STUDI PENERAPAN HTTP DIGEST AUTHENTICATION UNTUK PENGGUNA PADA PROXY SERVER DENGAN DATABASE LDAP

Muhammad Fiqri Muthohar – NIM : 13506084

Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung

Jl. Ganesha 10, Bandung

E-mail : fiqri@arc.itb.ac.id , viqronization@gmail.com

Abstrak

Makalah ini membahas mengenai penerapan sistem autentikasi menggunakan metode digest untuk pengguna pada proxy server yang ada di sebuah jaringan komputer. Hal ini disebabkan pada sistem autentikasi yang berjalan selama ini sering terjadi adanya pencurian akun pengguna baik karena kelalaian pengguna sendiri maupun karena adanya pengendusan (sniffing) terhadap data autentikasi pengguna di jaringan saat diantarkan dari pengguna ke server autentikasi. Salah satu upaya yang dapat dilakukan untuk mencegah pencurian data pengguna melalui pengendusan ini adalah dengan penggunaan metode autentikasi digest ini.

Pada makalah ini yang digunakan sebagai aplikasi proxy server adalah aplikasi squid sebuah aplikasi proxy dan cache server yang bersifat open source dan database yang digunakan untuk menyimpan data pengguna adalah LDAP(Lightweight Directory Access Protocol). Sedangkan, mode digest yang diterapkan untuk autentikasi adalah dengan menggunakan algoritma MD5(Message Digest 5). Selain itu, juga akan dibahas masalah-masalah yang timbul dengan adanya penerapan sistem autentikasi ini.

***Kata kunci**: Sistem Autentikasi, MD5, Squid, Proxy Server, Digest Authentication, LDAP.*

1. Pendahuluan

Pada sebuah sistim proxy server yang menggunakan database LDAP sebagai media penyimpanan data user yang ada, kebanyakan sistem autentikasi yang digunakan tidak dengan pesan yang terenkripsi hanya berupa plainteks biasa. Hal ini dapat menjadi salah satu masalah karena data yang dikirim merupakan data yang sifatnya rahasia dan tidak boleh diketahui oleh orang lain selain pemilik akun tersebut. Data plainteks tersebut dapat diketahui dengan melakukan pengendusan (*sniffing*) data pada jaringan dimana sistem tersebut berjalan. Jika data berhasil diendus maka data-data penting user, dalam hal ini adalah *username* dan *password*, dapat diketahui oleh pengendus data dan kemudian dapat menggunakannya untuk dipakai sebagai user pada sistim proxy server tersebut.

Dengan kondisi yang demikian tentu membuat pengguna tidak nyaman, terutama ketika terjadi pembatasan penggunaan akun pengguna hanya dapat dipakai pada satu alamat IP pada satu saat yang bersamaan. Ketika akun dan password pengguna digunakan oleh orang lain tentu saja user tidak akan dapat menggunakan akunnya untuk dipakai oleh dirinya yang sebenarnya memiliki hak penggunaan tersebut. Selain itu, dapat juga akun pengguna digunakan oleh orang lain yang memiliki hasil pengendusan data akun milik pengguna untuk melakukan akses yang tidak diperbolehkan oleh sistem sehingga akun pengguna diblokir oleh sistim, sedangkan hal ini tentu tidak diinginkan terjadi oleh pengguna yang sebenarnya karena pada kenyataannya dia tidak menggunakan akunnya yang melanggar apa yang telah diatur oleh sistim yang ada.

Salah satu cara menangani masalah ini adalah dengan mengenkripsi pesan autentikasi yang membawa data rahasia milik user dari komputer yang sedang digunakan oleh user ke server berada.

