

# [Root-me]PHP - Unserialize overflow

## PHP - Unserialize overflow - Medium (40 pts)

Bài cung cấp source code nên chúng ta sẽ cùng phân tích qua một tí về source code: #

Chúng ta có một class User, trong class có các thuộc tính protected: username, password, \_logged, email.

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

Các hàm trong class:

- setLogged(): cập nhập trạng thái login
- isLogged(): kiểm tra xem user đã login hay chưa
- getUsername(): lấy username
- getPassword(): lấy password

Các hàm chính:

### storeUserSession:

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

Hàm này thực hiện serialize object được tạo từ thông tin username và password do người dùng nhập vào, sau đó thực hiện thay đổi Null\_byte\*Null\_byte thành \0\0\0 và cập nhập \$\_SESSION['user'] thành giá trị trên

### getUserSession()

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

Trước tiên nếu như `isset($_SESSION['user'])` thì sẽ thực hiện lấy giá trị `$_SESSION['user']` thay đổi `\0\0\0` thành `null_byte*null_byte`, sau đó thực hiện unserialize giá trị trên thu được object user. Ngược lại nếu như `$_SESSION['user']` chưa được set thì sẽ tạo object user với username là 'guest' và password là ''. Kết quả trả về là object user.

### Đoạn code chính

```

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()) === 'b3b7b663909f8e9b4e2a581337159e8';
    $currentUser->setLogged($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();
}

```

#### Method POST

Ở đây chúng ta thấy giá trị username và password được nhập vào với method POST.

Sau đó, sẽ thực hiện tạo một object thuộc class User với giá trị username và password phía trên.

Nếu object này có username=admin và password = sha512() thì sẽ login thành công (gọi hàm setLogged() và bật \_logged lên true)

Gọi hàm storeUserSession() để set giá trị cho session.

#### Method GET:

Lấy thông tin của object user hiện tại bằng cách gọi đến hàm getUserSession()

Nếu như isLogged() thì flag sẽ được in ra.

Tóm lại, nhiệm vụ của chúng ta là cần phải làm cách gì đó để thuộc tính \_logged có giá trị là true.

Ở đây có một kiến thức chúng ta cần lưu ý là trong php, trong giá trị serialized, null\_byte\*null\_byte sẽ đứng trước thuộc tính protected. Đó là lý do tại sao có sự xuất hiện của việc thay đi thay lại null\_byte\*null\_byte->\0\0\0

Nhìn trong đoạn code thì có vẻ việc thay đi rồi thay lại thì khá bình thường nhưng thực chất là không.

Ví dụ ở đây mình nhập giá trị cho username là \0\0\0, thì lúc này lúc serialize thuộc tính username sẽ có số byte là 6. Nhưng sau khi giá trị serialize được lưu vào session và sau đó trước khi unserialize thì tất cả \0\0\0 sẽ được thay thế thành null\_byte\*null\_byte tức chỉ 3 byte, mà trong giá trị serialized vẫn bị 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 xảy ra và không thể unserialize.

Mình thực hiện dựng lại ở local để test:

- serialized object:

```
0:4: "User": 4: {s:12: "_username";s:7: "\0\0\0";s:12: "_password";s:2: "hi";s:10: "_logged";b:0;s:9: "_email";s:0: "";
```

- Giá trị sau thay thế:

```
0:4:"User":4:{s:12:"\0\0\0 username";s:7:"\0\0\0";s:12:"\0\0\0 password";s:2:"hi";s:10:"\0\0\0 logged";b:0;s:9:"\0\0\0_email";s:0:";}
```

- Giá trị sau thay thế để unserialize;