Studi dan Implementasi Sistem Keamanan Berbasis Web dengan Protokol SSL di Server Students Informatika ITB

Hary Fernando - 13505113[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*if15113@students.if.itb.ac.id*

***Abstract*—Keamanan internet sangat penting untuk menjaga kerahasiaan, otentikasi, integritas dari data yang dikirimkan. Demikian juga dengan kemanaan dalam penggunaan internet, apalagi dengan berkembangnya *e-commerce* yang mengutamakan keamanan dalam bertransaksi. Karena itu diperlukan suatu protokol yang menyediakan layanan keamanan untuk komunikasi antar jaringan. Protokol kemanan yang penting adalah SSL (*Secure Socket Layer*) yang dapat menghubungkan server dan klien dengan aman dan terenkripsi serta mencegah penyadapan dan hal lain yang tidak diinginkan.**

Pada makalah ini penulis mencoba untuk mengimplementasikan protokol SSL (*Secure Socket Layer*) pada suatu server di Informatika ITB yaitu server Students Informatika ITB.

Index Terms*—HTTPS, sertifikat digital, SSL, *Secure Socket Layer

I. LATAR BELAKANG

Seiring dengan pekembangan internet yang semakin pesat, maka diperlukan suatu mekanisme dalam melakukan pengaturan keamanan. Salah satu cara yang dapat ditempuh adalah dengan penggunaan sertifikat digital. *Public Key Infrastructure* (PKI) bertujuan untuk menciptakan, mengatur, mendistribusikan penggunaan sertifikat digital ini. Mungkin kita sering melihat penggunaan setifikat digital pada web, tetap penggunaan sertifikat digital tidak hanya terbatas pada layanan web, melainkan juga email, dan aplikasi lainnya.

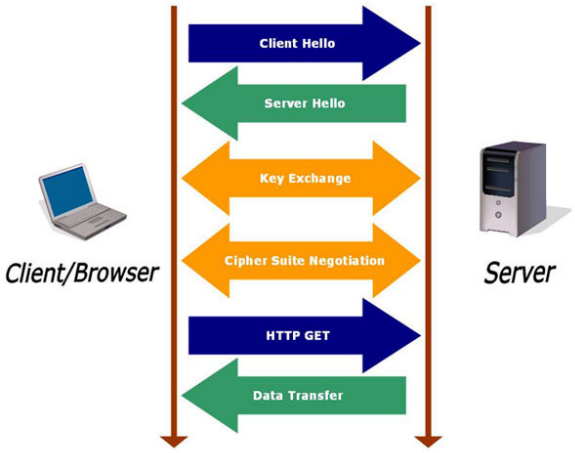
Berikut akan dibahas tentang *Secure Socket Layer* (SSL), *Transport Layer Security* (TSL), *Public Key Infrastructure* (PKI) dan langkah implementasi yang dilakukan untuk membangun sebuah layanan SSL pada situs server Students Informatika ITB.

Diharapkan hasil dari makalah ini adalah sebagai sarana belajar bagi penulis dan tentunya agar hasilnya dapat digunakan oleh pihak yang terkait, seperti mahasiswa informatika ITB sendiri.

II. SSL

Secure Socket Layer adalah suatu protokol yang diciptakan oleh Netscape untuk memastikan keamanan dalam bertransaksi di internet antara *webserver* dan *browser* dari klien.

Protokol ini menggunakan sebuah badan yang biasa disebut CA (*Certificate Authority*) untuk mengidentifikasi memverifikasi pihak-pihak yang bertransaksi.