2. Proxy Server

Proxy server adalah sebuah komputer server atau aplikasi komputer yang bertindak sebagai perantara dalam permintaan *resource* dari klien ke server yang lainnya. Seorang klien melakukan koneksi ke proxy server, kemudian klien meminta beberapa servis seperti meminta file, koneksi, halaman web, atau resource lainnya yang berada pada server lain. Proxy server kemudian dapat melakukan pengecekan terhadap permintaan yang dilakukan oleh klien, apakah sesuai dengan konfigurasi aturan penyaringannya atau tidak. Sebagai contoh, proxy server dapat melakukan penyaringan terhadap alamat IP klien, protokol yang digunakan, alamat web yang akan diakses oleh pengguna, dan lain sebagainya. Apabila permintaan dari klien disetujui oleh proxy server, dalam hal ini permintaan klien lolos dari aturan pernyaringan, maka proxy server

akan menyediakan resource yang diminta dengan melakukan koneksi kepada server yang sesuai dengan resource yang diminta dan proxy server akan bertindak meminta resource tersebut atas nama pengguna/klien yang meminta resource tersebut ke server yang menyediakan resource tersebut.

Sebuah proxy server dapat juga secara opsional mengubah permintaan dari klien maupun respon dari server dan terkadang proxy server juga dapat menyediakan permintaan tanpa harus melakukan kontak ke suatu server tertentu. Dalam kasus ini, proxy server men-cache respon dari server remote dan mengembalikan secara langsung untuk permintaan konten yang sama.

Sebuah proxy server memiliki dua tujuan :

- Untuk menjaga mesin di belakangnya tetap anonym (terutama untuk alas an keamanan)
- Untuk mempercepat akses terhadap sebuah resource melalui mekanisme cache. Hal ini biasanya digunakan untuk menyimpan cache dari halaman web dari sebuah web server.

Sebuah proxy server yang melanjutkan permintaan dan balasan tanpa modifikasi sama sekali biasanya disebut dengan gateway atau juga sering disebut *tunneling proxy*.

Sebuah proxy server dapat diletakkan di komputer lokal dari user maupun di berbagai tempat diantara user dan server tujuan atau internet. Sebuah reverse proxy adalah sebuah proxy yang digunakan sebagai front-end untuk mempercepat dan cache permintaan resource tersebut (seperti halaman web).

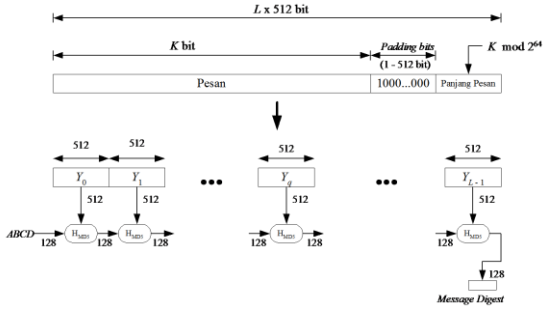
3. Message Digest 5 (MD5)

Message digest 5 (MD5) adalah sebuah algoritma message digest. Algoritma ini mengambil input dengan panjang berapapun dan menghasilkan keluaran "*fingerprint*" atau "*message digest*" sepanjang 128bit dari masukan yang diterimanya. Secara komputasi tidak mungkin dihasilkan dua buah pesan yang memiliki message digest yang sama, atau juga menghasilkan sebuah pesan dari sebuah message digest tujuan yang telah ditentukan sebelumnya. Algoritma MD5 ini ditujukan untuk aplikasi tandatangan digital, dimana sebuah file yang berukuran besar harus dikompresi melalui proses yang aman sebelum dienkripsi dengan menggunakan sebuah kunci privat rahasia dalam sebuah sistim kriptografi kunci publik seperti RSA.

Algoritma MD5 ini didesain untuk dapat berjalan cukup cepat pada mesin 32bit. Sebagai tambahan, algoritma MD5 tidak membutuhkan tabel substitusi yang besar, algoritma ini dapat dikodekan secara kompak.

Algoritma MD5 adalah kelanjutan dari algoritma message digest MD4. MD5 sedikit lebih lambat dari MD4, tetapi lebih konservatif dalam desainnya. MD5 dibuat karena ditemukannya celah keamanan pada algoritma MD4. Celah keamanan ini ditengarai terbuka karena kecepatan dari algoritma MD4 ini. Maka dari itu desain dari MD5 ini lebih lambat daripada MD4 demi mencapai keamanan dari hasil message digestnya..

