Guide Book

For Phield

- A Basic Security Tool -

About Phield Tool

Phield is a PHP library specifically designed for the CodeIgniter framework, integrating a variety of foundational security functions aimed at enhancing the overall security of applications.

Tool Set

Please place the **phield.php** file in the **application/libraries** folder.

Tool Apply

Phield requires loading the library file in the application before using its functions.

There are two ways to load the library:
1. Global Loading
Add **$autoload['libraries'] = array('library_name')** in **application/config/autoload.php** file.

2. On-Demand Loading
In the location where it is needed, use **$this->load->library('phield')**.

Feature: HSTS

1. Feature Introduction

HTTP Strict Transport Security (HSTS) is a web security policy that enforces all HTTP requests to be redirected to HTTPS, ensuring that data transmission between the browser and the server is always encrypted.

Once HSTS is enabled, the browser will automatically switch all subsequent requests to HTTPS, protecting data from eavesdropping and tampering.

2. Feature Apply

To ensure the feature functions fully, we need to configure both the server and the web application.

2-1. Server Configuration

Using Apache as an example, add the following line to **conf/extra/httpd-ssl.conf**:

```
Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains"
```

The parameter values can be adjusted based on requirements.

2-2. Web Application Configuration

2-2-1. Set the base_url

Modify the following setting in **application/config/config.php**:

```
$config['base_url'] = 'https://' . 'web_url';
```

2-2. Configure the .htaccess File

Add the following lines to **.htaccess** file:

```
RewriteCond %{HTTPS} !=on
```

```
RewriteRule ^(.*)$ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]
```

3. Vulnerability Prevented by the Feature

HSTS enforces the use of HTTPS connections, effectively addressing common security threats related to data transport security.

One of the most common attacks is the Man-in-the-Middle Attack (MITM). This attack occurs when an attacker intercepts, modifies, or forges information between two communicating parties. The attacker can steal sensitive data, alter the transmitted content, or even impersonate one of the parties to carry out malicious actions, thus posing significant risks to users or systems.

Feature: Input Filter

1. Feature Introduction

Input Filter is a security mechanism used to validate and filter user input data, preventing the injection of malicious code. By enabling the Input Filter feature, it effectively prevents attackers from embedding malicious code in input, thereby protecting against common security threats such as Cross-Site Scripting (XSS) and SQL Injection attacks.

Once the Input Filter is enabled, illegal symbols in user input will be removed and replaced with [removed].

2. Feature Apply

Please use **$input = $this->phield->input_filter($input)** to filter the input.

3. Vulnerability Prevented by the Feature

Input Filter addresses and mitigates two major vulnerabilities by filtering user input: XSS attacks and SQL injection attacks.

Cross-Site Scripting (XSS) occurs when an attacker injects malicious JavaScript code into a page, often through unprotected forms, URL parameters, or comment systems. XSS attacks can lead to session hijacking or the leakage of sensitive information.

SQL injection is an attack where an attacker embeds malicious SQL code into user input, allowing them to control SQL queries executed by the application. This can lead to data theft, database corruption, or gaining administrator privileges. SQL injection is one of the most common attacks in database applications.