Gambar 1 Prinsip Kerja SSL

Secara umum, cara kerja protokol SSL adalah sebagai berikut (lihat Gambar 1) :

:

1. Klien membuka suatu halaman yang mendukung protokol SSL, biasanya diawali dengan "https://" pada browsernya.
2. Kemudian *webserver* mengirimkan kunci publiknya beserta dengan sertifikat server.
3. *Browser* melakukan pemeriksaan : apakah sertifikat tersebut dikeluarkan oleh CA (*Certificate Authority*) yang terpercaya? Apakah sertifikat tersebut masih valid dan memang berhubungan dengan alamat situs yang sedang dikunjungi?
4. Setelah diyakini kebenaran dari *webserver* tersebut, kemudian *browser* menggunakan kunci public dari *webserver* untuk melakukan enkripsi terhadap suatu kunci simetri yang dibangkitkan

secara random dari pihak klien. Kunci yang dienkripsi ini kemudian dikirimkan ke webserver untuk digunakan sebagai kunci untuk mengenkripsi alamat URL (*Uniform Resource Locator*) dan data http lain yang diperlukan.

5. Webserver melakukan dekripsi terhadap enkrispi dari klien tadi, menggunakan kunci privat server. Server kemudian menggunakan kunci simetri dari klien tersebut untuk mendekripsi URL dan data http yang akan diperlukan klien.
6. Server mengirimkan kembali halaman dokumen HTML yang diminta klien dan data http yang terenkripsi dengan kunci simetri tadi.
7. Browser melakukan dekripsi data http dan dokumen HTML menggunakan kunci simteri tadi dan menampilkan informasi yang diminta.

II. SSL DAN TSL

SSL merupakan protokol keamanan yang terkenal dan banyak digunakan, namun SSL bukan satu-satunya protokol yang ada untuk keamanan transaksi web. Salah satu protkol yang juga penting dan menjadi pengganti dari SSL adalah TLS.

TLS (*Transport Layer Security*) adalah protokol keamanan dari IETF (*Internet Engineering Task Force*) sebagai pengganti untuk protokol SSL v3.0 yang dikembangkan oleh Netscape. TSL didefinisikan di dalam suatu *Request for Comment*, yaitu pada RFC2246. Banyak protokol pada layer aplikasi yang menggunakan TLS untuk menciptakan koneksi yang aman, antara lain HTTP, IMAP, POP3, dan SMTP

Protokol TSL ini dibangun oleh dua layer :

1. Protokol TLS *record*, digunakan untuk melindungi kerahasian dengan menggunakan enkripi algoritma kunci simetri
2. Protokol TLS *handshake*, yang melakukan autentikasi antara server dan klien dan melakukan negosiasi terhadap algorima enrkripsi dan kinci kriptografi yang akan digunakan di dalam transaksi sebelum protokol aplikasi mengirimkan atau menerima data.

Prinsip kerja protokol TSL adalah sebagai berikut : Pertama-tama, protokol TLS *handshake* melakukan pertukaran kunci dapat menggukaan algoritma kunci asimetrik, seperti RSA atau Diffie-Hellman antara klien dan server. Kemudian protokol TLS *record* membuka saluran yang terenkripsi menggunakan algoritma kunci simetrik seperti RC4, IDEA, DES, atau Triple-DES sehingga informasi dapat dikirimkan dengan aman.

Selain itu, protokol TLS *record* juga bertanggung jawab untuk memastikan bahwa komunikasi antara klien dan server tidak berubah di tengah jalan, oleh karena itu digunakan juga fungsi hash, seperti MD5 dan SHA.

III. PUBLIC KEY INFRASTRUCTURE

Public Key Infrastructure adalah kumpulan perangkat keras, perangkat lunak, orang, peraturan dan prosedur yang diperlukan untuk membuat, mengatur, mendistribusikan, menggunakan, menyimpian dan membatalkan sertifkat digital. PKI terdiri atas beberapa komponen yaitu :

1. pengguna (pemohon sertifikat dan pemakai sertifikat)
2. sertifikat digital
3. CA (*Certificate Authority*)
4. Direktori (menyimpan sertifikat digital dan Certificate Revocation List)

Di dalam kriptografi dan keamanan komputer, terdapat banyak CA yang terpercaya yang digunakan untuk menandatangani sertifikat yang diminta oleh pengguna. Standar yang ditetapkan untuk sertifikat dan yang paling umum digunakan adalah X.509 yang ditetapkan oleh ITU *(International Telecommunication Union)*. Terdapat beberapa versi standar dari X.509 yaitu v1, v2, dan v3. *Field-field* yang terdapat dalam sertifikat standard X.509 dapat dilihat pada Tabel I.

