Сигурност на PHP уеб приложения

PHP Web Application Security

Топ 10 уязвимости при уеб приложенията

Open Web Application Security Project
http://www.owasp.org

- 1. Cross Site Scripting (XSS)
- 2. SQL Injection
- 3. Malicious File Execution

Топ уязвимости при уеб приложенията

- 4. Insecure Direct Object Reference
- 5. Cross Site Request Forgery (CSRF)
- 6. Information Leakage and Improper Error Handling
- 7. Broken Authentication and Session Management
- 8. Insecure Cryptographic Storage
- 9. Insecure Communications
- 10. Failure to Restrict URL Access

Cross Site Scripting (XSS)

- Какво може да доведе до XSS атака?

  Уеб приложение използва данни предоставени от потребителя, без да ги валидира или филтрира.

- До какво може да доведе една XSS атака?

  Изпълнение на нежелани скриптове в браузъра на потребителя - жертва на атаката.

- Кои видове уеб приложения са засегнати?

  Всички, които визуализират динамично съдържание предоставено от потребителя.

Видове XSS атаки

- Non-persistent (Reflected)
  Когато данни, предоставени от уеб клиент (най-често под формата на параметри в HTTP заявка), се визуализират директно в браузъра на потребителя.
- Persistent (Stored)
  Когато данни, предоставени от потребителя, се съхраняват на сървъра и се използват за генериране на динамични страници без да се филтрират.
- DOM Based
  Изпълняват се изцяло от страна на клиента. Когато JavaScript код използва директно данни от DOM обекти, като document.location.

Пример

```
<form action="comment.php" method="POST">
  Име: <input type="text" name="name" />
  Коментар: <textarea name="comment"></textarea>
  <input type="submit" value="Добави коментара" />
</form>
```

```
<?php
  echo "<div class='comment'>
  echo "  <p>$name написа:</p>";
  echo "  <blockquote>$comment</blockquote>";
  echhp "</div>";
?>
```

Как да се предпазим?

- **Валидиране на входните данни**

  Всички входни данни трябва да се проверяват за коректност – тип, формат и т.н.

  ```
  <?php if( isValidName($name) ) { … }?>
  ```

- **Филтриране на изходните данни**

  Всички изходни данни трябва да се филтрират за специални символи преди да се визуализират.

```
<?php echo htmlentities( $comment ); ?>
```

SQL Injection

- Какво може да доведе до SQL инжекция?

  Изпълнение на SQL заявки, които се конструират динамично с входни данни без валидация.

- До какво може да доведе една SQL инжекция?

  Неправомерен достъп и модификация на данни. Неправомерна автентикация и оторизация и др.

- Кои видове уеб приложения са засегнати?

  Всички, които изпълняват динамично генерирани заявки.

Пример

Форма за вход в сайт, чрез която потребителят въвежда потребителско име и парола:

```
<form action="login.php" method="POST">
  Потребител: <input type="text" name="username" />
  Парола: <input type="text" name="password" />
  <input type="submit" value="Вход" />
</form>
```

Пример

Заявка, която търси потребител с въведените име и парола:

```
<?php
  $username = $_POST['username'];
  $password = md5($_POST['password']);
  $sql = "SELECT COUNT(id) FROM users WHERE “;
  $sql.= "username='$username'";
  $sql.= "AND password='$password'";
?>
```

Как да се предпазим?

- **Валидиране на входните данни**

  Всички входни данни трябва да се проверяват за коректност – тип, формат и т.н.

```
<?php if( isValidUsername($username) ) { … }?>
```

- **Филтриране на данните, участващи в заявки**

  Всички данни, които участват при конструирането за заявки, трябва да се филтрират, като се премахнат специалните за SQL символи.

```
mysql_real_escape_string($_POST['username']);
```

Malicious File Execution

- Какво може да доведе до атаката?

  Най-често възможност за качване на файлове от потребителя – снимки, документи и т.н.

- До какво може да доведе подобна атака?

  Неправомерното изпълнение на файлове дава на атакуващия почти неограничени възможности за злоупотреба.

- Кои видове уеб приложения са засегнати?

  Всички, които приемат файлове качвани от потребителя.

Пример

Форма, чрез която потребителят може да качи изображение в jpg формат:

```
<form action="upload.php" method="POST"
enctype="multipart/form-data">
  Изберете изображение:
  <input type="file" name="picture" />
  <input type="submit" value="Качване" />
</form>
```

Пример

Скрипт, който проверява и записва изображението:

```
<?php
  $picture = $_FILES['picture'];
  $filename = "pictures/" . basename($picture['name']);
  if($picture['type'] != "image/jpg") {
    echo "Невалиден формат! Моля, опитайте отново.";
    exit; }
  else {
    move_uploaded_file($picture['tmp_name'],$filename);
  }
?>
```

Как да се предпазим?

- **Съхраняване извън публичната директория**

  Когато файловете се съхраняват извън публичната директория, те не са достъпни и атакуващият няма да има възможност да изпълни тяхното съдържание.

- **Преименуване на файловете**

  Файлове да се преименуват при тяхното качване, така че атакуващият да не знае под какво име са записани.

```
<?php
  $picture = $_FILES['picture'];
  $picture_info = getimagesize($picture['tmp_name']);
  $filename = "../pictures/" . time() . ".jpg"); … ?>
```

Софтуер за автоматично тестване сигурността на PHP приложения

- **PHP Security Scanner**
  http://sourceforge.net/projects/securityscanner/
- **Spike PHP Security Audit**
  http://sourceforge.net/projects/phpsecaudit/
- **PIXY**
  http://pixybox.seclab.tuwien.ac.at/pixy/

Софтуерът за автоматично тестване не може да гарантира сигурността на едно PHP приложение, той е само допълнително средство.

Полезни връзки

- Secure file upload in PHP web applications
  http://www.scanit.be/uploads/php-file-upload.pdf
- OWASP Top 10 – 2010: The ten most critical
  web application security risks
  http://owasptop10.googlecode.com/files/OWASP Top 10 - 2010.pdf
- SQL Injection Attacks by Example
  http://unixwiz.net/techtips/sql-injection.html
- PHP and the OWASP Top Ten Security Vulnerabilites
  http://www.sklar.com/page/article/owasp-top-ten

Полезни връзки

- PHP Security Manual
  http://us3.php.net/manual/en/security.php
- Cross Site Scripting (XSS) Cheat Sheet
  http://ha.ckers.org/xss.html
- Improving web application security
  http://www.cgisecurity.com/lib/Threats_Countermeasures.pdf
- PHP Security Guide
  http://shiflett.org/php-security.pdf
- A Guide to Building Secure Web Applications and Services
  http://prdownloads.sourceforge.net/owasp/OWASPGuide2.0.1.pdf