# Analisis Keamanan Web Service Zimbra Collaboration Suite

**Rangga Wisnu Adi Permana - 13504036** [1)]

1) Program Studi Teknik Informatika ITB, Bandung 40132, email: if14036@students.if.itb.ac.id

***Abstract*** *– Dengan maju kemajuan teknologi, email menjadi salah satu media komunikasi yang penting. Penyedia layanan email atau yang sering disebut dengan mail server menjadi suatu aplikasi penting pada sebuah perusahaan atau instansi lainnya. Zimbra Collaboration Suite adalah salah satu contoh dari mail server tersebut. Zimbra collaboration Suite merupakan mail server yang dipadukan dengan beberapa aplikasi sehingga disebut dengan collaboration. Zimbra Collaboration Suite cukup populer karena dengan beragamnya fitur yang disediakan dan user interface yang menarik. Sebuah instansi yang menggunakan mail server pada sistemnya, pada umumnya tidak hanya menggunakan sebuah aplikasi yang berupa mail server saja. Fitur yang disediakan oleh sebuah mail server atau collaboration pada umumnya tidak sesuai atau kurang memenuhi kebutuhan dari instansi tersebut. Pada Zimbra Collaboration Suite disediakan suatu API yang dapat berfungsi untuk menambahkan fitur pada mail server tersebut secara manual dengan membangun aplikasi diluar mail server tersebut yang dapat berkomunikasi dengan mail server tersebut, atau dapat berfungsi untuk memadukan aplikasi yang berjalan pada suatu instansi dengan mail server tersebut. API yang disediakan oleh Zimbra Collaboration Suite tersebut berupa Web Service.*

***Kata Kunci****: mail server, web service, SOAP, SHA-1, XML.*

## 1. PENDAHULUAN

Komunikasi merupakan kebutuhan manusia, salah satu cara melakukan komunikasi ini adalah dengan melakukan pengiriman pesan. Dilatarbelakangi oleh kebutuhan manusia tersebut, teknologi komunikasi dewasa ini maju dengan pesat. Dengan kemajuan teknologi tersebut, manusia dapat melakukan pengiriman pesan dengan mudah di mana saja dan kapan saja dengan menggunakan berbagai media.

Dengan kemajuan teknologi, sekarang ini manusia dapat melakukan pengiriman dan penerimaan surat melalu media elektronik yang disebut dengan *email*. Dewasa ini, *email* menjadi salah satu media komunikasi yang penting dan merupakan salah satu media komunikasi dengan pengguna terbanyak. *Email* menyediakan kemudahan dalam melakukan komunikasi, selain dapat melakukan pengiriman pesan dengan cepat dan jangkauannya yang sangat luas, yaitu semua tempat yang memiliki koneksi internet, biaya pengiriman pesan dengan menggunakan *email* sangatlah murah, hanya membutuhkan koneksi internet saja.

Untuk melakukan komunikasi dengan *email* dibutuhkan suatu perangkat lunak yang disebut dengan *mail server*. Sekarang ini *mail server* bukan lagi perangkat lunak yang sulit untuk dicari, bahkan sekarang ini banyak sekali berbagai nama *mail server* yang beredar dipasaran, salah satunya adalah Zimbra Collaboration Suite.

Sekarang ini Zimbra Collaboration Suite cukup populer karena memiliki *user interface* yang menarik. Namun bukan hanya itu saja yang menjadi keunggulan mail server ini, mail server ini juga menyediakan fitur-fitur menarik lainnya seperti sistem kalender dan *rss feeder*. Dengan semakin arsitektur perangkat lunak dengan berbasis SOA (Service Oriented Atchitecture), Zimbra Collaboration Suite juga menyediakan suatu Web Service agar dapat berkomunikasi dengan perangkat lunak lain.

Yang akan dibahas pada makalah ini adalah melakukan eksplorasi dan pembelajaran dari metode Web Service yang disediakan oleh Zimbra Collaboration Suite. Selain itu juga akan dilakukan analisis dari Web Service yang telah disediakan oleh Zimbra Collaboration suite. Versi Zimbra Collaboration Suite yang akan dibahas pada makalah ini adalah versi 4.5.4 dan yang merupakan versi *open source* yang dimana beberapa fitur dibatasi.