Tabel 1 *Field-Field* yang terdapat dalam X.509

Field	Arti
Version	Versi X.509
Serial Number	Nomor ini plus nama *CA* secara unik digunakan untuk mengidentifikasi sertifikat
Signature Algorithm	Algoritma yang digunakan untuk menandatangani sertifikat.
Issuer	Nama pemberian X.509 untuk *CA*
Validity period	Waktu awal dan akhir periode valid
Subject name	Entitas (individu atau organisasi) yang disertifikasi
Public Key	Kunci publik subjek dan *ID* dari algoritma yang menggunakannya.
Issuer ID	*ID* opsional yang secara unik mengidentifikasi *certificate's issuer*.
Subject ID	*ID* opsional yang secara unik mengidentifikasi *certificate's subject*
Extensions	Bayak ekstensi yang telah didefinisikan.
Signature	Tanda-tangan sertifikat (ditandatangani dengan kunci privat CA).

III. LANGKAH IMPLEMENTASI

Implementasi yang dilakukan disesuaikan dengan kondisi yang ada saat ini di Informatika ITB. Oleh karena server Students Informatika ITB menggunakan sistem operasi Fedora Linux, maka implementasi akan dikerjakan pada sistem ini serta perangkat-perangkat lunak yang diperlukan juga akan disesuaikan dengan yang ada pada server Students Informatika ITB.

A. Penambahan Modul OpenSSL

Untuk membuat suatu server linux mendukung SSl,

salah satu caranya adalah dengan menggunakan program OpenSSL. Sebelumnya diperiksa terlebih dahulu apakah OpenSSL telah terinstall di server ini.

```
[root@students ~]# whereis openssl
```

Jike belum terdapat OpenSSL, perlu dilakukan penambahan paket openSSL. OpenSSL dapat diperoleh di http://www.openssl.org/source. Langkah untuk melakukan instalasi OpenSSL pada server Students Informatika ITB adalah:

```
[root@students ~]# yum install openssl
```

Setelah OpenSSL terinstall, maka server telah siap untuk ditambahkan modul SSL agar dapat berjalan dan mendukung enkripsi sesuai dengan yang diharapkan.

B. Membuat Kunci Privat

Hal selanjutnya yang harus dilakukan adalah membuat kunci privat. Jika ingin mendapatkan setifikat digital dari lembaga CA yang terpercaya, maka kita harus dilakukan pembelian terlebih dahulu. Batasan masalah pada makalah ini adalah jika tidak terdapat CA untuk memeverifikasi, maka akan dibuat self-signed certificate, tapi jika terdapat free SSL Certificate, maka dapat digunakan yang SSL yang free tersebut. Oleh karena pada implementasi ini tidak ditekankan pada pembelian sertifikat ke CA, maka akan dicoba digunakan *self-signed certificate*, yaitu sertifikat digital yang ditanda tangani sendiri.

Pertama-tama buat direktori kerja :

```
[root@students ~]# mkdir /root/ssl

[root@students ssl]# cd /root/ssl
```

Untuk menjaga keamanan dan mencegah pihak-pihak yang tidak berkepentingan dalam menggunakan kunci privat server secara bebas maka dapat digunakan perintah berikut ini untuk menjaga privasi, sehingga hanya akun root yang dapat melakukan akes, baca dan tulis terhadap folder tersebut

```
[root@students ssl]# chmod 700 /root/ssl
```

Pada Implementasi ini akan dibuat sebuah kunci privat untuk server dengan panjang 1024 dan dienkripi menggunakan algoritma Triple-DES :

```
[root@students ssl]# openssl genrsa -des3 -
out server.key 1024
```

Kemudian sistem akan mengeluarkan output sebagai berikut :

```
Generating RSA private key, 1024 bit long
modulus

.....++++++

..++++++

e is 65537 (0x10001)

Enter pass phrase for server.key: xxxx

Verifying - Enter pass phrase for
server.key: xxxx
```

Program ini akan meminta masukan *pass-phrase*, dan kemudian menyimpan kunci privat dari server di file server.key. Masukan *pass-phrase* tersebut jangan sampai lupa. Jika kunci hilang atau *pass-phrase* lupa, maka sertifikat tersebut menjadi tidak berguna lagi.

