

BAB 11

Keamanan WEB

11.1 Pendahuluan

Pembahasan tentang web programming belum lengkap apabila belum mempelajari tentang keamanan dalam aplikasi. Fasilitas yang melimpah, fungsi yang sangat banyak tidak akan berarti apabila aplikasi kita gagal dalam hal pengamanan data.

Pada bab ini, kita akan mempelajari bagaimana mengamankan komunikasi antara server dan client melalui SSL. Kita juga akan mempelajari tentang 10 celah keamanan pada aplikasi web dan mempelajari bagaimana cara menanggulangnya.

11.2 SSL

SSL telah menjadi standar de facto pada komunitas untuk mengamankan komunikasi antara client dan server. Kepanjangan dari SSL adalah Secure Socket Layer; SSL adalah sebuah layer protocol yang berada antara layer TCP/IP standar dengan protocol di atasnya yaitu application-level protocol seperti HTTP. SSL memungkinkan server untuk melakukan autentikasi dengan client dan selanjutnya mengenkripsi komunikasi.

Pembahasan tentang operasi SSL pada bab ini bertujuan agar kita mengetahui penggunaan teknologi ini untuk mengamankan komunikasi antara server dengan client.

11.2.1 Mengaktifkan SSL pada aplikasi.

Untuk mengetahui keuntungan SSL pada aplikasi, kita perlu melakukan konfigurasi server untuk menerima koneksi SSL. Pada servlet container yang berbeda akan berbeda pula cara untuk melakukannya. Disini kita akan belajar tentang melakukan konfigurasi Sun Application Server 8.1

11.2.2 Certificates

Salah satu bagian yang perlu kita konfigurasi untuk membangun komunikasi SSL pada server adalah sebuah security certificate. Bisa kita bayangkan sebuah certificate dalam hal ini seperti sebuah passport : dimana memiliki informasi-informasi penting pemilik yang bisa diketahui oleh orang lain. Sertifikat tersebut biasanya disebar oleh Certification Authorities (CA). Sebuah CA mirip seperti passport office : dimana CA bertugas untuk melakukan validasi sertifikat pemilik dan menandai sertifikat agar tidak dapat dipalsukan.

Sampai saat ini sudah banyak Certification Authorities yang cukup terkenal, salah satunya adalah Verisign. Menentukan pemilihan CA adalah tanggung jawab atau wewenang dari seorang admin untuk memberikan sebuah sertifikat keamanan yang berlaku pada server.

Apabila pada suatu kasus ditemukan tidak adanya certificate dari CA, sebuah certificate temporer (sementara) dapat dibuat menggunakan tools dari Java 1.4 SDK. Perlu Anda catat bahwa client biasanya tidak melanjutkan transaksi yang memerlukan tingkat keamanan yang tinggi dan menemukan bahwa certificate yang digunakan adalah certificate yang kita buat.

11.2.3 Membuat certificate private key

Untuk menyederhanakan permasalahan ini, akan lebih mudah bila dengan melakukan operasi dimana certificate disimpan. Hal ini dapat ditemukan do direktori %APP_SERVER_HOME%/domains/domain1/config.

Buka directory menggunakan command line. Selanjutnya panggil command berikut ini:

```
keytool -genkey -alias keyAlias  
-keyalg RSA -keypass keypassword  
-storepass storepassword  
-keystore keystore.jks
```

- *keyAlias* – adalah alias atau ID dimana certificate ini akan menunjuk kepada siapa.
- *keypassword* – adalah password untuk private key yang digunakan dalam proses enkripsi.
- *storepassword* – adalah password yang digunakan untuk keystore.

Dalam hal ini mungkin sedikit membingungkan dimana dibutuhkan dua password untuk membuat sebuah certificate. Untuk mengatasinya, bisa kita ingat bahwa key yang dimasukkan disebut juga keystore. Keystore dapat menyimpan satu atau beberapa key. Keypassword merupakan password dari private key yang akan digunakan pada certificate, sedangkan storepassword merupakan password dari key yang ada di dalam keystore. Pada direktori yang sedang kita operasikan sudah memiliki sebuah keystore file dengan sebuah password, sehingga kita perlu menset nilai storepass menjadi : changeit.

