Analisis Keamanan HTTPS Menggunakan SSL/TSL

Muhammad Anis / 13508068
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
if18068@students.if.itb.ac.id

***Abstract*—Perkembangan dunia internet saat ini semakin pesat. Salah satu aspek yang juga semakin berkembang adalah dunia web. Saat ini pengguna website semakin banyak dan aplikasi yang ditawarkan pun semakin banyak pula. Pesatnya perkembangan ini tentu harus diikuti oleh factor keamanan informasi dimana data-data pengguna harus dijamin aman. Maraknya kasus pencurian data-data pribadi seperti password,email, nomor kartu kredit menjadi salah satu alasan mengapa faktor keamanan data menjadi sangat penting. Oleh karena itu, dikembangkan suatu protokol dimana data akan dienkripsi terlebih dahulu sebelum dikirim ke dunia internet. Penggunaan TSL/SSL pada dunia web merupakan salah satu teknik yang digunakan untuk menjaga keamanan data. Protokol yang disebut HTTPS ini merupakan protokol HTTP biasa dimana data yang melaluinya harus dienkripsi sedemikian rupa sehingga tidak mudah dicuri oleh pihak yang tidak memilki hak atas data tersebut.**

***Index Terms*—SSL.TSL,HTTPS,Keamanan data**

1. PENDAHULUAN

Perkembangan dunia website saat ini semakin pesat Peekembangan ini telah mencapai titik dimana kehidupan *virtual* di dunia web juga mempengaruhi kehidupan nyata sehari-hari. Penggunaan data-data pribadi yang mungkin tiap harinya dikirim melalui paket data dari suatu tempat ke tempat lain sudah tidak dapat dihindari. Terkadang data-data pribadi ini sangat penting sehingga tidak boleh ada pihak yang tidak memiliki hak untuk mendapatkannya. Salah satu contoh adalah penggunaan jasa *PayPal.* Data yang ditransfer melalui website ini bukan data yang boleh dilihat oleh orang lain, karena data yang ditransfer adalah data kartu kredit yang jika orang lain mengetahuinya maka akan sangat merugikan pengguna internet tersebut. Hal ini diperburuk mengingat pengambilan atau pencurian data melalui internet sangat mudah untuk dilakukan. Sebuah serangan internet yang biasa disebut *Sniffing Attack* dan *Man in The Middle Attack* sangat mudah dilakukan untuk mencuri paket data dari satu tempat ke tempat lain.

Salah satu strategi yang digunakan untuk mencegah hal ini adalah penggunaan metode enkripsi untuk mencegah orang yang mungkin mencuri paket data tersebut untuk mengetahui informasi apa yang sebenarnya dikirim oleh pengirim tersebut. Namun tentu saja pihak yang benar-benar ingin dituju harus dapat mendekripsi ulang data tersebut. Metode enkripsi dekripsi melali jalur internet yang biasa digunakan adalah protokol SSL/TSL.Penggunaan web server yang menggunakan protokol SSL/TSL sebagai bagian dari protokol HTTP tersbut dinamakan HTTPS.

