Analisis Penggunaan Fungsi Hash MD5 di Javascript sebagai Alternatif dari Penggunaan HTTP Secure Untuk Melakukan Autentikasi

Budi Satrio - 13504006
Program Studi Teknik Informatika ITB, Bandung 40132, email: if14006@students.if.itb.ac.id

Abstract *– Metode pengamanan autentikasi user dalam web yang umum digunakan adalah dengan menggunakan protokol HTTP Secure (HTTPS) untuk mengenkripsi data dari client menuju server dan sebaliknya. Tetapi, walaupun metode ini termasuk paling aman dari beberapa mekanisme pengamanan data, terdapat beberapa kekurangan pada HTTPS. Diantaranya adalah tidak semua server mempunyai sertifikat untuk mengamankan data melalui HTTPS, ditambah dengan lamanya koneksi yang dibutuhkan saat melalui protokol ini.*

Untuk menutupi kekurangan tersebut, banyak cara alternatif yang diajukan untuk melakukan autentikasi user. Salah satu alternatif penggunaan HTTPS yang penulis ajukan adalah dengan menggunakan fungsi hash MD5 pada javascript untuk mengenkripsi data dari browser saat melakukan autentikasi. Kunci dari metode tersebut adalah adanya sebuah string acak, disebut challenge, yang dikirimkan dari server setiap kali user merequest form. String acak tersebut akan di-hash bersama dengan password untuk kemudian dikirim ke server. Server kemudian akan membandingkan hash dari server dengan hash dari browser. Karena sifat dari string tersebut adalah random, maka ini akan memperkecil kemungkinan seseorang mengetahui hash dari password sebenarnya dan menyulitkan serangan man in the middle attack.

Makalah ini akan membahas mengenai penggunaan string acak tersebut untuk melakukan autentikasi user disertai dengan implementasi, contoh studi kasus, serta analisis kekuatan dan kelemahannya.

Kata Kunci*: HTTP, HTTPS, hash, MD5, javascript, string acak, challenge, autentikasi user, browser.*

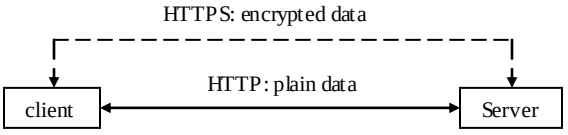
1. PENDAHULUAN

Internet terdiri dari berbagai macam situs web yang saling terhubung satu dengan yang lain. Setiap situs dalam internet diakses dengan menggunakan protokol HTTP (*Hyper Text Transfer Protocol*). HTTP sendiri merupakan suatu protokol untuk melakukan transfer informasi dalam bentuk hypertext [4]. Dalam HTTP, setiap kali kita melakukan *request* dan *retrieve* sebuah halaman web, akan digunakan 2 metode yang paling terkenal untuk melakukan pengiriman data. Metode tersebut adalah dengan POST dan GET.

POST adalah jenis pengiriman data melalui form. Sedangkan GET adalah jenis pengiriman data melalui link dalam url. Untuk POST, data yang dikirmkan tidak akan terlihat pada browser karena pengiriman melalui form, sedangkan untuk GET, data yang dikirimkan akan terlihat jelas pada url form action. Untuk melakukan autentikasi user, metode yang paling banyak digunakan adalah POST.

Pengamanan pengiriman data POST untuk melakukan autentikasi user suatu situs internet sangat diperlukan, karena dalam internet terdapat berbagai macam kemungkinan penyadapan data untuk disalahgunakan. Oleh karena itu dibutuhkan suatu cara untuk mengamankan data tersebut. Salah satu cara yang umum digunakan pada situs-situs besar di internet adalah dengan menggunakan HTTPS.

Jika menggunakan HTTPS, proses *request* dan *retrieve* data akan berjalan pada protokol yang mengenkripsi setiap pengiriman data, yaitu HTTPS. Perbedaan antara HTTPS dan HTTP dapat dijelaskan dengan gambar sebagai berikut:

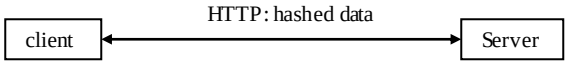


Gambar 1: Perbedaan HTTPS dan HTTP

Secara umum, pengamanan untuk pengiriman data dengan menggunakan HTTPS sudah menjadi standar internasional, namun terdapat kekurangan-kekurangan dari metode ini. Karena itu muncul cara alternatif untuk mengamankan data selain dengan HTTPS.

