UNIVERSITY OF TARTU
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
Institute of Computer Science
Software Engineering Curriculum

Ejiroghene John Okugbeni

Security Implementation of Mission Control System for ESTCube-1 Satellite

Master's Thesis (30 ECT)

Supervisor(s): M.Sc. Urmas Kvell

Tartu 2014

Security Implementation of Mission Control System for ESTCube-1 Satellite

Abstract:

ESTCube-1 is Estonia's first satellite project built by university students. ESTCube-1 Mission Control System (MCS) software is also developed as part of this educational project. Mission Control System is a modular system, comprised of various components in multiple servers, of which most of them are running on default or basic security configuration settings and in some cases, data is not protected well enough in the present state. Some of the components communicate over unsecured network thereby making its data vulnerable. As this thesis title "Security Implementation of Mission Control System for ESTCube-1 Satellite" implies, there is need for a systematic approach about the entire data security of the mission and my aim is to improve the security of ESTCube-1 Mission Control System.

The following steps are taken in the thesis: establish a good understanding ESTCube-1 MCS architecture, understand the possibilities of security configurations of all used technologies, analyse the effect of a possible selection of security methods, implement the chosen solutions in a sandbox environment, test and verify the operating of the complete MCS with the implemented solution.

The results show security implementations done on the various components allow the connection between components are secure and data in motion are encrypted. Access to the data at rest are restricted, some are encrypted and only privileged users can gain access. Mission Control System accessibility over the Internet is more secure and access to the hardware tightened.

In conclusion, the Mission Control System can certainly be accessed via the Internet securely as long as the user has valid certificates. Other access means are through other means like VPN and SSH Tunnelling. The original system configuration providedESTCube-1 MCS with just adequate security that would be befitting for a production environment, with the security solution found in current thesis, the system could be elevated for enterprise-level usage.

Keywords: ESTCube-1, Mission Control System, CubeSat, Hummingbird, Atlassian Crowd, Mongodb, Oracle 11g, Apache ActiveMQ, Jetty Web Server

Satelliidi ESTCube-1 missioonijuhtimissüsteemi turvalisuse parendamine

Lühikokkuvõte

ESTCube-1 on Eesti esimene satelliit ja ühtlasi onta ehitatud tervenisti üliõpilaste poolt. ESTCube-1 paljudestallsüsteemidest on üks osa missioonijuhtimissüsteemist (ingl. k.*Mission Control System*- MCS). Missioonijuhtimise tarkvara on modulaarne, moodulid võivad asuda erinevates serverites. Praeguses seadistusestöötab enamik moodulitest vaikimisi konfiguratsiooniseadetes ja mõnel juhul ei ole andmed piisavalt kaitstud – näiteks suhtlevad osad komponendidilma turvalise võrguühenduseta. Käesoleva töö eesmärk on süstemaatiliselt läheneda missioonijuhtimise süsteemi kui terviku turvalisusele ja leida lahendus senisest paremini turvatud süsteemi seadistamiseks.

Töö koosneb järgnevatest sammudest:kirjeldada ESTCube-1 missioonijuhtimissüsteemi arhitektuuri, analüüsida kõikide süsteemi moodulite turvalahenduste võimalusi, rakendada leitud terviklahendus missioonijuhtimissüsteemi turvalahendustetestkeskkonnas, katsetadaja kontrollida süsteemi tööd uues seadistuses.

Töös valitud lahendus võimaldab turvalisiühendusi erinevate moodulite vahel ja krüpteerib salvestatud andmed. Andmetele juurdepääsu saab piirata ka kasutajapõhiselt.

Kokkuvõttes võib missioonijuhtimissüsteemi tarkvara panna tööle avatud ligipääsuga üle Interneti. Seni kasutatud lahendus tugines VPN ja SSH tunnelitele, mis on küll sobiliksüsteemi arenduseks, aga käesolev lahendus võimaldab süsteemile turvalise ligipääsu satelliidi opereerimise igapäevatöös.

Võtmesõnad: ESTCube-1, Mission Control System, CubeSat, Hummingbird, Atlassian Crowd, Mongodb, Oracle 11g, Apache ActiveMQ, Jetty Web Server

Table of Contents

Abbreviations and Acronyms

AMQ	-Active Messaging Queue
BSON	-Binary-JSON
DOS	-Denial of Service
ESTCube-1	-Electric Sail Test Cube
EXTROC	-External Procedure
HTTP	-Hypertext Transfer Protocol
HTTPS	-Hypertext Transfer Protocol Secure
JAAS API	-Java Authentication Authorization Service Application Program Interface
JSON	-JavaScript Object Notation
LDAP	-Lightweight Directory Access Protocol
LUKS	-Linux Unified Key Setup
MCS	-Mission Control System
PAM	-Pluggable Authentication Module
SDC	-Space Data Chain
SDR	-Software-Defined Radio
SPNEGO	-Simple and Protected GSSAPI Negotiation
SSL	-Secure Socket Layer
SSO	-Secure Single Sign-on
TC	-Telecommand
TCP	-Transmission Control Protocol
TCPS	-Transmission Control Protocol Secure
TM	-Telemetry
TNC	-Terminal Node Controller
UDP	-User Datagram Protocol
USB	-Universal Serial Bus
VPN	-Virtual Private Network
WAR	-Web Application Archive
XML	-Extensible Markup Language

Introduction

ESTCube-1 is Estonia's first satellite, which is a CubeSat built entirely by students. ESTCube-1 was launched in May 2013. Since the satellite was built by students for educational purpose, it provides an avenue for Space Engineering students (and other students as well) to get good education on Space Technology, also paves the way for other Space projects in future.

There are not a lot of information on security on Mission Control Systems for these kind of satellites. Nonetheless, caution is taken when dealing security of data in transit or who has privilege to access some resources[1]. Security solutions of Mission Control System are either powered by third party software, they are not robust enough or are being operated within an internal network and not exposed to the Internet, this would be the reason why there is lack of information in this area.

ESTCube-1 project is comprised of many sub-systems and components, some of which are located in different physical locations. This project contains two sub-systems (Ground Station and Mission Control System) that have physically components located in different places and they are connected to each other via some form of communication protocols. Mission Control System (MCS) is a subsystem in ESTCube-1 that handles the Mission Control of the satellite ESTCube-1. MCS is composed of different ready-made components and some own-developed ones as well; these makeup the entire system. There exists some instructions for setting up some parts of the system, but they have not been analysed as a whole to the level of configuration for an enterprise system whereby all the components are connected and working together. Hummingbird[1] is an open-source platform for building, monitory and controlling both satellites and ground stations. Hummingbird provides framework on which Mission Control System

[1]http://www.hbird.de/

(MCS) is built, it is specifically designed for ESTCube-1 mission with Technical support from CGI Groups.

This research topic "Security Implementation of Mission Control System for ESTCube-1 Satellite" is very important to the ESTCube-1 project because of its potentiality to change the security status of the Mission Control System of now and in future. Like every other projects, be it hardware or software, there are security concerns that must be addressed in order to get a safe, secure, confidential system; not to forget its integrity. Most of MCS components are using default security setting or at most basic security settings for its operation. The main problem MCS is facing is the lack of adequate security solutions for its components that would ensure the overall security of system when tackled from the Tartu University's network and from the Internet.

The solutions to these problems can be successfully achieved in this thesis by making sure these following steps are accomplished:

- ✓ Understand the whole system's architecture
- ✓ Check for available security solution possibilities for these technologies
- ✓ Analyse the different security solution options available
- ✓ Implementation of the chosen security solutions in a MCS test environment
- ✓ Test and verify the solution in the MCS test environment

ESTCube-1 project involves students from various fields of study working on different parts of the project. The novelty of this project for the field of Computer Sciences in general is the integration of Software Development Technology with Space technology, which delivers a robust system. Since some of the activities are software related, the field of Computer Sciences (Software Engineering) would be of great use relating to software security issues in this project, also the fact that this system as a whole could make up for an Enterprise-Level system.

1 ESTCube-1 Architecture

1.1 Components diagram

The architecture for ESTCube-1's Mission Control System comprises of many components from different servers in different locations (logical, physical and geographic). Figure 1.1 below shows all the components' interconnectivity, sub-systems and the overall systems. The figure reveals sub-systems namely: Ground Station hardware, Ground Station Software, Back-End, Front-End, Data Storage, etc. and their components.

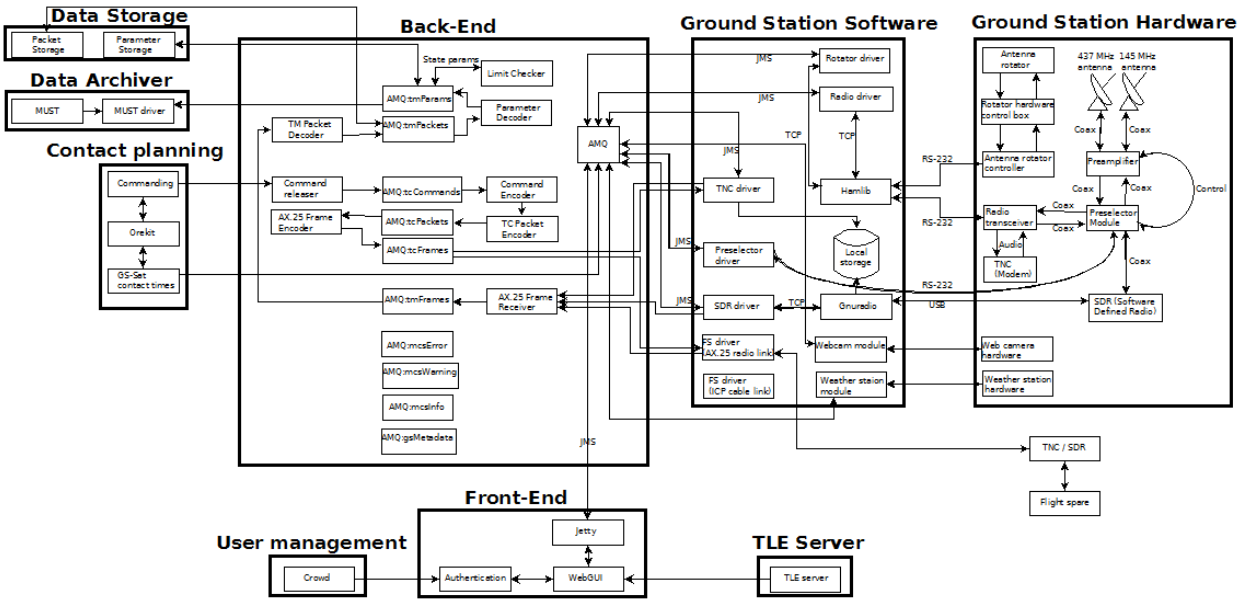


Figure 1.1.Mission Control Software architecture for ESTCube-1 showing the communication between various subsystems and components, the subsystems includes Ground Station hardware, Ground Station Software, Back-End, Front-End, Data Storage, Data Archiver and so on.

1.2 Components in University network

Currently, MCS components are located in the University of Tartu's internal network. The figure below depicts the current structure for the components under the umbrella of Tartu University's

secured network. Figure 1.2 depicts the whole system and components inside Tartu University internal network.

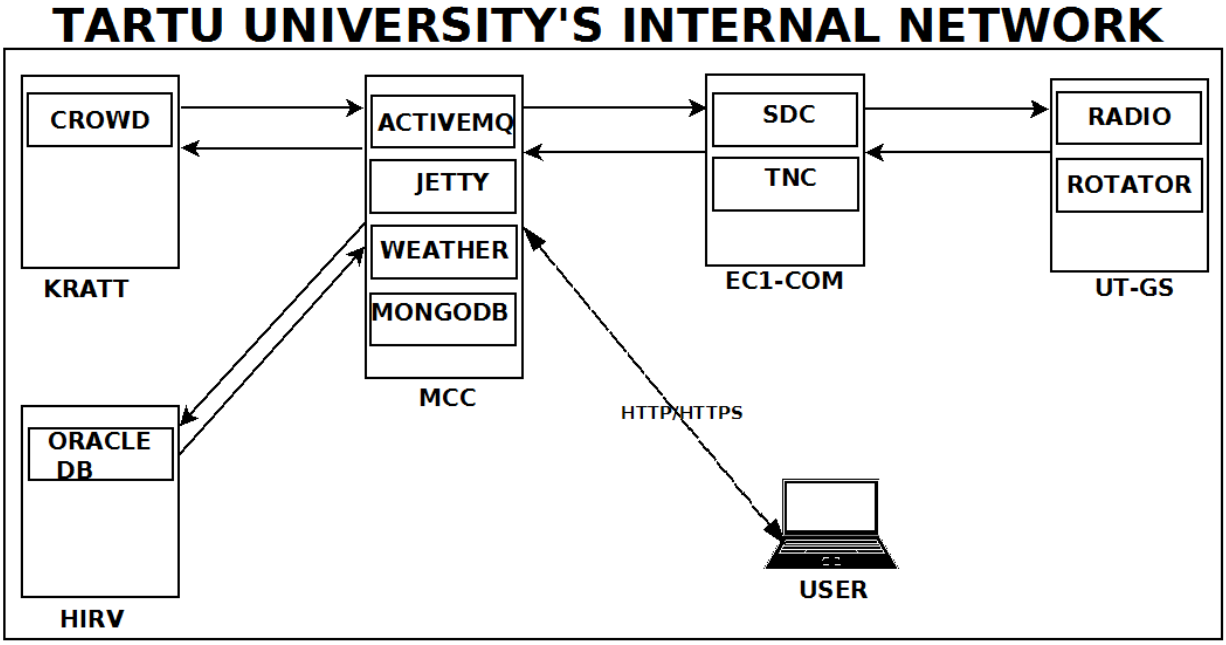


Figure 1.2. Showing components diagram inside Tartu University's internal network with the flow of transmission running from right to left.