Menggunakan enrkipsi dengan algoritma Triple-DES seperti cara di atas, server akan menunggu admin untuk memasukan *pass-phrase* setiap kali aplikasi webserver dinyalakan kembali (*restart*) atau sistem di-*reboot*. Salah satu opsi adalah dengan menggunakan kunci privat yang tidak terenkripsi, hal ini dapat dilakukan dengan langkah berikut :

```
[root@students ssl]# openssl genrsa -out
server.key
```

Akan keluar output sebagai berikut

```
[root@students ssl]# openssl genrsa -out
server.key

Generating RSA private key, 512 bit long
modulus

........++++++++++++

..............++++++++++++

e is 65537 (0x10001)
```

Namun jika telah terlanjut menggunakan Triple-DES, admin dapat menghilangkan algoritma Triple-DES tesebut dengan cara :

```
[root@students ssl]# openssl rsa -in
server.key -out server.key.tak-terenkripsi
```

Hal ini tentu memiliki kelemahan karena kunci privat tidak dienkripi di server, tetapi pada sistem yang keamanan telah dijaga oleh firewall serta pengamanan folder dan pengamanan sejenis lainnya dapat menggunakan cara ini untuk menyederhanakan permasalahan dimana setiap kali menjalankan ulang aplikasi, maka admin harus memasukan *pass-phrase*. Terlebih jika yang dapat melakukan akses terhadap server hanya root. Namun, jika kemanan merupakan hal yang penting, maka kita dapat memilih untuk tetap menggunakan Triple-DES, dan memasukan *pass-phrase* setiap kali aplikasi direstart.

C. Membuat CSR *(Certificate Signing Request)*

Setelah terbentuk kunci privat, selanjutnya membuat

CSR (*Certificate Signing Request*). Idealnya, CSR akan dikrimkan ke sebuah CA (Certificate Authority) seperti Thawte atau Verisign, kemudian mereka akan melakukan verifikasi terhadap identitas dari pihak yang meminta sertifikat tersebut untuk ditanda tangani. Karena kita tidak memiliki sertifikat yang dapat ditanda tangani oleh CA tersebut, maka opsi yang lain adalah dengan self-sign CSR.

Perintah berikut akan membuat sebuah file CSR :

```
openssl req -new -key server.key -out
server.csr
```

Kemudian kita akan diminta untuk mengisi berberapa hal yang berkaitan dengan domain dan lokasi tempat kita berada, seperti di bawah ini :

```
[root@students ssl]# openssl req -new -key
server.key -out server.csr

You are about to be asked to enter
information that will be incorporated

into your certificate request.

What you are about to enter is what is
called a Distinguished Name or a DN.

There are quite a few fields but you can
leave some blank

For some fields there will be a default
value,

If you enter '.', the field will be left
blank.

-----

Country Name (2 letter code) [GB]:ID

State or Province Name (full name)
[Berkshire]:Jawa Barat

Locality Name (eg, city) [Newbury]:Students
Informatika

Organization Name (eg, company) [My Company
Ltd]:

Organizational Unit Name (eg, section)
[]:students.if.itb.ac.id

Common Name (eg, your name or your server's
hostname) []:students.if.itb.ac.id

Email Address
[]:if15113@students.if.itb.ac.id



Please enter the following 'extra'
attributes

to be sent with your certificate request

A challenge password []:

An optional company name []:
```

Akan terbentuk file server.csr yang berisi informasi tentang situs kita dalam hal ini server students Informatika ITB.

D. Menanda-tangani Sertifikat

Hal selanjutnya yang harus dilakukan adalah menanda tangani sertifikat yang telah dibuat tadi dengan kunci privat yang juga tadi dibuat sendiri. Karena itulah cara ini disebut *self-sign certificate*.

