28-11-2024

Ugarteburu torlojueak web orrialdearen zaurgarritasuenen txostena

Zaurgarritasunen txostena

RHEM

Indizea

[1.Register formularioa 2](#_Toc183690575)

[1. SQL Injection 2](#_Toc183690576)

[2. Cross-Site Scripting (XSS) 5](#_Toc183690577)

[3. Artxiboen igoera – Artxiboen kargan ahultasuna 6](#_Toc183690578)

[4. CSRF aurkako protekzioa 7](#_Toc183690579)

[2.Login formularioa 7](#_Toc183690580)

[1. SQL Injection 7](#_Toc183690581)

[2. Brute-force protekzio falta 8](#_Toc183690582)

[3. CSRF Protekzio falta 8](#_Toc183690583)

[3.Search formularioa 9](#_Toc183690584)

[1. SQL Injection 10](#_Toc183690585)

[2. CSRF Protekzio falta 10](#_Toc183690586)

3. XSS Eraso protekzio falta………………………………………………………………………………………..11

Zaurgarritasunen txostena

Txosten honek emandako PHP formularioetan aurkitutako ahultasunak eta horiek zuzentzeko gomendioak deskribatzen ditu. Kode kalteberaren kokapen zehatza eta urrakortasun bakoitza zuzentzeko behar diren hitzez hitzeko aldaketak zehazten dira.

# 1.Register formularioa

## 1. SQL Injection

Aurkitutako kalteberatasuna: Emandako kodean, prestatutako sententzia bat erabiltzen bada ere, formularioko datuak ez dira erabat baliozkotuak edo sanitizatuak izango datu-basean txertatu aurretik. Prestatutako sententziak SQL injekzioa prebenitzen laguntzen duen arren, sarrera-eremuak ez balidatzea arrisku potentziala da.

Kodean kokatzea: Urrakortasuna datu-basean txertatzeko datuak prestatzen diren atalean dago. Hurrengo blokea kode kaltebera da:

```  
$sql = "INSERT INTO users (username, password, izena, abizena, hiria, lurraldea, herrialdea, postakodea, telefonoa, irudia) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";  
$stmt = mysqli\_prepare($conx, $sql);  
mysqli\_stmt\_bind\_param($stmt, "ssssssssss",   
 $data['email'],   
 $hashed\_password,   
 $data['firstname'],   
 $data['lastname'],   
 $data['city'],   
 $data['stateProv'],   
 $data['country'],   
 $data['postcode'],   
 $data['telephone'],   
 $data['imagen']);  
```  
  
Gomendioa: Eremu bakoitzerako baliozkotzeak eta sanitizazioa inplementatzea, erabili aurretik. Adibidez, mezu elektroniko bat baliozkotzeko, 'filter\_var ($data [' email '], FILTER\_VALIDATE\_EMAIL)' erabil dezakezu.  
Kodean hitzez hitz aldatzea:  
  
```  
 if (!filter\_var($data['email'], FILTER\_VALIDATE\_EMAIL)) {

$error['email'] = "Emailaren formatua ez da egokia.";

}

if ($data['password'] !== $data['password2']) {

$error['password'] = "Pasahitzak ez dira berdinak.";

}

if (strlen($data['password']) < 8) {

$error['password'] = "Pasahitza 8 karaktere edo gehiago izan behar ditu";

}  
```

Baita datuak parametrizatuko ditugu datu basera insertatzerakoan SQL Injection prebenitzeko:

```

$hashed\_password = password\_hash($data['password'], PASSWORD\_DEFAULT);

$sql = "INSERT INTO users (username, password, izena, abizena, hiria, lurraldea, herrialdea, postakodea, telefonoa, irudia)

VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";

$stmt = $conx->prepare($sql);

$stmt->bind\_param("ssssssssss",

$data['email'],

$hashed\_password,

htmlspecialchars($data['firstname'], ENT\_QUOTES, 'UTF-8'),

htmlspecialchars($data['lastname'], ENT\_QUOTES, 'UTF-8'),

htmlspecialchars($data['city'], ENT\_QUOTES, 'UTF-8'),

htmlspecialchars($data['stateProv'], ENT\_QUOTES, 'UTF-8'),

htmlspecialchars($data['country'], ENT\_QUOTES, 'UTF-8'),

htmlspecialchars($data['postcode'], ENT\_QUOTES, 'UTF-8'),

htmlspecialchars($data['telephone'], ENT\_QUOTES, 'UTF-8'),

$data['imagen']);

if (!$stmt->execute()) {

die('Error: ' . $stmt->error);

} else {

echo '<meta http-equiv="refresh" content="0;url=index.php">';

exit;

}