Banyak cara alternatif yang diajukan untuk melakukan autentikasi user. Salah satunya yang menjadi makalah ini adalah penggunaan fungsi hash MD5 pada javascript.

Jika menggunakan fungsi hash MD5 pada javascript, proses *request* dan *retrieve* data akan tetap berjalan di protokol HTTP, namun data yang dikirimkan akan di hash terlebih dahulu pada setiap pengiriman data. Secara mudahnya dapat dilihat pada gambar sebagai berikut:



Gambar 2: Penggunaan MD5 untuk hashing data pada HTTP

Pada makalah ini akan dibahas mengenai penggunaan fungsi hash MD5 pada javascript sebagai alternatif penggunaan HTTPS saat melakukan autentikasi user sebuah situs, contoh studi kasus implementasinya, serta analisis kekuatan dan kelemahan dari metode ini.

2. HTTP SECURE

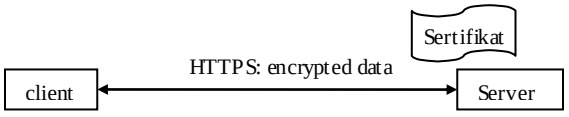
HTTP Secure atau biasa kita sebut dengan HTTPS adalah versi aman dari HTTP, protokol komunikasi untuk transfer data dalam bentuk hypermedia[4]. HTTPS ditemukan oleh Netscape Communications Corporation untuk menyediakan autentikasi dan komunikasi tersandi dan penggunaan dalam komersi elektris.

2.1. Konsep Dasar

HTTPS disebut sebagai versi aman dari protokol HTTP karena data sesi disandikan dengan menggunakan protokol SSL (*Secure Socket Layer*) atau protokol TLS (*Transport Layer Security*). Kedua protokol tersebut memberikan perlindungan yang memadai dari serangan *eavesdroppers*, dan *man in the middle attacks*. Pada umumnya port HTTPS adalah 443, berbeda dengan port 80 pada HTTP.

Tingkat keamanan HTTPS tergantung pada ketepatan implementasinya pada browser web, perangkat lunak server, dan didukung oleh algoritma penyandian yang aktual. HTTPS menggunakan sertifikat yang disediakan oleh server untuk melakukan penyandian data. Sertifikat antara server satu dengan server lain dapat berbeda, namun sertifikat-sertifikat tersebut harus mendapatkan persetujuan dari badan pemberi sertifikat (*certificate authority*) yang berada di atasnya.

Data yang dikirimkan dengan menggunakan HTTPS akan terlebih dahulu dienkripsi berdasarkan key dan metode yang ada pada sertifikat. Sertifikat tersebut biasanya disediakan oleh browser sebagai default, namun beberapa sertifikat ada yang harus diinstall dari situs. Hal ini menjadikan hanya pihak yang berhak saja yang mampu membaca data yang dikirim berdasarkan key dari sertifikat. Metode inilah yang disebut dengan SSL.



Gambar 3: Sertifikat yang diperlukan untuk menentukan penyandian data

2.2. Kelebihan HTTPS

Beberapa kelebihan dari HTTPS sehingga masih digunakan sampai sekarang adalah:

1. Jalur yang digunakan untuk transfer data sangat aman karena terenkripsi.
2. Sertifikat yang didapatkan dari badan pemberi sertifikat dijamin keamanannya oleh badan tersebut.
3. Dapat menangkal serangan *eavesdroppers*, dan *man in the middle attacks*.
4. Banyak situs-situs besar yang menggunakan metode ini, sehingga komunitas pengguna HTTPS cukup banyak.

