6letter-juggler – PHP Type Juggling

```php
if (strcmp($_GET['login'], 'unknownuser') == 0 &&
strcmp($_GET['password'], 'unknownpassword') == 0)
{ // do stuff as authenticated user }
```

The web application is written in PHP, and probably uses a line of code similar to the above for authenticating the user

6letter-juggler – PHP Type Juggling

```
if (strcmp($_GET['login'], 'unknownuser') == 0 &&
strcmp($_GET['password'], 'unknownpassword') == 0)
{ // do stuff as authenticated user }
```

However, there's a vulnerability in PHP when using the “loose” comparison operator ==, such that data of different types can be compared to each other, e.g., integers compared to strings

6letter-juggler – PHP Type Juggling

```
"123abc" == 123    // true
```

However, there's a vulnerability in PHP when using the “loose” comparison operator ==, such that data of different types can be compared to each other, e.g., integers compared to strings

6letter-juggler – PHP Type Juggling

```
"123abc" == 123    // true
```

When PHP compares data of different types, there can be some strange results, such as the string of numbers and letters comparing true for the number 123 in the example above

PHP Type Juggling – Strcmp

```php
<?PHP
if (strcmp('hello', []) == 0) {
echo "Returned true!";
} else {
    echo "This won't execute\n";
}
?>
```

The PHP Strcmp function is used to compare two strings, but type juggling problems come up if you use the Strcmp function to compare strings to other data types

PHP Type Juggling – Strcmp

```
if (strcmp($_GET['login'], 'unknownuser') == 0 &&
strcmp($_GET['password'], 'unknownpassword') == 0)
{ // do stuff as authenticated user }
```

So if the PHP app is checking passwords by using the Strcmp function with loose comparisons, we can bypass it by supplying an array instead of a string for the password