1.3 Components over the Internet

Figure 1.3 shows MCS without the internal network of Tartu University and now connected to the Internet, with most components communicating over the Internet. For the whole system to function properly in this new network (the Internet), it has to be configured securely using SSL configuration for particular components. The figure below shows red ovals as the components that must be secured in order for the system to use the connectivity of the Internet.

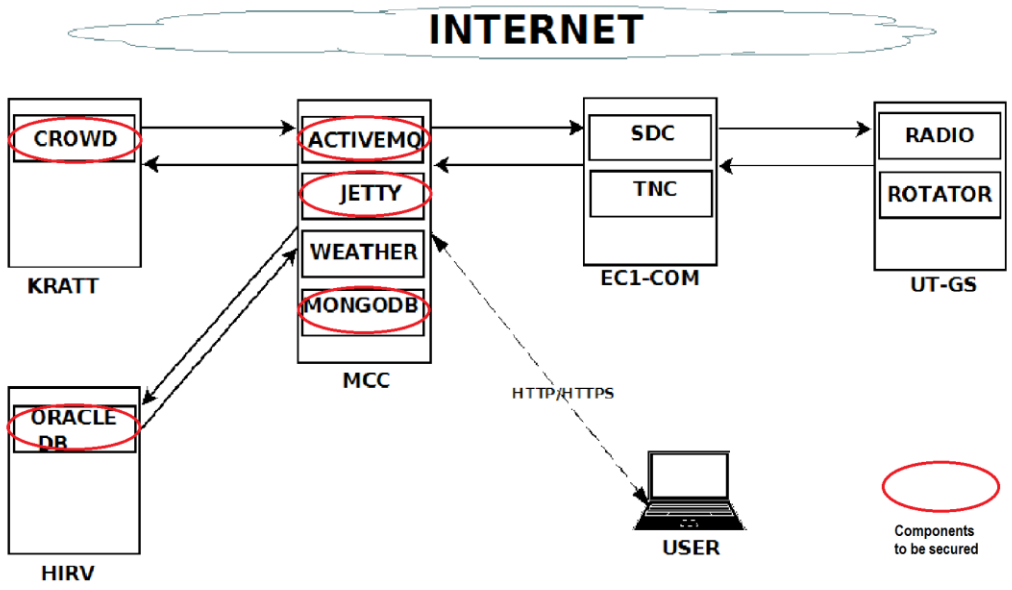


Figure 1.3. Components diagram having red ovals on them are the important components that security implementations must be carried out on.

1.4 Mission Control System Architecture

This is a simplified view of the entire ESTCube-1 architectural system showing the sub-sections of the project, components and connectivity, as well as the end-users. The Satellite (ESTCube-1) communicates with the Ground Stations over an amateur radio protocol called AX.25; Ground Station communicates with the Back-End components over the internet; the Back-End then communicates with the End-User(Browser) via the internet as well.

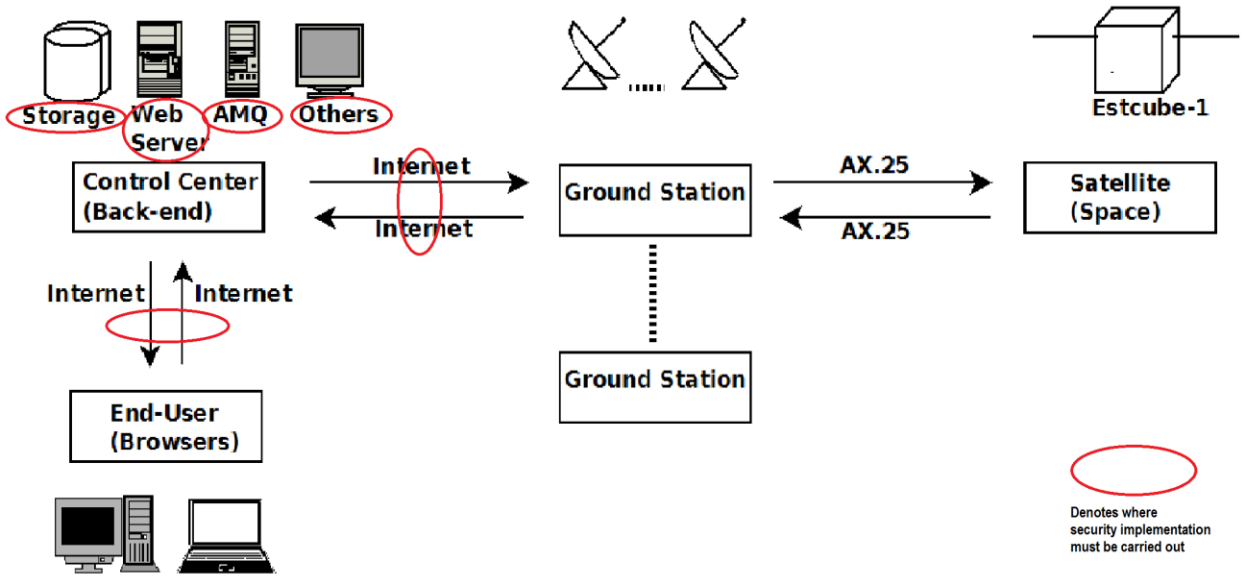


Figure 1.4. System Architecture revealing the Satellite, Ground Stations, Control Centre comprising of databases, webserver and others, and finally the End-User. The red ovals identify the components that must be secured.

1.5 How Mission Control System works

The current state of MCS is represented in the figure below showing how the components are connected. In Figure 1.5 ESTCube-1 sends packets to the Ground station via AX.25 protocol, which is then sent to TNC and SDR hardware after that TNC communicates this to TNC driver via Serial connection, TNC driver sends the packets to the Archiver via AMQ and they are later stored in the Oracle database. From the SDC to AMQ, then to WebServer via TM and finally to GUI (downlink). Up-link from GUI to the WebServer, from Webserver to AMQ via TC, from here to the SDC and then to TNS driver and to TNS and finally to ESTCube-1. The SDR gets packets to the SDR driver using USB, TCP, Serial, the SDR driver then sends packets to AMQ, AMQ sends to Archiver via TCP and Archiver to Oracle database (downlink). Then from SDC to AMQ, then to WebServer via TM and finally to GUI (downlink). For up-link- from GUI to the WebServer, from Webserver to AMQ via TC, from SDC to SDR driver, from there to SDR and finally to ESTCube-1.

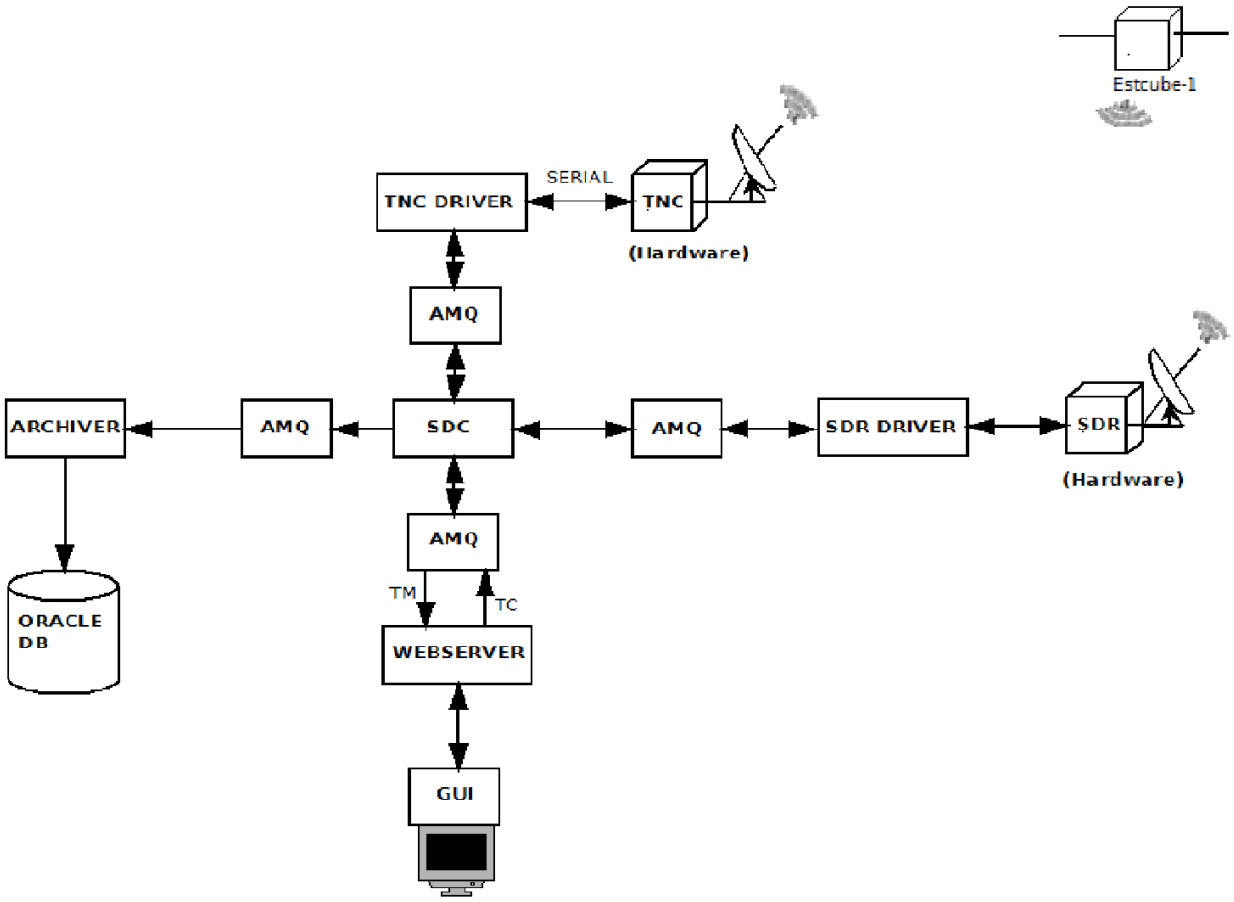


Figure 1.5. View of how packets are moving from and to ESTCube-1 as well as the communication protocols.

2 Available security solutions possibilities

2.1. ActiveMQ

ActiveMQ is an open source message broker with JMS (Java Message Service) client support designed by Apache. ActiveMQ, which is a Message Oriented Middleware (MOM) was designed for sending messages between two or more components in the same application or different applications[2]. ActiveMQ broker is a Publish-Subscribe messaging negotiator that publishes messages regardless of the subscriber and the subscriber receives messages regardless of the publisher and all these are done in classes.

After installation of ActiveMQ, it started-off with the default configuration settings, which has no security configurations in place. So with these default settings in place ActiveMQ is not secured a bit; the broker, users and other parts that connects to ActiveMQ must be securely configured.

2.1.1 Authentication

Security in ActiveMQ are mainly implemented as plug-ins. The plug-ins were made to be easier to customize and configure through the <plugin> element in the XML configuration file of ActiveMQ. There are two authentication plug-ins available to users in ActiveMQ:

- ***Simple authentication plug-in*** - This method allows the credentials to be taken directly from ActiveMQ's XML configuration file or from a properties file. This is the easiest way of securing a broker, since all the authentication credentials are in the XML configuration file of the brokers. ActiveMQ provides the simple authentication plug-in as a functionality that is part of its distribution. Figure 2.1.1A shows ***simple authentication plug-in*** granting three ***users*** access to ActiveMQ. The users have usernames and passwords for the authentication. The attribute named "groups" determines what category the users are in respect to the authorization.

```
<!-- Configure authentication; Username, passwords and groups -->
<simpleAuthenticationPlugin>
    <users>
        <authenticationUser username="system" password="${activemq.password}"
            groups="users,admins"/>
        <authenticationUser username="user" password="${guest.password}"
            groups="users"/>
        <authenticationUser username="guest" password="${guest.password}" groups="guests"/>
    </users>
</simpleAuthenticationPlugin>
```

Figure2.1.1A. Example of a simple authentication plug-in in ActiveMQ.