```

## 2. Cross-Site Scripting (XSS)

Aurkitutako kalteberatasuna: Erabiltzaileek sartutako datuak (adibidez, 'email', 'firstname', 'lastname', etab.) zuzenean HTMLn agertzen dira, XSSa prebenitzeko inolako neurririk gabe. Horri esker, erasotzaile batek beste erabiltzaile batzuen nabigatzailean exekutatzen den JavaScript kode maltzurra injektatu dezake.

Kodean kokatzea: Datuak ihes egin gabe injektatzen diren inprimakiaren zatian dago zaurgarritasuna. Hurrengo blokea kode kaltebera da:

```  
<input type="text" name="data[email]" value="<?php echo $data['email']; ?>" />  
<input type="text" name="data[firstname]" value="<?php echo $data['firstname']; ?>" />  
<input type="text" name="data[lastname]" value="<?php echo $data['lastname']; ?>" />  
```  
  
Gomendioa: Erabili 'htmlspecialchars ()' funtzioa xss-a inplementatzeko baklio duten sinbiloak ezabatzeko HTMLn erakutsi aurretik.

Kodigoan aldaketak:  
  
```  
<p>

<label>Email/username: </label>

<input type="email" name="data[email]" value="<?php echo htmlspecialchars($data['email'] ?? '', ENT\_QUOTES, 'UTF-8'); ?>" required />

<?php if ($error['email']) echo '<p>' . htmlspecialchars($error['email'], ENT\_QUOTES, 'UTF-8'); ?>

</p>

<p>

<label>Izena: </label>

<input type="text" name="data[firstname]" value="<?php echo htmlspecialchars($data['firstname'] ?? '', ENT\_QUOTES, 'UTF-8'); ?>" required pattern="[A-Za-zÀ-ÿ\s]+" title=" Bakarrik letrak jarri ahal dira" />

<?php if ($error['firstname']) echo '<p>' . htmlspecialchars($error['firstname'], ENT\_QUOTES, 'UTF-8'); ?>

</p>  
```

Erebai backend-ean hau aplikatuko dugu segurtasun agehiago bermatzeko:

## 3. Artxiboen igoera – Artxiboen kargan ahultasuna

Aurkitutako kalteberatasuna: Fitxategiaren luzapena baliozkotu arren, ez da edukia egiaztatzen, eta, horri esker, erasotzaile batek fitxategi maltzurrak igo ditzake. Gainera, fitxategiaren izena ez da sanitarioa, eta horrek zeharkako direktorioaren eraso bat edo lehendik dauden fitxategiak gainidazteko aukera eman dezake.

Kodean kokatzea: Fitxategien karga maneiatzen den atalean dago kalteberatasuna:

```  
if (isset($\_FILES['imagen']) && $\_FILES['imagen']['error'] == 0) {  
 $valid\_extensions = ['jpg', 'jpeg', 'png', 'gif'];  
 $file\_extension = pathinfo($\_FILES['imagen']['name'], PATHINFO\_EXTENSION);  
 if (!in\_array($file\_extension, $valid\_extensions)) {  
 $error['imagen'] = "Argazki formatua ez du balio.";  
 }  
 if ($\_FILES['imagen']['size'] > 5000000) {  
 $error['imagen'] = "Artxiboa oso handia da.";  
 }  
```  
  
Gomendioa: Fitxategiaren edukia balidatu 'getimagesize ()' erabiliz, irudi balioduna dela ziurtatzeko. Sortu ausazko fitxategi-izenak zeharkako direktorioko gainidazketak eta erasoak saihesteko.