Pada makalah ini juga terdapat hasil uji coba penggunaan Web Service Zimbra Collaboration Suite Dengan adanya makalah ini, diharapkan pembaca dapat mendapatkan informasi berharaga seputar penerapan Web Service dan juga keamanan komunikasi antar perangkat lunak.

## 2. LANDASAN TEORI

### 2.1. Web Service dan SOAP

*Web Service* adalah desain suatu sistem yang memungkinkan suatu mesin untuk berinteraksi dengan mesin lain dalam suatu jaringan.

*Web Service* biasanya digunakan sebagai API (*Applicaton Programming Interface*) pada suatu aplikasi yang bekerja di jaringan internet. Bahkan *Web Service* sering disebut sebagai Web API.

SOAP (*Simple Object Access Protocol*) adalah salah satu jenis protokol yang digunakan untuk melakukan pertukaran informasi yang berbasis XML.

XML (*Exstensible Markup Language*) adalah suatu format teks yang fleksibel, yang dimana pengguna dapat mendefinisikan sendiri elemen-elemen yang terdapat pada teks XML tersebut. Karena XML berbasis teks, maka memungkinkan untuk dijadikan sebagai media komunikasi atau *sharing* data pada beberapa sistem yang berbeda. Salah satu penggunaan XML yang sering dipakai dewasa ini adalah pada *blog*, pada umumnya situs penyedia layanan *blog* menyimpan tulisan *blog* dalam bentuk XML. Penggunaan lainnya adalah pada situs berita, situs berita dewasa ini pada umumnya menyediakan layanan untuk menyimpan berita dalam format RSS yang merupakan contoh penggunaan XML.

Pada SOAP pesan dikirimkan dalam format XML dan response dari pesan itu pun dikirimkan dengan format XML juga. Hal ini memungkinkan pesan SOAP mudah untuk dikenali karena XML merupakan *file* berisi plain teks. SOAP ini pada umumnya memakai protokol HTTP dan HTTPS dalam melakukan pengiriman pesan. SOAP terdiri dari 3 bagian yaitu *envelope* yang digunakan untuk mendefinisikan *framework* yang digunakan untuk mendefinisikan pesan yang dikirim dan cara pemrosesan pesan tersebut, aturan *encoding* dan konvensi yang digunakan untuk menggambarkan pemanggilan *procedure* dan *response,* untuk lebih jelas mengenai struktur XML dari SOAP dapat dilihat pada **Gambar 1**. SOAP memungkinkan untuk digunakan bersama protokol lainnya.

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelo
pe/">
        <SOAP-ENV:Header>

        </SOAP-ENV:Header>

        <SOAP-ENV:Body>

        </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

**Gambar 1 Struktur XML dari SOAP**

## 2.2. Ajax

Ajax (Asynchronous JavaScript and XML) adalah salah satu teknik dalam pembangunan suatu aplikasi berbasis internet.

Ajax memungkinkan untuk melakukan komunikasi antara *client* dan server yang memakai bahasa yang berbeda karena ajax menggunakan format XML yang *plain* teks dalam pengiriman pesan. Dengan pemakaian ajax, *refresh* halaman dapat dilakukan secara sebagian dan secara otomatis jika terdapat *trigger* tertentu. Ajax juga memungkinkan untuk mengubah isi halaman tanpa perlu mengganti halaman, hal ini dikarenan dengan ajax memungkinkan untuk melakukan *server side scripting.*

Teknologi ajax dan XML ini merupakan dasar dari terbentuknya teknologi Web 2.0, yang sekarang ini sedang populer, contoh situs dengan teknologi Web 2.0 ini adalah situs penyedia layanan *blogging* yang dimana user dapat menentukan isi *content* dari suatu halaman web.