2.3. Kelemahan HTTPS

Beberapa kelemahan dengan menggunakan metode HTTPS dapat ditunjukkan sebagai berikut:

1. Tidak semua web browser mempunyai kemampuan untuk menginstall sertifikat dengan baik. Oleh karena itu sertifikat harus mempunyai *browser recognition rate*.
2. Kerja komputer semakin tinggi karena melibatkan enkripsi, maka client dan server membutuhkan waktu, resource, dan *bandwidth* lebih untuk melakukan kegiatannya.
3. Kebutuhan *bandwidth* bertambah karena data yang dikirim lebih besar daripada data yang tidak dienkripsi.
4. Proses mendapatkan sertifikat yang cukup rumit, sehingga membuat para pembuat situs individu sulit untuk membuat sertifikat yang sesuai dengan kerja dari situsnya.

3. FUNGSI HASH MD5 PADA JAVASCRIPT

Penggunaan fungsi hash MD5 pada javascript ini didasarkan pada kemampuan fungsi hash untuk menghasilkan string acak tak bermakna dari sebuah string yang mempunyai makna. MD5 sendiri adalah mekanisme hash yang mempunyai masukan sebuah string dan keluaran adalah 128 bit string acak. Jadi, dengan fungsi hash MD5 ini setiap field dari form yang akan dikirimkan melalui web dimasukkan dalam fungsi hash yang kemudian akan dicocokkan dengan hash form dari server.

3.1. Konsep Umum

Secara umum, inti dari penggunaan metode ini adalah bagaimana caranya untuk mengamankan data POST untuk autentikasi user pada sebuah situs dengan menggunakan fungsi hash. Diharapkan dengan menggunakan metode ini, data yang telah di hash akan sulit diketahui nilai asli dari data tersebut [5].

Fungsi hash yang digunakan pada metode ini adalah MD5 karena beberapa latar belakang pemilihan sebagai berikut:

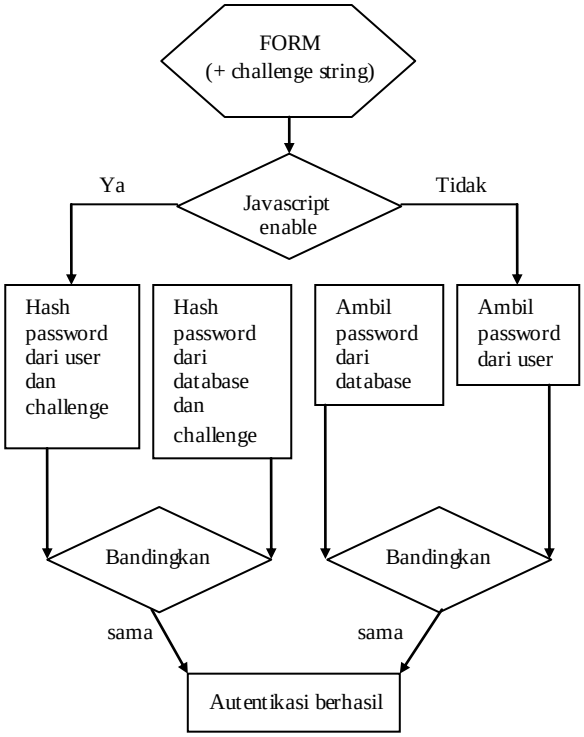
1. MD5 mudah diimplementasikan pada Javascript.

2. MD5 merupakan fungsi hash yang cepat.
3. Banyak library-library MD5 pada Javascript.

MD5 adalah fungsi hash satu arah yang dibuat oleh Ronald Rivest pada tahun 1991. MD5 merupakan perbaikan dari MD4 setelah MD4 berhasil diserang oleh kriptanalis. Algoritma MD5 menerima masukan berupa pesan dengan ukuran sembarang dan menghasilkan *message digest* yang panjangnya 128 bit [1].

3.2. Mekanisme Autentikasi Dengan Menggunakan Fungsi Hash MD5 Pada Javascript

Mekanisme autentikasi dengan metode ini diawali ketika user akan mengirimkan username dan passwordnya menuju server. Server akan membuat sebuah string acak, disebut *challenge*, yang akan dikirimkan setiap kali client merequest form. *Challenge* ini akan digunakan untuk membedakan hash password dari form satu dengan yang lain. Jika dengan cara biasa, password akan di-hash dahulu baru dikirim menuju server. Tetapi dengan cara ini, password yang telah di-hash akan ditambahkan dengan *challenge* yang berasal dari server. Diharapkan dengan cara ini, serangan *man in the middle attack* dapat diatasi [3]. Fungsi hash sendiri digunakan untuk membingungkan *eavesdropper* mengetahui pesan sebenarnya [2].



