Penggunaan Fungsi *Hash MD5* dalam Enkripsi *PHP Session* dan *Cookies*

Luthfi Hamid Masykuri and 13512100[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]luthfi_hamid_m@itb.ac.id

***Abstrak*—Makalah ini membahas tentang aplikasi fungsi hash MD5 dalam PHP Session dan Cookies. Fungsi hash adalah fungsi yang bagus digunakan dalam enkripsi suatu informasi. Di dalam makalah ini akan dijelaskan bagaimana fungsi hash MD5 bekerja. Kemudian dijelaskan juga PHP Session dan Cookies. Selanjutnya implementasi enkripsi fungsi hash MD5 dalam PHP Session dan Cookies. Enkripsi yang bersifat satu arah ini akan membantu dalam mengamankan suatu informasi.**

***Kata Kunci*—enkripsi, hash, session, cookie.**

I. PENDAHULUAN

Dunia teknologi informasi di masa sekarang mengalami kemajuan yang sangat pesat. Terutama dalam teknologi internet. Semuanya sekarang identik dengan dunia *online*. Mulai dari belanja, surat-menyurat, forum dan masih banyak lagi fasilitas *online.*

Dunia *online* erat kaitannya dengan *website.* Karena konten-konten yang ada disematkan dalam *website* agar bisa diakses dengan mudah.

Dalam beberapa situs kita sering diminta untuk melakukan *login* dengan akun yang kita miliki di situs tersebut. Dalam *login* ini kita bisa melakukan *auto login.* Proses ini dalam sebuah *website* erat kaitannya dengan *PHP Session* dan *Cookies.*

Tapi dua hal tersebut dapat menjadi celah orang *jahil* untuk mencuri akun kita. Tentu hal tersebut sangat berbahaya. Apalagi kalau akun kita diacak-acak oleh orang tak bertanggung jawab.

Misalnya saja akun *facebook* kita yang sudah kia dicuri kata sandinya. Kemudian akun kita digunakan untuk mem-*post* sesuatu di luar kehendak kita yang bisa saja *postingan* tersebut bisa menjatuhkan nama baik kita. Lebih ekstirm lagi jika yang dicuri adalah informasi akun rekening *online* bank. Uang yang ada di dalam rekening tersebut bisa dikuras oleh pencuri.

Penulis di sini akan membahas bagaimana mengamankan informasi yang di simpan dalam *session* dan *cookies* yang menyimpan data *login.* Di mana data *login* ini akan digunakan situs untuk memproses *auto login.*

Informasi tersbut akan diamankan dengan cara dienkripsi. Banyak metode enkripsi untuk megamankan informasi. Penulis akan menggunakan metode funsi *hash MD5.* Fungsi *hash* ini banyak digunakan dalam enkripsi informasi. Penggunaan fungsi ini dalam membangun website pun cukup mudah, karena secara *default* dalam bahasa *PHP* fungsi ini sudah ada.

II. LANDASAN TEORI

A. Fungsi Hash

Hash adalah teknik klasik dalam ilmu komputasi yang sudah digunakan. *Hash* merupakan suatu metode yang secara langsung mengakses *records* dalam suatu table dengan melalukan transformasi yang mengambil input ukuran dengan ukuran tertentu dan mengembalikan sebuah string berukuran tetap yang disebut sebagai nilai *hash*. Fungsi *hash* dapat dirumuskan sebagai berikut :

$$h = H(x)$$

h : nilai *hash* yang panjangnya tetap
H : fungsi *hash*
x : input dari fungsi

Dengan adanya rumus sederhana di atas, maka yang dibutuhkan untuk fungsi *hash* adalah sebagai berikut

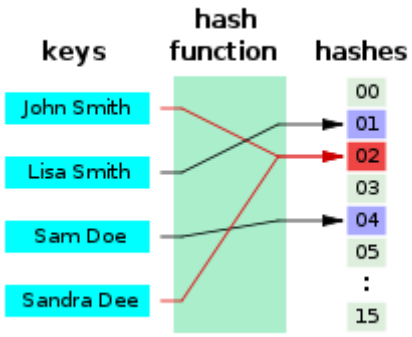
1. Input dengan panjang sembarang
2. Hasilnya mempunyai keluaran dengan panjang yang tetap
3. H(x) umumnya mudah dikalkulasi untuk sembarang nilai x
4. H(x) adalah satu arah
5. H(x) tidak pernah bermasalah dengan yang lain

Fungsi *hash* merupakan fungsi yang bersifat satu arah. Hal ini berarti akan sulit sekali mencari nilai balikan dari hasil fungsi ini. Sehingga kita sulit untuk menemukan nilai input x yang memenuhi persamaan H(x) = h.