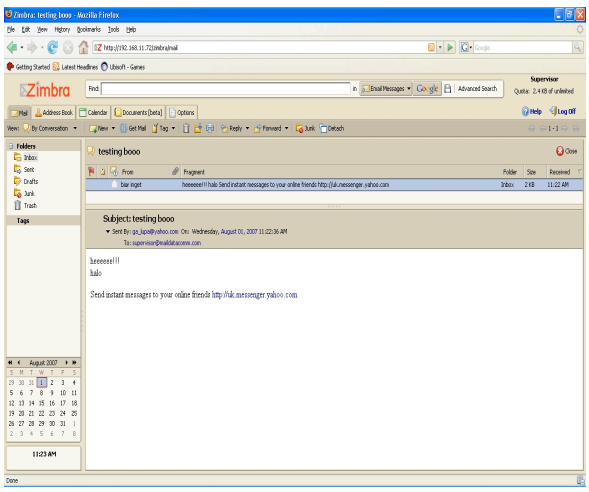
# 3. Zimbra Collaboration Suite

## 3.1. Pengertian

Zimbra Collaboration Suite adalah suatu *mail server* yang dibangun dengan menggunakan bahasa pemrograman Java dan menggunakan teknologi pendukung ajax.

Zimbra Collaboration Suite dibangun oleh sebuah perusahaan Zimbra Inc, yang kemudian perusahaan tersebut dibeli oleh Yahoo! Secara garis besar terdapat dua buah versi, versi *open source* dan versi komersial, dimana versi *open source* memiliki beberapa keterbatasan fitur.

Zimbra Collaboration Suite menggunakan teknologi ajax, sehingga Zimbra Collaboration Suite dapat memiliki tampilan yang menarik, seperti melakukan *drag and drop* email dan *pop-up screen shot* sebuah URL pada sebuah email yang menjadi mungkin karena teknologi ajax. Dalam **Gambar *2*** dapat dilihat contoh tampilan dari Zimbra Collaboration Suite.




**Gambar 2 Contoh Tampilan ZCS**

Untuk penyimpanan basis data, mail server ini menggunaka MySQL. Mail server ini dapat berjalan pada sistem operasi yang berbasis UNIX, seperti Red Hat dan Suse.

Zimbra Collaboration Suite memiliki banyak fitur, diantaranya adalah pengiriman email, sistem kalender dan *sharing* dokumen. Untuk memungkinkan aplikasi lain dapat mengakses fitur-fitur tersebut, Zimbra Collaboration Suite menydiakan suatu Web Service sebagai API dengan aplikasi lain.

### 3.2. Zimbra Web Service

Komunikasi dengan Zimbra Collaboration Suite dapat dilakukan melalui *web service* yang telah disediakan oleh Zimbra. Zimbra Collaboration Suite menyediakan dua jenis *web service* yang dapat digunakan, yaitu dengan REST (*Representational State Transfer)* dan SOAP (*Simple Object Access Protocol*).

REST dapat digunakan untuk mengambil dokumen dari Zimbra Collaboration Suite,contoh dokumennya dapat berupa kalender yang sudah berisi jadwal pengguna yang bersangkutan ataupun. Penggunaan REST ini adalah memakai URL diamana dokumen tersebut disimpan, jadi dapat diakses langsung melalui *web browser* dengan mengetikan alamat dokumen tersebut disimpan, contohnya adalah sebagai berikut :

**http://localhost:7070/zimbra/user1@a.com/calendar**

Contoh tersebut dapat berjalan dimana "user1" sudah memiliki izin untuk memasuki dokumen, hal tersebut dapat dilakukan dengan cara melakukan log-in secara manual terlebih dahulu. Jika belum memiliki izin maka pada URL tersebut harus ditambahkan password pada nama pengguna.

Untuk melakukan akses *web service* melalui SOAP dapat dilakukan dengan membangun suatu aplikasi yang bekerja sebagai *client web service* tersebut yang akan melakukan *request* yang berupa perintah kepada Zimbra Collaboration Suite. Setiap kali melakukan request, Zimbra Collaboration Suite akan mengirikan *response* sesuai dengan *request* yang diberikan. Untuk melakukan request yng berkaitan dengan perubahan didalam *mail server,* seperti melakukan pengiriman email atau melakukan perubahan isi dari sistem kalender yang terdapat pada mail server tersebut dibutuhkan suatu kunci otorisasi yang disebut dengan "*authToken*".

*AuthToken* bekerja seperti tiket untuk melakukan identifikasi pengguna. Dengan menggunakan *authToken* ini, pengguna tidak perlu mengirimkan password untuk melakukan komunikasi dengan Zimbra Collaboration Suite.