Perintah untuk untuk menanda tangani sertifikatnya adalah sebagai berikut :

```
[root@students ssl]# openssl x509 -req -days
365 -in server.csr -signkey server.key -out
server.crt

Signature ok

subject=/C=ID/ST=Jawa Barat/L=Students
Informatika/O=My Company
Ltd/OU=students.if.itb.ac.id/CN=students.if.
itb.ac.id/emailAddress=if15113@students.if.i
tb.ac.id

Getting Private key
```

Perintah di atas, berarti bahwa sertifikat valid selama 365 hari. Bagian ini dapat disesuaikan dengan kondisi masing-masing. Sebaiknya masa berlaku sertifikat tidak terlalu singkat tetapi tidak juga tidak terlalu lama.

Setelah semua tahap diatas dilakukan akan didapatkan beberapa file seperti di bawah ini :

```
server.crt

server.csr

server.key.tak-terenkripsi

server.key
```

File server.crt adalah sertifkat yang telah ditanda tangani, file server.csr adalah sertifikat yang belum ditanda tangani yang berisi informasi tentang suatu badan atau institusi, dan file server.key, merupakan kunci privat dari server.

E. Memasukkan Sertifikat ke Server

Server students Informatika ITB menggunakan Apache server untuk layanan HTTP-nya. Oleh karena itu, langkah selanjutnya yang perlu dilakukan adalah mengaktifkan mod_ssl pada apache.

```
yum install mod_ssl
```

Setelah mod_ssl di install di apache, langkah selanjutnya melakukan penambahan file sertifikat yang telah kita tanda-tangani tadi berserta kuncinya ke konfigurasi Apache.

```
cp server.crt /etc/httpd/conf/server.crt

cp server.key /etc/httpd/conf/server.key
```

Kemudian, pada konfigurasi apche, tambahkan dukungan konfigurasi SSL ke Apache dengan melakukan

penambahan baris berikut ke file httpd.conf :

```
Include /etc/httpd/cond.d/ssl.conf
```

Kemudian beberapa perubahan agar SSL dapat berjalan dengan baik. Hasil yang diharapkan akan nampak seperti ini dalam file httpd.conf berikut ini

```
<VirtualHost _default_:443>

  DocumentRoot "/var/www/"

  ErrorLog logs/error_log

  TransferLog logs/access_log


  ScriptAlias /cgi-bin/ "/var/www/chem6/cgi-
bin/"

  <Directory "/var/www/chem6/cgi-bin">

    AllowOverride None

    Options None

    Order allow,deny

    Allow from all

  </Directory>


  SSLEngine on


  SSLCertificateFile /etc/httpd/conf/my-
server.cert

  SSLCertificateKeyFile /etc/httpd/conf/my-
server.key

  <Files ~ "\.(cgi|shtml|phtml|php3?)$">

    SSLOptions +StdEnvVars

  </Files>

  <Directory "/var/www/chem6">

    SSLOptions +StdEnvVars +CompatEnvVars

  </Directory>

  SetEnvIf User-Agent ".*MSIE.*" \

  nokeepalive ssl-unclean-shutdown \

  downgrade-1.0 force-response-1.0

  CustomLog logs/ssl_request_log \

  "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x
\"%r\" %b"

</VirtualHost>
```

Sedangkan pada file ssl.conf dilakukan beberapa perubahan, antara lain menambahkan

```
SSLCertificateFile
/etc/httpd/conf/server.crt
```

Pada langkah ini, sertfikat telah terhubung dengan Apache dan selanjutnya file konfigurasi tersebut dapat disimpan.

Langkah selanjutnya adalah jalankan kembali layanan HTTP dengan langkah :

```
apachectl -restart
```

Atau dapat juga digunakan :

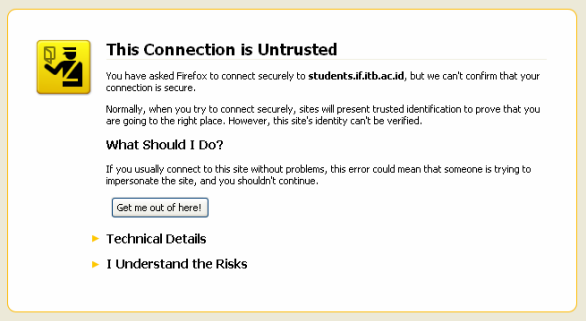
```
/etc/init.d/apache restart
```

F. Testing

Langkah berikutnya adalah melakukan testing atau pengujian terhadap fitur yang telah ditambahkan pada bagian sebelumnya.