2. LANDASAN TEORI

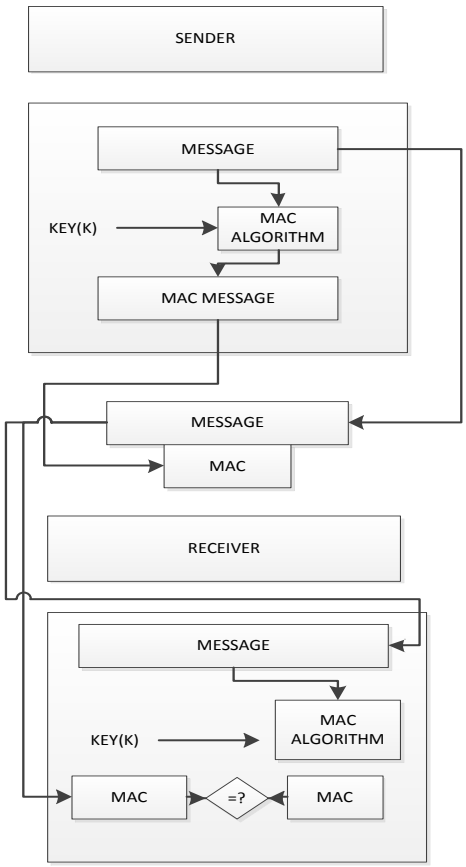
2.1 Kriptografi Simetrik

Kriptografi simetrik adalah teknik enkripsi dekripsi dimana untuk mengenkripsikdan dan mendekripsikan data menggunakan kunci yang sama atua menggunakan kunci yang dapat ditransform dari satu bentuk ke bentuk lain. Algoritma kriptografi simetrik dapat dibagi menjadi dua yaitu *stream cipher* dan *block cipher*. *Stream Cipher* mengnekripsi bit-bit dari pesan, sedangkan *Block Cipher* membagi-bagi bit pesan berdasarkan suatu panjang blok tertentu. Algoritma kriptografi simetrik yang cukup terkenal adalah AES (Rijndael).

2.2 *Message Authentication Code*

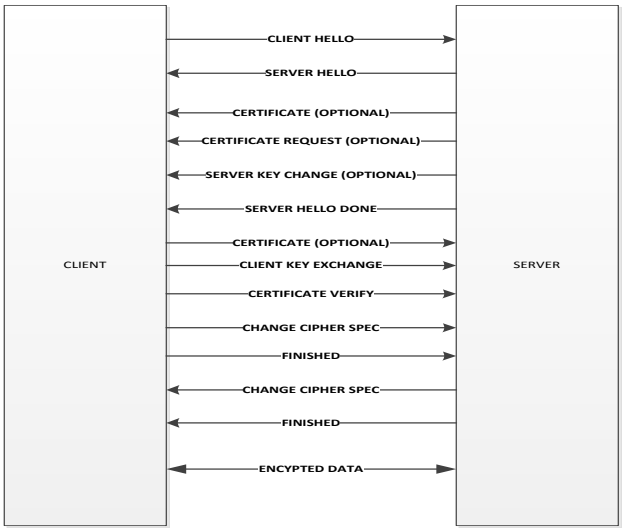
Message Authentication Code (MAC) adalah teknik kriptografi yang digunakan untuk menambahkan suatu informasi pada suatu pesan untuk mengecek kebenaran dari pesan tersebut. Sebuah fungsi algoritma MAC menerima sebuah kunci dan pesan dengan panjang tertentu dan menghasilkan sebuah pesan yang akan digabung dengan pesan aslinya yang dapat digunakan untuk mengecek adanya perubahan pada pesan asli yang dikirimkan.

Perbedaan antara *Message Auhtentication Code* dengan tanda tangan digital adalah hasil dari fungsi MAC sama-sama dihasilkan dan diverifikasi menggunakan kunci yang sama. Berikut adalah diagram alir penggunaan *Message Authentication Code:*



2.3 TLS/SSL

TLS (Transport Layer Security) merupakan kelnjutan dari SSL (Secure Socket Layer). TLS/SSL ini adalah protokol yang menyediakan jalur komunikasi yang aman antara dua *host* yang berkomunikasi melalui jalur internet atau biasa disebut menggunakan jalur *socket.* Keamanan yang disediakan disini adalah pengenkripsian paket data pada Network Layer (atau biasa dikenal dengan OSI Layer atau TCP/IP) yang berada diatas Transport Layer. Proses pengenkripsian pada protokol ini menggunakan *symmetric crypthography* dan *Message Authentication Code.* Setiap paket data yang dikirim melalui protokol TLS/SSL ini akan dienkripsi terlebih dahulu menggunakan kunci yang telah disepakati antara *client* dan *server* sehingga hanya kedua pihak tersebut yang dapat mengenkripsi atau mendekripsi paket data tersebut. Berikut adalah skema dasar protokol SSL/TSL :