3.1. Algoritma MD5



Secara garis besar langkah-langkah pembuatan message digest adalah:

a. Penambahan bit-bit pengganjal(padding bits)

Pesan ditambah dengan sejumlah bit pengganjal sedemikian sehingga panjang pesan (dalam satuan bit) kongruen dengan 448 modulo 512. Jika panjang pesan 448 bit, maka pesan tersebut ditambah dengan 512 bit menjadi 960 bit. Jadi, panjang bit-bit pengganjal adalah antara 1 sampai 512. Bit-bit pengganjal terdiri dari sebuah bit 1 diikuti dengan sisanya bit 0.

b. Penambahan nilai panjang pesan semula

Pesan yang telah diberi bit-bit pengganjal selanjutnya ditambah lagi dengan 64 bit yang menyatakan panjang pesan semula. Jika panjang pesan > 2^{64} maka yang diambil adalah panjangnya dalam modulo 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.

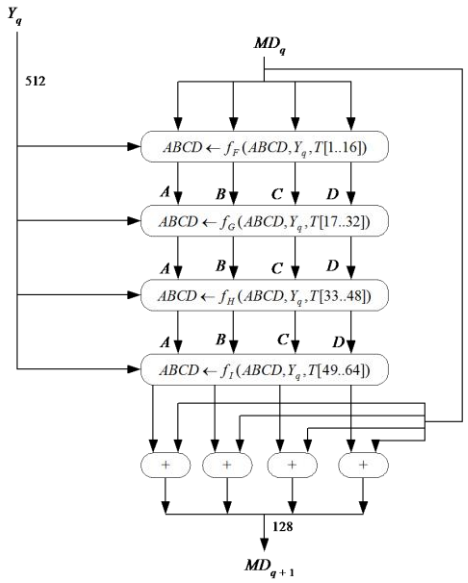
c. Inisialisasi penyangga (buffer) MD

MD5 membutuhkan 4 buah penyangga(*buffer*) yang masing-masing panjangnya 32 bit. Total panjang penyangga adalah 4 x 32 bit = 128 bit. Keempat penyanggga 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:

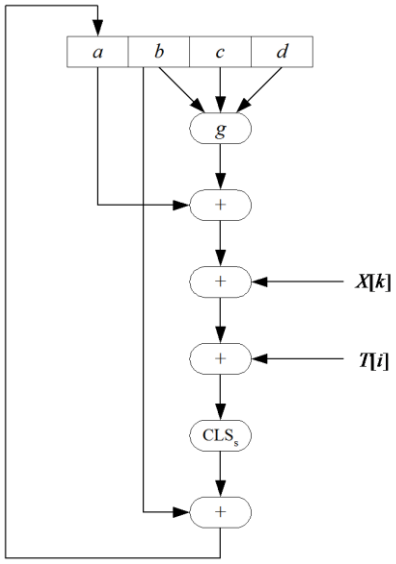
A = `01234567`
B = `89ABCDEF`
C = `FEDCBA98`
D = `76543210`

d. Pengolahan pesan dalam blo berukuran 512 bit

Pesan dibagi menjadi L buah blok yang masing-masing panjangnya 512 bit (Y_0 sampai Y_{L-1}). Setiap blok 512 bit diproses bersama dengan penyangga MD menjadi keluaran 128 bit, dan ini disebut proses H_{MD5}.



Pada gambar, Y_q menyatakan blok 512 bit ke-*q* dari pesan yang telah ditambah bit-bit pengganjal dan tambahan 64 bit nilai panjang pesan semula. MD_q adalah nilai message digest 128 bit dari proses H_{MD5} ke-*q*. pada awal proses, MD_q berisi nilai inisialisasi penyangga MD. Proses H_{MD5} terdiri dari 4 buah putaran, dan masing-masing putaranmelakukan operasi dasar MD5 sebanyak 16 kali dan setiap operasi dasar memakai sebuah elemen T. Jadi setiap putaran memakai 16 elemen tabel T. Fungsi-fungsi f_F, f_G, f_H, dan f_I masing-masing berisi 16 kali operasi dasar terhadap masukan, setiap operasi dasar menggunakan elemen tabel T.