Gambar 4: Mekanisme autentikasi dengan fungsi hash MD5 pada javascript

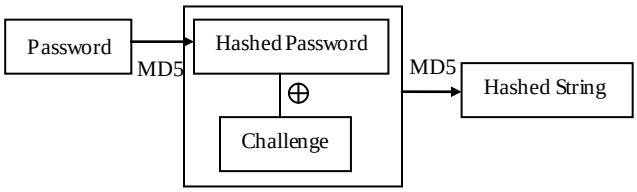
Metode ini hanya dapat bekerja di browser yang mempunyai kemampuan javascript, namun tentunya semua browser saat ini sudah mendukung javascript.

3.3. Mekanisme Hashing Data

Mekanisme hashing data yang digunakan dalam metode ini adalah melakukan hash dua kali pada password dan *challenge*. *Challenge* akan dibuat setiap kali client merequest form, sehingga dijamin tidak ada *challenge* yang sama untuk setiap kali form direquest.

Password pertama kali akan dihash dengan menggunakan MD5, kemudian ditambahkan dengan *challenge* dan kemudian dihash lagi untuk dikirimkan kepada server.

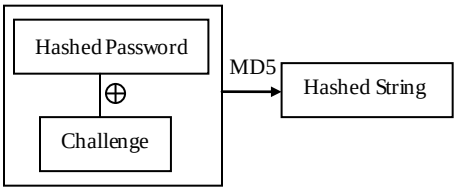
Mekanisme hash pada client dapat digambarkan sebagai berikut:



Gambar 5: Mekanisme hash pada client

Sedangkan pada server, password yang disimpan dalam bentuk hash diambil dari database kemudian ditambahkan dengan challenge. String hasil penambahan ini akan dihash untuk kemudian dibandingkan dengan hash dari client.

Mekanisme hash pada server dapat digambarkan sebagai berikut:



Gambar 6: Mekanisme hash pada server

String yang telah di-hash inilah yang akan dibandingkan dengan string sama yang dihasilkan oleh server. Perlu diketahui bahwa server tidak pernah menyimpan password sebenarnya dari user, karena server hanya menyimpan password yang telah di hash saja.

3.3. Implementasi Fungsi Hash MD5 Pada Javascript

Untuk mengimplementasikan metode penggunaan fungsi hash MD5 pada javascript ini, penulis menggunakan fungsi MD5 yang sudah tersedia di internet, sehingga hanya tinggal memanggil saja untuk menggunakannya. Fungsi MD5 ini didapatkan dari situs http://pajhome.org.uk/site/legal.html.

Berikut *pseudocode* algoritma hash untuk metode ini:

```
Hash(form_action,login_url){
 url = form_action+"?";
 if(javascript_enable){
  pass = get password;
  hashPass = MD5(pass);
  challenge = get_challenge;
  temp = hashPass+challenge;
  hashString = MD5(temp);
  foreach setiap elemen form {
   if(elemen_form=password){
    url += hash;
   }
  }
  beri tanda di url bahwa hash berhasil;
  beri tanda di url bahwa ada javascript;
  submit form;
  matikan form agar tidak bisa diulangi lagi
 }
}
```

Dari *pseudocode* diatas, dibuat implementasi keseluruhan algoritma dalam javascript sehingga didapatkan kode program sebagai berikut:

```
function hash(form,login_url) {
 var url;
 if (arguments.length > 1 && login_url != "")
 {
  url = login_url;
 } else {
  url = "http://localhost/login";
 }
 url += "?";
 if (valid_js()) {
  var passwd = form.passwd.value;
  var hash1 = MD5(passwd);
  var challenge = form[".challenge"].value;
  var hash2 =hash1 + challenge;
  var hash;
  if(form.passwd.value){
   hash=MD5(hash2);
  } else {
   hash="";
  }
  var js = 0;
  for(i=0; i<form.elements.length; i++){
   if(form.elements[i].name.length <=0) {
    continue;
   }
   if(i > 0){
    url += "&";
   }
   url += form.elements[i].name;
   url += "=";
   if(form.elements[i].name == "passwd"){
    url += hash;
   }   else   if   (form.elements[i].type   ==
"checkbox" && !form.elements[i].checked) {
    url += "";
   }   else   if   (form.elements[i].type   ==
"radio" && !form.elements[i].checked) {
    url += "";
   }   else   if   (form.elements[i].name   ==
".save"){
    url += "1";
   }   else   if   (form.elements[i].name   ==
".js"){
    js = 1;
    url += "1";
   } else {
    url += escape(form.elements[i].value);
   }
 }
 url += "&.hash=1";
 if(js == 0){
  url += "&.js=1";
 }
 url += "&.md5=1";
 location.href=url;
 form.onsubmit=null;

 return false;
 }
return true;
}
```