Password ini dapat diganti menggunakan keytool seperti ini:

```
keytool -keystore keystore.jks -storepass newPassword
```

11.2.4 Membuat cerificate

Setelah kita selesai membuat key yang akan digunakan oleh certificate sekarang kita dapat membuat file certificate itu sendiri:

```
keytool -export -alias keyAlias  
-storepass storepassword  
-file certificateFileName  
-keystore keystore.jks
```

Pada baris diatas dijelaskan bahwa keytool digunakan untuk membuat certificate file menggunakan private key yang disebut juga *keyAlias* yang berada pada keystore.

11.2.5 Mengatur certificate

Agar aplikasi server dapat mengenali certificate yang sudah kita buat, kita perlu menambahkannya pada daftar dari trusted certificates. Server memiliki file bernama cacerts.jks yang di dalamnya terdapat certificates. Kita dapat menambahkan certificate kita dengan menggunakan keytool berikut ini:

```
keytool -import -v -trustcacerts
-alias keyAlias
-file certificateFileName
-keystore cacerts.jks
-keypass keypassword
```

11.2.6 Membuat secure HTTP listener

Setelah kita sudah berhasil membuat certificate dan mendaftarkannya untuk aplikasi server, sekarang kita akan membuat sebuah HTTP listener yang dapat digunakan untuk membuat komunikasi yang aman.

Untuk melakukannya, langkah pertama login ke administration console. Selanjutnya klik tab Configuration dan buka HTTP Service :

The screenshot displays the Sun Java System Application Server Admin Console in a Mozilla Firefox browser. The address bar shows the URL `http://localhost:4849/admingui/TopFrameset`. The console interface features a navigation tree on the left with the following structure:

- Resources
 - JDBC
 - Persistence Managers
 - JMS Resources
 - JavaMail Sessions
 - JNDI
 - Connectors
 - Configuration
 - Web Container
 - EJB Container
 - Java Message Service
 - Security
 - Transaction Service
 - HTTP Service
 - HTTP Listeners
 - Virtual Servers
 - server
 - __asadmin
 - ORB
 - Thread Pools
 - Admin Service
 - Connector Service

The main content area displays a welcome message and a 'Common Tasks' section with the following items:

- Search Log Files**: Search the log files of your application server instances.
- Deploy Enterprise Application (.ear)**: An enterprise application is packaged in an EAR file, a type of archive file that contains any type of J2EE standalone modules, such as WAR and EJB JAR files.
- Deploy Web Application (.war)**: A Web application is packaged in a WAR file, a type of archive file that contains components such as servlets and JSP pages.
- Create New JDBC Connection Pool**: A connection pool is used to provide database access to deployed applications.
- View Monitoring Data**: Monitor various aspects of the application server.

Below the 'Common Tasks' section, there are several links and notices:

- Stay connected. [Sign up](#) to become a registered user today.
- Need help? Learn how to get [free help](#) from Sun.
- Try [NetBeans 4.1](#), a full featured J2EE 1.4 based IDE.
- Join [Project GlassFish](#) - Sun's open source application server

The console also includes a 'Documentation' section with links to:

- Quick Start Guide**: Basic steps for starting the application server, and packaging and deploying applications. It also provides information about the Admin Console and command-line tools.
- Administration Guide**: How to create server instances and clusters, how to configure them, and how to deploy applications to them.
- Developers Guide**: How to configure your

The browser window title is 'Sun Java(TM) System Application Server Platform Edition 8.1 Admin Console - Mozilla Firefox'. The status bar at the bottom shows 'Done' and 'Disabled'.

Selanjutnya, klik pada HTTP Listener, dan pada kolom kanan klik tombol New.

The screenshot displays the Sun Java System Application Server Admin Console interface. The browser window title is "Sun Java(TM) System Application Server Platform Edition 8.1 Admin Console - Mozilla Firefox". The address bar shows "http://localhost:4849/adingui/TopFrameset". The page header includes navigation links: HOME, VERSION, UPGRADE, REGISTRATION, LOGOUT, and HELP. The user information is "User: admin Server: localhost Domain: domain1".

The main content area is titled "Application Server > Configuration > HTTP Service > HTTP Listeners". The central panel is "Create HTTP Listener", which includes a description: "An HTTP listener is a listen socket that has an IP address, a port number, a server name, and a default virtual server." and a note: "* Indicates required field".