Kodean aldaketa:  
  
```  
if (isset($\_FILES['imagen']) && $\_FILES['imagen']['error'] == 0) {  
 $image\_info = getimagesize($\_FILES['imagen']['tmp\_name']);  
 if ($image\_info === false) {  
 $error['imagen'] = 'El archivo no es una imagen válida.';  
 }  
 $file\_extension = pathinfo($\_FILES['imagen']['name'], PATHINFO\_EXTENSION);  
 $new\_file\_name = uniqid() . '.' . $file\_extension;  
 $path = "perfiles/" . $new\_file\_name;  
 move\_uploaded\_file($\_FILES['imagen']['tmp\_name'], $path);  
 $data['imagen'] = $new\_file\_name;  
```

## 4. CSRF aurkako protekzioa

Aurkitutako kalteberatasuna: Formularioak ez du CSRF erasoen aurkako babesik inplementatzen. Horri esker, erasotzaile batek formulario maltzur bat sor dezake, zerbitzariari datuak bidaltzeko erabiltzailearen baimenik gabe.

Kodean kokatzea: Inprimakiak ez du CSRF tokenik eskaera erabiltzailearengandik berarengandik datorrela egiaztatzeko.

```  
<form action="<?php echo $\_SERVER['PHP\_SELF'] . "?action=register"; ?>" method="POST" enctype="multipart/form-data">  
```  
  
Recomendación: Implementar un token CSRF para proteger el formulario. Este token debe ser generado por el servidor y validado al procesar la solicitud.

Cambio literales en el código:  
  
```  
$token = bin2hex(random\_bytes(32));  
$\_SESSION['csrf\_token'] = $token;  
<input type="hidden" name="csrf\_token" value="<?php echo $token; ?>" />  
```

# 2.Login formularioa

## 1. SQL Injection

Aurkitutako kalteberatasuna: Erabiltzailearen sarrera-balioekin zuzenean eraikitako SQL kontsultari esker, SQL injekzioa ahalbidetzen da (). Erasotzaile batek komando maltzurrak exekutatu ditzake eremu hauetan zehar.  
Kodean duen kokapena: Zaurgarritasuna "mysqli\_query" funtzioarekin eraikitako SQL kontsultan dago.  
  
```  
$creds = mysqli\_query($conx,"SELECT \* FROM users WHERE username='".$\_POST['username']."' AND password='".md5($\_POST['password'])."'");  
```  
Gomendioa: SQL injekzioa prebenitzeko, prestatutako sententziak erabiltzea.  
Kodean hitzez hitz egindako aldaketak:  
  
```  
$stmt = $conx->prepare("SELECT \* FROM users WHERE username = ? AND password = ?");  
$hashed\_password = hash("md5", $\_POST["password"]);   
$stmt->bind\_param("ss", $\_POST["username"], $hashed\_password);  
$stmt->execute();  
$result = $stmt->get\_result();  
```

## 2. Brute-force protekzio falta

Aurkitutako zaurgarritasuna: Formularioak saioa hasteko saiakera mugagabeak ahalbidetzen ditu, eta hori indar gordineko eraso baten bidez leher daiteke.

Kodean kokatzea: Ahulezia login-saiakerak mugatzeko mekanismorik ez izatean datza:

```  
if(isset($\_POST["submit"])) {  
 $creds = mysqli\_query($conx, ...);  
}  
```  
  
Gomendioa: Huts egindako saiakera-kontagailu bat eta blokeo-denbora bat inplementatzea.  
Kodean hitzez hitz egindako aldaketak:  
  
```  
if (!isset($\_SESSION["login\_attempts"])) {  
 $\_SESSION["login\_attempts"] = 0;  
}  
  
if ($\_SESSION["login\_attempts"] >= 3) {  
 die("Saiakera gehiegi, saiatu geroago..");  
}  
  
if (!$result || $result->num\_rows === 0) {  
 $\_SESSION["login\_attempts"]++;  
 header("Location: ".$\_SERVER["PHP\_SELF"]."?action=login");  
 exit;  
}  
```

## 3. CSRF Protekzio falta

Aurkitutako zaurgarritasuna: Formularioak ez du inolako mekanismorik ezartzen erasotzaile batek erabiltzailearen izenean eskaera maltzurrak bidal ez ditzan.