2.4 HTTPS

HypertText Transfer Protocol Secure (HTTPS) pada awalnya dikembangkan oleh Netscape. HTTPS adalah protokol HTTP yang dilengkapi dengan protocol SSL/TSL yang dapat memberikan komunikasi yang aman antara *client* dan *server,* dalam hal ini adalah web server yang menggunakan protokol SSL/TSL. Protokol HTTPS ini mengenkrip dan mendekrip halaman yang diminta oleh klien serta halaman yang dikirim oleh web server. Pengenkripsian ini juga termasuk pengiriman data-data pada saat mengrimimkan form ke web. Sebagai perbedaan dengan protokol HTTP biasa, penggunaan web yang menggunakan protoko, HTTPS dimulai dengan URL https://. Selain itu, port yang digunakan pun berbeda dengan port yang biasa digunakan oleh port http biasa. HTTPS didesain untuk mencegah serangan-serangan yang dapat mencuri data-data pribadi seperti *Man In The Middle Attack* serta *Eavesdropping Attack.*

3. IMPLEMENTASI HTTPS

3.1 Membuat HTTPS Server dengan UNIX-Like Server menggunakan OpenSSL

Untuk pembahasan kali ini, penulis akan membuat sebuah server http yang dilengkapi dengan SSL/TSL atau HTTPS. Untuk system operasi yang akan digunakan kali ini adalah system operasi FreeBSD. Berikut adalah langkah-langkah untuk menginstall sebuah HTTPS server pada FreeBSD menggunakan OpenSSL

- Install OpenSSL

Pertama-tama kita akan menginstall openssl melalui koleksi ports dari FreeBSD

```
#cd /usr/ports/security/openssl
#cp Makefile makefile.old
#echo EXTRACONFIGURE+=no-idea >>
Makefile
#make install clean
#rehash
```

Setelah selesai menginstall, kita harus menambahkan konfigurasi pada beberapa file:

```
#cp /etc/make.comf /etc/make.conf/old
#echo "WITH_OPENSSL_PORT=YES" >>
/etc/make.conf
#mv /etc/ssl/opens;.cnf
/etc/ssl/openssl.cnf.old
#cd /usr/local/openssl
#cp openssl.cnf.sample openssl.cnf
```

Kemudian kita harus mengenerate sertifikat untuk server tersebut

```
#cd /usr/local/openssl
#cp misc/CA.pl certs
#sed –I .old 's/365/1095/' openssl.cnf
#cd /usr/local/openssl/certs
#setenv OPENSSL /usr/local/bin/openssl
# ./CA.pl –newca
```

- Install HTTP Server
  Setelah menginstall OpenSSL kita juga harus menginstall http server pada server tersebut. Untuk software yang digunakan sebagai HTTP server adalah Apache 2.2 Langkah-langkah menginstall Apache pada FreeBSD adalah sebagai berikut:

```
#cd /usr/ports/www/apache22
#make config; make install clean
#rehash
```

Setelah itu kita harus mengkonfigurasu file httpd.conf sesuai dengan kebutuhan kita, termasuk untuk mengadakan fungsi openSSL pada Apache22

3.2 HTTPS Server dengan JAVA

Bahasa pemrograman JAVA telah menyediakan *Application Programming Interface* (API) untuk membuat koneksi soket menggunakan protokol SSL/TSL. Berikut adalah implementasi sebuah HTTPS Server sederhana mengunakan API dari bahasa pemrograman JAVA.

Pertama-tama kita akan mengenerate sertifikiat untuk autentukasi. Sertifikat dibuat untuk membuat jalur komunikasi yang aman antara client dan server. Cara mengenerate sertifikat menggunakan J2SE adalah sebagai berikut (dilakukan menggunakan sistem operasi Windows) :

- Buka command prompt
- Pastikan jalur folder bin pada jre telah masuk pada variable sistem PATH pada Windows
- Lalu ketik perintah berikut pada prompt:
- C:\> `keytool -genkey -keystore <nama_file> -keyalg <nama_algoritma> -alias qusay`
- Ganti <nama_file> dengan nama yang Anda inginkan dan <nama_algoritma> dengan algortima yang Anda inginkan. Algoritma yang biasa digunakan adalah RSA

Berikut adalah kode pembuatan sebuah server HTTP dengan menggunakan protokol SSL/TSL (dalam kasus ini digunakan SSL v3.0)