- ***JAAS authentication plug-in*** - This method uses JAAS API (Java Authentication and Authorization Service Java Authentication and Authorization Service Application Service Interface) implementation and it produces a more powerful, customizable authentication solution. ActiveMQ uses the same techniques for verifying user's credentials through a pluggable authentication API. The ***javax.security.auth.spi.LoginModule*** interface and changing of ActiveMQ configuration settings are all needed for the implementation of this authentication method. Out of the box, ActiveMQ is equipped with modules that can authenticate users by the property files, Secure Socket Layer (SSL) certificates and Lightweight Directory Access Protocol (LDAP). Figure 2.1.1B shows the file contents of JAAS module that is used to configure users and groups, which ActiveMQ will be able to recognize with ***PropertiesLoginModule***in placewith the ***activemq-domain*** being very important here as well as the ***PropertiesLoginModule***.

```
## Create a file login.config for configuring JAAS users and groups
## that has PropertiesLoginModule which ActiveMQ would recognize

activemq-domain {
    org.apache.activemq.jaas.PropertiesLoginModule required
        debug=true
        org.apache.activemq.jaas.properties.user="users.properties"
        org.apache.activemq.jaas.properties.group="groups.properties";

};
```

```
    <!-- Configure PAASauthentication -->
    <plugins>
        <jaasAuthenticationPlugin configuration="activemq-domain" />
    </plugins>
```

Figure 2.1.1B. JAAS login module for configuration of users and groups that ActiveMQ could recognize with *PropertiesLoginModule*and*activemq-domain*.

2.1.2 Authorization

Authorization is one more layer of security on top of authentication. This is level of control that gives a client the authority to access resources after the user has been authenticated. This is only granted to the client with that permission. Authorization for ActiveMQ has two levels, which have more control than the simple authentication: Operational-level authorization and message-level authorization.

- ***Operation-level authorization*** - Operation-level authorization or user-level operation consists of three main elements: **read**-gets messages from the JMS destination; **write**-sends messages to the JMS destination; **admin**-administers the JMS destination; controls how operations are performed. ActiveMQ XML configuration file makes it easier to add operations' authorization to some destinations. Figure 2.1.2A defines JAAS authorization plug-in, which points to the ***activemq-domain*** configuration. The destination to be secured is firstly defined when configuring the map authorization entries. The configuration is done in the ***topic*** attribute or the ***queue*** attribute. The second step is to declare which ***users***and or ***groups*** have the permission in that destination. The figure below shows the topics ">" can be fully accessed by the admins, the topic "**users.**>" can

be accessed by users and the topic "**guest.>**" can be accessed by guests, users. The operations and their descriptions are as follows: *read*- browse and consume from destination, *write*- send messages to destination, *admin*- creates destination if not existing and controls the creation of new destination in the ***topic/queue*** hierarchy.

```
<plugins>
    <jaasAuthenticationPlugin configuration="activemq-domain" />
    <authorizationPlugin>
    <map>
      <authorizationMap>
        <authorizationEntries>
          <authorizationEntry topic=">" read="admins" write="admins" admin="admins" />
          <authorizationEntry topic="USERS.>" read="users" write="users" admin="users" />
          <authorizationEntry topic="GUEST.>" read="guests" write="guests,users" admin="guests,users" />

          <authorizationEntry topic="ActiveMQ.Advisory.>" read="guests,users" write="guests,users" admin="guests,users"/>
        </authorizationEntries>
      </authorizationMap>
    </map>
  </authorizationPlugin>
</plugins>
```

Figure 2.1.2A. JAAS authorization plug-in pointing to the activemq-domain configuration with *read*, *write* and *admin* operations.

- ***Message-level authorization*** - Message-level authorization provides means of granting access to some messages in the destination. If the host and the broker are running on the same host then it allow application to consume messages. For this to happen, an interface that shows the implementation should be created and it should be a simple interface that defines a method *isAllowedToConsume()*. The method can access the message and consumption connection as well. The class and package should be compiled to a ***.jar*** file and placed in *{ACTIVEMQ_HOME}/lib/* directory for use. Then, the policy should be configured so that it can create the *AuthorizationPolicy* instance class in the ActiveMQ XML configuration file. When the broker starts, the *AuthorizationPolicy* class instance is started in the XML element <*messagesAuthorizationPolicy*>.

2.1.3 Build a custom security plug-in

Since ActiveMQ plug-in API is very flexible, thus allowing easier integration with other third party plug-ins. A class *BrokerFilter* in ActiveMQ makes it possible to intercept operation in the broker-level. Adding some custom functions can be done by extending the class *BrokerFilter* and

overriding a method for that operation. Implementation of a custom security handling plug-in and Implementation of JAAS login module are available custom options.

To implement custom plug-in, a class called *IPAuthenticationBroker* must be created to override the *BrokerFilter.addConnection()* method thereby limiting connection access to the broker on the IP-address. A check if the IP address can be granted access to connect or not is done when the method is implemented. Figure 2.1.3A shows the custom implementation of the broker. The method shown below checks if the IP addresses have the right to access the broker and thus is allowed to be connected but if it does not then an exception is thrown as shown in the red rectangle below. Once the plug-in has been implemented, then configuration and installation of the ActiveMQ broker process in order for the plug-in to work takes place. A custom implementation plug-in where a custom class instance is created and it returns the new broker for the next plug-in.

```
public class IPAuthenticationBroker extends BrokerFilter {
    List<String> allowedIPAddresses;
    Pattern pattern = Pattern.compile("^/([0-9\\.]*):(.*)");
    public IPAuthenticationBroker(Broker next, List<String>
            allowedIPAddresses) {
        super(next);
        this.allowedIPAddresses = allowedIPAddresses;
    }
    public void addConnection(ConnectionContext
            context, ConnectionInfo info) throws Exception {

String remoteAddress = context.getConnection().getRemoteAddress();
    Matcher matcher = pattern.matcher(remoteAddress);

    if (matcher.matches()) {
        String ip = matcher.group(1);
        if (!allowedIPAddresses.contains(ip)) {
            throw new SecurityException("Connecting from IP address "
                + ip + " is not allowed" );
        }
        } else {
            throw new SecurityException("Invalid remote address "
                + remoteAddress);
        }
        super.addConnection(context, info);
    }
}
```

Figure 2.1.3A. Custom implementation of a broker extending the class *BrokerFilter*.A method for checking if the IP addresses have the right to access the broker is shown in the red rectangle.

2.1.4 Certificate-based security

Certificate-based security solution uses underlying technologies needed for its execution.

- User Datagram Protocol (UDP) that sends and receives data packets but does not guarantee data packet's order-ness and data might have duplicates when they arrive. Another characteristic is that it does not guarantee the data packet delivery since it uses connectionless protocol.
- Transmission Control Protocol (TCP) sends and receives data in a stream-oriented manner with the data packet order-ness guaranteed no duplicates of data and guarantees reliability of packet delivery with no packet lost on the way due to its use of active reliable connection.

This leads us to Secured Socket Layer protocol (SSL), a secure protocol designed for transiting encrypted data over protocol like the TCP network. SSL protocol was designed to make use of a public key and a private key so that the communication channel would be secure. ActiveMQ provides an SSL transport connector that would ensure secure communication between the client and the broker over the TCP network. For SSL to work, the keys and certificates to be used must be configured appropriately.

```
<transportConnectors>
    <transportConnector
                name="tcp"
                uri="tcp://localhost:61616?trace=true"/>
    <transportConnector
                name="udp"
                uri="udp://localhost:61618?trace=true" />
</transportConnectors>

      <transportConnector name="ssl" uri="
      ssl://localhost:61617?trace=true" />
   </transportConnectors>
```

Figure 2.1.4. UDP, TCP and SSL transport connectors configuration is shown, in which ActiveMQ is set to use secure SSL connection over TCP.

This brings us to the Certificate-based security for securing the broker in ActiveMQ. It works by using certificates over SSL transport and using necessary plug-ins to secure the broker. This can

also be configured to grant different access rights to clients according to their certificates. Both the *keystores*(stores your private certificates and their private keys) and the *truststores*(stores the certificates of other trusted applications) should be created and made available.

2.2 Jetty Web Server

Jetty a Java-based open source project that provides HTTP server, HTTP client and java.servlet container[3]. Jetty offer machine-to-machine communication for large framework of software. The default Jetty installation comes without any security configuration implemented, so upon first starting it uses the default settings. The rest of this section will be about the important parts to be secured like the communication, connection and authentication, as well as the available solutions there are for Jetty.

2.2.1 SSL configuration

By default, Jetty web server comes with standard configurations like the use of *http* default base directory for managing multiple instances but with the ability to create more bases to handle more installations. To ensure the server's connection is secured, some modules will be added to the configuration file. The security configuration properties are in the *start.ini* file, while the others are located in *http.mod*, *https.mod* and *ssl.mod* files.

Firstly, a new base directory should be created for use and the *SSL*, *HTTP*, and *WEBAPP DEPLOYMENT* modules will be added to the start file in that *base* directory. Then the webserver's .*WAR* (Web Application Archive) file should be placed in the *webapps/* folder of the base directory (***my-base*** in this case). Next, the web application's *keystore* is copied and placed in the *ect/* folder of the base directory. The *start.ini* file is then edited to allow the SSL settings by changing this ***–module=ssl***, defining a secure port for redirection ***jetty.secure.port=8443***, setting up the **keystore** and **truststore*jetty.keystore=etc/keystore***and***jetty.truststore=etc/keystore***respectively. The default password should then be changed. Set the –module=server parameter for the initialization of the module server, the *–module=http* and *–module=deploy* should be set also. When all these changes has be done, Jetty could be started.

Alternatively, starting a new module can been done directly from the shell by adding the module to the command when starting Jetty: ***java -jar ../jetty-distribution/start.jar --add-to-start=https***

2.2.2 Authentication and Authorization (Security Realms)

Authentication in Jetty can be done in several ways called standard authentication mechanism, they are: ***Basic***, ***Digest***, ***Form***, ***Client-Cert***; and also using plug-ins mechanisms like ***JASPI*** or ***SPNEGO***. Figure 2.2.2 below shows the standard authentication mechanisms in red rectangles.

The authentication mechanism is done by declaring the <*login-config*> element in the ***web.xml*** descriptor or annotator files. The <*login-config*> element in the *web.xml* file is compared to the realm name in the jetty xml configuration file. The LoginService possesses unique name, each user is given their own authentication information and associated roles. A LoginService in Jetty performs the authentication by granting each user with authenticated information access into the webapp. This protection could be done specifically for one webapp or for many webapps. Jetty supports configuration of different types of authentication mechanism: HashLoginService, JDBCLoginServic, DataSourceLoginService, JAASLoginService and SpnegoLoginService.

Authorization comes into play after the LoginService does the authentication and access request to certain resources are made. When a request for resources is made, the LoginService then checks to see if the *user* has the permission to access that resource based on the granted role.

```
<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>Test Realm</realm-name>
</login-config>


<login-config>
  <auth-method>DIGEST</auth-method>
  <realm-name>Test Realm</realm-name>
</login-config>


<login-config>
  <auth-method>FORM</auth-method>
  <realm-name>Test Realm</realm-name>
  <form-login-config>
     <form-login-page>/logon.html?param=test</form-login-page>
     <form-error-page>/logonError.html?param=test</form-error-page>
  </form-login-config>
</login-config>
```

Figure 2.2.2. Standard authentication mechanisms showing the Basic, Digest and Form types in red rectangles with their matching realm names.

