

# Apache

- [\[SECURITY\] http header & cookie](#)
- [\[Security\] Content-Security-Policy](#)
- [\[SECURITY\] Header !\[\]\(1207edb9a08751d3d55970560645ed23\_img.jpg\)](#)

# [SECURITY] http header cookie

Cookie Name	Description	Secure Flag	HttpOnly Flag
PHPSESSID	PHP Session ID	Yes	Yes
XSRF-TOKEN	CSRF Token	No	Yes
Laravel session_id	Laravel Session ID	No	No

XSRF-TOKEN is a Laravel session cookie

“

## PHPSESSID Secure Flag

- **php.ini** configuration

```
session.cookie_secure = On
```

- **PHP** configuration

```
session_start() // session cookie is secure:
```

```
ini_set('session.cookie_secure', 1);  
ini_set('session.cookie_httponly', 1); // HttpOnly flag  
session_start();
```

- **.htaccess** configuration
- **.htaccess** configuration PHP session cookie is secure:

php\_value session.cookie\_secure On

php\_value session.cookie\_httponly On

# [Security] Content-Security-Policy

Content-Security-Policy header is not set. It restricts resources the page can load and prevents XSS attacks.

seo □□ □ □ □□□ □□□ □ □□ □□□□.

# Content-Security-Policy(CSP)☐?

CSP HTTP □□ □ □□□, □□□□ □□□ □ □□□ □□ □ □□ □□□ □ □□ □□□.

1. **XSS**(`<script>alert('XSS')</script>`)  
`<script>` is JavaScript code that runs in the browser.
2. `<img src=x onmouseover=alert('XSS')>`  
This code (URL, CSS, etc.) runs in the browser.
3. **Clickjacking**  
`<iframe src=...>`

# Content-Security-Policy

CSP □ □ □ □ □ □ □ □ :

-  
-    .
- HTTPS    HTTP  .



# 🔒 🌐: Content-Security-Policy 🌐 🌐

## Apache 🌐 🌐 CSP 🌐

Apache🌐Content-Security-Policy 🌐.htaccess 🌐httpd.conf, apache2.conf)🌐 🌐 🌐 🌐 🌐.

```
<IfModule mod_headers.c>
Header set Content-Security-Policy "default-src 'self'; script-src 'self' https://www.googletagmanager.com
https://d3js.org https://cdnjs.cloudflare.com https://cdn.jsdelivr.net 'unsafe-inline' 'unsafe-eval'; style-src 'self'
https://fonts.googleapis.com https://cdn.jsdelivr.net 'unsafe-inline'; font-src 'self' https://fonts.gstatic.com
https://cdn.jsdelivr.net data:; connect-src 'self' https://www.google-analytics.com; img-src 'self' data:
https://cdn.jsdelivr.net;"
</IfModule>
```

### default-src 'self'

🌐 🌐 🌐 🌐 🌐 (🌐, 🌐, 🌐)🌐 🌐 🌐 🌐.

### self

🌐 🌐 🌐 🌐 🌐 🌐.

### unsafe-inline

🌐 🌐 🌐 (🌐 🌐 🌐).

### unsafe-eval

eval() 🌐setTimeout(string) 🌐 🌐 🌐 🌐.

### data:

🌐 URI🌐 🌐 🌐, 🌐 🌐 🌐 🌐.

\* 500 server error가 발생함.

## 2. Content Security Policy (CSP)

1. app/Http/Middleware에 class를 생성함.
2. handle function을 생성함. \$csp 값을 설정함.

```
public function handle(Request $request, Closure $next)
{
    $response = $next($request);

    $csp = "default-src 'self'; " .
        "script-src 'self' https://www.googletagmanager.com https://d3js.org https://cdnjs.cloudflare.com
https://cdn.jsdelivr.net 'unsafe-inline' 'unsafe-eval'; " .
        "style-src 'self' https://fonts.googleapis.com https://cdn.jsdelivr.net 'unsafe-inline'; " .
        "font-src 'self' https://fonts.gstatic.com https://cdn.jsdelivr.net data:; " .
        "connect-src 'self' https://www.google-analytics.com; " .
        "img-src 'self' data: https://cdn.jsdelivr.net;";

    $response->headers->set('Content-Security-Policy', $csp);

    return $response;
}
```

3. app/Http/Kernel.php에 등록함.

```
protected $middleware = [ \App\Http\Middleware\ContentSecurityPolicy::class, ];
```



## Content Security Policy

- `unsafe-inline` allows inline scripts, which is not recommended.
- `nonce` and `hash` are used to specify a unique key for the CSP. The key is generated and used to hash the content of the script.

## Nonce (key)

The nonce is a unique key generated for each script.

