Broken Authentication pada Aplikasi PHP yang menggunakan Loose Comparison Hash

Achmad Fahrurrozi Maskur 13515026
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl Ganesha 10 Bandung 40132, Indonesia
13515026@std.stei.itb.ac.id

***Abstrak* — Bahasa Pemrograman PHP merupakan salah satu bahasa yang banyak digunakan dalam membuat suatu aplikasi berbasis web. Dalam pengembangannya, salah satu aspek penting yaitu keamanan data pengguna khususnya pada bagian autentikasi. Dalam mencapai suatu autentikasi yang baik, kata sandi terlebih dahulu di-*hash* kemudian disimpan di dalam basis data. Namun terdapat beberapa kesalahan yang dapat terjadi apabila dilakukan autentikasi pada PHP dengan menggunakan *loose comparison* pada kata sandi yang di-*hash* yang disebut dengan *magic hashes*. Pada makalah ini akan dibahas bagaimana kelemahan tersebut dapat terjadi dan apa itu *magic hashes*.**

Kata kunci — PHP; Authentication; Hash; Magic Hashes

I. PENDAHULUAN

Bahasa pemrograman PHP merupakan bahasa yang cukup mudah dipahami dan juga popular pada saat ini, banyak aplikasi web yang dibangun dengan menggunakan bahasa pemrograman PHP. PHP cukup mudah dipahami dikarenakan oleh sifat yang cukup dinamis yang dimiliki oleh PHP dibuktikan dengan tidak dibutuhkannya pendeklarasian tipe pada saat mendeklarasikan *variable*.

Dibalik kemudahan PHP tersebut, terdapat beberapa hal yang harus "dikorbankan" seperti aspek kerapihan kode maupun keamanan dari aplikasi tersebut. Salah satu hal yang cukup menarik yaitu adanya *type juggling* yang dapat menentukan tipe dari suatu *variable* secara otomatis.

Selain itu PHP juga memberikan opsi pada operasi perbandingan. Terdapat dua opsi, yaitu perbandingan yang longgar atau disebut *loose comparison* dan perbandingan yang ketat atau disebut *strict comparison*.

Dari fitur-fitur yang disediakan oleh PHP ini, dapat ditemukan suatu kerentanan yang fatal yang menyebabkan terjadinya *broken authentication*. Selanjutnya akan dibahas pada Bab II terkait teori-teori dasar yang akan digunakan.

II. DASAR TEORI

A. PHP: Hypertext Preprocessor

PHP merupakan satu dari sekian banyak bahasa pemrograman yang cukup populer khususnya dalam pengembangan aplikasi berbasis web. PHP pertama kali dikembangkan oleh Rasmus Lerdorf pada tahun 1995. PHP merupakan bahasa pemrograman *server-side* yang didesain untuk pengembangan web.

PHP disebut *server-side* karena diproses pada komputer server penyedia web, bukan pada klien yang mengakses web tersebut. Sehingga klien tidak dapat melihat kode sumber dari bahasa pemrograman PHP, berbeda dengan pemrograman *client-code* seperti Javascript.

Menurut Wikipedia pada tahun 2014, sekitar 82% dari web yang ada di dunia menggunakan PHP. Hal ini yang menyebabkan PHP merupakan bahasa pemrograman yang cukup populer, termasuk karena kemudahan dalam menggunakan bahasa pemrograman PHP. PHP juga menjadi basis dari bahasa pemrograman aplikasi CMS (*content management system*) yang populer seperti Wordpress, Joomla, dll.

B. PHP Type Juggling and Comparison

PHP tidak membutuhkan atau tidak mendukung pendefinisian tipe pada saat mendeklarasikan *variable*, berbeda pada bahasa pemrograman pada umumnya yang dimana tiap *variable* harus dideklarasikan tipe-nya baik itu *interger*, *string*, atau yang lainnya. Oleh karena itu, PHP memiliki fitur yang bernama *type juggling*.