The "General Settings" section contains the following fields:

- Name:** SecureListener
- Listener:** Enabled
- Security:** Enabled
- Network Address:** 0.0.0.0 (with a note: "Dotted-pair or IPv6 notation; also 0.0.0.0, any, or ANY or INADDR_ANY (all IP addresses); for SSL must be 0.0.0.0 if used by more than one server")
- Listener Port:** 8091 (with a note: "Can be 1-65535; ports 1-1024 require superuser privileges")
- Default Virtual Server:** server (with a note: "Use Virtual Servers page to define a new virtual server")
- Server Name:** (with a note: "Use alias if server uses alias; append colon and port to send with URL")

The "SSL" section includes the field:

- Client Authentication:** Enabled

The left sidebar shows a tree view of the configuration hierarchy, with "HTTP Listeners" selected under "HTTP Service". The bottom status bar shows "Done" and "Disabled".

Sun Java(TM) System Application Server Platform Edition 8.1 Admin Console - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://localhost:4849/adingui/TopFrameset

Customize Links Free Hotmail Windows Media Windows Yahoo Messenger H... Object Oriented Dat...

HOME VERSION UPGRADE REGISTRATION LOGOUT HELP

User: admin Server: localhost Domain: domain1

Sun Java™ System Application Server Admin Console

Sun™ Microsystems, Inc.

- JDBC
- Persistence Managers
- JMS Resources
- JavaMail Sessions
- JNDI
- Connectors
- Configuration
 - Web Container
 - EJB Container
 - Java Message Service
 - Security
 - Transaction Service
 - HTTP Service**
 - HTTP Listeners**
 - Virtual Servers
 - server
 - _asadmin
 - ORB
 - Thread Pools
 - Admin Service
 - Connector Service

General Settings

* Name:

Listener: Enabled

Security: Enabled

* Network Address:
Dotted-pair or IPv6 notation; also 0.0.0.0, any, or ANY or INADDR_ANY (all IP addresses); for SSL must be 0.0.0.0 if used by more than one server

* Listener Port:
Can be 1-65535; ports 1-1024 require superuser privileges

* Default Virtual Server:
Use Virtual Servers page to define a new virtual server

Server Name:
Use alias if server uses alias; append colon and port to send with URL

SSL

Client Authentication: Enabled
Requires the client to authenticate itself to the server

Certificate NickName:
Takes a single value, identifies the server's keypair and certificate

SSL3 / TLS

SSL3: Enabled

Done Disabled

Pada screen diatas merupakan hasil dari klik dari New button dengan disertai contoh nilai yang sudah terisi.

Lakukan restart pada server. Konfigurasi baru kita dapat kita coba dengan mengakses alamat :

`https://serverAddress:listenerPort/index.html`

Untuk dapat menggunakan komunikasi yang aman antara client dan server, lakukan redirect pada user ke secure listener port ketika mengakses aplikasi Anda.

11.3 10 Celah keamanan pada aplikasi web

Open Web Application Security Project (OWASP) adalah project open source yang dibangun untuk menemukan penyebab dari tidak amannya sebuah software dan menemukan cara menanganinya. Ada 10 celah kewanaman aplikasi web yang ditemukan dan rekomendasi mereka tentang menanganinya sebagai sebuah standard keamanan minimal dari aplikasi web.

Berikut ini adalah 10 celah tersebut dan cara agar kita dapat mengatasi masalah tersebut.

I. Unvalidated input

Semua aplikasi web menampilkan data dari HTTP request yang dibuat oleh user dan menggunakan data tersebut untuk melakukan operasinya. Hacker dapat memanipulasi bagian-bagian pada request (query string, cookie information, header) untuk *membypass* mekanisme keamanan.

Berikut ini tiga jenis penyerangan yang berhubungan dengan masalah ini:

- Cross site scripting
- Buffer overflows
- Injection flaws

Ada beberapa hal yang dapat dicatat ketika menangani validasi pada aplikasi kita. Pertama, adalah tidak baik pada aplikasi web untuk percaya pada client side scripting. Script tersebut biasanya menghentikan form submission apabila terdapat sebuah input yang salah. Akan tetapi, script tersebut tidak dapat mencegah hacker untuk membuat HTTP requestnya sendiri yang terbebas dari form. Menggunakan client side validation masih bisa membuat aplikasi web yang mudah diserang.