Untuk melakukan pengujian dapat dilakukan dengan mengunjungi https://students.if.itb.ac.id, perhatikan huruf "s" pada https yang menyatakan bahwa situs tersbut mendukung *secure* http.

Studi kasus dilakukan pada web browser Mozilla Firefox. Bila terdapat error, seperti pada Gambar 2, jika keluar halaman tersebut, hal ini menanyakan apakah Anda percaya terhadap halaman ini atau tidak.



Gambar 2 browser memberikan peringantan tentang untrusted site

Untuk sekedar uji coba Anda dapat saja menekan tombol, "I understand the Risks" dan percaya bahwa ini adalah situs yang aman, bukan alamat phising. Tetapi, perlu diingat, untuk dunia nyata sebaiknya Anda tidak begitu saja percaya terhadap sertifikat-sertifikat seprti ini jika ingin berhubungan dengan aman.

G. Lain-lain

Hal lain yang perlu dilakukan adalah penbambahan mime cert. Hal ini dilakukan jika server belum mendukung tipe cert, maka dapat ditambahakn di /etc/mime.types hal berikut ini :

```
application/x-x509-ca-cert      crt
```

Dengan demikian implementasi protokol SSL pada server Students Informatika ITB telah dilakukan

IV. HASIL IMPLEMENTASI

Salah satu hasil dari eksperimen ini adalah bahwa telah berhasil diterapkan protokol SSL dan juga telah dilakukan pengujian sistem students.if.itb.ac.id. Hasil yang diperoleh cukup baik serta semua prosedur dapat dengan baik.

Namun terdapat berberapa *warning* saat suatu klien baru yang mencoba melakukan akses terhadap server . *Warning* ini terjadi dikarenakan server masih menggunakan sertifikat yang ditanda tangani sendiri (*self-signed certificate)*. Hal ini tentu tidak baik dan tidak cocok jika diterapkan pada sistem komersil.

Diharapkan untuk sistem yang berhubungan dengan klien dan digunakan secara luas, lebih baik membeli ke CA yang terpercaya seprti Thawte dan Verisign. Sehingga klien lebih percaya terhadap transaksi yang dilakukan.

V. KESIMPULAN

Implementasi penggunaan SSL pada server students Informatika ITB telah berhasil dikembangkan dan diharapkan dapat digunakan sebagai media belajar, studi kasus dan keperluan lainnya.

REFERENCES

[1] Spesifikasi SSL 2 http://www.mozilla.org/projects/security/pki/nss/ssl/draft02.html

[2] Spesifikasi SSL 3.0 http://www.freesoft.org/CIE/Topics/ssl-draft/3-SPEC.HTM

[3] Free SSL Certificate Trial http://www.verisign.com/ssl/buy-ssl-certificates/free-ssl-certificate-trial

[4] SSL versus TLS What's the difference? http://luxsci.com/blog/ssl-versus-tls-whats-the-difference.html

[5] mod_ssl: The Apache Interface to OpenSSL www.modssl.org

[6] mod_ssl - Apache HTTP Server http://httpd.apache.org/docs/2.0/mod/mod_ssl.html

[7] OpenSSL http://www.openssl.org

[8] Generating an SSL Certificate with Apache+mod_ssl, Slacksite.com

[9] Transport Layer Security http://www.topbits.com

[10] Munir, Rinaldi, “Kriptografi”, Informatika ITB.

[11] Secure Sockets Layer (SSL) technology protects your Web site and makes it easy for customers to trust you. http://securesocketslayercertificates.com/securesocketslayercertificates-english-eu.html

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, 12 Mei 2010

Hary Fernando
13505113