Kodean duen kokapena:  
  
```  
<form action="<?php echo $\_SERVER["PHP\_SELF"]."?action=login"; ?>" method="post">  
```  
  
Gomendioa: Inprimakiari token CSRF bat gehitzea eta eskabidea prozesatzean egiaztatzea.  
Kodean hitzez hitz egindako aldaketak:

```  
if (empty($\_SESSION['csrf\_token'])) {

$\_SESSION['csrf\_token'] = bin2hex(random\_bytes(32));

}

if (!isset($\_POST['csrf\_token']) || $\_POST['csrf\_token'] !== $\_SESSION['csrf\_token']) {

die("CSRF token inválido.");

}

Input hidden bat berifikatzen duena formularioa bidaltzerakoan zure sesiokoa dela.:

<input type="hidden" name="csrf\_token" value="<?php echo $\_SESSION['csrf\_token']; ?>" />

```

# 3.Search formularioa

## 1. SQL Injection

Index.php kodean, ez da erabiltzen ari SQL kontsultarako prestatutako sententzia bat, non datu-basean $\_GET ['keyword'] parametroa erabiliz bilatzen den. Kontsulta hau SQL injekzioarekiko kaltebera da.

```  
$stmt = mysqli\_query($conx,"SELECT \* FROM produktuak WHERE ". "izena LIKE " . $search . " OR " . "deskripzioa LIKE " . $search );  
```  
Gomendioa: Bind\_param bidez prestatutako sententziak erabili, SQL injekzioa saihesteko.

Egindako aldaketak:  
  
```  
$stmt = $conx->prepare("SELECT \* FROM produktuak WHERE izena LIKE ? OR deskripzioa LIKE ?");

$search = '%' . $testua . '%';

$stmt->bind\_param("ss", $search, $search);

$stmt->execute();

$result = $stmt->get\_result();  
```

## 2. CSRF Protekzio falta

Bilaketa-formularioak ez du CSRF tokenik ezartzen, eta, beraz, aplikazioa Cross-Site Request Forgery (CSRF) motako erasoen aurrean zaurgarria da.

Kodean duen kokapena:  
  
```  
<form name="search" method="get" action="<?php echo $\_SERVER['PHP\_SELF'] . '?action=search'; ?>" id="search"> <input type="text" value="" name="keyword"/> <input type="submit" name="search" value="Bilatu"/> </form>```  
  
Gomendioa: Formularioan CSRF token bat inplementatzea, CSRF motako erasoak saihesteko.

Kodean egindako aldaketak:

```  
if (empty($\_SESSION['csrf\_token'])) { $\_SESSION['csrf\_token'] = bin2hex(random\_bytes(32)); }

if ($\_POST['csrf\_token'] !== $\_SESSION['csrf\_token']) { die("CSRF token inválido."); }

Input hidden bat berifikatzen duena formularioa bidaltzerakoan zure sesiokoa dela.:

<input type="hidden" name="csrf\_token" value="<?php echo $\_SESSION['csrf\_token']; ?>" />

```  
  
3.XSS Erasoei bulnerablea

Erabiltzaileak emandako balioei ez zaie behar bezala ihes egiten ari orrian erakutsi aurretik, eta horrek Cross-Site Scripting (XSS) motako erasoekiko kaltebera uzten du aplikazioa.

´´´

echo "<h3>Ongi Etorri " . $\_SESSION['username'];

´´´

Gomendioa: Erabiltzaileen sarrera-datuei ihes egitea htmlspecialchars () erabilita, script maltzurrak exekutatzea prebenitzeko.

´´´

echo "<h3>Ongi Etorri " . htmlspecialchars($\_SESSION['username'], ENT\_QUOTES, 'UTF-8');

´´´

Baita ere, bilaketa-formularioak ez du behar bezala baliozkotzen keyword eremua. Arazo kritikoa ez den arren, jasotako datuak seguruak direla eta espero den formatuan daudela ziurtatzea gomendatzen da.

´´´

<input type="text" value="" name="keyword"/>

´´’

Gomendioa: Balidazio- edo sanitizazio-patroi bat erabiltzea, jasotako balioa espero zena izango dela bermatzeko. <input type="text" value="" name="keyword" pattern="[A-Za-z0-9\s]+" title="Solo se permiten caracteres alfanuméricos y espacios"/>