Kedua, beberapa aplikasi menggunakan pendekatan "negative" (negative approach) pada validasinya : Aplikasi mencoba mendeteksi jika terdapat elemen yang berbahaya pada request parameter. Masalah dari jenis pendekatan ini adalah hanya bisa melindungi dari beberapa serangan yaitu : hanya serangan yang dikenali oleh validation code yang dicegah. Ada banyak cara dimana hacker dapat *membypass* keamanan dari unvalidated input; Masih ada kemungkinan dimana cara yang baru tidak dikenali oleh aplikasi dapat *membypass* validasi dan melakukan perusakan. Adalah cara yang lebih baik untuk menggunakan pendekatan "positive" (positive approach) yaitu : membatasi sebuah format atau pola untuk nilai yang diijinkan dan memastikan input tersebut sesuai dengan format tersebut.

II. Broken Access Control

Banyak aplikasi yang mengkategorikan user-usernya ke dalam role yang berbeda dan level yang berbeda untuk berinteraksi dengan content yang dibedakan dari kategori-kategori tersebut. Salah satu contohnya, banyak aplikasi yang terdapat user role dan admin role : hanya admin role yang diijinkan untuk mengakses halaman khusus atau melakukan action administration.

Masalahnya adalah beberapa aplikasi tidak efektif untuk memaksa agar otorisasi ini bekerja. Contohnya, beberapa program hanya menggunakan sebuah checkpoint dimana hanya user yang terpilih yang dapat mengakses : untuk proses lebih lanjut, user harus membuktikan dirinya terotorisasi dengan menggunakan user name dan password. Akan tetapi, Mereka tidak menjalankan pengecekan dari checkpoint sebelumnya : dimana apabila user berhasil melewati halaman login, mereka dapat bebas menjalankan operasi.

Masalah lain yang berhubungan dengan access control adalah:

- Insecure Ids – Beberapa site menggunakan id atau kunci yang menunjuk kepada user atau fungsi. ID dapat juga ditebak, dan jika hacker dapat mudah menebak ID dari user yang terotorisasi, maka site akan mudah diserang.
- File permissions – Kebanyakan web dan aplikasi server percaya kepada external file yang menyimpan daftar dari user yang terotorisasi dan resources mana saja yang dapat dan/atau tidak dapat diakses. Apabila file ini dapat dibaca dari luar, maka hacker dapat memodifikasi dengan mudah untuk menambahkan dirinya pada daftar user yang diijinkan.

Langkah-langkah apa saja yang dapat dilakukan untuk mengatasinya? Pada contoh-contoh tadi, kita dapat mengembangkan filter atau komponen yang dapat dijalankan pada sensitive resources. Filter atau komponen tadi dapat menjamin hanya user yang terotorisasi dapat mengakses. Untuk melindungi dari insecure Ids, kita harus mengembangkan aplikasi kita agar tidak percaya pada kerahasiaan dari Ids yang dapat memberi access control. Pada masalah file permission, file-file tersebut harus berada pada lokasi yang tidak dapat diakses oleh web browser dan hanya role tertentu saja yang dapat mengaksesnya.

III. Broken Authentication dan Session Management

Authentication dan session management menunjuk kepada semua aspek dari pengaturan user autentikasi dan management of active session. Berikut ini beberapa hal yang perlu diperhatikan :

- Password strength – Aplikasi kita harus memberikan level minimal dari keamanan sebuah password, dimana dapat dilihat dengan cara melihat panjang dari password dan kompleksitasnya. Contohnya sebuah aplikasi dimana terdapat user baru yang akan mendaftar : aplikasi tidak mengijinkan password dengan panjang 3-4 karakter atau kata-kata simpel yang dapat mudah ditebak oleh hacker.
- Password use – Aplikasi kita harus membatasi user yang mengakses aplikasi melakukan login kembali ke sistem pada tenggang waktu tertentu. Dengan cara ini aplikasi dapat dilindungi dari serangan brute force dimana hacker bisa menyerang berulang kali untuk berhasil login ke sistem. Selain itu, log in yang gagal sebaiknya dicatat sebagai informasi kepada administrator untuk mengindikasikan kemungkinan serangan yang terjadi.
- Password storage – password tidak boleh disimpan di dalam aplikasi. Password harus disimpan dalam format terenkripsi dan disimpan di file lain seperti file database atau file password. Hal ini dapat memastikan bahwa informasi yang sensitif seperti password tidak disebarkan ke dalam aplikasi.