Kode javascript diatas akan dipanggil saat user melakukan submit form. Jika terjadi kemungkinan user merefresh halaman, maka akan direquest *challenge* baru. Submit form melebihi satu kali juga tidak akan bisa dilakukan.

3.4. Lingkungan Pembangunan Aplikasi

Untuk membangun aplikasi dengan metode ini, lingkungan pembangunannya adalah sebagai berikut:

- Prosesor: Intel Core Duo Processor @ 1.60 Ghz, 533 FSB, 2Mb L2 Cache.
- Memory: 768MB DDR2.
- VGA: Intel Graphic Media Accelerator 950.
- OS: Windows XP Service Pack 2.
- Notepad++ untuk menuliskan kode Javascript.
- Browser Internet Explorer 7 dan Opera 9.25 untuk menguji studi kasus.
- Wireshark 0.99 untuk mengetahui transfer data yang dikirimkan dari client dan server.

3.5. Kekuatan Metode Penggunaan Fungsi Hash MD5 Pada Javascript

Kekuatan dari metode ini dapat dijabarkan sebagai berikut:

1. Transfer data masih menggunakan protokol HTTP, sehingga tidak perlu ada transisi perpindahan protokol HTTPS.
2. Dengan menggunakan hash, *eavesdropper* tidak akan mengetahui data aslinya. Hal ini berarti metode ini mampu melindungi data asli dari penyadapan.
3. Dengan adanya *challenge* berupa string acak yang akan berubah setiap kali user merequest form, maka hal ini secara otomatis akan melindungi serangan *man in the middle attack*.
4. Proses hash di client akan mempercepat eksekusi karena server hanya perlu melakukan satu kali proses hash password dari database dan *challenge*.
5. Proses hash hanya digunakan pada field yang diperlukan saja sehingga akan mengefektifkan penggunaan *bandwidth*.
6. Beberapa situs terbesar di dunia menggunakan metode mirip seperti ini, contohnya Yahoo.

3.6. Kelemahan Metode Penggunaan Fungsi Hash MD5 Pada Javascript

Kelemahan metode ini adalah sebagai berikut:

1. Ketergantungan kepada browser yang

memiliki kemampuan javascript. Jika tidak ada javascript, maka metode ini tidak akan dapat digunakan.

2. Data yang dikirimkan tetap dapat disadap oleh *eavesdropper*, walaupun data yang disadap tidak memiliki arti.
3. Terdapat kemungkinan modifikasi fungsi javascript oleh penyadap.
4. Proses hash di client menghabiskan resource komputer client, walaupun hanya sedikit.

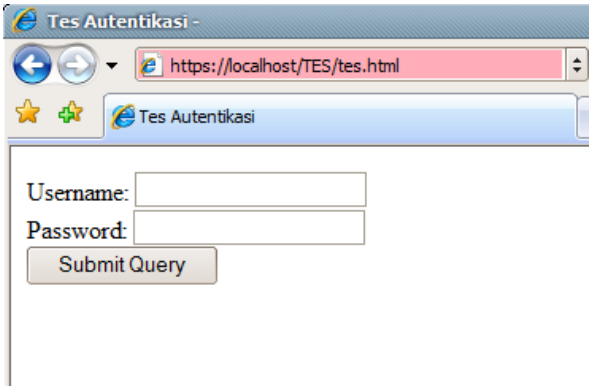
4. STUDI KASUS PENGGUNAAN FUNGSI HASH MD5 PADA JAVASCRIPT

Untuk melakukan autentikasi user secara sederhana, penulis membuat suatu halaman web yang mempunyai masukan berupa username dan password kemudian mengujinya dengan metode HTTPS dan metode penggunaan fungsi hash MD5 pada javascript.