Jika diberikan sebuah pesan x untuk dikompresi dalam suatu fungsi *hash*, kita akan sulit mencari pesan y yang akan menyamai dari nilai H(x) yang berarti sulit untuk mencari H(x) = H(y). Maka dapat kita katakana bahwa fungsi *hash* x tidak mungkin bertabrakan dengan fungsi *hash* y. Bisa dikatakan akan sangat sulit untuk menemukan kesamaan 2 pesan dalam fungsi *hash.* Jadi fungsi *hash* adalah fungsi yang bersifat *strongly collision-free* (bebas dari tabrakan).

Collision sendiri merupakan gejala tabrakan yang terjadi

ketika dua input menghasilkan hash yang sama. Tabrakan ini dapat dihindari dengan *Collision Resolution. Collision Resolution* merupakan proses untuk menangani kejadian dua atau lebih key di-*hash* ke alamat table yang sama. Cara yang dilakukan jika terjadi *collision* adalah mencari lokasi yang kosong dalam tabel *Hash* secara terurut. Cara lainnya adalah dengan menggunakan fungsi Hash yang lain untuk mencari lokasi kosong tersebut.



Ilustrasi *collision* dalam suatu *hash table*. Terlihat di atas 2 *key* yang berbeda mengasilkan *hash* yang sama. Sumber : http://id.wikipedia.org/wiki/Hash

Penggunaan fungsi hash ini erat kaitannya dengan dunia kriptografi karena sifat satu arah yang sulit untuk dikembalikan. Contoh penggunaan yang paling terkenal dari fungsi *hash* adalah *MD2*, *MD5*, dan *SHA*.

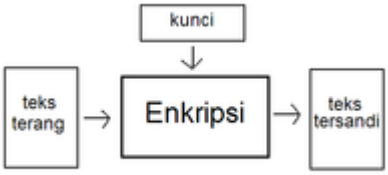
B. Enkripsi

Sebelum mengenal lebih jauh tentang enkripsi kita harus tahu tentang Kriptografi. Kriptografi adalah ilmu dan seni untuk menjaga kerahasiaan berita (Oleh Bruce Schneier - *Applied Cryptography*). Selain pengertian tersebut terdapat pula pengertian ilmu yang mempelajari teknikteknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan data, keabsahan data, integritas data dan autentifikasi dat (Oleh A. Menezes, P. van Oorschot and S. Vanstone - *Handbook of Applied Cryptography*).

Dalam kriptografi ada 4 hal yang mendasar yang juga merupakan aspek keamanan informasi, yaitu :

- Kerahasiaan, merupakan aspek yang digunakan untuk menjaga isi dari sebuah informasi dari siapapun kecuali memiliki otoritas dari informasi ini.
- Integritas data, berhubungan dengan penjagaan dari perubahan isi informasi. Sistem keamanan harus menjaga data dari perubahan isi data, sehingga data harus sama dengan kedaan semula ketika ingin diakses.
- Autentifikasi, beruhubungan dengan identifikasi. Artinya dalam penyaluran informasi Antara dua pihak harus memperkenalkan diri. Informasi yang dikirimkan juga harus diautentifikasi keasliannya, isi datanya, waktu pengiriman, dan lain-lain.
- Non-repudiasi, merupakan usaha untuk mencegah terjadinya penyangkalan terhadap pengiriman/atau terciptanya suatu informasi oleh yang mengirimkan / membuat.

Di dalam kriptografi, kita mengenal istilah enkripsi. Enkripsi adalah proses mengamankan informasi dengan membuat informasi tersebut tidak dapat dibaca tanpa bantuan pengetahuan khusus.



Secara singkat, proses enkripsi adalah proses mengubah teks terang menjadi teks tersandi. Sumber: http://id.wikipedia.org/wiki/Enkripsi

Di berbagai negara enkripsi telah digunakan untuk mengamankan informasi. Di pertangahan tahun 1970-an, enkripsi kuat dimanfaatkan untuk pengamanan oleh secretariat pemerintah Amerika Serikat pada domain public, dan saat ini enkripsi telah digunakan pada system secara meluas.

