Session

Einführung

Mit Sessions hat man die Möglichkeit, bestimmte Daten während einer Folge von Aufrufen eurer Website festzuhalten. Jedem Besucher wird dazu eine einzigartige Session-ID zugeordnet, eine mögliche Session-ID könnte wie folgt aussehen: *dadafb244bbcb4bb84116d38f0ebd077*.

Diese Session-ID wird vom Browser des Besuchers bei jedem Seitenaufruf erneut an den Webserver gesendet, so dass der Webserver die Möglichkeit hat, den Besucher zu identifizieren und Werte von vorherigen Seitenaufrufen zu laden. Die Session-ID wird vom Browser dabei entweder als GET-Parameter an die URL angehängt oder beim Besucher lokal in einem Cookie gespeichert. Das Speichern als Cookie ist aus Sicherheitsgründen zu bevorzugen und wird von PHP auch präferiert. Als Webentwickler muss man sich zum Glück häufig nicht um die Weitergabe der Session-ID kümmern, PHP nimmt uns dort eigentlich die gesamte Arbeit ab.

Auf dem Server ist dann für jede Session-ID lokal ein Speicher eingerichtet, der beliebige Variablen für den Besucher beinhalten kann. So können wir in der Session für einen Besucher beispielsweise abspeichern, mit welchem Benutzernamen sich dieser eingeloggt hat, welche Waren in seinem Warenkorb liegen usw.

Der Benutzer hat keine Möglichkeiten, die Variablen in seiner Session zu sehen oder zu manipulieren. Er besitzt nur die Information, welche Session ID ihm vom Webserver zugewiesen wurde.

Sicherheit

Sessions bieten zwar keine 100%ige Sicherheit, dennoch sind sie relativ sicher. Bei jedem Seitenaufruf teilt der Browser dem Webserver mit, welche Session-ID dieser besitzt. Nun kann ein böswilliger Besucher seine Session-ID einfach manipulieren und könnte die Ses-

sion-ID von einem anderen Besucher angeben. Der Dieb könnte sich so als jemand anderes in einer Community ausgeben.

Wie bekommt der Angreifer denn die Session ID heraus?

Wie eingangs erwähnt werden Session-IDs entweder über die URL mittels dem Parameter *?PHPSESSID*übergeben oder als Cookie beim Besucher gespeichert. Sollten die Session-IDs per URL übertragen werden, dann kann es passieren dass der Benutzer diese URL kopiert und beispielsweise diese per E-Mail oder Facebook an Freunde und Bekannte weitersendet. Rufen die Freunde dann den Link auf, so benutzen diese die Session-ID weiter und sind mit einem anderen Account eingeloggt. Dies ist gerade problematisch, wenn man den Link auf einer öffentlichen Seite postet.

Werden stattdessen Cookies verwendet, was der Standard ist und bei allen Besuchern passiert die nicht explizit Cookies deaktiviert haben, muss der Angreifer an diesen Cookie gelangen um die Session zu stehlen. Dies geht beispielsweise über einen Trojaner auf dem Rechner oder über das Abhören der Leitung.

Session ID durch Zufall erraten

Natürlich kann der Dieb auch eine Session-ID erraten, allerdings ist dies sehr unwahrscheinlich. Es ist wahrscheinlicher, dass im Lotto mehrmals hintereinander die gleichen Zahlen gezogen werden als eine Session ID per Zufall zu erraten. Darüber sollte wir uns eigentlich keine Sorgen machen.

Zusammenfassend ist zu sagen, dass Sessions schon recht sicher sind und wir uns um den Diebstahl einer Session-ID meistens keine Sorgen machen müssen. Sehr kritische Operationen, wie z.B. das Löschen des Accounts oder ähnliches, sollten aber mit einer zusätzlichen Passwortabfrage nochmals gesichert sein.

Erste Schritte - Sessions registieren

Möchten wir in einem Script auf Sessions zurückgreifen, müssen wir, bevor wir irgendeine Ausgabe machen, den Befehl *session_start();* aufrufen:

```
<?php
Session_start();
?>
```

Es empfiehlt sich, diesen Code immer ganz oben der Scripts stehen zu haben. Falls ihr die Fehlermeldung *Cannot send session cookie - headers already sent by* erhaltet, dann gab es irgendwo vor *ses sion_start()* eine Ausgabe. Eine leere Zeile oder sogar ein Leerzeichen vor <?php reichen bereits aus.

Möchten wir einen Wert / eine Variable über mehrere Seitenaufrufe hinweg in der Session speichern, dann geht dies wie folgt:

```
<?php
$_SESSION['name'] = "wert";
?>
```

Diesen Wert können wir später, auch auf anderen Seiten, wie folgt ausgeben:

```
<?php
$name = $_SESSION['name'];
echo $name;
?>
```

Wichtig, immer wenn irgendwo mit Sessions gearbeitet wir, muss zuvor *session_start()* ausgeführt worden sein.
Eine Session-Variable kann wie jede andere Variable in PHP verwendet werden. Es können darin Zahlen, Zeichenketten oder sogar Arrays abspeichern werden.

Überprüfen ob Session-Variable registriert ist

Oft empfiehlt es sich zu überprüfen, ob eine gewisse Session-Variable bereits registriert wurde. Dies geht mittels der Funktion *isset($variable)*

```
<?php
session_start();
if (!isset($_SESSION['visited'])) {
  echo "Du hast diese Seite noch nicht besucht";
  $_SESSION['visited'] = true;}
else {
  echo "Du hast diese Seite zuvor schon aufgerufen";}
?>
```

Hier wird zuerst überprüft ob es die Session-Variable schon gibt. Falls es sie nicht gibt, wird sie registriert.

Würden wir anstatt **!isset** nur **isset** schreiben, könnten wir überprüfen, ob die Session registriert wurde. Dies lässt sich bei Logins verwenden:

```
<?php
session_start();
if (isset($_SESSION['username'])) {
  echo "Herzlich Willkommen ".$_SESSION['username'];
} else {
  echo "Bitte erst einloggen";
}
?>
```

Sessions löschen

Um alle Session-Daten der Sitzung zu löschen verwendet man *session_destroy()*

```
<?php
session_destroy();
?>
```

Dieser Befehl löscht **alle** Daten der Session und kann nützlich sein wenn sich der Benutzer z.B. aus eurem System ausloggen möchte. Denkt dran, ihr müsst zuvor *session_start()* ausführen bevor ihr *session_destroy()* ausführen könnt.

Um eine einzelne Session-Variablen zu löschen könnt ihr *unset($variable)* verwenden:

```
<?php
unset($_SESSION['name']);
?>
```