Kode HTML yang digunakan untuk contoh tersebut adalah sebagai berikut:

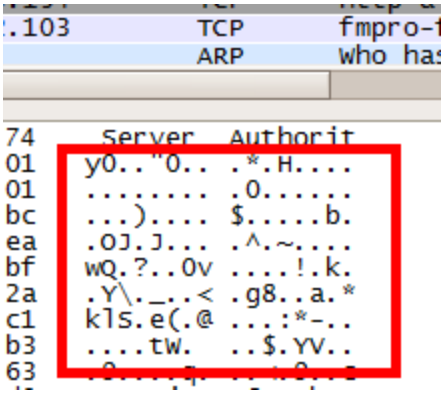
```
// file tes.html
<html>
 <head>
  <title>Tes Autentikasi</title>
 </head>
 <body>
  <form action="login.php" method="post">
   Username: <input type="text"
name="username"> <br>
   Password: <input type="passwd"
name="password"> <br>
   <input type="submit">
  </form>
 </body>
</html>
```

Jika menggunakan HTTPS, maka aksi dari server hanya mengambil data kemudian mencocokkannya di database. Saat melakukan login, browser akan tampak sebagai berikut:



Gambar 7: Tampilan HTTPS pada browser

Saat user melakukan submit login dan passwordnya, data yang dikirimkan akan terlihat dalam wireshark sebagai berikut:

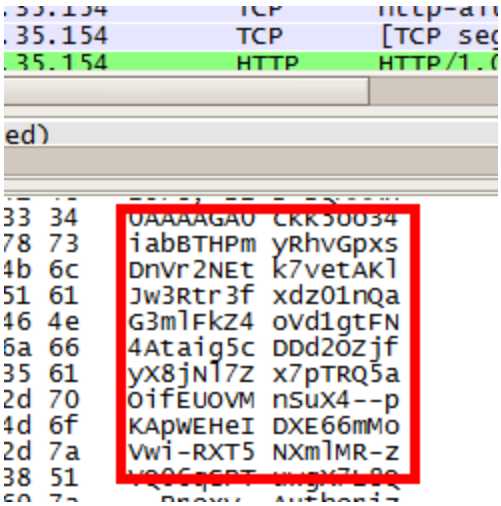


Gambar 8: Data yang dikirimkan dengan menggunakan HTTPS

Berikut properti yang lain yang penting untuk diperhatikan:

Ukuran data yang dikirimkan	551 bytes per frame, total 10 frame
Lama koneksi	1.91 detik
Proses eksekusi	0.75 detik

Jika menggunakan fungsi hash MD5 pada javascript, server melakukan hash password dari database yang ditambah dengan *challenge* kemudian akan dibandingkan dengan password yang dihash dengan *challenge* dari client. Saat user melakukan submit login dan passwordnya, data yang dikirimkan akan terlihat sebagai berikut:



Gambar 9: Data yang dikirimkan dengan menggunakan metode ini

Berikut properti yang lain yang penting untuk diperhatikan:

Ukuran data yang dikirimkan	458 bytes per frame, total 8 frame
Lama koneksi	1.02 detik
Proses eksekusi	0.87 detik

5. PEMBAHASAN

Dari hasil percobaan-percobaan diatas, terdapat beberapa hal dari metode ini yang perlu untuk dikaji, diantaranya mengenai analisis kekuatan dan kelemahan dari penggunaan metode ini.

Dengan menggunakan fungsi hash MD5 pada javascript, terdapat beberapa kekuatan dari metode ini dilihat dari berbagai sudut pandang. Pertama, dari sudut pandang browser. Browser tidak perlu melakukan perpindahan protokol ke HTTPS, namun tetap pada protokol HTTPS saja sehingga akan memudahkan pengaksesan. Browser juga tidak perlu repot menginstall sertifikat dari server yang harus selalu diupdate setiap tahun. Kedua, dari sudut pandang server. Server tidak mempunyai keharusan untuk menyediakan sertifikat yang sangat rumit untuk didapatkan. Ketiga, dari sudut pandang user sendiri. Password user tidak pernah keluar dari user, karena server hanya menyimpan hash dari password tersebut. Keempat, dari sudut pandang penyerang. Dengan adanya *challenge* yang sekaligus juga berfungsi untuk melakukan validasi form, hal ini akan mencegah terjadinya *man in the middle attack*. Hash yang digunakan juga akan menyulitkan *eavesdropper* mengetahui pesan aslinya.