*AuthToken* dapat diperoleh melalui 2 cara yaitu :

1. Memakai pre auth

Preauth adalah suatu kunci otorisasi yang dapat digunakan untuk meminta *authToken* dari semua pengguna Zimbra Collaboration Suite. Preauth ini dapat di-*generate* dari komputer *server* dimana Zimbra Collaboration Suite ter-*install*. Dalam melakukan permintaan *authToken* dengan metode ini hanya diperlukan kunci preauth dan *user account* dari pengguna *mail server*.

2. Memakai Password

Dalam metode ini nama pengguna dan *password* harus dikirim ke *mail server* untuk kemudian akan diberikan atau di-*generate authToken* dari *mail server*. Waktu berlakunya *authToken* yang ada dapat diatur melalui *mail server* secara langsung dengan masuk sebagai *admin*.

Yang dilakukan oleh *mail server* Zimbra Collaboration Suite dalam memperoleh *authToken* tersebut hamper sama. Yang dilakukan adalah melakukan melakukan perhitungan dengan menggunakan SHA-1 HMAC pada nama pengguna yang diinginkan, lama berlakunya *authToken* tersebut dan dapat juga ditambahkan waktu ekspirasi dari *authToken* dengan menggunakan sebuah kunci rahasia.

Jika yang dilakukan adalah menggunakan pre auth, maka pre auth tersebutlah yang menjadi kunci rahasia tersebut. Pre auth akan bekerja sebagai kunci rahasia yang dapat digunakan untuk semua pengguna. Dan jika yang dilakukan adalah menggunakan password, maka kunci rahasia tersebut akan berhubungan dengan password pengguna yang bersangkutan.

*AuthToken* yang dihasilkan merupakan hasil perhitungan SHA-1 HMAC yang ditulis dalam bentuk HexString. Contoh *authToken* adalah berikut :

**0_a6df1b4259436946bbc826cefca1e22405b40548_69643d33363a64353364376231662d373033612d343964632d616466642d6339646531633332633063643b6578703d31333a313138343134323435373134303b**

### 3.3. Uji Coba

Uji Coba yang dilakukan difokuskan untuk melakukan testing *web service* Zimbra Collaboration Suite dengan menggunakan SOAP. Versi Zimbra Collaboration Suite yang digunakan adalah versi 4.5.4. Setelah melakukan sedikit percobaan pada versi sebelumnya, terdapat *bug* pada penggunaan pre auth, dimana validasi kunci rahasia yang berfungsi untuk semua pengguna seperti tidak dilakukan validasi atau semua kunci dapat diterima.

Uji coba yang dilakukan adalah melakukan percobaan penggunaan pre auth dan juga melakukan permintaan *authToken* dengan menggunakan password dan melakukan *request* ke *mail server* dengan menggunakan *authToken* yang sudah didapatkan. Tujuan percobaan ini adalah mempelajari langkah-langkah penggunaan *web service* Zimbra Collaboration Suite agar dapat dipelajari penanganan keamanannya dan dapat dianalisis.

Untuk melakukan uji coba, penulis membangun suatu aplikasi sederhana yang akan bekerja sebagai client dengan menggunakan bahasa pemrograman java, dengan lingkungan pembangunan sebagai berikut :

1) Perangkat keras berupa laptop
2) Sistem operasi Windows XP Service Pack 2.
3) Netbeans IDE 5.5 dengan jdk 6 dan Sun Java Sistem Application Server Platform Edition 9.0.
4) Memakai jsp dan xampp.
5) Mozilla Firefox 2 sebagai *web browser*.

Zimbra Collaboration Suite sendiri dipasang pada suatu komputer server yang berjalan dengan sistem operasi Linux RedHat 4 AS.

Java dapat men*support* pembuatan komunikasi antar aplikasi, dan salah satunya adalah melalui SOAP. Dalam Zimbra Collaboration Suite 4.5, tidak disediakan WSDL(Web Service Descriptor Language), oleh karena itu perlu diketahui format yang diterima oleh *mail server* tersebut. Dan SOAP yang dikirim tidak dibangun dengan menggunakan WSDL.