2.2.3 Limiting Form Content configuration

Jetty process form contents are sent to the server in a map that webapps uses. If a very large amount of these form contents or form keys is sent to the server, it could cripple the server because the server would have to utilize large amount of memory and CPU power, which will later lead to a shutdown of the system. To avoid this, a limit should be set on the amount of contents it can accept as shown in Figure 2.2.3. Form-Content limits configuration for a single webapp and server showing its *MaxFormContentSize*and *MaxFormKeys* attributes. Limits are set to indicate the intended allowed amount. These are handled with methods: *ContextHandler.setMaxFormContentSize(intmaxSizeInBytes;); and ContextHandler.setMaxFormKeys(intformKeys);*

```
<!-- Form Limit for webapp     -->
<Set name="maxFormContentSize">200000</Set>
<Set name="maxFormKeys">200</Set>
```

```
<!-- Form limit configuration for the server     -->
 <Call name="setAttribute">
   <Arg>org.eclipse.jetty.server.Request.maxFormContentSize</Arg>
   <Arg>100000</Arg>
  </Call>
 <Call name="setAttribute">
   <Arg>org.eclipse.jetty.server.Request.maxFormKeys</Arg>
   <Arg>2000</Arg>
  </Call>
```

Figure 2.2.3. Form content limits configuration for a single webapp and server with *MaxFormContentSize*and *MaxFormKeys* attributes.

2.2.4 Secure password obfuscation

Passwords are used for securing most systems and are stored as plain text, obfuscated,checksummed or encrypted according to the security needs. Obfuscation methods make it possible for the system to retrieve the password from the *keystore* when needed, while being protected from casual viewing. When a password is entered into the system, it is compared with the stored password and the result determines whether an authentication will be granted or not.

Various secured passwords are generated as seen in the Figure 2.2.4 below. The blue rectangle shows a user called “me” and accompanying password “blah” in red rectangle, which is in a plain text format. It’s then encrypted to produce what is in the big red rectangle, which are the three different secured password types (OBF, MD5,CRYPT) for the same plain text password. The password can be copied and pasted in Jetty xml file instead of plain text passwords. The prefixes OBF:, MD5: and CRYPT: must be copied as well before usage.

```
export JETTY_VERSION=9.0.0.RC0
java -cp lib/jetty-util-9.1.3.v20140225.jar org.eclipse.jetty.util.security.Password me blah
blah
OBF:20771x1b206z
MD5:639bae9ac6b3e1a84cebb7b403297b79
CRYPT:me/ks90E221EY
```

Figure 2.2.4. Secure password generation for a plain text password format with three formats to choose from (OBF, MD5,CRYPT).

2.2.5 JAAS configuration

JAAS plug-in is used for both Authentication and Authorization, a java version of Pluggable Authentication Module (PAM) framework is implemented by JAAS. With PAM framework JAAS can easily do new authentication plug-ins or updates authentication plug-ins without disrupting the application itself. Using JAAS in Jetty requires the ***org.eclipse.jetty.jaas.JAASLoginService*** to be declared and a ***jaas-login*** module configuration file should be created, and must be specified in the Jetty command. Steps to get this done:

- A JAASLoginService in *web.xml* should be configured to match the name of the realm
- A matching JAASLoginService with the same name as the one in previous step should be created
- Setup LoginModule in configuration file, the application name should be the same name used in the JAASLoginService
- Jetty with support for ***jaas*** should be called by adding the jaas-related jars to Jetty container's *classpath* and setting the system properties ***java.security.auth.login.config***. The default JAASLoginService works fine and it is flexible enough to allow other configuration settings.

2.2.6 Spnego configuration support

The Spnego (Simple and Protected GSSAPI Negotiation) mechanism allows Windows or Active Directory based network users to be authenticated. This type of authentication and authorization support is done through Java JDK 6 and above. This is a very sensitive and fragile setup to use because it requires all configuration to be precise or it will not work. Good knowledge of Windows network and Active Domain Controller is required to configure, test and maintain the services for its users. Figure 2.2.6 shows Spnego configuration commands (a) and (b) for running and debugging respectively, as well as enabling Spnego authentication using role name as seen in the red rectangle.

```
Configuration Spnego requires these commands (a)
-Djava.security.krb5.conf=\path\to\jetty\etc\krb5.ini /
-Djava.security.auth.login.config=\path\to\jetty\etc\spnego.conf /
-Djavax.security.auth.useSubjectCredsOnly=false

Commands used to debug Spnego (b)
-Dorg.eclipse.jetty.LEVEL=debug /
-Dsun.security.spnego.debug=all
```

Figure 2.2.6A. Commands needed for the configuration and debugging of Spnego authentication for Jetty. Enabling Spnego authentication using a role name is shown with the red rectangle.

Enabling Spnego authentication in a webapp requires the role-name of the network to be changed to fit the corresponding one as shown below in Figure 2.2.6B. For the webapp to utilize this mechanism, a UserRealm must be created either by programming or in *jetty.xml* or in the context file. Amongst other important configuration files that comes with Jetty distribution are some Spnego configuration files: *spnego.properties* (configures user realm with runtime properties), *krb5.ini* (configures kerberos setup) and *spnego.conf* (configures the connection between gssapi and kerberos).

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Secure Area</web-resource-name>
    <url-pattern>/secure/me/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <!-- this is the domain that the user is a member of -->
    <role-name>My-HOST</role-name>
  </auth-constraint>
</security-constraint>
<login-config>
  <auth-method>SPNEGO</auth-method>
  <realm-name>Test Realm</realm-name>
  <!-- optionally to add custom error page -->
  <spnego-login-config>
    <spnego-error-page>/loginError.html?param=foo</spnego-error-page>
  </spnego-login-config>
</login-config>
```

Figure 2.2.6B. Enabling Spnego authentication using the role-name.

2.3 Mongodb

Mongodb is a NoSQL document-oriented database system that uses an open data format called Binary-JSON (BSON) which is based-off JSON to store data[4]. Mongodb like other NoSQL databases out there by default is without security password protection; this means anyone can access the database. Since security was not the main goal in mind when designing this database,

it lacks in the security department[5]. In order to have a secure database, the default settings must be changed ranging from the network connection using SSL to the database to the encryption of stored data in the database.

2.3.1 SSL configuration

SSL support for Mongodb allows encrypted connection to mongod instances. The default Mongodb distribution does not ship with SSL support but can be configured to allow that locally by using--*ssl* option to the database from the command line. SSL configuration for mongod and mongos instances that requires a client's certificate must have mongo shell started with --*ssl* and –*sslPEMKeyFile*and the *.pem* file that holds the certificate and keys needed for SSL connection to occur[6].

2.3.2 Access control

Mongodb comes equipped with access control capabilities, which by default is installed in the admin database, and it restricts the right to change or modify the structure of Mongodb. Mongodb supports authentication but by default it does not come with any, which make it easy for anyone with access to the network connection to gain access and issue commands[4].

The Authentication Mongodb offers is on a database level only, it does not apply to the whole database. Authentication for Mongodb is disabled by default. Users who are "*superusers*" have the *userAdmin* or *userAdminAnyDatabase* roles that enables them to create, modify and grant privileges to themselves, as well as other users. To enable the authentication, Mongodb has to start with this command (--*auth*) attached to it or set the command after Mongodb has started with the commands shown in Figure 2.3.2A. In Figure 2.3.2B, the user being authenticated was successful therefore the appearance of the number 1, otherwise it would be 0.

```
> auth=true
true
> setParameter = enableLocalhostAuthBypass=0
0
>
```

Figure 2.3.2A. Authentication activation for a database after starting, the setParameter telling the database not to enable localhostbypass authentication.

```
> db.auth("jon", "123joN")
1
```

Figure 2.3.2B. A user being authenticated, the number 1 indicates a successful authentication.

Authorization is done based on role privilege once the user has been authenticated, user can then gain access to some information. Figure 2.3.2C shows how a user is created: firstly, the admin user is called and then user's credentials are created. Notice the password changed, it was encrypted with MD5 hash and it is more secured. The Figure 2.3.2D below shows that the authenticated users do have the authority to perform these certain tasks. Mongodb authenticates users by "roles" privileges that are defined when a user is created or assigned to the user afterwards.

```
> use admin
switched to db admin
> show users
{
        "_id" : ObjectId("533c2485885b7190c908c636"),
        "user" : "jon",
        "pwd" : "201937884be87a5b7c2a857acf477fb3",
        "roles" : [
                "userAdminAnyDatabase"
        ]
```

Figure 2.3.2C. Creating a new user SuperUser with userAdminAnyDatabse role and encrypting the password with MD5 hash.

```
> use admin
switched to db admin
> show collections
system.indexes
system.users
>
> use jon
switched to db jon
> show collections
> db
jon
>
```

Figure 2.3.2D. Authorized users admin and jon are granted access to the resource "*show collections*".

2.3.3 Kerberos authentication for Mongodb

Kerberos authentication is a standard authentication protocol that is supported by Mongodb for large systems. Kerberos authentication works by configuring the *mongod* and *mongos* instances of Mongodb with their valid *keytab files* respectively. Mongodb support for Kerberos

Authentication exists for both Linux and Windows OS and are configured almost in the same manner. The configuration procedures go: start the *mongod.exe* without Kerberos support, then later connect (with privileges that can create a user) *mongod.exe* instance via *mongo.exe*. Next add the Kerberos principals (allows request for external source to authenticate) to the Mongodb for every user that would be authenticated. After that, the *mongo.exe* should be restarted with the Kerberos support this time around, making sure the parameter *authenticationMechanisms* is set to GSSAPI. Lastly, connect *mongo.exe* shell client to *mongod.exe* and then authenticate externally.

2.3.4 Network security

Avoiding the possibility of access to the database through the Internet is a good step to securing Mongodb. Access to Mongodb by default is allowed through all available network interfaces. This poses a possible threat to the Mongodb security. This could be rectified by limiting network access to the programs (*mongod* and *mongos*) via configurations settings.

- VPN- Virtual Private Network provides a secure tunnel connection over network with encryption. It (VPN) provides different choices of encryption protocols that offers certificate validation and authentication.
- Port- Mongodb comes with a default listening **port 27017** for all running *mongod* and *mongos* instances. Although changing the port does not fully protect Mongodb, however it makes it easier to filter connections to the Mongodb instances and limiting access from unknown/untrusted clients.
- Firewall- Firewalls prevent or limit hosts access to a system. Mongodb supports Firewall on both Windows and Linux systems. Windows Server's ***netsh***is responsible for managing Windows' Firewall. Linux systems use the ***netfilter*** firewall to filter traffic to the ports using ***iptables*** interface for access. It is safe to use these security options for securing Mongodb.

- Bind-ip- This configuration setting allows the limitation of network interfaces from which both *mongod* and *mongos* instances listens for incoming connections. Limiting the amount of incoming connections to listen to is done by specifying the IP addresses with commas separating them, while the programs are running. Both *mongod* ***–blind_ip*** and *mongos* ***–bind_ip*** activates this option.
- NoHttpInterface- This configuration settings option makes it possible to disable the default status page for "home" that is usually in port 27017. Changing this setting would make it impossible to view the read-only status interface. To enable that, run these commands: ***mongod – nohttpinterface*** and ***mongos –nohttpinterface***.
- Rest- Mongodb's*mongod* by default disables the *rest* setting which is responsible for the administrative *rest* interface. Activating this option will make the read-only status interface interactive. Authentication for this interactive rest interface is not supported, so it is not really secured therefore access to this interface should be limited to only known/trusted clients. Activation is done with *mongod –rest* in the command line.

2.3.5 Data encryption

A scenario where all other forms of security fails and an attacker gains access into the database and the stored data are encrypted. The data would thereby be useless to the intruder or simply difficult to decrypt. A full database encryption for Mongodb is not available yet but there are some steps to take to ensure some level of encryption. Encryption of selective (sensitive) stored data can be done on the application layer. Third party encryption solutions exists for this problem, amongst them are MongoDirector.com encryption solution, Gazzang and LUKS (Linux Unified Key Setup). For the encryption of moving data, SSL is most highly recommended.

2.3.6. Audit system activities

Auditing for Mongodb is supported but only in the MongoDB Enterprise edition. The auditing involves the tracking and writing events of all mongod server and mongo router processes to the console, a syslog, a JSON file and BSON file. The log messages can then be viewed to see who accessed what and what operations were done and so on. All auditable operations are recorded by default but with the help of a built-in filter option, the events to be recorded could be selected or filtered.

2.4 Oracle database

Oracle 11g is an object-oriented relational database system or simply ORDBMS for storing data. Oracle stores data in tablespaces logically in the form of datafiles. Oracle uses the standard SQL for storing, retrieving and manipulating of data[7]. Oracle database concerns itself with security, so there are some available security solutions already in place after installation. Although these are the basic security possibilities, they are not too bad for a beginning. More security solutions will be added to the already present ones to make the database more secure. Amongst them would be enhancing of security settings, security configuration settings and many more.

2.4.1 Enhance security settings

While it is advisable to have Oracle database installed and ran in a safe environment with some protections or behind firewall for security, some actions could be performed to make the database more secure.

- By default Oracle installations come with some passwords for both users and administrators, they should be changed. When left with default passwords, anyone with access to the OS can simply gain access to the database through the *sysdba* or *system* users accounts who have administrative privileges and can do many things. Figure 2.4.1 shows *sysdba* connecting to the database without username and password.