Issue lain yang berhubungan : password tidak boleh dalam bentuk hardcoded di dalam source code.

- Session ID Protection – server biasanya menggunakan session Id untuk mengidentifikasi user yang masuk ke dalam session. Akan tetapi jika session ID ini dapat dilihat oleh seseorang pada jaringan yang sama, orang tersebut dapat menjadi seorang client.

Salah satu cara yang dapat digunakan untuk mencegah terlihatnya session ID oleh seseorang pada suatu jaringan yang sama adalah menghubungkan komunikasi antara sever dan client pada sebuah SSL-protected channel.

IV. Cross site scripting

Cross site scripting terjadi ketika seseorang membuat aplikasi web melalui script ke user lain. Hal ini dilakukan oleh penyerang dengan menambahkan content (seperti JavaScript, ActiveX, Flash) pada request yang dapat membuat HTML output yang dapat dilihat oleh user lain. Apabila ada user lain yang mengakses content tersebut, browser tidak mengetahui bahwa halaman tersebut tidak dapat dipercaya.

Cara yang bisa digunakan untuk mencegah serangan cross site scripting adalah dengan melakukan validasi data masuk dari user request (seperti header, cookie, user parameter, ...). Cara negative approach tidak digunakan : mencoba untuk memfilter active content merupakan cara yang tidak efektif.

V. Buffer overflows

Penyerang dapat menggunakan buffer overflows untuk merusak aplikasi web. Hal ini dilakukan karena penyerang mengirimkan request yang membuat server menjalankan kode-kode yang dikirimkan oleh penyerang.

Kelemahan buffer overflow biasanya sulit dideteksi dan sulit dilakukan oleh hacker. Akan tetapi penyerang masih bisa mencari kelemahan ini dan melakukan buffer overflow pada sebagian aplikasi web.

Terima kasih atas desain dari Java environment, dimana aplikasi yang berjalan pada J2EE server aman dari jenis serangan ini.

Untuk memastikan keamanan, cara yang paling baik adalah melakukan pengawasan apabila terdapat patch atau bug report dari produk server yang digunakan.

VI. Injection flaws

Salah satu kelemahan yang populer adalah injection flaw, dimana hacker dapat mengirimkan atau meng*inject* request ke operating system atau ke external sumber seperti database.

Salah satu bentuknya adalah SQL injection. Berikut ini salah satu contoh dari SQL injection :

```
http://someServer/someApp/someAction?searchString=jedi
```

URL diatas akan memproses pencarian dengan kata kunci 'jedi'. Implementasi dimana tidak ada validasi input adalah seperti SQL code berikut ini :

```
select * from someTable where someField='value'
```

dimana *value* adalah nilai dari parameter searchString yang ada pada HTTP request.

Bagaimana jika, hacker melakukan input dari URL seperti ini :

```
http://someServer/someApp/someAction?searchString=jedi'%20AND%20true;%20DROP%20DATABASE;'
```

SQL query yang terbentuk adalah seperti ini :

```
select * from someTable where someField='jedi' AND true; DROP DATABASE;"
```

Statement awal pasti akan diterima dimana terdapat klausa AND TRUE. Dan statement selanjutnya yaitu DROP DATABASE juga akan dieksekusi yang akan memberikan kerusakan pada aplikasi.

Serangan ini bisa mungkin terjadi karena input yang tidak divalidasi. Ada dua cara yang bisa dilakukan untuk mencegah serangan ini yaitu:

- Daripada menggunakan statement SELECT, INSERT, UPDATE dan DELETE statement, bisa dibuat fungsi yang melakukan hal serupa. Dengan menggunakan fungsi diharapkan ada pengamanan terhadap parameter. Selain itu dengan adanya fungsi, parameter yang masuk harus sama dengan tipe data dari parameter yang dideklarasikan.
- Hak akses dalam aplikasi juga harus dibatasi. Contohnya, jika aplikasi hanya bertujuan untuk melihat data, tidak perlu diberikan hak akses untuk melakukan INSERT, UPDATE atau DELETE. Jangan menggunakan account admin pada aplikasi web untuk mengakses database. Hal ini juga dapat meminimalkan serangan dari hacker.

