CROSS-SITE-SCRIPTING (XSS)

Sicherheitslücken in Webseiten für XSS Angriffe verhindern

Handout

In diesem Handout warden die am Vortrag gezeigten Informationen und Angriffsszenarien dokumentiert. Es wird erklärt was XSS ist, was für Angriffsarten es gibt und wie man dagegen vorgehen kann.

Trottmann Sandro
Sandro.trottmann@casasoft.ch

Cross-site-scripting (XSS)

Was ist Cross-Site-Scripting

Cross-Site-Scripting nutzt Sicherheitslücken einer Webseite oder einer Webanwendung aus, um Daten der bestehenden Seite zu verändern. Cross-Site-Scripting zählt somit zu den aktiven Angriffen. Mit Cross-Site-Scripting wird zum Beispiel ein Script in eine vermeintlich sichere Seite eingeschleust. Hierbei gibt es verschieden Varianten von Cross-Site Scripting.

Unterschiedliche Arten von XSS

Reflektiert oder nicht-persistent

Das nicht persistente Cross-Site-Scripting, ist ein Angriff, der nicht beständig ist. Das heisst es werden zum Beispiel Sicherheitslücken von Formularen ausgenutzt. Es kann so zum Beispiel über eine URL (bei Formularen mit GET-Methode) die Übergabe Werte angepasst werden, oder auch bei POST-Methode kann eingegriffen werden (siehe Beispiel). Hat das Formular keine Validierung ist es möglich ein Script über diese Parameter mitzugeben. Da der Schadcode (das Script) jedoch nicht permanent auf der Webseite integriert ist, spricht man hier von einem nicht persistenten Cross-Site-Angriff.

Persistent oder beständig

Persistent oder beständiges Cross-Site-Scripting Attacken sind Angriffe, bei denen man versucht einen Schadcode, wie zum Beispiel ein Script, permanent auf einer Webseite zu platzieren. Beliebige Angriffsziele sind hier Gästebücher. Wird das Formular für einen Eintrag ins Gästebuch nicht validiert, kann der Schadcode direkt im Gästebuch (auf der Seite) eingebettet werden. Immer wenn nun die Seite aufgerufen wird, wird das Script ausgeführt.

Was ist mit Cross-Site-Scripting möglich, warum ist JS gefährlich?

Javascript ist eine vollumfängliche Programmiersprache. In Verbindung mit PHP und html ist hier sehr vieles möglich. Ob Login Daten klauen (Phishing) bis Sessions übernehmen (Session-Highjacking) oder auch Seiten Weiterleitung, mit Javascript können sehr böse Dinge angestellt werden und diese gilt es zu verhindern. Mit wenigen Schutzvorkehrungen können wir schon sehr viel bewirken. Mögliche Angriff- und Schutzmöglichkeiten wollen wir uns ansehen.

Beispiel Cross-Site-Scripting Angriffe und mögliche Lösungsansätze

Nicht-persistenter Cross-Site-Scripting Angriff

Wie oben geschrieben wird hier das Script einmalig eingeschleust und befindet sich nicht dauerhaft im Code.

Wir gehen mal von folgendem Formular aus:

```
<form method="GET" action="<?php echo $_SERVER['PHP_SELF']; ?>">
    <div>
        <label for="username">Username:</label>
        <input id="username" type="text" name="username" value="<?php echo $_GET['username']; ?>" />
    </div>
    <div>
        <label for="pass">Pass:</label>
        <input id="pass" type="text" name="pass" value="<?php echo $_GET['pass']; ?>" />
    </div>
    <div>
        <input type="submit" id="send">
    </div>
</form>
```

Hier befindet sich die Sicherheitslücke im form Tag:
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method= "posts">

Wir können nun die URL wie folgt anpassen und somit ein Javascript einschleusen:
http://meinewebsite.ch/index.php/"><script>alert("test")</script>

Interessant ist dabei dieser Abschnitt: /"><script>alert("test")</script>
Die Zeichen /"> beenden nämlich den echo befehl im action Attribut und schliesst den form Tag.
Somit wird unser Script Teil hinten angehängt und ausgeführt.
Mit dem Inspector kann man dies gut sehen.

```
<form method="POST" action="/mysite/indexget.php/">
  <script>alert("test")</script>
  ">
  <div>…</div>
```

Klar hier ist es nur ein alert im script, aber mit Javascript können wie oben beschrieben sehr viele Dinge angestellt werden.
Damit wir dies verhindern können schauen wir uns den Lösungsansatz an.

Lösungsansatz

PHP gibt uns die Möglichkeit die Eingabe mit htmlspecialchars() abzufangen:
<form action="<?php echo htmlspecialchars($_SERVER['PHP_SELF']); ?>" method= "posts">
Nun wird der Script Teil nur noch als Zeichen interpretiert und es wird nicht mehr ausgeführt. Der Script teil wird lediglich als Text wahrgenommen, die Seite funktioniert jedoch noch und das Script wird nicht ausgeführt.

```
<form action="/mysite/index.php/"><script>alert("test")</script>" method="posts">
  <p>…</p>
```

Persistent Cross-Site-Scripting Angriff

Nun gehen wir von einer Seite aus mit Gästebuch Einträge. Wir haben ein einfaches Formular mit den Feldern Name und einer Nachricht. Die Variablen werden aus der POST-variable geholt und in einer separaten Variable gespeichert und direkt ausgegeben.

```
$name = isset($_POST['name']) ? $_POST['name'] :'';
$message = isset($_POST['message']) ? $_POST['message'] :'';
```

Dies macht es uns nun sehr einfach ein script einzuschleusen, indem wir javascript im Feld eingeben:

Bemerkung:
```
Tolle Seite!!
<script>alert("du wurdest gehackt");</script>
```

Wenn wir nun das Formular absenden ist der Code bereits in der Seite. Und noch besser er ist auf den ersten Blick nicht ersichtlich.

Gästebucheinträge

Hans Muster
Toller Laden

Sandro
Tolle Seite!!

Erst mit dem Inspector sehen wir unser Script. Dieses Script wird nun bei jedem Seiten Aufruf ausgeführt.

```
<p>
  <b>Sandro</b>
  <br>
  Tolle Seite!!
  <script>alert("du wurdest gehackt");</script>
```

Lösungsansatz

Dies können wir nun wieder mit htmlspecialchars abfangen:
$name = isset($_POST['name']) ? htmlspecialchars($_POST['name']) :'';

Der Inhalt der Variable Namen wird nun nur noch als String interpretiert. Dies machen wir natürlich auch für die Nachricht. Das Script wird ausgegeben, da es nur noch als Text interpretiert wird.

Gästebucheinträge

Hans Muster
Toller Laden

Sandro
<script>alert("test");</script>