Type juggling lah yang menentukan seperti apa tipe dari suatu *variable*. Dapat dilihat pada kode dibawah ini contoh bagaimana PHP melakukan *type juggling*.

```php
<?php
  $var = "1";           // $var = string (ASCII 49)
  $var *= 2;            // $var = integer (2)
  $var = $var * 1.3;    // $var = float (2.6)
  $var = 5 * "1 Halo"; // $var = integer (5)
  $var = 5 * "2 Anak"; // $var = integer (10)
?>
```

Akibat dari adanya *type juggling*, maka PHP memiliki dua macam tipe *comparison* (perbandingan). Perbandingan yang pertama adalah *loose comparison*, dan yang kedua yaitu *strict comparsion*.

Loose comparison dilakukan dengan menggunakan dua buah karakter sama dengan (== atau !=). *Loose comparison* membandingkan kedua nilai dengan sifat yang *loose* atau dalam Bahasa Indonesia-nya yaitu longgar. Maksud dari longgar yaitu

kedua nilai bisa saja bernilai TRUE walaupun memiliki tipe yang berbeda. Contohnya yaitu *integer 1* dengan *string "1"* akan bernilai TRUE apabila dibandingkan dengan *loose comparison*. Berikut tabel perbandingan dengan menggunakan *loose comparison*.

Loose comparisons with ==

	TRUE	FALSE	1	0	-1	"1"	"0"	"-1"	NULL	array()	"php"	""
TRUE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE
FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE
1	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
0	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE
-1	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
"1"	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
"0"	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
"-1"	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
NULL	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	TRUE
array()	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE
"php"	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE
""	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE

Gambar 1. Tabel perbandingan dengan menggunakan *loose comparison.*

Berbeda dengan *loose comparison*, *strict comparison* dilakukan dengan menggunakan tiga buah karakter sama dengan (=== atau !==). *Strict comparison* membandingkan kedua nilai dengan sifat yang *strict* atau dalam Bahasa Indonesia-nya yaitu ketat. Maksud dari ketat yaitu kedua nilai harus memiliki tipe bentukan yang sama terlebih dahulu sebelum dibandingkan nilainya. Contohnya yaitu *integer 1* dengan *string "1"* akan bernilai FALSE karena keduanya memiliki tipe yang berbeda. Berikut tabel perbandingan dengan menggunakan *strict comparison*.

Strict comparisons with ===

	TRUE	FALSE	1	0	-1	"1"	"0"	"-1"	NULL	array()	"php"	""
TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
1	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
0	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
-1	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
"1"	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
"0"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
"-1"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
NULL	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE
array()	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
"php"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE
""	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE

Gambar 2. Tabel perbandingan dengan menggunakan *strict comparison.*

C. Security (autentikasi)

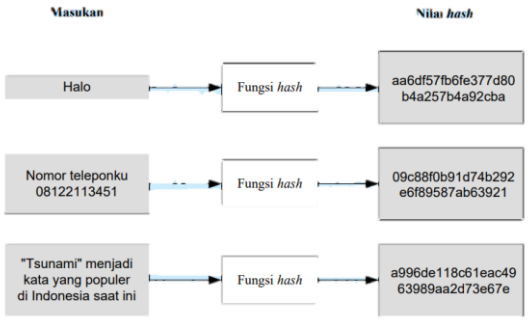
Dalam pembangunan sebuah aplikasi berbasis web, khususnya aplikasi yang bukan web statis, autentikasi merupakan salah satu hal yang cukup penting. Autentikasi digunakan untuk membuktikan bahwa orang tersebut benar dapat masuk ke dalam sistem.

Umumnya, autentikasi terbagi menjadi tiga faktor, yaitu faktor pengetahuan (*something what you **knows***), faktor kepemilikian (*something what you **have***), serta faktor sifat (*something what you **are***). Contoh dari faktor pengetahuan yaitu kata sandi, PIN, nama ibu kandung, dll. Contoh dari faktor kepemilikan yaitu kartu, nomor telepon, dll. Sedangkan faktor sifat yaitu seperti sidik jari, dan hal-hal yang berbau biometric.

Aplikasi berbasis web umumnya menggunakan faktor yang paling sederhana yaitu faktor pengetahuan seperti kata sandi. Biasanya suatu aplikasi berbasis web menggunakan dua parameter untuk melakukan autentikasi yaitu username/email dan kata sandi. Agar kata sandi pengguna tidak mudah diketahui oleh pengguna lain, maka kata sandi terlebih dahulu di-*hash* dan kemudian hasilnya disimpan dalam basis data aplikasi.