Class httpsServer.cs

```
package sslserver;

import java.io.FileInputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.security.KeyStore;
import javax.net.ServerSocketFactory;
import javax.net.ssl.KeyManagerFactory;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLServerSocket;

public class httpsServer implements Runnable {
      String keyFile = "generatedFiles";
      char password[] = "123456".toCharArray();
      char generatedKey[] =
"654321".toCharArray();

      //port number
      public static final int HTTPS_PORT = 443;
      public ServerSocket getServer(){
         try{
            KeyStore ks =
KeyStore.getInstance("JKS");
            ks.load(new
FileInputStream(keyFile),password);
            KeyManagerFactory kmf =
KeyManagerFactory.getInstance("SunX509");
            kmf.init(ks,generatedKey);
            SSLContext sslcontext =
SSLContext.getInstance("SSLv3");
            ServerSocketFactory ssf =
sslcontext.getServerSocketFactory();
            SSLServerSocket serversocket =
(SSLServerSocket)ssf.createServerSocket(HTTP
S_PORT);
            return serversocket;
```

```
        }catch(Exception e){
            System.err.println(e.getMessage());
        }
        return null;
    }

  public void run() {
    ServerSocket connection;
    try{
        connection  = getServer();
        while(true){
            Socket client = connection.accept();
            ClientHandler ch = new
ClientHandler(client);
            Thread t2 = new Thread(ch);
            t2.start();
        }
    }catch(Exception e){
        System.err.println(e.getMessage());
    }
  }

}
```

Class clientHandler.cs

```
package sslserver;

import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.Socket;
import java.util.StringTokenizer;

class ClientHandler implements Runnable{

  Socket client;
  BufferedReader clientBr;
  DataOutputStream clientDOS;
  ClientHandler(Socket client) {
    this.client = client;
    try{
        clientBr = new BufferedReader(new
InputStreamReader(this.client.getInputStream()))
;
        clientDOS = new
DataOutputStream(this.client.getOutputStream())
;
    }catch(Exception e){
        System.out.println(e.getMessage());
    }
  }

  public void run() {
    try{
      // get a request and parse it.
     String request = clientBr.readLine();
     System.out.println( "Request: "+request );
     StringTokenizer st = new StringTokenizer(
request );
       if ( (st.countTokens() >= 2) &&
        st.nextToken().equals("GET") ) {
         if ( (request =
          st.nextToken()).startsWith("/") )
           request = request.substring( 1 );
         if ( request.equals("") )
           request = request + "index.html";
         File f = new File(request);
         genFiles(clientDOS, f);
       } else {
         clientDOS.writeBytes( "400 Bad
Request" );
       }
       client.close();
    }catch(Exception e){

    }
  }

  private void genFiles(DataOutputStream out,
File f) throws IOException {
    try {
      DataInputStream in = new
        DataInputStream(new
FileInputStream(f));
      int len = (int) f.length();
      byte[] buf = new byte[len];
      in.readFully(buf);
      in.close();
      out.writeBytes("HTTP/1.0 200 OK\r\n");
      out.writeBytes("Content-Length: " +
        f.length() +"\r\n");
      out.writeBytes("Content-
Type:text/html\r\n\r\n");
      out.write(buf);
      out.flush();
    } catch (Exception e) {

out.writeBytes("<html><head><title>error</title
>"+
        "</head><body>\r\n\r\n");
      out.writeBytes("HTTP/1.0 400 " +
e.getMessage() + "\r\n");
      out.writeBytes("Content-Type:
text/html\r\n\r\n");
      out.writeBytes("</body></html>");
      out.flush();
    } finally {
      out.close();
    }
  }

}
```