C. PHP Session dan Cookies

Untuk mengenal *PHP Session* dan *Cookies* kita harus tahu mengenai bahasa pemrograman *PHP*. *PHP Hypertext Preprocessor* adalah bahasa skrip yang dapat ditanamkan atau disisipkan ke dalam *HTML (Hyper Text Markup Languange). PHP* banyak dipakai untuk membuat situs web yang dinamis.

Logo yang menjadi *trademark* dari *PHP*. Sumber : www.php.net

Pada awalnya *PHP* kepanjangannya dalah *Personal Home Page* (Situs Personal). *PHP* pertama kali dibuat oleh Rasmus Lerdorf pada tahun 1995. Waktu itu masih bernama *Form Interpreted (FI)*, yang wujudnya berupa sekumpulan skrip yang digunakan untuk mengolah data formulir dari web.

Selanjutnya Rasmus merilis *source code* tersebut untuk umum dan menamakannya *PHP/FI*. Dengan merilis *source code* tersebut, banyak *programmer* yang tertarik untuk ikut mengembangkannya.

Selanjutnya pada tahun 1998, *PHP* mencapai versi 3.0 dan singkatannya menjadi akronim berulang *PHP: Hypertext Preprocessing* yang berlaku hingga saat ini.

Saat ini *PHP* sudah mencapai versi 5. Dalam versi sekarang inti dari interprester *PHP* mengalami perubahan yang bersar. Sekarang *PHP* juga mendukung model pemrobraman berorientasi objek demi mendukung perkembangan bahasa pemrograman ke arah paradigm berorientasi objek.

Dalam *PHP* dikenal adanya fungsi *session. Session*

adalah suatu cara agar suatu *variable* dapat diakses di banyak halaman web. *Session* biasanya berupa *file* yang tersimpan di server.

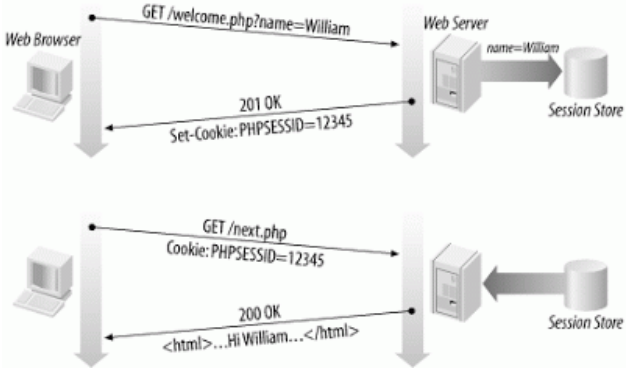


Diagram proses *session* dalam sebuah *website.*
Sumber : http://bonsaiteknologi.wordpress.com

Dalam fungsi *session* terdapat beberapa perintah atau fungsi, yaitu :

1. session_start(), berfungsi untuk memulai atau mengaktifkan session.
2. session_register(), berfungsi untuk mendaftarkan suatu *variable* ke dalam session. Parameter dari fungsi ini adalah nama *variable* yang akan disimpan di *session.* Fungsi ini bisa digantikan dengan perintah $_SESSION[‘nama_variable’].
3. session_unregister(), berfungsi untuk mengapus suatu *variable* yang disimpan di *session.*
4. session_is_registered(), berfungsi untuk memeriksa apakah suatu *variable* ada atau terdaftar dalam *session.*
5. session_unset(), berfungsi untuk menghapus semua *variable* yang terdaftar dalam *session.*
6. session_destroy(), berfungsi untuk menutup atau menghapus *session* beserta *file session*-nya.

Sedangkan *cookies* adalah paket informasi yang dikirim oleh *server* ke peramban dunia Web kemudian dikirim kembali oleh browser setiap kali ia mengakses *server.*

Sebenarnya *cookies* mirip dengan *session.* Hanya saja jika *session* disimpan di *server*, sedangkan *cookies* di simpan di sisi *client (browser web).*

Cookies sering digunakan untuk fitur *autologin* pada sebuah web. Ini yang kadang menjadi sasaran para *hacker* untuk mencuri informasi akun target. Hal ini bisa terjadi karena kurangnya enkripsi pada *cookies* yang diberi oleh *programmer* dari *website.*

III. HELPFUL HINTS

A. Cara Kerja Algoritma MD5

Enkripsi *MD5* bekerja dengan setiap pesan yang akan dienkripsi akan dicari dahulu berapa banyak bit yang terdapat pada pesan. Anggap pesan sebanyak x bit. Di sini x adalah bit non negatif integer, x bisa saja nol dan tidak harus kelipatan delapan. Berikut adalah langkah-langkah dalam enkripsi *MD5* yang dikutip dari Diktat Mata Kuliah IF 3058 Kriptografi Teknik Informatiak Institut Teknologi Bandung oleh Rinaldi Munir.