Pada studi kasus diatas, dapat dihitung bahwa setiap penambahan field yang di-hash, maka jumlah data yang dikimkan menjadi bertambah banyak sekitar 512 bit. Hal ini berarti semakin banyak data yang di-hash, data yang dikirmkan akan menjadi lebih panjang. Selain itu, proses transfer pengiriman akan semakin lama. Maka dari itu, untuk lebih mengefektifkan penggunaan *bandwidth*, field yang perlu di-hash adalah field yang diperlukan saja,

Walaupun metode ini masih mempunyai kekurangan seperti ketergantungan pada kemampuan browser manampilkan javascript dan masih dapat disadapnya hash data, tetapi jika dibandingkan dengan metode yang menggunakan HTTPS, metode penggunaan hash MD5 pada javascript ini sudah mencakup keseluruhan kebutuhan untuk mengamankan data dari client ke server dan sebaliknya. Tidak hanya itu, metode ini akan lebih mengefektifkan penggunaan waktu, resource dan *bandwidth* untuk melakukan autentikasi user. Metode ini juga memungkinkan server yang tidak memiliki sertifikat bisa melakukan autentikasi user secara aman.

Secara umum dapat dikatakan bahwa metode penggunaan fungsi hash MD5 pada javascript ini cukup aman untuk digunakan sebagai alternatif dari penggunaan HTTPS.

6. KESIMPULAN

Kesimpulan yang dapat diambil dari penggunaan metode ini adalah sebagai berikut:

1. Terdapat alternatif penggunaan HTTPS untuk melakukan autentikasi user, yaitu dengan menggunakan fungsi hash MD5 pada javascript.
2. Metode penggunaan fungsi hash MD5 pada javascript ini hanya bisa berjalan pada browser yang mendukung javascript. Tetapi pada saat ini hampir semua browser sudah mendukung penggunaan javascript.
3. Untuk lebih mengefektifkan kerja metode ini dan meminimalkan penggunaan *bandwidth*, field yang dihash hanyalah yang penting saja seperti field password.
4. Proses eksekusi metode ini lebih cepat daripada HTTPS dikarenakan proses hash lebih cepat daripada proses encrypt di HTTPS. Perpindahan protokol pada HTTPS akan membutuhkan waktu lebih banyak daripada tetap menggunakan protokol HTTP.
5. Dengan menggunakan metode ini, kemungkinan terjadi penyadapan oleh pihak luar tetap ada, namun penyadap hanya bisa mendapatkan informasi hash dari password dan *challenge*. Karena *challenge* dibuat secara random setiap kali ada request, maka hampir dapat dipastikan tidak mungkin penyadap dapat mengetahui password sebenarnya.
6. Metode ini aman terhadap serangan *man in the middle attack* karena adanya *challenge* yang dibuat oleh server setiap kali ada request.
7. Password yang asli tidak akan pernah disimpan di server, karena server hanya menyimpan hash dari password sebenarnya. Ini berarti password tidak pernah keluar dari user.
8. Penggunaan fungsi hash MD5 di javascript untuk melakukan autentikasi user cukup aman digunakan sebagai alternatif penggunaan HTTPS.

DAFTAR REFERENSI

[1] Munir, Rinaldi, *Diktat Kuliah IF5054 Kriptografi*, Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung, 2007.

[2] Berson, Thomas A. "Differential Cryptanalysis Mod 2^{32} with Applications to MD5". *EUROCRYPT*: 1992.

[3] Hans Dobbertin, Cryptanalysis of MD5 compress. Announcement on Internet, May 1996.

[4] Tim BL., Basic HTTP, http://www.w3.org/Protocols/HTTP/HTTP2.html, 1992

[5] Johnston, P. A., Login System, http://pajhome.org.uk/crypt/md5/auth.html, Oktober 2005.