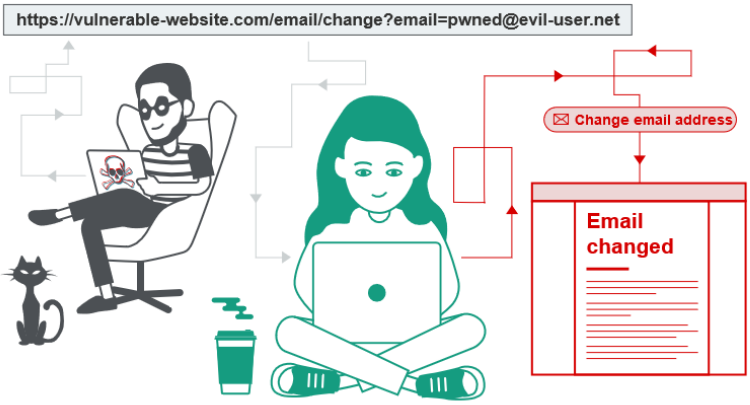
Cross Site Request Forgery (CSRF)

Was ist es?

CSRF ist eine Website Schwachstelle, welche es Angreifern erlaubt den Benutzern der Website Aktionen zu durchführen welche sie nicht gewollt hätten.



Was ist der Schaden einer CSRF Attacke?

In einer erfolgreichen CSRF Attacke, bringt der Angreifer das Opfer dazu z.B. die E-Mail-Adresse oder Passwort zu wechseln. Abhängig von der Aktion, kann der Angreifer vollen Zugriff auf das Konto des Opfers bekommen. Falls das Opfer Admin Rechte auf der Website besitzt, kann der Angreifer die volle Kontrolle über die Website erlangen.

XSS vs. CSRF

XSS oder Cross-Site Scripting erlaubt es einen Angreifer JavaScript vom Browser des Opfers aus zu starten. Der Schaden einer XSS Schwachstelle sind generell mehr ernst zu nehmen wie CSRF Schwachstellen.

Wie sieht ein Angriff aus

Der Angreifer muss das Opfer nur dazu bringen seine schädliche URL anzuklicken. Das kann in Form von einer E-Mail (Social Engineering) oder einer eigen gehosteten Seite sein z.B. ein Blog.

Das Opfer muss bei einer schädlichen Seite nur einmal drauf gehen und ohne dass das Opfer irgendetwas anklickt wird ein GET / POST Request mit der schädlichen URL gesendet. Der Request wird vom Server verarbeitet und die Aktion ausgeführt, welche in der URL beschrieben wurde. In diesem Fall wird das Password des aktuellen Users geändert.

Einen einfachen GET Request kann man mit den folgenden HTML Element simulieren.

```
<img src="http://example.com/change.php?password=easyPassword">
```

Um einen POST Request zu senden braucht es dazu mehr.

```
<body onload="document.getElementById('csrf').submit()">
  <form id="csrf" action="http://example.com/change.php" method="POST">
    <input name="password" value="easyPassword" />
  </form>
</body>
```

Schutzmassnahmen

Man führt einen Token ein, welcher bei dem Server generiert wird und dem Client, im bestimmten Formular, geteilt wird.

```
<form action="change.php" method="post">
    <h3>Email</h3>
    <input type="email" name="email">
    <h3>Password</h3>
    <input type="password" name="password">
    <input type='hidden' name='key'value='70da9a1ab8'>
    <input type="submit">
</form>
```

Der Key kann mit «type='hidden'» versteckt werden. Der Key sollte länger wie 10 Chars sein und der Key mit der Client Session gebunden sein.

Fehler welche passieren können beim Implementieren

Laut dieser Quelle [1], sind das Fehler, welche beim Implementieren passieren können.

Validierung eines Tokens hängt von der Request Methode ab.

Einige Applikationen validieren die POST Methode richtig, überspringen aber die GET Methode. Der Angreifer kann da einfach die Methode von POST zu GET wechseln.

Validierung eines Tokens hängt davon ab ob er präsent ist

Der Angreifer kann hier den kompletten Parameter des Tokens überspringen.

Token ist nicht an User Session gebunden

Der Angreifer kann sich mit seinem Account einloggen und seinen Token nehmen und so seine schädliche URL erstellen.

[1] https://portswigger.net/web-security/csrf