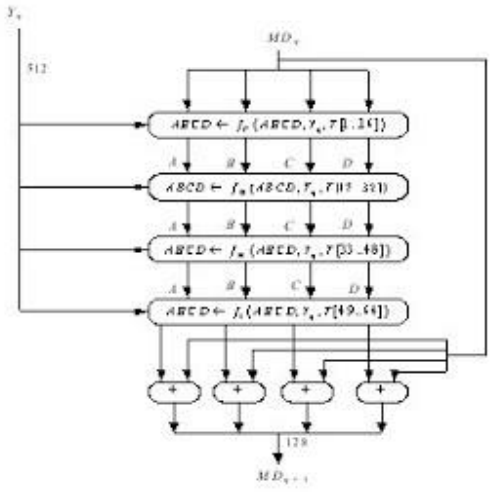
Langkah pertama dalam enkripsi ini adalah penambahan bit-bit pengganjal. Pesan yang akan dienkripsi ditambah sejumlah bit pengganjal sehingga pesan kongruen dengan 448 modulo 512. Jika pesan yang akan dienkripsi panjang pesannya 448 bit, maka pesan akan ditambah dengan 512 bit menjadi 960 bit. Jadi,panjang bit-bit pengganjal antara 1 sampai 512. Bit-bit penggajal terdiri dari satu bit 1 dan sisanya bit 0.

Langkah kedua adalah penambahan nilai panjang pesan. Dalam langkah ini pesan yang telah diberi bit-bit pengganjal selanjutnya ditambah lagi dengan 64 bit yang menyatakan panjang semula. Jika panjang pesan lebih dari 2^{64} maka yang diambil adalah panjangnya dalam mudulo 2^{64}. Dengan kata lain, jika panjang pesan semula adalah K bit, maka 64 bit yang ditambahkan menyatakan K modulo 2^{64}. Setelah ditambah dengan 64 bit, panjang pesan sekarang menjadi kelipatan 512 bit.

Langkah ketiga adalah inisialisasi penyangga md5. *MD5* membutuhkan 4 buah penyangga *(buffer)* yang masing-masing panjangnya 32 bit. Total panjang penyangga adalah 4 x 32 = 128 bit. Keempat penyangga ini menampung hasil antara dan hasil akhir. Keempat penyangga ini diberi nama A, B, C, dan D. Setiap penyangga diinisialisasi dengan nilai-nilai (dalam notasi *HEX*) sebagai berikut :

A = 01234567
B = 89ABCDEF
C = FEDCBA98
D = 76543210

Langkah keempat adalah pengolahan pesan dalam blok berukuran 512 bit. Dalam langkah ini pesan dibagi menjadi L buah blok yang masing-masing panjangnya 512 bit. Setiap blok 512-bit diproses bersama dengan penyangga *MD* menjadi keluaran 128-bit, dan ini disebut proses H_{MD5}. Gambaran proses H_{MD5} diperlihatkan pada gambar berikut:



Sumber : http://ilmu-kriptografi.blogspot.com/2009/05/mengenal-algoritma-md5.html

B. Menggunakan Fungsi MD5 di PHP

Seperti kita ketahui *Interpreter PHP* sudah menyediakan fungsi *hash MD5*. Jadi kita tinggal memanfaatkan fungsi tersebut tanpa harus membuat sendiri algoritmanya terlebih dahulu.