D. Fungsi Hash

Fungsi *hash* adalah fungsi yang menerima masukan berupa *string* yang panjangnya sembarang kemudian menghasilkan keluaran berupa suatu *string* yang baru. Keluaran tersebut memiliki panjang yang tetap untuk setiap masukan yang diberikan. Hasil dari keluaran fungsi *hash* bisa disebut dengan *message digest* atau *hash value* dalam bentuk hex, yaitu memiliki set karakter a-f dan 0-9.



Gambar 3. Contoh masukan dan hasil dari fungsi hash

Fungsi *hash* merupakan fungsi satu-arah (*one-way function*). Fungsi satu-arah yang dimaksud yaitu pesan yang sudah di-*hash* menjadi *message digest* tidak dapat dikembalikan menjadi seperti sebelumnya (*irreversible*).

Sifat lain dari fungsi *hash* yaitu untuk setiap masukan *string*, tidak mungkin ada *string* lain yang memiliki hasil *hash* yang sama. Sehingga hal ini yang menjadikan salah satu kegunaan dari fungsi *hash* yaitu untuk menjamin integritas dari sebuah data.

Namun pada saat ini, beberapa algoritma fungsi hash sudah ditemukan kolisinya, yaitu keadaan dimana dua buah *string* memiliki nilai *hash* yang sama. Beberapa fungsi hash yang cukup terkenal yaitu MD5, SHA-1, SHA-2.

E. Magic Hashes

Diberi nama *magic hashes* karena sifatnya yang unik. Uniknya yaitu terdapat pada *hash* yang dihasilkan. Karena *hash* memiliki set karakter yaitu a-f dan 0-9, maka bisa saja terdapat hash yang berbentuk "0e1245…" atau *hash* yang hanya memiliki angka serta diawali dengan 0e.

Magic hashes ditemukan pada tahun 2010 oleh Gregor Kopf, kemudian pada tahun 2013 oleh Tyler Borland dan dilanjut pada tahun 2014 oleh Michael Spacek dan Jos Wetzels. Permasalahan yang terjadi yaitu ketika sebuah *magic hashes* digunakan dalam melakukan perbandingan terhadap suatu *string* yang lain maka dapat terjadi sesuatu yang tidak diinginkan. Hal tersebut akan dibahas pada bab selanjutnya yaitu bab III.

III. Analisis dan Pembahasan

Hampir semua bahasa pemrograman menggunakan *syntax* dua sama dengan (==) untuk melakukan perbandingan, oleh karena itu, kebanyakan *software developer* sudah terbiasa menggunakan operasi tersebut, pun di bahasa pemrograman

PHP yang merupakan *loose comparison*. Hal ini menyebabkan banyaknya terjadi masalah. Salah satu nya yaitu adanya *magic hashes*.

Apabila *magic hashes* dibandingkan dengan menggunakan *loose comparison* pada sebuah string "0" atau integer 0 atau *magic hashes* lainnya yang berbeda, maka akan mengembalikan nilai TRUE. Hal ini dikarenakan *magic hashes* akan diinterpretasikan sebagai nilai 0.

```php
<?php

    $hash1 = hash("md5", "240610708", false);
    $hash2 = hash("md5", "QLTHNDT", false);

    print "hash1: $hash1\n";
    print "hash2: $hash2\n";

    if ($hash1 == $hash2) {
        print "hash1 dan hash2 sama\n";
    }

?>
```

Gambar 4. Kode sumber perbandingan antara dua *magic hashes*

```
um Crypto$ php tes.php
hash1: 0e462097431906509019562988736854
hash2: 0e405967825401955372549139051580
hash1 dan hash2 sama
```

Gambar 5. Hasil eksekusi dari kode gambar 4

Dapat dilihat pada contoh diatas, hash1 dan hash2 bernilai TRUE apabila dilakukan perbandingan. Hal ini disebabkan oleh hasil dari *hash* kedua *string* tersebut merupakan *magic hashes* karena diawali dengan 0e dan hanya memiliki angka. Sehingga, bahasa pemrograman PHP akan menginterpretasikan nilai dari *string* tersebut sama dengan nol.