Operasi dasar MD5 yang ada pada gambar dapat ditulis dengan sebuah persamaan dasar sebagai berikut:

$$a \leftarrow b + CLS_x(a + g(b,c,d) + X[k] + T[i])$$

yang dalam hal ini,

a, b, c, d = empat buah peubah penyangga 32 bit (berisi nilai penyangga A, B, C, D)

g = salah satu fungsi F, G, H, I

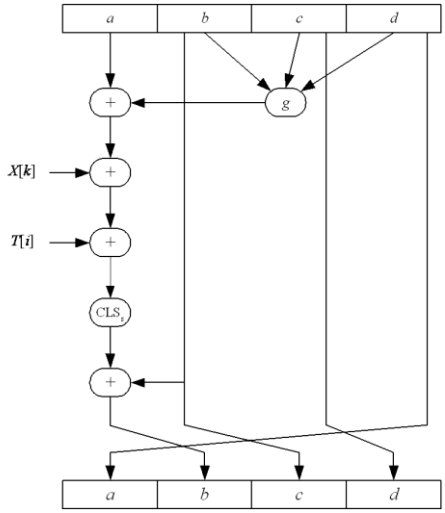
CLS_x = circular left shift sebanyak s bit

X[k] = kelompok 32 bit ke-*k* dari blok 512 bit message ke-q. Nilai k = 0 sampai 15

T[i] = elemen tabel T ke-*i* (32 bit)

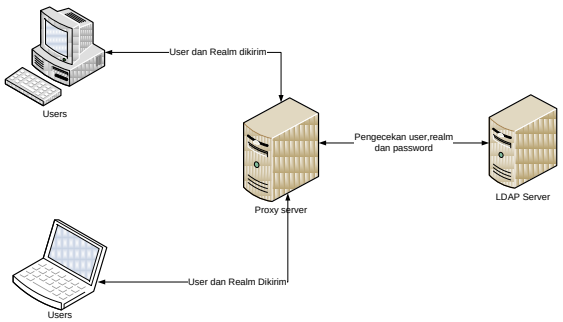
\+ = operasi penjumlahan modulo 2^{32}.

Setiap kali selesai satu operasi dasar, penyangga-penyangga itu digeser ke kanan secara sirkuler, sehingga prosesnya akan menjadi seperti gambar berikut ini.



Setelah putaran keempat, *a*, *b*, *c* dan *d* ditambahkan ke *A*, *B*, *C*, dan *D* dan selanjutnya algoritma memproses untuk blok data berikutnya (Y_{q+1}). Keluaran akhir dari algoritma MD5 adalah hasil penyambungan bit-bit di *A*, *B*, *C*, dan *D*.

4. Implementasi



Selama ini jika menggunakan LDAP sebagai database pengguna maka password akan dikirim ke server proxy sebagai plainteks.

Implementasi dari digest authentication ini dilakukan melalui konfigurasi di server untuk dapat diterapkan pada seluruh sistim yang ada. Sebagai prasyarat ada beberapa komponen server yang harus tersedia sebelum dilakukan implementasi ini, yaitu:

a. Aplikasi proxy server squid.

b. Aplikasi LDAP server

Untuk menerapkan metode autentikasi ini dilakukan perubahan pada beberapa hal di LDAP server yaitu:

a. Penambahan attribute "ProxyDigest" dengan tujuan untuk mengarahkan kolom password pada authentication Squid.

b. Membuat hash MD5 dari '**UID:"Squid proxy-caching web server":userPassword**' (sebut saja HASH)

c. Mengisi attribute "ProxyDigest" dengan **'Squid proxy-caching web server:HASH'**