```
C:\Windows\system32>sqlplus / as sysdba

SQL*Plus: Release 11.2.0.1.0 Production on Mon May 12 23:03:46 2014

Copyright (c) 1982, 2010, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> alter user SYS identified by "123joN";

User altered.

SQL> alter user SYSTEM identified by "123joN";

User altered.
```

Figure 2.4.1. Connecting to the database without username and password, altering the user's password to remove it from its default state.

- Privileges are very powerful and should be granted carefully. If a user with administrative rights to perform tasks, like create tables,create users and so on, should a grant a privilege to a new user, the new user with the privileges could also grant that administrative right to other users. This is dangerous because it makes it easy to abuse this right. Privileges should be monitored from time to time.
- During a fresh installation of Oracle database, many components come bundled in the package. Oracle database configuration assistant has two installation options: Typical and Custom. The custom option should be picked so that unwanted components could be unselected and needed ones are installed. For example, the spatial feature (Spatial MDSYS schema), if installed and not properly managed could lead to vulnerabilities that are unknown to the administrator[8]. Therefore, if a component is not really needed, its installation should not be necessary.

2.4.2 Security updates installation

Security updates are great at fixing security loopholes and most time enhances the security of that system. Oracle offers security updates as well, which tackles the security issues that might be available. Security updates are usually offered to registered members with account on this link: *metalink.oracle.com*. Next is the security updates should be from the Operating Systems on which Oracle database runs. Most Operating Systems if not all, frequently or occasionally offer security updates to their users to patch any security problems that were discovered, some are done automatically and others manually.

2.4.3 Data encryption

Oracle supports encryption to some extent using Transparent Data Encryption (TDE), which is a file level encryption solution for data at rest on hard drives and backup media. In Oracle, this could be enabled through the Advanced Security Option. TDE supports encryption for both columns and tablespaces, so encryption could be done on the table's column or an entire tablespace. If data is entered into an encrypted column or tablespace, TDE automatically encrypts the data and when data is queried from the encrypted column or tablespace it decrypts it before sending it out.

2.4.4 OS authentication bypass

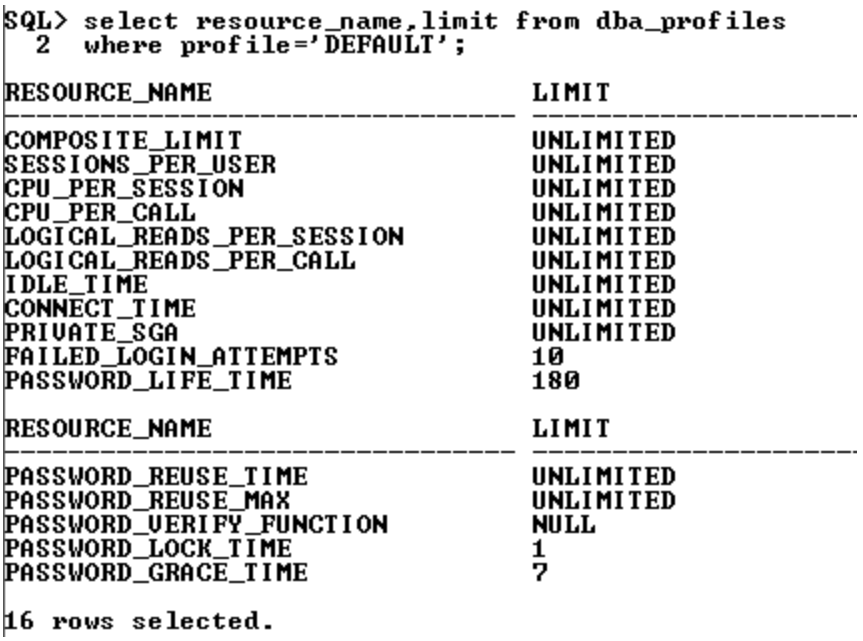
A user gains access to the Oracle database without any form of authentication by Oracle because the authentication was done in the OS level. Although, it might not be considered dangerous to some extent if the database is behind a firewall but it is not a safe practice because anyone with access to the machine automatically have access to the database as shown in Figure 2.4.1 above.

2.4.5 Security configuration enhancement

Password parameters management is a vital part of security for Oracle database system. Looking at the **dba_profiles**, some resources' limits are ***unlimited*,** these are somewhat risky to the database. Figure 2.4.5 shows various default profiles and their limits, including password parameters' management in the dba_profiles.

- Failed_login_attempts - the amount to failed attempts login before the account is locked
- Password_life_time - the amount in day/s a password could be used for authentication before it expires
- Password_reuse_time - the password can be used for the amount of day for authentication
- Password_reuse_max - the amount of times a password must be changed before using the original one
- Password_verify_function - the passwords' complexity will be verified to ensure it is not weak. A free program Oracle Password Checker (Cracker) can check for weak passwords
- Password_lock_time - the account would be locked in day/s after several failed logins. If not changed, it remains in its default day of 1.
- Password_grace_time - the amount of day/s the account will be locked for after the password expires.

The profiles can be changed with a command like this: ***alter profile default limit***password_life_time 100*;*



```
SQL> select resource_name,limit from dba_profiles
  2  where profile='DEFAULT';

RESOURCE_NAME                    LIMIT
-------------------------------- ----------------------
COMPOSITE_LIMIT                  UNLIMITED
SESSIONS_PER_USER                UNLIMITED
CPU_PER_SESSION                  UNLIMITED
CPU_PER_CALL                     UNLIMITED
LOGICAL_READS_PER_SESSION        UNLIMITED
LOGICAL_READS_PER_CALL           UNLIMITED
IDLE_TIME                        UNLIMITED
CONNECT_TIME                     UNLIMITED
PRIVATE_SGA                      UNLIMITED
FAILED_LOGIN_ATTEMPTS            10
PASSWORD_LIFE_TIME               180

RESOURCE_NAME                    LIMIT
-------------------------------- ----------------------
PASSWORD_REUSE_TIME              UNLIMITED
PASSWORD_REUSE_MAX               UNLIMITED
PASSWORD_VERIFY_FUNCTION         NULL
PASSWORD_LOCK_TIME               1
PASSWORD_GRACE_TIME              7

16 rows selected.
```

Figure 2.4.5. The default profiles, limits and the password parameters management in dba_profiles.

2.4.6 Network security

The network security here would include Oracle SSL configuration, Listener's configuration and closing of unused or unnecessarily open ports that may connect to the database.

- Oracle network security configuration for the SSL protocol ensures the security of data on transmission over network. SSL ensures data integrity, confidentiality, encrypts data and provides authentication for access using the Oracle Internet Directory's message digest. In Oracle environment SSL components are Certificates authentication, Certificates and Wallets. The SSL support uses Certificates that are stored in the Wallet and the certificates are taken from the Wallet when needed for Authentication. SSL works by going through *tnsname.ora* configuration file, checks the *listener.ora* file for the TCPS protocol and then connect through a secure port[9].
- The listener.ora configuration file consist of an Oracle created EXTROC entry, which make it possible for making external procedural calls from the database. If the database does not make any of these external procedural calls, for security reasons it is advisable to remove it from the listener.ora file (ADDRESS = (PROTOCOL = IPC)(KEY =

EXTPROC1521)). Another option for securing the listener is by setting an administrative restriction on the listener, adding ADMIN_RESTRICTIONS_LISTENER_<non-default-listener-name>=ON would make users with "write" permission unable to carry out changes to the listener.ora file. Restricting IP addresses that can gain access to the database will provide a layer of protection. In a scenario where other protections are bypassed, this can protect the database. Add these commands (tcp.validnode_checking = ON and tcp.invited_nodes = (hostname1, hostname2, hostnameN) to sqlnet.ora file to set the permitted IP addresses. All these files could be found in this path ~dbhome_1\NETWORK\ADMIN[9].Benefits of securing the Listener includes avoiding:**1.** *Stopping the listener and creating large file that consumes CPU and disks resources thereby causing Denial-of-service (DoS) attacks.* **2.** *Writing to the file system through the listener.* **3.** *Running codes called from the database or at the OS level.* **4.** *Extracting the database's configuration information for further use*[10].

- All unused ports that could communicate with the database which are not in use should be closed or blocked. Leaving ports that are not in use open could lead to a point of entry for attackers. View and close available port using the Command line or Fusion Middleware Control[9].

2.5 Crowd

Atlassian Crowd is a management tool that allows secure single sign-on into many web apps for user. Crowd's SSO product is designed to integrate into third party applications and systems[11]. Lightweight **D**irectory **A**ccess **P**rotocol (LDAP)[2] is an open, simpler version of the industrial standard X.500 protocol[3], which is used for accessing information directories. It actually

[2]http://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol

[3]http://en.wikipedia.org/wiki/X.500

supports TCP/IP and it can be used in different types of servers. LDAP is mostly used for single sign on that allow a user to utilize one password for many services[12].

2.5.1 Authentication and Authorization

Authentication is verifying a user identity to confirm who they are and this is usually done typically by using a ***username*** and ***password***. Authentication could also be done by connecting to an LDAP sever, and then bind the connection to the application user. The DN (distinguishing name) of the user to be authenticated should be searched for, making sure the user exists or is valid and finally bind the connection to the DM user and use the supplied password[13].

Authorization is done when attributes of the user's account are searched and re-bound as the application user, then checks for the user and requests attributes showing if user is authorized[13]. Authorization mostly come in the form of belonging to **groups**, having **roles** and **permissions**

Figure 2.5.1 shows a successfully authenticated user with through the LDAP's "Authentication Test" login page.

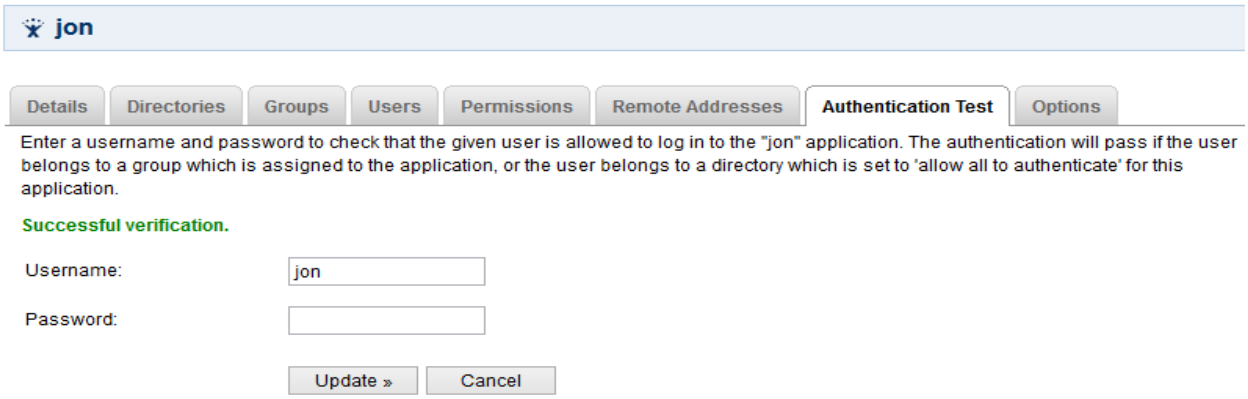


Figure 2.5.1 show a successfully authenticated user with through the LDAP's Authentication Test login.

2.5.2 Change the Crowd's port

Crowd uses a default port of 8095. Figure 2.5.2 shows 8095 being used as the connector for http and it is the main port. This could be changed to a new port to make it less conspicuous and

thereby giving Crowd less changes of being attacked using this means. In addition, changing the crowd.url port, demo.url port, openidserver.url and running the build.xml script against the build.xml configuration file goes a long way in protecting Crowd[14].