Class main.cs

```
package sslserver;

import java.io.PrintStream;
import javax.net.ssl.SSLServerSocket;
import javax.net.ssl.SSLServerSocketFactory;
import javax.net.ssl.SSLSocket;
import javax.net.ssl.SSLSocketFactory;

public class Main {

    public static void main(String[] args) {
        // TODO code application logic here
        httpsServer https = new httpsServer();
        Thread t = new Thread(https);
        t.start();

        PrintStream out = System.out;
      out.println("\nDefault SSL socket factory:");
      try {
        // Generating the default
SSLServerSocketFactory
        SSLServerSocketFactory ssf =
(SSLServerSocketFactory)
          SSLServerSocketFactory.getDefault();

System.out.println("\nSSLServerSocketFactory
class: "
          +ssf.getClass());
        String[] dcsList =
ssf.getDefaultCipherSuites();
        out.println("   Default cipher suites:");
        for (int i=0; i<dcsList.length; i++) {
          out.println("      "+dcsList[i]);
        }

        // Genearting SSLServerSocket
        SSLServerSocket ss
          = (SSLServerSocket)
ssf.createServerSocket();
        System.out.println("\nSSLServerSocket
class: "
          +ss.getClass());
        System.out.println("   String:
"+ss.toString());

        // Generating the default SSLSocketFactory
        SSLSocketFactory sf =
          (SSLSocketFactory)
SSLSocketFactory.getDefault();
        out.println("\nSSLSocketFactory class: "
          +sf.getClass());
        dcsList = sf.getDefaultCipherSuites();
        out.println("   Default cipher suites:");
        for (int i=0; i<dcsList.length; i++) {
          out.println("      "+dcsList[i]);
        }

        // Genearting SSLSocket
        SSLSocket s
          = (SSLSocket) sf.createSocket();
        System.out.println("\nSSLSocket class:
"+s.getClass());
        System.out.println("   String:
"+s.toString());
      } catch (Exception e) {
        System.err.println(e.toString());
      }
    }

}
```

Setelah mejalankan main.java, tersebut cukup membuka di browser https://localhost/<nama_file>

4. ANALISIS KEAMANAN HTTPS

4.1 HTTPS vs HTTP

Seperti telah disebutkan diatas, bahwa data-data yang melalui protokol https akan dienkrip terlebih dahulu sebelum diteruskan melalui jalur internet. Kita akan melihat perbedaan antara data yang dikirim melalui protokol HTTP dan HTTPS menggunakan perangkat lunak *Wireshark (*www.wireshark.org).

- Penangkapan data HTTP menggunakan *Wireshark*
  - GET / HTTP/1.1
    Host: 167.205.34.51
    User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:2.0.1) Gecko/20100101 Firefox/4.0.1
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
    Accept-Language: en-us,en;q=0.5
    Accept-Encoding: gzip, deflate
    Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
    Keep-Alive: 115
    Connection: keep-alive
    Cookie: PHPSESSID=38o9jv6jii6qk106lt2o2m1iu3
    Cache-Control: max-age=0
- Penangkapan data HTTPS menggunakan *Wireshark*
  - UNIX Server
    Data dari server ini akan ditulis dengan dalam hexa karena adanya karakter-karakter yang tidak bisa ditulis:

    URL : https://167.205.34.52/

160301010610000102010024a306a311478
8526872b321a90021b7f8151c9966e663da
c97e39bac33634b80b9471f1ed5f5213097a
fc2ed4c6ecbf958226fead8c6933c443901f2
bdc70c21be968d6461cd51588d962c959d3
9b206bbfd8e2ea9c9436105215288c89d95
dd8f3386f212dd3b44d8f097df4f9eead599
bb2ce4f6ef57e136fa86be707084bf9f88955
94d693d2e65d6ebb3619d181b4c049a276f
b51827a48dd10852390625830440cd1b4e8
bf32e9f32d348b19cc6a3760b2e8706e07fd
0118df9fceeffbc384773d49fa0bdb74b9646
0b0edbd0cfda532bf4385fc0cd60a943f517a
abb10b8623a6d6daa9c294b818afcb8b8b6
3b5bcb0435166f8a8998171d990727f0914
030100010116030100 3044c2126318a3d0b
5e1b93399e07afa9058b929a4459defbfef2d
8028afd1038606fa1240bf9cacde388fdde35
0fa377c

Sedikit teks yang bisa diprint:

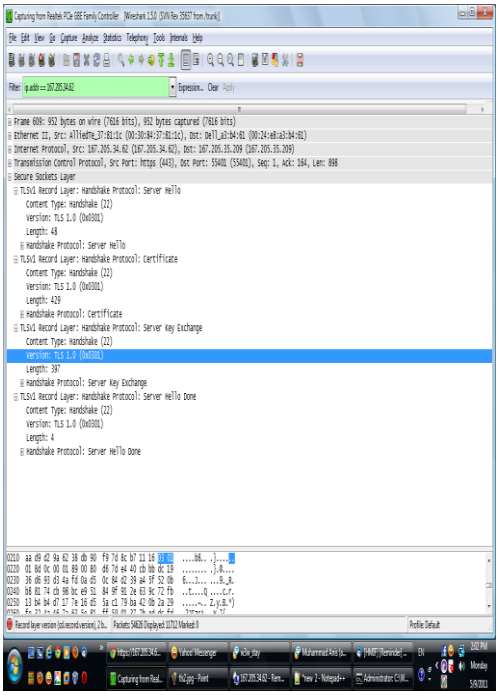
$GRhr!!fc~964q_R z.&i3C+phFbY
k6R(]8o!-M
}YOn~okKU]n6IvQzH#bX0D2-
4j7`pnw=Itd`2C
?Qz#cCQfqr 0Dc3zX)E-(@8P7|

- JAVA Server

URL:
https://167.205.34.62:4453/index.html
Isi dari index.html :

```
<html>
<body>
<h1> Hello World !</h1>
<body>
</html>
```

Berikut adalah *printscreen* dari tangkapan paket data komunikasi http yang melalui komunikasi data https



0,M%+=823JeI9#00 R0 *H010U
localhost0091110234847Z1911082348
47Z010U localhost00
*H0%'jm_u#cAXc[syu||n'\j2z#+)(/f`k(
Wg;H"8H:@WG5F=qbScgz(40
*HjIlh_o'74fA$3UTqO!]huRAA|{J$O5
ojh@HD)Ej\)|V.KS[.BlKh-L=b8}}@6J
9_RtQ.cr~ZyB*)2JFzc^Y7{3F;rx[xt}3bt}h
9>$!y>..}9q?[Sduv+:>n9)LK(w^xtlC:'6ae#
k^D
J`;d$
//|6cs/lR(1s]IJ8hR;eaqO!0"7zu)2GG="
<0s+JQN"/3F

4.2 Analisis HTTPS dari Segi Kriptografi

HTTPS yang menggunakan protokol SSL/TLS sebagai tambahan untuk menjamin keamanan sudah sangat baik dan sulit untuk dipecahkan dari segi kriptografi. Hal ini disebabkan karena adanya pertukaran kunci antara server dan client yang digunakan sebagai dasar untuk enkripsi data. Kunci yang digunakan salah satunya menggunakan algoritma RSA untuk membangkitkan dua kunci yang dapat digunakan oleh client dan server. Kekuatana algoritma RSA terletak pada sulitnya memfaktorkan sebuah bilangan menjadi faktor prima yang tepat. Apaagi jika bilangan tersebut adalah sebuah bilangan yang sangat besar. Sebagai tambahan pada pembuatan server HTTPS menggunakan JAVA API, digunakan faktor tambahan untuk mengecek kebenaran dari sertifikat yang akan dikirimkan kepada client. Faktor tersebut adalah digunakannya dua buah password untuk mengecek keaslian sertifikat. Pada contoh pembuatan HTTPS server diatas kedua password tersebut dimasukkan secara

eksplisit ke dalam server, untuk memudahkan pengecekan jalan tidaknya HTTPS server yang diinginkan.

Dapat dilihat dari perbedaan tangkapan paket data antara protocol HTTP dan HTTPS pada contoh diatas. Protocol HTTP mentransmisikan datanya menggunakan plain teks sehingga jika dicuri oleh orang lain akan sangat membahayakan jika mengandung data-data rahasia. Sedangkan pada protocol HTTPS, data yang ditransmisikan sudah dienkripsi menggunakan sertifikat dari klien dan server sehingga akan sangat sulit untuk diketahui apa saja yang dikirimkan melalui paket data tersebut.

Selain itu, komunikasi antara client dan server ini juga dijamin kebenaran datanya karena setiap pertukaran data akan diikuti dengan suatu model *hashing*, dimana integritas suatu paket data yang dikirim akan dicek menggunakan teknik *Message Authentication Code* yang juga telah dijelaskan pada bagian landasan teori.