Jika ingin melakukan komunikasi melalui SOAP ini, yang pertama harus dilakukan adalah mengetahui alamat *web service* SOAP pada aplikasi yang telah menyediakan *web service*. Selanjutnya harus dipelajari mengenai format XML yang dapat diterima oleh aplikasi penyedia layanan terebut.

Untuk meng-*generate file* SOAP dan melakukan komunikasi dengan penyedia layanan tersebut, pada Java harus melakukan *import library* berikut:

```
import javax.xml.soap.*;

import javax.xml.transform.*;

import javax.xml.transform.stream.*;

import java.net.*;
```

Setelah melakukan *import file-file* tersebut, komunikasi melalui SOAP dapat dilakukan. *Library* tersebut dapat digunakan untuk membentuk suatu file XML untuk dikirim melalui SOAP atau dapat juga mengirimkan langsung file XML berupa file teks yang sudah terbentuk. Dalam hal ini, penulis menggunakan cara pertama yaitu membentuk file XML dari kode.

Aplikasi yang dibangun hanya melakukan pengiriman dan penerimaan file XML, oleh karena itu tidak menggunakan *user interface*, response SOAP diuliskan pada *command prompt*.

Untuk melakukan percobaan dengan menggunakan pre auth, dibutuhkan kunci rahasia yang akan berguna untuk menghasilkan HMAC yang dapat bekerja untuk semua pengguna, kunci tersebut dapat didapatkan dengan memintanya pada komputer server. Setelah mendapatkan kunci rahasia tersebut, maka dibutuhkan mengirimkan pre auth tersebut untuk mendapatkan *authToken.* Contoh XMLnya adalah sebgai berikut :

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
 <SOAP-ENV:Header>
  <context xmlns="urn:zimbraAccount"/>
 </SOAP-ENV:Header>
 <SOAP-ENV:Body>
   <AuthRequest xmlns="urn:zimbraAccount">
  <account>john.doe@domain.com</account>
  <preauth timestamp="1135280708088" expires="0}">b248f6cfd027edd45c5369f8490125204772f844
</preauth>
 </AuthRequest>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Jika menggunakan password dibutuhkan komunikasi dengan SOAP yang didalamnya terdapat perintah “authRequest”, contoh XML untuk melakukan permintaan tersebut adalah sebagai berikut :

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
 <SOAP-ENV:Header>
  <context xmlns="urn:zimbraAccount"/>
 </SOAP-ENV:Header>
 <SOAP-ENV:Body>
  <AuthRequest xmlns="urn:zimbraAccount" >
  <account>victor@maildatacomm.com</account>
  <password>victor2</password>
  </AuthRequest>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Setelah pesan tersebut diterima oleh server maka akan diberikan *response* yang didalamnya terdapat *authToken.* Berikut adalah contoh penggunaan *authToken* untuk melakukan pengiriman email :

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

<SOAP-ENV:Header>

<context xmlns="urn:zimbra">

<authToken>0_de9810192c3a2e17d2713daadfe5165abd4e9b06_69643d33363a65333332643833312d633033332d343038362d623961612d3461353835616433306439373b6578703d31333a313138353930313230303334333b</authToken>

</context>

</SOAP-ENV:Header>

<SOAP-ENV:Body>

<SendMsgRequest xmlns="urn:zimbraMail">

 <m>
        <e t="t" a="bravo@maildatacomm.com" p="Bravo" />
   <su>testing dolo</su>

   <mp ct="text/html">
                <content>
                        <html>
                                <head>
                                </head>
                                <body>
                                        bener ga sih
                                </body>
                        </html>
                </content>
   </mp>

 </m>

</SendMsgRequest>

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

## 3. ANALISIS

Berikut akan dibahas analisis *web service* Zimbra Collaboration Suite berdasarkan eksplorasi dan uji coba yang telah dilakukan.

### 3.1 Analisis REST

Penggunaan REST pada Zimbra Collaboration Suite dapat dilakukan dengan mudah tanpa membangun suatu file XML ataupun sebuah aplikasi, namun penggunaannya tidak cukup praktis karena membutuhkan pengiriman password dan penggunaan metode GET menjadikan penggunaan REST ini tidak terlalu aman.