```
# The http port you wish to run crowd from, ie: http://localhost:8095/crowd
crowd.tomcat.connector.port=8095

# Tomcat requires a unique port for shutdown
crowd.tomcat.shutdown.port=8020

# Crowd context root
crowd.url=http://localhost:8095/crowd

# Demo context root
demo.url=http://localhost:8095/demo

# OpenID server context root
openidserver.url=http://localhost:8095/openidserver
```

Figure 2.5.2 Connectors configuration for http, Tomcat, Openid server and its main port.

2.5.3 Spring security

Spring Security [4] is a powerful and high customizable authentication and authorization framework for J2EE applications. This approach enables Spring Security to handle the authentication and authorization requests to Crowd's applications that supports it[15].

2.5.4. Change all default passwords

Users - both administrator and new user should change their passwords upon first logins. This is a good practice as it makes users utilize stronger and safer passwords. By default, after installation the passwords for Crowd - Openid-server, Crowd, Demo, and Username are all "password". Easily guessable passwords like these are a threat to the security of Crowd and other applications [11].

[4]http://projects.spring.io/spring-security/

2.5.5. Security updates installation

Updates are vital to all applications as most consists of various problems they address. Some updates may just be bug fixes, security fixes or some additional features to the application. For most applications with vulnerabilities, security fixes are usually issued to address those issues and, these may sometimes come as a major update or a simple patch or hotfix.

3Analyse the different security solution options available

To better understand how these security solutions would be applied to ESTCube-1's Mission Control System (MCS), the system would be divided into levels with the components forming the levels, as seen in Figure 3 below. First things first, the components must be installedand the

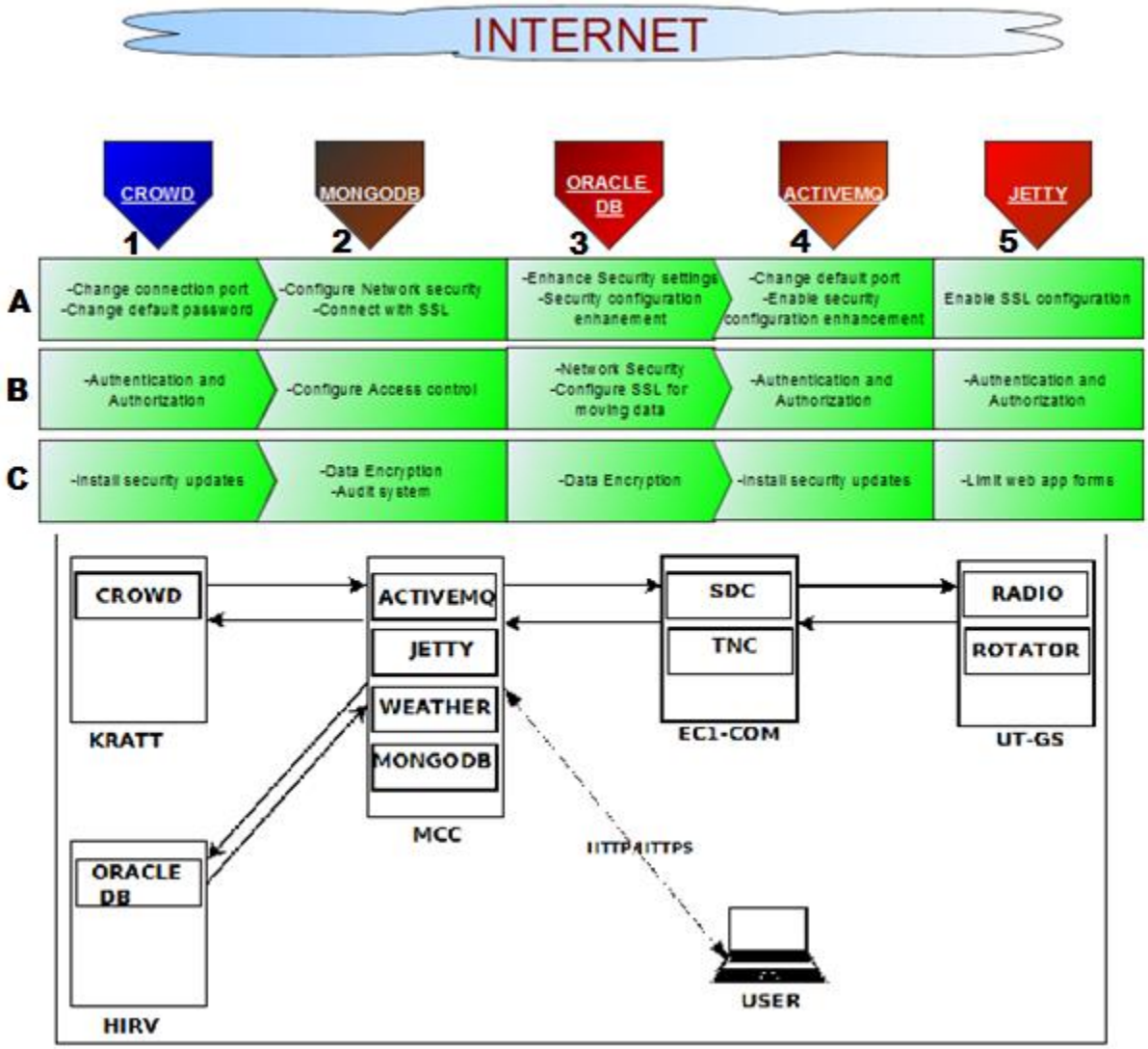


Figure 3. Layer-by-layer implementation of security solutions revealing the levels and their components, having MSC secured in the middle.

necessary linkages between the components are made appropriately. For MCS to function well without hiccups, it is advisable to assume the following order for installation, running and implementation of the security solutions: Crowd > Mongodb > Oracle db> ActiveMQ > Jetty. Figure 3 illustrates the level-by-level security solutions implementation to be enforced, the alphabets **A**, **B**, **C** being the levels and the numbers **1**, **2**, **3** being the component's number and direction. The full MCS in focus is not a secured system; it has security problems that needs urgent attention. It is only right to view the security problems from a broader perspective (the system as a whole), which also includes the various components that make up the system. Fixing these issues or implementing these security measures would be done in four levels starting from the basic security implementations and moving deeper to the core.

LEVEL A – After installations of all the components, the first approach to ensuring a secure system would be to secure the network connection of components in level **A**, starting from numbers **1** through number **5**. For **Crowd** it would be to change the connection port from default port to another one to make it unknown to others, and changing the default password, which would make its access controlable. Next would to configure **Mongodb**'s Network security, as well as to connect with SSL to ensure a secure connection to the database. For **Oracle** the Enhance security settings should be configured to ensure access control and also to enable Security configuration enhancement to manage the password parameters communication between Oracle clients and Oracle. The **ActiveMQ** would firstly be changed from its default port to hide it and then enable configuration of Certificate-based security to allow connection for ActiveMQ to be safe and secure from the beginning. Finally, **Jetty** first would change its default connection port and then implement SSL configuration that would provide secure connection from the start.

LEVEL B – this level already have some protection and now would be to take the security up a notch by following the same numbering pattern. **Crowd** in this stage should be configured for Authentication and Authority to enable access control, also implementing Spring security configuration as a form of access control. The **Mongodb** now requires implementation of Access control in order to restrict activities to the database and implementing Kerberos authentication to do the same as Access control, just a tad safer. **Oracle** would be configured to enable Network security to secure connection to port and Data encryption of moving data to ensure the integrity

and confidentiality of the data in motion using the SSL protocol. Here in **ActiveMQ**, the configuration of Basic Authentication and Authorization mechanism should be done to control how the users can access it or LDAP authentication implementation as well doing similar thing but with more security. In Jetty, implementation of in-built Authentication and Authorization (Security Realm) mechanism or using one from JAAS configuration or Spnego authentication configuration for a safer and more reliable means.

LEVEL C –The final level for **Crowd** would be the installation of application/security updates as soon as it is available. For **Mongodb** on this level would be to implement Data encryption as last stage of security and Audit system activities to know what is happening and had happened in the system. In **Oracle**, it would be to implement Data encryption of data at rest- for protecting the stored data form access ifit happens to be the final security option available, another one is to carry out Security updates installation, always. For **ActiveMQ**, install available security updates if/when available. Finally, for **Jetty** is to ensure the configuration of Web-Form limit for both webapps and server that would prevent influx of information to Jetty, which would cause denial of service attacks.

3.1 ActiveMQ

ActiveMQ's default settings have little or no security protection in place and it is highly vulnerable to intrusions and attacks. Table in AppendixI below highlights the list of possible solutions to make ActiveMQ more secured. The first solution which is **Authentication**[2] has the benefit of being the simplest form of security to be applied, it is customizable and does support plug-ins. A big advantage is that it is a default feature but it still supports other third party products. The bad thing about it is that the user's credentials are stored in plain-text format and this is not suitable for product or large systems. **Authorization**[2] on the other hand is easy to configure, supports JAAS configuration, has two levels of authorization and it does not depend on any transport protocol. The configuration files and credentials can be accessed easily from the system is a disadvantage. **LDAP authentication**[2] uses different technologies to authenticate and authorize users. Its support for SSL, its flexibility, support of plug-ins and being secure are good traits. The negative thing here is that it stores the directory locally and unprotected. **Custom security plug-ins**[2][16] might not be expensive, they are secure and

flexible-all good traits. They can take long time to build and can be complex depending on the amount of the codes involved. **Certificate-basedsecurity**[2][16]issafe and secure, commercial or private options to choose from, supports OpenSSL. The cons are that the commercial CA are costly, the private CA have trust issues with browsers, applications and Operating Systems. The self-signed certificate has maximum of one-year life span, is slower and uses low hash and cipher technology and has no support for advance PKI function.

3.2 Jetty Web Server

Jetty web server is an http sever and servlet container based on java used for adding network and web connectivity to java apps via set of JARS. A complete Jetty web server installation can be accessed, manipulated and controlled without permission because all settings and features are in their default states. Below are the measures to be put in place to ensure a secure Jetty web server as well as the good and the bad things associated with them. In AppendixII, the first solution, which is SSL configuration, provides Jetty with integrity, makes it trustworthy, confidential, encrypts data and authenticates target's identity. SSL certificates expire, difficulties in installation, hinders performance and 100% protection not guaranteed-are some of the disadvantages of this solution. The setting up of **Authentication and Authorization** for Jetty proves to make it safer and secure. Some of its pros are the fact that there are many plug-ins supported, does not use a lot of memory, has large community, has LDAP support and encrypts data. The issues with this solution are that the page load time may slow down, adds pressure on sever and its complexity. **Webapp form limit configuration** is flexible, customizable and simple. These are the advantages of this solution. On the other hand, it is a nice place through which attackers can carry out a DOS. Secure password obfuscation provides confidentiality, protects passwords and secures access to Jetty. The disadvantages are the inability to easily change the used algorithm for the passwords and the results are irreversible. **JAAS configuration** is flexible, secure, supports LDAP and multiple plug-ins. Bad about this is that the credentials are locally stored and might not be encrypted which will lead to others being able to access them. **Spnego configuration** offers security based on the standard GSSAPI mechanism, supports three main browsers, LDAP and SSO support and it is flexible. Disadvantages are no support for remote access, restricted browser support and requires a good

knowledge to test and configure. If not set properly or a little mistake is made, the whole system would fail.

3.3 Mongodb

Mongodb is an open source NoSQL database solution with support for many platforms. Like most NoSQL databases, its default installation has minimal or no security implemented for use. In AppendixIIIare shownthe various security features to be implemented to make Mongodb more secured for use both in testing environment and in production environment. The very first feature in this list would be the **SSL configuration**, which provides secure connection over the internet for Mongodb. Its advantages are data encryption, secure communication,and that it can use many valid certificates and support for many language drivers. Its downside is that SSL does not come pre-installed and has performance issues when implemented. Refer to the table in Appendix III for more. **Access control**, which restricts or minimizes access to the database, would be the next in line to make a more secure Mongodb. The advantages are the control over the database and the restriction of unauthorized access. The bad news is that after installation there are no authentication and authorization. Installed databases have no password authentication and new created users have a read-only access to all databases. **Kerberos authentication**, which can as well be a part of the "access control" solution, has these pluses: it is industry's standard, has support for Windows and Linux, is secure and supports many language drivers. The bad news is that there is no support for some older versions of Mongodb. **Network security** in this context contains the network filtering capabilities available in both Linux and Windows and the ability to allow SSL. Its advantages are secured communications, REST interface being disabled and support for Linux and Windows platforms. The **Data encryption**'s support for third party solution and encryption of sensitive data is a good factor. The disappointing factor is the fact that not the whole database can be encrypted and that it costs. Finally, the **Audit system activities** are very nice way of monitoring who had access to what operations and what activities took place. The good news is the knowledge of the operations going on in the database and the options to view these logs. The bad news is there is no *syslog* support for Windows, which is not a big deal though; also, this is for only the enterprise version of Mongodb.

3.4 Oracle database

Oracle database is an object-oriented database management system built by Oracle Corporation that uses the relational database framework for accessing data through SQL. An Oracle database installation does have some default security features in place but these are not considered secure. AppendixIV shows some of security implementations that must be put in place to ensure a more secure Oracle database. The first one in this list is the **Enhance security settings**, which talks about the changing of default passwords that are included in the installation, control of users' privileges and installation of only necessary components. Not to leave out authentication and authorization of users. This makes the database safer and it is a safe practice. Next is the **Security updates installation**. This is vital to the security of the database because it brings security fixes to bugs or backdoors. The security updates should be applied to the operating system also for better security on both levels. The sad news here is that some security fixes happen to be slow. The **Data encryption** will be encrypting data in motion and data at rest. The benefits of data in motion's encryption are the support for encryption mechanisms like OAS, IPSEC, Stunnel and SSH Tunnelling. While that of data at rest are eCrypts, RMAN, DBMS_CRYPTO and TDE for column encryption as well as tablespaces. Its disadvantages are the weight it puts on the system, database and it takes time. **OS authentication**means when the database allows authentication through the operating system.The good thing with it is that it is fast and time efficient but the bad thing with this is that it is not the safest form of authentication, while anyone with access to the machine can active do many things to the database. **Security configuration enhancement** deals with password parameters management to ensure a better and safer configuration for the database. Advantages of these tunings are good control over the access of the database, a safer database and knowing that better passwords are used for authentication. Network security entails the securing the network communication by encrypting the database's connection using SSL, changing the listeners' configurations as well as the tnsname configuration. The default connection ports must be changed too. Good about this is the knowing that the connections are only the approved ones and that the data are safe and secure. Bad is that the configuration can be confusing, which might be frustrating to users.

3.5 Crowd

Crowd is an Atlassian product for managing SSO into various web applications. The security-implemented features are shown in AppendixV. For **Authentication and Authorization**, Crowd uses LDAP technology, which is based on X.500 to bind the connection to a webapp user. The advantages of this authentication and authorization is the ease it delivers thanks to Active Directory and LDAP support, its access control and the one credential for multiple apps ability. Disadvantage is that credentials are stored locally in the directory. **Changing Crowd's ports** is a way of making Crowd more secure. All the default ports are known by anyone with knowledge of Crowd and could be used in a bad way. Changing the ports makes the ports unknown to intruders and it makes Crowd hidden. **Spring security configuration** is another way to secure Crowd by authenticating and authorizing access to Crowd. The advantages are the customization, security and encryption it offers. The drawback is that it works over unprotected networks. **Changing all default passwords** upon initial logins is a good practice because most parameters use the default passwords after installation. The negative part is the passwords can be used to access Crowd. **Application updates installation** might be a fix for any vulnerability found in Crowd, or addition of more features. Advantages of installing updates are the ease of the procedure; it might increase performance and security, and fix old ones. The disadvantage is that updates might alter existing configurations and settings.

4Implementation of chosen solutions on MCS test environment

4.1 Crowd

- CHANGE CONNECTION PORTS

Connection ports are changed from their default port number to a desired port by editing the *build.properties* file. Next is to change the server.xml file to match the one in the *build.properties* file.Figure 4.1A shows the default SSL port is changed from 8443 to 8444.

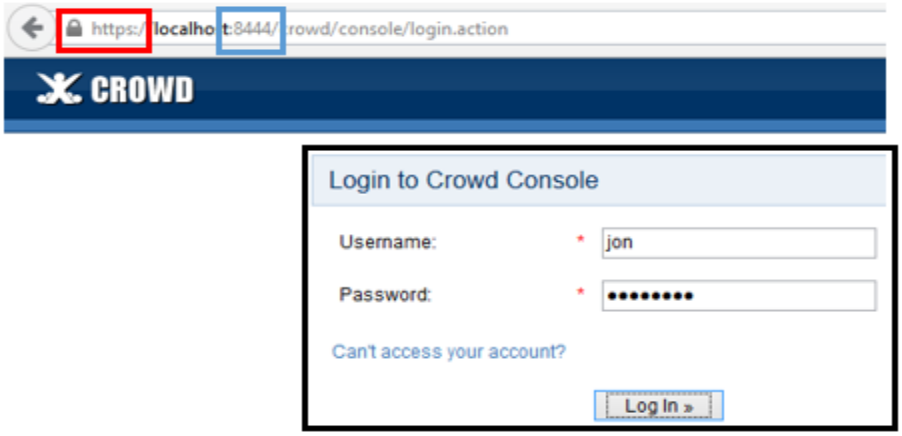


Figure 4.1A. Shows a web browser with information on SSL configured in red rectangle, changed port number in blue rectangle and login process for a user in black rectangle

- SSL CONFIGURATION

SSL was configured in the *server.xml*file to use new connector port, SSL protocol, Keystore file and keystore password was provided. Figure 4.3.2 is the SSL configuration file in server.xml showing the connection port, SSL protocol, Keystore file and Keystore password. Figure 4.1B shows SSL being actively used in the web browser.

