den HTML-Code einer Webanwendung kommt ausgesprochen häufig vor. Dadurch lässt sich die anfangs aufgestellte Behauptung, praktisch jede nicht explizit abgesicherte dynamische Webanwendung sei mit Hilfe von Cross-Site Scripting angreifbar, untermauern. Fast jede Webanwendung nimmt Eingaben vom Nutzer entgegen und verarbeitet sie. Sobald diese Eingaben auf beliebige Weise ihren Weg zurück in den HTML-Code der generierten Webseiten finden, ist es möglich, einen XSS-Angriff durchzuführen.

In diesem Kapitel haben Sie die Grundlagen des Cross-Site Scripting kennen gelernt und sich durch ein einfaches Beispiel mit der praktischen Anwendung vertraut gemacht. Die hier erworbenen Grundlagen sind elementar für Ihr Verständnis von Cross-Site Scripting. Daher ist es wichtig, dass Sie alle Verständnisfragen oder technischen Probleme klären, bevor Sie sich den folgenden Kapiteln zuwenden.

Wir werden auf den hier gezeigten Vorgängen aufbauen, wenn wir im Laufe dieses TecFeeds sowohl die Möglichkeiten des Cross-Site Scripting erforschen als auch häufig angewandte Schutzmechanismen umgehen.

### Die drei grundlegenden XSS-Typen

Nun, da Sie wissen, worum es sich beim Cross-Site Scripting handelt und wie einfache Angriffe durchgeführt werden können, ist es an der Zeit, die verschiedenen Arten von XSS-Angriffen genauer zu betrachten.

#### REFLEKTIERTES CROSS-SITE SCRIPTING

Bei Schwachstellen dieser Art wird der zu injizierende Schadcode der Webanwendung mit dem Aufruf übergeben und umgehend an den Nutzer zurückgesandt. In freier Wildbahn werden derartige Schwachstellen meist wie folgt ausgenutzt:

Der Angreifer verbreitet eine präparierte URL, die als einen Bestandteil den zu injizierenden Code enthält, auf beliebigem Wege – üblicherweise durch E-Mails oder Einträge in Foren. Ein Opfer wird auf die URL aufmerksam und klickt auf den entsprechenden Link. Die anfällige Webseite wird nun vom Browser des Opfers aufgerufen. Durch die entsprechenden Parameter wird Skriptcode injiziert und an das Opfer zurückgesandt. Der Browser verarbeitet das ihm zugesandte HTML-Dokument und stößt dabei auf den injizierten Skriptcode.

Diese Form des Cross-Site Scripting ist bei Weitem die häufigste. Auch unser Beispiel funktioniert so.