### 3.2 Analisis SOAP

Untuk melakukan komunikasi melalui SOAP dibutuhkan 2 langkah penting berikut :

1. Melakukan permintaan *authToken*
2. Menggunakan *authToken* untuk melakukan *request.*

Jika permintaan *authToken* yang dilakukan dengan menggunakan pre auth, kunci rahasia didapatkan dengan meminta langsung dari komputer server, sehingga cukup aman, namun hal ini juga dapat menjadi kelemahan besar apabila pemilik kunci rahasia tersebut merupakan orang yang tidak bertanggung jawab, dimana orang tersebut dapat mengakses semua pengguna yang memiliki *account*. Oleh karena pemilik pre auth haruslah pihak yang benar-benar berhak memiliknya.

Jika permintaan *authToken* dilakukan dengan penggunaan kunci rahasia yang dibangun dari password, maka hanya pengguna yang bersangkutan yang dapat mengakses *account*-nya. Namun metode ini memiliki kelemahan, dimana pada metode ini pengguna harus mengirimkan passwordnya terlebih dahulu sehingga apabila jalur pengirimannya disadap, password akan mudah diketahui, ditambah file yang dikirimkannya merupakan file XML yang merupakan file teks yang mudah dibaca. Penangan masalah ini dapat diatasi dengan menggunakan jalur yang berbeda dan lebih aman ketika melakukan permintaan *authToken* dari ketika melakukan *request* lainnya, seperti melakukan penggunaan SSL atau dilakukan pada jaringan internal yang lebih aman. Selain itu dapat juga dengan menambahkan program pada *mail server* untuk melakukan dekripsi, dimana sebelum pesan XML dari *client* dikirim pesan tersebut dienkripsi terlebih dahulu dan didekkripsi oleh *mail server* ketika menerima file XML tersebut.

*AuthToken* sendiri cukup aman karena dikomputasi menggunakan SHA-1 HMAC. Jadi walaupun terlihat oleh pihak yang tidak bertanggung jawab, password akan sulit dipecahkan. Selain itu terdapat jangka waktu valid dari *authToken* tersebut sehingga akan membuat makin sulit untuk dipecahkan.

Penggunaan metode *authToken* ini dapat memenuhi 3 dari empat aspek kriptografi yaitu kerahasiaan, dimana password diubah agar tidak diketahui, kemudian otentikasi, karena merupakan pengganti password dan nirpenyangkalan, karena jika password berubah maka *authToken* pun akan berubah. Untuk aspek keempat yaitu integritas data tidak dipenuhi karena dengan *authToken* yang sama dapat melakukan *request* yang berbeda-beda.

## 4. KESIMPULAN

Dari keseluruhan isi makalah ini, dapat diambil kesimpulan sebagai berikut:

1. *Web service* dapat digunakan sebagai API suatu aplikasi dengan aplikasi lainnya.

2. XML dapat digunakan untuk memudahkan dalam melakukan file sharing antar aplikasi melalui internet.

3. Metode penanganan *web service* dengan menggunakan REST yang dilakukan Zimbra Collaboration Suite kurang aman karena menggunakan metode GET yang dimana pengiriman data dilakukan melalui URL.

4. Metode penanganan *web service* dengan menggunakan SOAP yang dilakukan Zimbra Collaboration Suite dilakukan dengan cukup baik, semua bergantung dengan pemakaian.

5. Sebaiknya pengiriman password untuk semua keperluan tidak dilakukan dengan melakukan pengiriman berupa plainteks dan dilakukan melalui jaluran yang aman.

6. Metode pemakaian *authToken* pada komunikasi SOAP memenuhi 2 dari 4 aspek kriptografi; yaitu otentikasi, nirpenyangkalan dan kerahasiaan.

## DAFTAR PUSTAKA


[1]Munir, Rinaldi, *Diktat Kuliah IF5054 Kriptografi*, Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung, 2006

[2]Tanenbaum, Andrew S, *Computer Network Fourth Edition*, Prentice Hall PTR, New Jersey, USA 2003

[3]http://java.sun.com/ diakses pada Desember 2007

[4]http://tools.ietf.org/html/rfc2104 diakses pada Januari 2008

[5]http://www.w3.org/XML diakses pada Januari 2008

[4]http://www.zimbra.com diakses pada Desember 2007

[5]http://wiki.zimbra.com diakses pada Desember 2007