```
<Connector port="8444" maxHttpHeaderSize="8192" maxThreads="150"
minSpareThreads="25" maxSpareThreads="75" enableLookups="false"
disableUploadTimeout="true" acceptCount="100" scheme="https" secure="true"
clientAuth="false" sslProtocol="TLS" SSLEnabled="true"
keystoreFile="C:\jdk1.7.0_51\bin\tomcaz.jks" keystorePass="tomz12345"/>
```

Figure 4.1B. SSL configuration showing the connection port, SSL protocol, Keystore file and Keystore password.

- AUTHENTICATION AND AUTHORIZATION

Authentication is done when a user wants access to the webapp through the web browser. The user is asked to present valid login credentials, with which the user would be verified. Figure 4.1A above shows a login realm for Crowd asking for a username and password. Authorization is done by assigning roles to a given user or group or given permissions. Figure 4.1C below shows how permissions are being taken away from users in a specific directory. This could be done in the group as well as with users.

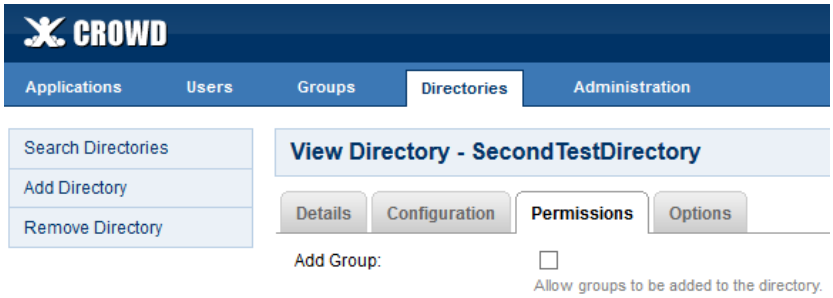


Figure 4.1C. Authentication done by checking or unchecking the boxes in the Permissions tab of a directory.

- APPLICATION/SECURITY UPDATES INSTALLATION

Security updates installation in Crowd are mainly offered through “Crowd Security Advisories and Fixes” channel. The page gives information on any vulnerabilities that was discovered in Crowd (as well asthe vulnerability’s severity, risk assessment and fix) and offers “Security Patches” if available, alternatively the solutions are in the form of application upgrades.

4.2 Mongodb

- NETWORK SECURITY

A different connection port was chosen to run the Mongodb server (mongod) and client (mongo). As seen in Figure 4.2A, the red rectangle shows the non-default port chosen to run the server and the rest of the command line contains other parameter needed for running the server in a secure way.

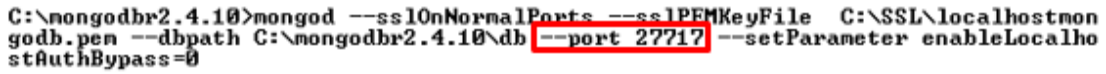


Figure 4.2A. A non-default port chosen to run the server as indicated in the red rectangle.

A Firewall (*netsh* Windows and *Iptables* for Linux) are used to configure limit from hosts to the system. Below in Figure 4.2B shows Mongodb connection port, Mongodb server and Mongodb client configured to be inside the Firewall, the red rectangles indicating outcome of the implementation and the blue rectangles show the rule names that are added.

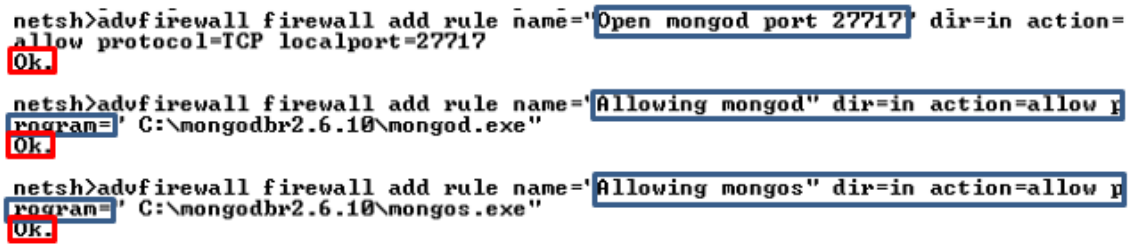


Figure 4.2B. The connection port, server and client configured to be inside the Firewall, the red rectangles indicates the status and blue rectangles shows the rule.

Bind_ip was used to bind *mongod* and *mongo* the specific ip-addresses. While the Rest interface was disabled not to allow connection to server through a web browser. Figure 4.2C shows the *mongodb.config* file having the *bind_ip*in red rectangle and rest in blue rectangle as well as other commands

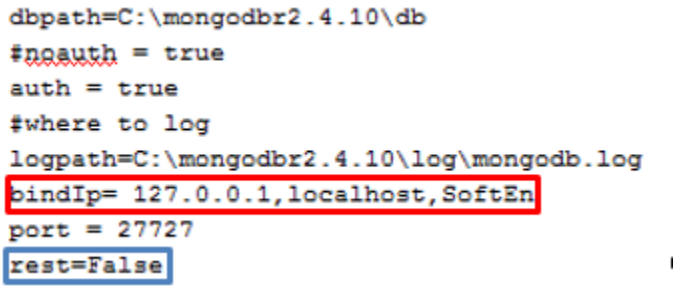


Figure 4.2C. *Mongodb.config* file with *bind_ip* option in red rectangle and *rest* in blue rectangle and other commands.

- ACCESS CONTROL

Authentication and Authorization: Authentication was done on the client end and it is shown in the server end. Figure 4.2D shows an authenticated user with the accompanying credentials in the server side after it was authenticated on the client side. Authorization was done on the user inside the database after being authenticated in the form of "roles" assigned to the user.

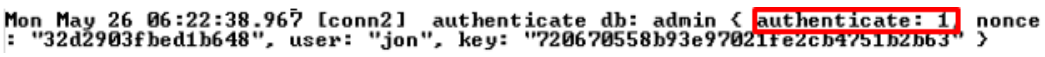


Figure 4.2D. An authenticated user with accompanying credentials in the server side after it was authenticated on the client side, the "1" in the red rectangle indicates the user was successful.

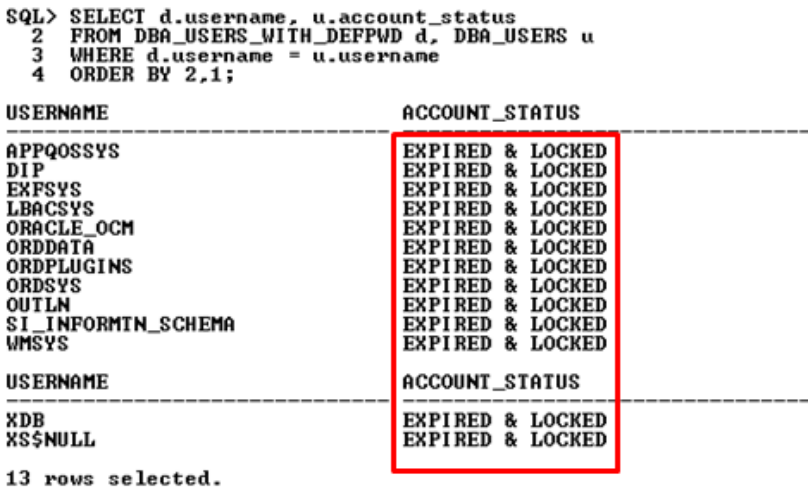
- DATA ENCRYPTION

Data encryption for this version of Mongodb is not officially supported but there are some third party paid and Open Source options to supplement for that. Since encryption of the whole database is not supported, only parts can be done with the official one. For Windows, the available options are for encrypting the whole drive. TrueCrypt and BitLocker Drive encryption tools are used for full drive encryption. Depending on the version of Windows owned will determine if it would be available for use. For Linux there are Linux Unified Key Setup (LUKS), Gazzang and MongoDirector.com.

4.3 Oracle database

- ENHANCE SECURITY SETTINGS

On gaining access to Oracle, the first thing done was to check the accounts to see which ones have or are using a default password associated with them. Figure 4.3A below displays all users which have default password set, they are ordered by theirs various usernames and the "Account_status" being "Expired & Locked" as shown in the red rectangle[17].