Dalam *PHP* fungsi *hash MD5* cukuk dengan memanggil dengan fungsi md5() dengan parameter inputan sebuah string dengan panjang sembarang. Berikut adalah contoh penggunnan sederhana penggunaan fungsi *md5()* dalam *Interpreter PHP* :

```
<?php
        $text = "test";
        $enkripsi = md5($text);
        echo "Input : ".$text."<br>";
        echo "Hasil : ".$enkripsi;
?>
```

Dari sepotong *source code* di atas variabel *$text* berisi string "test". String ini akan dienkripsi dengan fungsi *md5()* yang akan dimasukan ke dalam variabel *$enkripsi*. Hasil output dari sepotong kode di atas adalah sebagai berikut :

```
Output program :
Input : test
Hasil : 098f6bcd4621d373cade4e832627b4f6
```

Dari hasil output di atas bisa kita lihat input hanya 4 karakter hanya 4 karakter, namun hasilnya menjadi string dengan panjang 32 karakter. Sekarang kita coba dengan mengganti sedikit input string dengan "Test" (huruf pertama menjadi kapital.

```
<?php
        $text = "Test";
        $enkripsi = md5($text);
        echo "Input : ".$text."<br>";
        echo "Hasil : ".$enkripsi;
?>

Output program :
Input : Test
Hasil : 0cbc6611f5540bd0809a388dc95a615b
```

Dari hasil output program di atas terlihat sekali perbedaan output daripada sebelumnya. Hanya dengan mengganti huruf pertama menjadi kapital, output menjadi sangat berbeda.

Sekarang kita coba dengan input string dengan panjang lebih dari 32 karakter. Berikut adalah *source code* dari program tersebut :

```
<?php
        $text =
        "0123456789abcdefghijklmnopqrstuvwxyz";
        $enkripsi = md5($text);
        echo "Input : ".$text."<br>";
        echo "Hasil : ".$enkripsi;
?>

Output program :
Input : 0123456789abcdefghijklmnopqrstuvwxyz
Hasil : e9b1713db620f1e3a14b6812de523f4b
```

Input string di atas panjang nya adalah 36 karakter. Input ini lebih dari 32 karakter. Tapi kita lihat hasil outputnya tetap 32 karakter. Ini menunjukan fungsi *hash md5()* merupakan fungsi yang konsisten menghasilkan output dengan panjang tetap 32 karakter dengan input string dengan panjang berapapun.

Fungsi *md5()* ini biasanya dipakai oleh *programmer* untuk mengenkripsi *password* user dalam suatu *website*. Dikarenakan fungsi *md5()* bersifat satu arah. Seorang admin *website* tidak bisa mengetahui *password user*. Sehingga privasi dari *user* pun tetap terjaga.

C. Fungsi MD5 untuk Ekripsi PHP Session

Session biasanya digunakan sebuah *website* untuk menyimpan data *login user* pada satu sesi. Ketika *user* pertama kali *login* web akan menyimpan data *login* seperti *username* dan *password*, serta kadang *status login.* Sehingga setiap membuka *website* tersebut *user* tak perlu mengulangi terus proses *login* asalkan browser belum ditutup.

Secara sederhana modul *login* menggunakan *session* ini bisa terbagi menjadi 4 bagian. Bagian pertama adalah *login page*, bagian kedua adalah bagian autentifikasi data dan penyimpanan *session*, bagian 3 halaman setelah user *login*, serta bagian 4 adalah bagian *logout user*.

Bagian pertama, berikut merupakan potongan contoh *source code* dari halaman *user*.

```
<html>
<head>
   <title>Contoh Login</title>
</head>
<body>
   <form action="p_login.php" method="post">
      Username : <br>
      <input type="text" name="username"> <br>
      Password : <br>
      <input type="password" name="password"> <br>
      <input type="submit" name="submit">
   </form>
</body>
</html>
```

Selanjutnya bagian proses *login* sebagai berikut :

```
<?php
  if (isset($_POST['submit'])){
    if (($_POST['username'] == 'user')&&
        (md5($_POST['password']) ==
         "1a1dc91c907325c69271ddf0c944bc72"))
    {
```

```
    session_start();
    $_SESSION['username'] = $_POST['username'];
    $_SESSION['password'] = md5($_POST['password']);
    header("location:user.php");
    }else{
      echo "login gagal";
    }
  }
?>
```

Kemudian pada bagian halaman user setelah *login* terdapat perintah yang akan memverifikasi adanya *login session* atau tidak. Berikut adalah potongan *souce code* dari halaman *user.*

```
<?php
 session_start();
 if ((isset($_SESSION['username']))&&
     (isset($_SESSION['password'])))
 {
   if (($_SESSION['username'] == 'user')&&
   ($_SESSION['password'] ==
        "1a1dc91c907325c69271ddf0c944bc72"))
   {
    echo 'Ini adalah halaman user <br>';
      echo '<a href="logout.php">Logout</a>';
   }else{
    header("location:login.php");
   }
 }else{
   header("location:login.php");
 }
?>
```