Untuk melakukan **HASH** dilakukan dengan cara (berikut ini adalah contoh pengimplementasian):

a. Via BASH :

```
REALM="Squid proxy-caching web
server" HASH=`echo -n
"UID:$REALM:userPassword" | md5sum |
cut -f1 -d' '` ldapmodify -x -D
"cn=manager,dc=arc,dc=itb,dc=ac,dc=i
d" -w "password" << EOF
dn: UID= xxxx,ou=people,dc=arc,
dc=itb,dc=ac,dc=id
l: $REALM:$HASH
EOF
```

b. Via Perl (untuk semua user sekaligus):

```
 $ct++;
 print ("$ct".".");
print ($uid,"\n");
my $hash = `md5 -qs $uid:"Squid
proxy-caching web server":$pwd|cut
-f1 -d' '`;
$nhash = "Squid proxy-caching web
server:".$hash;
$dn=
"AccountID=$uid,ou=people,dc=arc,
dc=itb,dc=ac,dc=id";
if ($alamat eq ''){
      $msg1 = $ldap->modify($dn,
add => {ProxyDigest => "$nhash" });
}else{
      $msg2 = $ldap->modify($dn,
replace => {ProxyDigest =>
"$nhash"});          }
```

5. Kendala yang dihadapi

Penerapan HTTP digest authentication ini bukan tanpa kendala, karena pada dasarnya autentikasi HTTP pada awalnya dikembangkan tidak untuk menangani masalah autentikasi yang terenkripsi namun hanya melalui plainteks saja.

Akibat dari implementasi dari autentikasi HTTP pada browser umumnya hanya mendukung autentikasi yang biasa bukan yang terenkripsi sehingga muncullah kendala untuk pengimplementasian dari HTTP digest authentication ini pada sistim yang ada.

Jika jenis autentikasi ini dipaksakan kepada pengguna untuk digunakan maka cara yang harus ditempuh adalah memastikan semua pengguna memiliki web browser yang telah mendukung tipe autentikasi ini.

Sampai tulisan ini dibuat, beberapa web browser yang telah dapat menggunakan autentikasi HTTP jenis ini adalah:

- Amaya
- Gecko-based: Mozilla application suite, Mozilla firefox, netscape 7+
- KHTML- dan Webkit-based : Konqueror, Google Chrome, Safari
- Tasman-based: Internet Explorer for Mac
- Trident-based: Internet Explorer 7+
- Presto-based : Opera, Opera Mobile, Opera Mini, Nintendo DS browser, Nokia 770 browser, Sony Mylo 1's browser, Wii internet Channel browser

Sedangkan untuk Internet Explorer versi 5 dan 6 masih belum sepenuhnya memenuhi standar implementasi dari metode autentikasi ini.

Sedangkan aplikasi-aplikasi seperti download manager rata-rata belum mampu untuk menggunakan metode autentikasi digest ini.

Jika terlalu banyak pengguna tidak dapat menggunakan metoda ini karena memang aplikasi yang mereka gunakan tidak mampu untuk melakukan metoda ini dan juga tidak mungkin memaksa pengguna dalam jumlah yang sangat besar untuk beralih ke aplikasi lain yang mampu mendukung metode ini, maka implementasi dari metode ini dapat dipertimbangkan kembali dengan mempertimbangkan keamanan data pribadi pengguna atau kenyamanan pengguna dari sistim yang ada.

6. Kesimpulan

Implementasi metode digest ini masih memiliki kekurangan yaitu dukungan aplikasi yang menggunakan autentikasi HTTP sampai saat ini sebagian besar hanya mendukung tipe dasar saja belum mampu menggunakan metoda autentikasi digest.

Untuk dapat menggunakan metoda autentikasi digest ini pengelola sistim harus memaksa pengguna untuk menggunakan aplikasi yang mendukung metode autentikasi ini. Namun, jika faktor kenyamanan dari pengguna menjadi prioritas ataupun banyaknya pengguna yang tidak dapat menggunakan metode autentikasi ini karena kurangnya dukungan aplikasi yang mereka gunakan maka rencana penerapan metode autentikasi ini dapat dikaji ulang jadi atau tidaknya penerapan metode autentikasi ini.

7. Daftar Pustaka

[1] http://en.wikipedia.org/wiki/Digest_access_authentication, waktu akses 20 Mei 2009 pukul 20.00

[2] http://wiki.squid-cache.org/KnowledgeBase/LdapBackedDigestAuthentication, waktu akses 18 Mei 2009 pukul 20.00

[3] Munir, Rinaldi. *Kriptografi*, Penerbit Informatika, 2006.

[4] Dokumentasi internal tim administrator jaringan ITB, 2009