4.3 Analisis HTTPS dari Segi Keamanan Jaringan

Meskipun protokol HTTPS ini dapat dikatakan aman untuk transmisi data karena kekuatan enkripsinya, ternyata protokol ini masih memilki celah jika dikaitkan dengan keamanan jaringan internet. Salah satu celah yang digunakan untuk mengalahkan protokol HTTPS ini adalah bahwa sebelum masuk ke jaringan yang aman (jaringan yang telah menggunakan SSL/TSL), biasanya pengguna diarahkan dari jaringan yang tidak aman (jaringan yang paket datanya dapat ditangkap dengan mudah dan tidak dienkripsi). Misalkan, untuk menuju suatu halaman https://, sorang pengguna harus menekan sebuah tombol di halaman http:// biasa. Atau pengguna biasanya mengetikkan http:// pada browser mereka untuk menuju suatu halaman https://. Hal ini akan mengakibatkan diadakannya suatu usaha untuk mengarahkan dari jaringan yang tidak aman menuju jaringan yang aman. Blok pemisah antara yang aman dan tidak aman tersebut lah yang sering dijadikan celah untuk mengakali protokol HTTPS.

Salah satu cara untuk memanfaatkan celah tersebut adalah dengan cara melakukan *Man In The Middle Attack.* Untuk makalah kali ini tidak dijelaskan secara lengkap bagaimana cara untuk melakukan hal tersebut. Secara garis besar, sebuah serangan MITM dilakukan dengan menjadikan suatu computer sebagai *proxy* yang menjembatani komunikasi antara client dan server sebenarnya yang ingin berkomunikasi. Hal ini dilakukan dengan cara melakukan *ARP Poisoning* atau *IP Spoofing* dimana sebuah computer klien dibuat menganggap sebuah computer yang sebenarnya tidak berhak mendapatkan data tersebut sebagai computer yang dituju sebenarnya.

5. KESIMPULAN DAN SARAN

Protokol SSL/TLS merupakan salah satu protokol yang menggunakan kekuatan algoritma-algoritma kriptografi sebagai landasan untuk membuat sebuah jalur komunikasi yang aman antara sebuah klien dan server. Kekuatan SSL/TSL ini diperkuat dengan mampunya protokol ini unttuk menghandle beberapa algoritma sekaligus tanpa terjadi bentrok satu sama lainnya. Kemampuan protokol SSL/TSL ini salah satunya dimanfaatkan untuk membuat sebuah jalur komunikasi yang aman pada komunikasi di website. Hal ini disebabkan karena maraknya pencurian data-data penting dan pribadi yang terjadi pada website.

Kemampuan protokol SSL/TSL dalam menjaga komunikasi HTTPS antara klien dan server dapat dikatakan cukup dan sudah sangat baik. Hal ini disebabkan karena untuk memecahkan dan mendekripsikan suatu pesan yang sudah dikodekan menggunakan protokol ini sulit untuk dipecahkan secara *brute force.* Namun,dari segi jaringan ternyata masih ada celah-celah yang dapat dimanfaatkan untuk mencuri paket-paket data tersebut sebelum data tersebut mask ke dalam jaringan yang aman.

Untuk mencegah hal ini, sebaiknya ditambahkan suatu mekanisme dimana sebuah computer tidak mungkin dianggap sebagai computer lain. Teknik yang digunakan sekarang yang berdasarkan penggunaan *MAC Address* dari tiap *interface* jaringan masih dapat dimanipulasi sedemikian sehingga berubah nilainya.

6. DAFTAR PUSTAKA

[1] Munir, Rinaldi. 2011. Bahan Kuliah IF3054 Kriptografi. Departemen Teknik Informatika, Institut Teknologi Bandung.

[2] http://java.sun.com/developer/technicalArticles/Security/secureinternet/

[3] http://www.zdnet.com/news/researcher-demonstrates-ssl-attack/271389

[4] http://www.herongyang.com/JDK/HTTPS-Server-Test-Program-HttpsHello.html

[5] http://en.wikipedia.org/wiki/Transport-Layer_Security

7. 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, 9 Mei 2011

ttd

Muhammad Anis / 13508068