Terlihat di atas bahwa untuk membuka halaman *user* harus terotentifikasi terlebih dahulu. Fungsi *isset()* digunakan untuk mengecek apakah adanya nama variabel tersebut atau tidak. Hal ini harus dilakukan untuk menghindari *error* ketika tidak adanya variabel tersebut karena *user* belum *login.*

Kemudian di bawah ini adalah *source code* dari bagian *logout user*.

```
<?php
  session_start();
  session_unset($_SESSION['username']);
  session_unset($_SESSION['password']);
  session_destroy();
  header("location:login.php");
?>
```

Pada modul-modul di atas tersebut nama variabel dalam *session* belum dienkripsi, jadi masih mudah terbaca. Untuk itu nama variabel pun akan dienkripsi. Sehingga *source code* menjadi sebagai berikut

p_login.php :

```
<?php
 if (isset($_POST['submit'])){
  if (($_POST['username'] == 'user')&&
      (md5($_POST['password']) ==
       "1a1dc91c907325c69271ddf0c944bc72"))
  {
   session_start();
   $_SESSION[md5('username')]=$_POST['username'];
   $_SESSION[md5('password')] =
                              md5($_POST['password']);
   header("location:user.php");
   }else{
    echo "login gagal";
   }
  }
?>
```

user.php :

```
<?php
 session_start();
 if ((isset($_SESSION[md5('username')]))&&
     (isset($_SESSION[md5('password')])))
 {
  if (($_SESSION[md5('username')] == 'user')&&
  ($_SESSION[md5('password')] ==
       "1a1dc91c907325c69271ddf0c944bc72"))
  {
   echo 'Ini adalah halaman user <br>';
     echo '<a href="logout.php">Logout</a>';
  }else{
   header("location:login.php");
  }
 }else{
  header("location:login.php");
 }
  ?>
```

logout.php :

```
<?php
  session_start();
  session_unset($_SESSION[md5('username')]);
  session_unset($_SESSION[md5('password')]);
  session_destroy();
  header("location:login.php");
?>
```

Source code di atas terlihat sedikit perbedaan yaitu penambahan fungsi *md5()* dalam parameter *session.* Hal ini menyebabkan *session* menjadi sulit untuk dienkripsi.

D. Penerapan MD5 dalan Cookies

Pada penjelasan sebelumnya diterangkan bahwa *cookie* adalah penyimpanan variabel sementara yang terletak pada sisi *client (browser).* Dalam penggunaan pada kasus modul *login*, *cookie* dapat menggantikan fungsi dari *session.*

Dalam menggunakan *cookie* dikenal juga masa berlaku *cookie* tersebut terhadap *server*. Kita pun bisa mengeset secara manual batas masa berlaku *cookie.* Berikut

merupakan transformasi modul-modul *login* yang telah menggunakan *cookie.*

```
p_login.php :
<?php
 if (isset($_POST['submit'])){
  if (($_POST['username'] == 'user')&&
   (md5($_POST['password']) ==
    "1a1dc91c907325c69271ddf0c944bc72"))
{
  setcookie("username", $_POST['username'],
              time()+3600);
  setcookie("password", md5($_POST['password']),
        time()+3600);  header("location:user.php");
}else{
 echo "login gagal";
}

user.php :
<?php
 if ((isset($_COOKIE['username']))&&
    (isset($_COOKIE ['password'])))
 {
  if (($_COOKIE ['username'] == 'user')&&
  ($_SESSION['password'] ==
      "1a1dc91c907325c69271ddf0c944bc72"))
  {
   echo 'Ini adalah halaman user <br>';
    echo '<a href="logout.php">Logout</a>';
  }else{
   header("location:login.php");
  }
 }else{
  header("location:login.php");
 }
?>

logout.php :
<?php
 setcookie("username", "", time()-3600);
 setcookie("password", "", time()-3600);
?>
```