Salah satu cara untuk melakukan pengecekan apakah web tersebut rentan terhadap hal ini yaitu pertama-tama dilakukan pendaftaran dengan kata sandi *magic hash*, kemudian dilakukan percobaan *login* menggunakan kata sandi *magic hash* yang berbeda. Apabila berhasil, maka web tersebut rentan terhadap serangan ini.

Untuk menemukan *magic hashes* tidaklah sulit, dapat dilakukan teknik *brute force* atau mencoba satu-satu *string* acak, kemudian cek apakah hasil dari *hash* memiliki awalan 0e dan hanya memiliki angka.

Beberapa fungsi *hash* sudah ditemukan sebuah *string* yang jika di-*hash* akan menghasilkan sebuah *magic hash*. Berikut tabel fungsi *hash* serta *magic hash* yang dihasilkan.

Tipe *hash*	Panjang *hash*	*String*	*Magic hashes (8byte pertama)*
MD2	32	505144726	0e015339…
MD4	32	48291204	0e266546…
MD5	32	240610708	0e462097…
SHA1	40	10932435112	0e077669…
RIPEMD128	32	315655854	0e251331…
RIPEMD160	40	20583002034	00e18390…
TIGER160,3	40	13181623570	00e47060…
CRC32	8	2332	0e684322
FNV164	16	8338000	0e738457…
HAVAL128,3	32	809793630	00e38549…
HAVAL128,4	32	71437579	0e016970…

Pada tahun 2015, Scott Arciszewski menemukan kerentanan *magic hash* pada CMS Joomla seperti yang dapat dilihat pada link github https://github.com/joomla/joomla-cms/issues/8326. Kerentanan tersebut menyebabkan terjadinya *bypass* terhadap pengecekan yang terjadi.

IV. Kesimpulan dan Saran

Broken Authentication merupakan hal yang sangat fatal pada proses *login*, namun hal ini disebabkan hanya oleh hal yang sangat remeh, yaitu pada perbedaan operasi perbandingan yang dilakukan. Sebaiknya selalu gunakan *strict comparison* untuk menghindari hal-hal yang tidak diinginkan.

Hal yang harus dilakukan untuk memperbaiki hal ini hanyalah menambahkan satu simbol sama dengan, dari "==" menjadi "===" dan "!=" menjadi "!==". Selain itu, juga dapat digunakan fungsi bawaan dari PHP yaitu "*hash_equals*" jika ingin membandingkan nilai *hash.*

V. Ucapan Terima Kasih

Puji syukur kita panjatkan kepada Tuhan Yang Maha Esa karena atas berkat dan rahmatnya penulis masih sempat menulis makalah ini, tidak lupa pula kita haturkan shalawat kepada Nabi Muhammad SAW. Terima kasih juga kepada kedua orang tua penulis yang tak pernah lupa untuk mendoakan penulis. Penulis juga berterima kasih kepada Bapak Rinaldi Munir, selaku dosen mata kuliah IF4020 Kriptografi yang telah ikhlas mengajar dan menginspirasi penulis untuk terus berkarya. Kepada teman-teman yang membantu dalam pembuatan makalah ini, penulis juga mengucapkan terima kasih atas bantuannya.

Referensi

[1] https://www.duniailkom.com/pengertian-dan-fungsi-php-dalam-pemograman-web/ diakses pada tanggal 3 Mei 2019.

[2] Munir, Rinaldi. 2018. Fungsi Hash. Program Studi Teknik Informatika Institut Teknologi Bandung.

[3] https://www.php.net/manual/en/language.types.type-juggling.php diakses pada tanggal 3 Mei 2019.

[4] https://www.php.net/manual/en/types.comparisons.php diakses pada tanggal 3 Mei 2019.

[5] https://www.whitehatsec.com/blog/magic-hashes/ diakses pada tanggal 15 April 2019.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 10 Mei 2019

Achmad Fahrurrozi Maskur
13515026