VIII. Insecure storage

Aplikasi web biasanya perlu menyimpan informasi yang sensitif seperti password, informasi kartu kredit, dan yang lain. Dikarenakan item-item tersebut bersifat sensitif item-item tersebut perlu dienkripsi untuk menghindari pengaksesan secara langsung. Akan tetapi beberapa metode enkripsi masih lemah dan masih bisa diserang.

Berikut ini beberapa kesalahan yang sering terjadi :

- Kesalahan untuk mengenkripsi data penting
- Tidak amannya kunci, certificate, dan password
- Kurang amannya lokasi penyimpanan data
- Kurangnya penghitungan dari randomisasi
- Kesalahan pemilihan algoritma
- Mencoba untuk menciptakan algoritma enkripsi yang baru

Berdasarkan skenario berikut ini : Terdapat sebuah aplikasi, dimana terdapat password pada user object. Akan tetapi, aplikasi menyimpan user object ke dalam session setelah user login. Permasalahan yang akan muncul pada skenario ini adalah password dapat dilihat oleh seseorang yang dapat melihat session dari user tersebut.

Salah satu cara yang dilakukan untuk menghindari kesalahan penyimpanan informasi yang sensitif adalah : tidak membuat password sebagai atribut dari kelas yang mewakili informasi user; Daripada mengenkripsi nomor kartu kredit dari user, akan lebih baik untuk menyimpannya setiap kali dibutuhkan.

Selain itu, menggunakan algoritma enkripsi yang sudah ada akan lebih baik daripada membuat algoritma sendiri. Anda cukup memastikan algoritma yang akan digunakan telah diakui oleh public dan benar-benar dapat diandalkan.

IX. Denial of Service

Denial of Service merupakan serangan yang dibuat oleh hacker yang mengirimkan request dalam jumlah yang sangat besar dan dalam waktu yang bersamaan. Dikarenakan request-request tersebut, server menjadi kelebihan beban dan tidak bisa melayani user lainnya.

Serangan DoS mampu menghabiskan bandwidth yang ada pada server. Selain itu dapat juga menghabiskan memory, koneksi database, dan sumber yang lain.

Pada umumnya sangat sulit untuk melindungi aplikasi dari serangan ini. Akan tetapi masih ada cara yang dapat dilakukan seperti membatasi resource yang dapat diakses user dalam jumlah yang minimal. Merupakan ide / cara yang bagus untuk membuat load quota yang membatasi jumlah load data yang akan diakses user dari sistem.

Salah satu contoh adalah pada implementasi bulletin board : adanya pembatasan user pada saat melakukan search, dimana operasi ini hanya dapat dilakukan setiap 20 detik. Dengan cara ini dapat dipastikan bahwa user tidak bisa menghabiskan koneksi dari database.

Solusi yang lain adalah mendesain aplikasi web dimana user yang belum terotorisasi hanya memiliki akses yang sedikit atau tidak memiliki akses ke content web yang berhubungan dengan database.

X. Insecure Configuration Management

Biasanya kelompok (group) yang mengembangkan aplikasi berbeda dengan kelompok yang mengatur hosting dari aplikasi. Hal ini bisa menjadi berbahaya, dikarenakan keamanan yang diandalkan hanya dari segi aplikasi : sedangkan dari segi server juga memiliki aspek keamanan yang perlu diperhatikan. Adanya kesalahan dari konfigurasi server dapat melewati aspek keamanan dari segi aplikasi.

Berikut ini adalah kesalahan konfigurasi server yang bisa menimbulkan masalah :

- Celah keamanan yang belum *patch* dari software yang ada pada server – administrator tidak melakukan *patch* software yang ada pada server.
 - Celah keamanan server dimana bisa menampilkan list dari direktori atau juga serangan berupa directory traversal.
 - File-file backup atau file contoh (sample file), file-file script, file konfigurasi yang tertinggal / tidak perlu.
 - Hak akses direktori atau file yang salah.
 - Adanya service yang seperti remote administration dan content management yang masih aktif.
 - Penggunaan default account dan default password.
 - Fungsi administrative atau fungsi debug yang bisa diakses.
 - Adanya pesan error yang informatif dari segi teknis.
 - Kesalahan konfigurasi SSL certificate dan setting enkripsi.
 - Penggunaan self-signet certificates untuk melakukan autentikasi.
 - Penggunaan default certificate.
 - Kesalahan autentikasi dengan sistem eksternal.
-