- apache

```
Header set Content-Security-Policy "default-src 'self'; script-src 'self' https://www.googletagmanager.com https://d3js.org https://cdnjs.cloudflare.com 'nonce-abc123'; style-src 'self' https://fonts.googleapis.com https://cdn.jsdelivr.net 'unsafe-inline'; font-src 'self' https://fonts.gstatic.com; connect-src 'self' https://api.example.com; img-src 'self' data;;"
```

- script

```
<script nonce="abc123"> console.log('Inline script with nonce'); </script>
```

- PHP Nonce

```
<?php
// php + javascript
$nonce = base64_encode(random_bytes(16));
header("Content-Security-Policy: script-src 'self' 'nonce-$nonce'");
?>
```

```
<script nonce="<?php echo $nonce; ?>">
    console.log('This script uses a nonce');
</script>
```

```
<script>
// only javascript
// Generate a random nonce
const array = new Uint8Array(16);
```

```
window.crypto.getRandomValues(array);
const nonce = btoa(String.fromCharCode.apply(null, array));

// Set the Content-Security-Policy header
const meta = document.createElement('meta');
meta.setAttribute('http-equiv', 'Content-Security-Policy');
meta.setAttribute('content', `script-src 'self' 'nonce-${nonce}'`);
document.head.appendChild(meta);

// Dynamically create a script element with the nonce
const script = document.createElement('script');
script.setAttribute('nonce', nonce);
script.textContent = "console.log('This script uses a nonce')";
document.body.appendChild(script);
</script>
```



# [SECURITY] Header

Header	OK	Notice	Warning	Critical	Recommendation
-----					
X-Frame-Options	0	0	77	0	X-Frame-Options header is not set. It prevents clickjacking attacks when set to 'deny' or 'sameorigin.'
X-Content-Type-Options	0	0	77	0	X-Content-Type-Options header is not set. It stops MIME type sniffing and mitigates content type attacks.
Referrer-Policy	0	0	77	0	Referrer-Policy header is not set. It controls referrer header sharing and enhances privacy and security.
Feature-Policy	0	0	77	0	Feature-Policy header is not set. It allows enabling/disabling browser APIs and features for security. Not important if Permissions-Policy is set.
Permissions-Policy	0	0	77	0	Permissions-Policy header is not set. It allows enabling/disabling browser APIs and features for security.
Server	0	0	77	0	Server header is set to known 'Apache.' It is better not to reveal used technologies.
Set-Cookie	70	65	70	0	Set-Cookie header for 'PHPSESSID' does not have 'SameSite' flag. Consider using 'SameSite=Strict' or 'SameSite=Lax.'  Set-Cookie header for 'XSRF-TOKEN' does not have 'HttpOnly' flag. Attacker can steal the cookie using XSS. Consider using 'HttpOnly' when cookie is not used by JavaScript.

“

# X-Frame-Options

📄:

- X-Frame-Options 📄 📄 📄.

📄:

Apache 📄 📄 📄 📄:

Header always set X-Frame-Options "DENY"

- **DENY:** iframe 📄 📄 📄.
- **SAMEORIGIN:** 📄 📄 📄 📄 iframe 📄 📄.

“

# X-Content-Type-Options

📄:

- X-Content-Type-Options 📄 📄 📄.

📄:

Apache 📄 📄 📄 📄:

Header always set X-Content-Type-Options "nosniff"

- MIME 📄 📄 📄.

# Referrer-Policy

📄:

- Referrer-Policy 📄 📄 📄.

📄:

Apache 📄 📄 📄 📄:

Header always set Referrer-Policy "no-referrer"

- **no-referrer:** 📄 📄 📄 📄.
- 📄 `strict-origin`, `strict-origin-when-cross-origin`, `same-origin` 📄.

“

## Feature-Policy / Permissions-Policy

📄:

- Feature-Policy 📄 Permissions-Policy 📄 📄 📄.

📄:

Apache 📄 📄 📄 📄:

Header always set Permissions-Policy "geolocation=(), camera=(), microphone=()"

- **Permissions-Policy:** 📄 📄 📄 (API) 📄 📄.
- 📄 `geolocation`, `camera`, `microphone` 📄.

# Server

🔍:

- Server 🇯🇵 Apache🇯🇵 🇯🇵 🇯🇵 🇯🇵.

🔍:

Apache 🇯🇵 🇯🇵 🇯🇵 🇯🇵:

```
ServerTokens Prod
ServerSignature Off
Header unset Server
```

- **ServerTokens Prod:** 🇯🇵 🇯🇵 🇯🇵.
- **ServerSignature Off:** 🇯🇵 🇯🇵 🇯🇵 Apache 🇯🇵 🇯🇵.
- **Header unset Server:** Server 🇯🇵 🇯🇵.

“

## Set-Cookie

🔍:

- SameSite [ HttpOnly 🇯🇵 🇯🇵.

🔍:

Apache 🇯🇵 🇯🇵 🇯🇵 🇯🇵:

```
Header always edit Set-Cookie ^(.*)$ "$1; SameSite=Strict; HttpOnly; Secure"
```

- **SameSite=Strict:** 🇯🇵 🇯🇵 🇯🇵 🇯🇵 🇯🇵.
- **HttpOnly:** JavaScript🇯🇵 🇯🇵 🇯🇵 🇯🇵.
- **Secure:** HTTPS 🇯🇵 🇯🇵 🇯🇵.



#### # X-Frame-Options []

Header always set X-Frame-Options "DENY"

#### # X-Content-Type-Options []

Header always set X-Content-Type-Options "nosniff"

#### # Referrer-Policy []

Header always set Referrer-Policy "no-referrer"

#### # Permissions-Policy []

Header always set Permissions-Policy "geolocation=(), camera=(), microphone=()"

#### # Server [] []

ServerTokens Prod

ServerSignature Off

Header unset Server

#### # Set-Cookie [] []

Header always edit Set-Cookie ^(.\*)\$ "\$1; SameSite=Strict; HttpOnly; Secure"