Source code modul-modul di atas menggunakan otentifikasi *cookie* dengan batas masa berlaku 1 jam terhadap *server*. Nama variabel *cookie* di atas belum dienkripsi. Berikut adalah versi dengan nama variabel terenkripsi.

```
p_login.php :
<?php
 if (isset($_POST['submit'])){
  if (($_POST['username'] == 'user')&&
   (md5($_POST['password']) ==
    "1a1dc91c907325c69271ddf0c944bc72"))
{
  setcookie(md5("username"), $_POST['username'],
              time()+3600);
  setcookie(md5("password"),
            md5($_POST['password']),time()+3600);
  header("location:user.php");
 }else{
  echo "login gagal";
 }

user.php :
<?php
 if ((isset($_COOKIE[md5('username')]))&&
    (isset($_COOKIE [md5('password')])))
 {
  if (($_COOKIE [md5('username')] == 'user')&&
  ($_SESSION[md5('password')] ==
      "1a1dc91c907325c69271ddf0c944bc72"))
  {
   echo 'Ini adalah halaman user <br>';
    echo '<a href="logout.php">Logout</a>';
  }else{
   header("location:login.php");
  }
 }else{
  header("location:login.php");
 }
?>

logout.php :
<?php
 setcookie(md5("username"), "", time()-3600);
 setcookie(md5("password"), "", time()-3600);
?>
```

Dengan enkripsi seperti di atas *cookie* menjadi sulit terbaca oleh orang lain. Terkadang *cookie* menjadi sasaaran paling empuk untuk mencuri informasi pada jaringan LAN yang sama.

IV. PERMASALAHAN

Dengan adanya fungsi *hash MD5* yang bersifat satu arah enkripsi dalam penyimpanan variabel *session* dan *cookie* menjadi lebih aman. Namun tidak ada keamanan yang sempurna. Keamanan ini kadang masih ada celah.

Fungsi *hash* walaupun sulit ternyata masih bisa di-*crack* untuk mendapatkan hasil balikannya. Cara paling primitif untuk *cracking* adalah menggunakan metode *bruteforce.* Kemungkinan terburuk cara ini adalah mengulang sampai 2^{128} kali (128 bit). Metode lain adalah menggunakan *collision* dengan *rainbow table*. Namun cara ini cukup sulit untuk dilakukan.

Untuk menanggulangi *cracking* sebaiknya *user* dalam memilih *password* yang sulit di tebak. Hal ini akan memperlambat pencurian informasi.

V. KESIMPULAN

Penggunaan fungsi *hash MD5* dalam enkripsi penyimpanan variabel *PHP Session* dan *Cookies* ternyata mampu mengurangi kemungkinan pencurian informasi dalam suatu web. Walaupun metode ini masih bisa dicari celahnya walaupun sulit.

VI. UCAPAN TERIMA KASIH

Alhamdulillah, saya ucapkan atas rahmat Allah SWT sehingga saya bisa menyelesaikan makalah ini. Tak lupa juga saya ucapkan terima kasih kepada orang tua tercinta. Terima kasih juga saya ucapkan kepada para dosen Labtek V tercinta dan juga para teman-teman saya yang selalu sedia membantu saya ketika kesulitan.

REFERENCES

[1] Eyubalzary, "Cara Mudah Menggunakan PHP Session", 2013, http://bonsaiteknologi.wordpress.com/2013/06/07/cara-mudah-menggunakan-php-session/ Diakses pada Minggu, 15 Desember 2013 jam 19:14.

[2] Fernanfo, Ricky Gilbert, "Penggunaan Fungsi Hash dalam Kriptografi", Makalah IF2153 Matematika Diskrit, Teknik Informatika Institut Teknologi Bandung, 2007.

[3] Hastomo, "Pengertian dan Sejarah PHP", 2013, http://hastomo.net/php/pengertian-dan-sejarah-php/ Diakses pada Minggu 15 Desember 2013 jam 16:12.

[4] Munir, Rinaldi, "Pengantar Kriptografi", Departemen Teknik Informatika Institut Teknologi Bandung, 2004.

[5] Putra, Aldy, "Pengertian Cookies Dan Kegunaannya Dalam Website", 2012, http://aldyputra.net/2012/11/pengertian-cookies-dan-kegunaannya-dalam-website/ Diakses pada Minggu, 15 Desember 18:33.

[6] Silalahi, Ivan, "Tutorial Cookie PHP : Cara Penggunaan Cookie dalam PHP Lengkap dengan Penjelasan", 2013, http://planetsphp.blogspot.com/2013/02/tutorial-cookie-php-cara-penggunaan.html diakses Minggu, 15 Desember 2013 jam 20:12.

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, 16 Desember 2013

Luthfi Hamid Masykuri
13512100