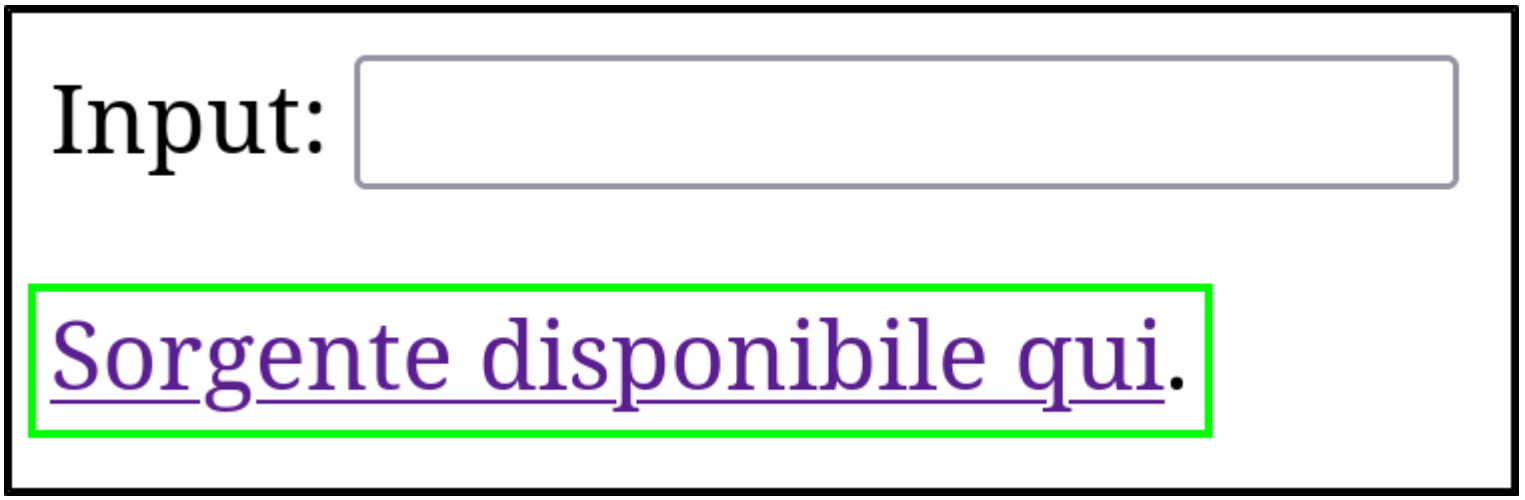
OLIcyber – ConfuseMe
PHP Code Analysis



In this challenge, the website includes an input field as well as a link to the site's PHP source code

OLIcyber – ConfuseMe
PHP Code Analysis

```php
if (isset($_GET['input'])) {
  $user_input = $_GET['input'];

  if ($user_input == substr(md5($user_input), 0,
    echo "Ce l'hai fatta! Ecco la flag: $flag";
```

In the source code, we see that if we supply user input that is equal to its own MD5 hash value, then we'll receive the flag

OLIcyber – ConfuseMe
PHP Code Analysis

```
└─$ echo -n 'apple' | md5sum
1f3870be274f6c49b3e31a0c6728957f
```

For example, if we hash the word `apple` using the MD5 method, the result is `1f3870be274f6c49b3e31a0c6728957f`

OLIcyber – ConfuseMe
PHP Type Juggling

```
if ($user_input == substr(md5($user_input)
```

In the PHP language, the loose comparison (==) operator can be abused to create two different hash values that, when evaluated by PHP are considered the same

OLIcyber – ConfuseMe
PHP Type Juggling

```php
<?php
$a = 240610708;
$b = "240610708";

if ($a == $b) {
    echo "In loose comparison, these are the same.\n";
```

The classic use of the PHP loose comparison operator is to compare different data types to each other, such as comparing strings of numbers against integer numbers

OLIcyber – ConfuseMe
PHP Type Juggling

```php
<?php
$a = "0e12345";
$b = 0e09876;

if ($a == $b) {
    echo "In loose comparison, these are the same.\n";
```

The quirk of PHP type juggling is that all numbers in scientific notation that begin with the number zero (0e###) are considered equal

OLIcyber – ConfuseMe
PHP Type Juggling

```
└─$ echo -n '0e215962017' | md5sum
0e291242476940776845150308577824
```

So to solve this challenge, we need to supply user input that both starts with the string 0e, followed by numbers, and returns the value 0e, followed by numbers when hashed with MD5

OLIcyber – ConfuseMe
PHP Type Juggling

```
└─$ echo -n 'apple' | md5sum
1f3870be274f6c49b3e31a0c6728957f
```

This means that solving this challenge naturally is impossible, but there's a vulnerability in earlier versions of the PHP language that makes this challenge solvable