Web: Saltfish

The content of the web page is the following PHP script:

```php
<?php
  require_once('flag.php');
  if ($_ = @$_GET['pass']) {
    $ua = $_SERVER['HTTP_USER_AGENT'];
    if (md5($_) + $_[0] == md5($ua)) {
      if ($_[0] == md5($_[0] . $flag)[0]) {
        echo $flag;
      }
    }
  } else {
    highlight_file(__FILE__);
  }
```

We need to carefully bypass every *if* statement and there are three. In order to solve this challenge I used an online PHP tester as debugger / tester.

BLOCK 1

The first block is just assigning the value contained in the GET request parameter “pass” in the variable $_. When we call the service and we assign any value to this variable, the block is bypassed.

BLOCK2

Here it’s a bit more complex and we need to understand what’s going on.
Two variables are involved:

- $_, the password that we provided;
- $ua: it is given by the parameter HTTP_USER_AGENT through the global variable $_SERVER.

For example, let’s see how an md5 looks like (I use an online PHP tester); for example, given the following code:

```
$tmp = MD5("a");
echo $tmp;
```

The result is:

```
0cc175b9c0f1b6a831c399e269772661
```

If we try to do the sum with a character, like with the following code:

```
$tmp2 = $tmp + "a";
echo $tmp2;
```

The result is:

0

Ok, that's suspicious: this code is full of those things called *type juggling*.
The comparison is between the addition of the md5 of the password and the first letter of the password with the md5 of the field contained in the user agent. Here we are exploiting odd PHP's behaviours, and this comparison returns True on several conditions, e.g., MD5("a") + "a" == Md5("a") is True.

BLOCK 3

In the last block we check if the first character is equal to the MD5 of the string concatenation between the first character of the password and the string flag.
Now, we can have a lot of thoughts, like how it works etc…
I'll go in an opposite direction, and my clues are:

- There must be at least one password that guarantees us the authentication;
- Only the first character of this password matters.

Say no more! Since we only need one character, why don't we just try all the possible characters? Yes, this is a brute-force.

I designed the following python3 code:

```
import requests
import string


for i in string.ascii_letters:
    url = f"http://127.0.0.1:124/?pass={i}"
    r = requests.get(url, headers={'User-Agent': str(i)})
    print(r.text)
```

Where the flag is obtain with the letter 'b':

35c3_password_saltf1sh_30_seconds_max