```
SQL> SELECT d.username, u.account_status
  2  FROM DBA_USERS_WITH_DEFPWD d, DBA_USERS u
  3  WHERE d.username = u.username
  4  ORDER BY 2,1;

USERNAME                       ACCOUNT_STATUS
------------------------------ --------------------------------
APPQOSSYS                      EXPIRED & LOCKED
DIP                            EXPIRED & LOCKED
EXFSYS                         EXPIRED & LOCKED
LBACSYS                        EXPIRED & LOCKED
ORACLE_OCM                     EXPIRED & LOCKED
ORDDATA                        EXPIRED & LOCKED
ORDPLUGINS                     EXPIRED & LOCKED
ORDSYS                         EXPIRED & LOCKED
OUTLN                          EXPIRED & LOCKED
SI_INFORMTN_SCHEMA             EXPIRED & LOCKED
WMSYS                          EXPIRED & LOCKED

USERNAME                       ACCOUNT_STATUS
------------------------------ --------------------------------
XDB                            EXPIRED & LOCKED
XS$NULL                        EXPIRED & LOCKED

13 rows selected.
```

Figure 4.3A. Displays all the users with default passwords, they are ordered by theirs various usernames and the "Account_Status" being "Expired and Locked".

For the users' account that is not in use or do not deserve certain privileges, the best thing to do is to remove or revoke its right to *EXECUTE*. In addition, some other means from which execution could take place are also revoked. As seen in Figure 4.3B showing privileges to the "Public" account stripped off its privilege to execute on certain channels. Only minimal or necessary privileges needed for getting jobs done should be granted to users[10].

```
SQL>
SQL> REVOKE EXECUTE ON dbms_job FROM PUBLIC;

Revoke succeeded.

SQL> REVOKE EXECUTE ON utl_tcp FROM public;

Revoke succeeded.

SQL> REVOKE EXECUTE ON utl_smtp FROM public;

Revoke succeeded.

SQL> REVOKE EXECUTE ON utl_http FROM public;

Revoke succeeded.
```

Figure 4.3B. Shows privileges to the "Public" account stripped off its privilege to execute on certain channels.

Next is checking for users with default password using the query that selects the usernames and password from the *dba_users*. As shown in Figure 4.3C with the result returning "*no rows selected*" indicating no such users exists.

```
SQL> SELECT username, password from dba_users where password='DEFAULT';
no rows selected
```

Figure 4.3C. The result returning "*no rows selected*", which is indicating no such users exists.

When creating the database, the "Custom database" option was chosen for installation so that only the needed components could be installed. This allows you to unselect the unwanted components that by choosing the default installation would be installed. Figure 4.3D shows all the default components in the "Database Components" and clicking on the button "*Standard Database Components"* reveals all the "*Standard Database Components*".

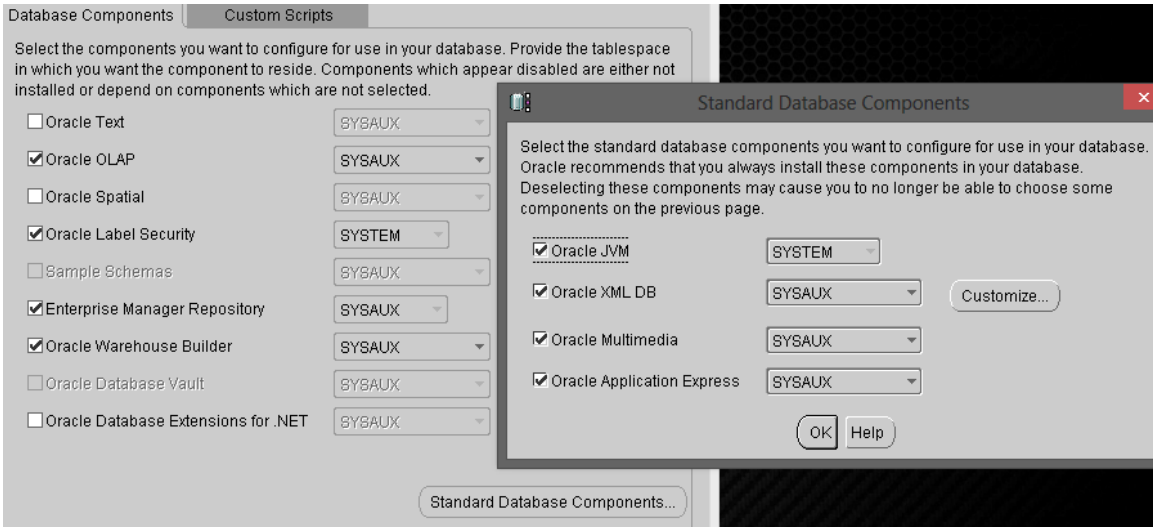


Figure 4.3D. All the default components in the "Database Components" and clicking on the button "*Standard Database Components*" reveal all the "*Standard Database Components*".

- SECURITY CONFIGURATION ENHANCEMENT

After checking the password parameters to see the default limits that exists, most of them were set to their default limit which is unlimited as seen in Chapter 2, Figure 2.4.5. The password

parameters are set to their respective values as shown below in Figure 4.3E, thus enforcing strict password parameter management for Oracle.

```
SQL> alter profile default
  2  limit password_reuse_max 3;

Profile altered.

SQL> select resource_name, limit from dba_profiles
  2  where profile= 'DEFAULT'
  3  ;

RESOURCE_NAME                    LIMIT
-------------------------------- ----------------------------------
COMPOSITE_LIMIT                  UNLIMITED
SESSIONS_PER_USER                UNLIMITED
CPU_PER_SESSION                  UNLIMITED
CPU_PER_CALL                     UNLIMITED
LOGICAL_READS_PER_SESSION        UNLIMITED
LOGICAL_READS_PER_CALL           UNLIMITED
IDLE_TIME                        UNLIMITED
CONNECT_TIME                     UNLIMITED
PRIVATE_SGA                      UNLIMITED
FAILED_LOGIN_ATTEMPTS            10
PASSWORD_LIFE_TIME               100

RESOURCE_NAME                    LIMIT
-------------------------------- ----------------------------------
PASSWORD_REUSE_TIME              60
PASSWORD_REUSE_MAX               3
PASSWORD_VERIFY_FUNCTION         VERIFY_FUNCTION_11G
PASSWORD_LOCK_TIME               1
PASSWORD_GRACE_TIME              7

16 rows selected.
```

Figure 4.3E.Password parameter management shows the changed parameter limits from their default state.

- NETWORK SECURITY

The default Oracle Listeners that comes installed shows its security being set to ON and can authenticate locally through the OS or by providing the listener's password. This is safer as access to the listener is authenticated and can restrict unwanted access. In Figure 4.3F, the listener status showing its security as being "*ON=Local OS Authentication*" and "*ON=Password or Local OS Authentication*", as well as its host and listening port.

```
Security                  ON: Local OS Authentication

LSNRCTL> change_password
Old password:
New password:
Reenter new password:
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=EXTPROC1521)))
Password changed for LISTENER
The command completed successfully
LSNRCTL> status
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=EXTPROC1521)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for 32-bit Windows: Version 11.2.0.1.0 - Produ
ction
Start Date                18-MAY-2014 11:26:56
Uptime                    0 days 7 hr. 58 min. 28 sec
Trace Level               off
Security                  ON: Password or Local OS Authentication
```

Figure 4.3F. Listener status showing it security as being "ON=Local OS Authentication" and "ON=Password or Local OS Authentication", as well as its host and listening port.

Next is to instruct the listener not to accept set of commands from any other sources apart from the ones it is given by editing the listener's configuration (*listener.ora*) file to include *ADMIN _ RESTRICTIONS_LISTENER = ON*.

Change the listeners' default password or simply change the old weak password to a new stronger one. This can either be done from the command prompt or using a GUI like that of Figure 4.3G

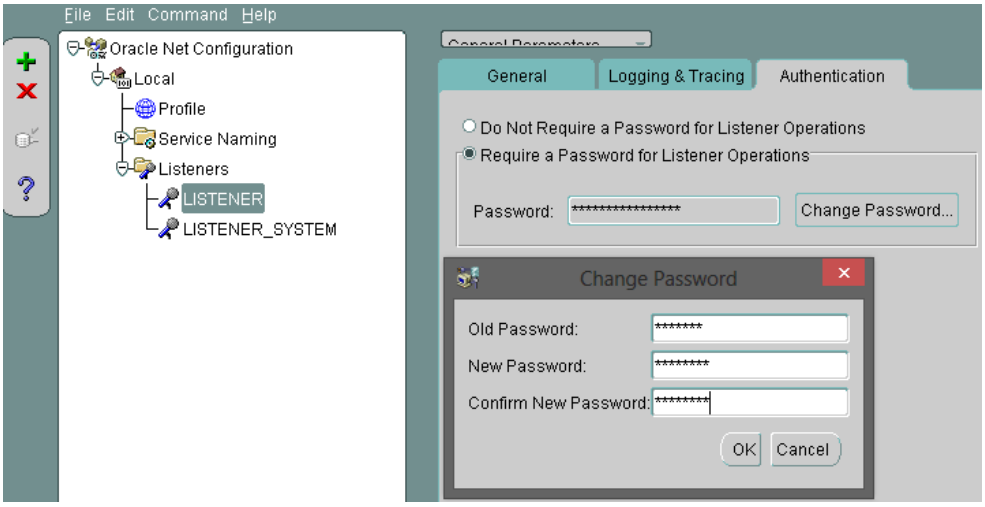


Figure 4.3G. The listeners' password being changed using the Oracle Network Configuration Manager GUI, as well as other parameters.

- CHANGE DEFAULT AND /OR UNUSED PORTS

In addition to the security implemented in the listeners', it is till advisable to change their default ports to something else to reduce the risk of attack by scanning for default and open ports.Most important ports to be changed are the Oracle Net Listener (1521) and the Oracle Net Listener/Enterprise Manager Repository (1526) ports. These are done by editing/changing the port number the *listener.ora* configuration file and restarting the listener. Method differs for others and is highly suggested to take ports from ranges 1521-1550 and 1600-1699for better protection.

- DATA ENCRYPTION (MOTION DATA-SSL)

After creating Wallets which contains the both the public-private key pairs using the Oracle Wallet Manger and/or OpenSSL[5], the Certificates are then to imported into the Wallet and saved to its directory while allowing it to use "Auto login". It would still require authentication though. Figure 4.3H shows the Wallet with various certificates and an overview of the Trusted Certificates' parameters. Next, Oracle Net Configuration is opened and then Profiles and to the SSL tab where Credentials Configuration for both Server and Client are chosen. After this is done, the Network Configuration is saved and the Listers are reloaded and changes visible in listener.ora, sqlnet.ora and tnsname.ora files. Figure 4.3I reveals Oracle Net Configuration showing the setting up of SSL with Oracle advanced Security option.

[5]https://www.openssl.org/

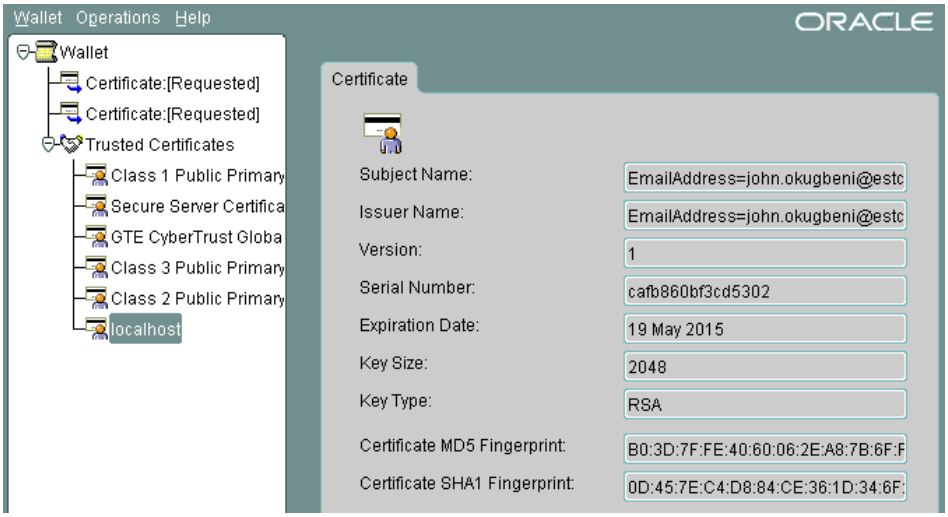


Figure 4.3H. Wallet with all its certificates as well as a clear view of the Trusted Certificates' parameters.

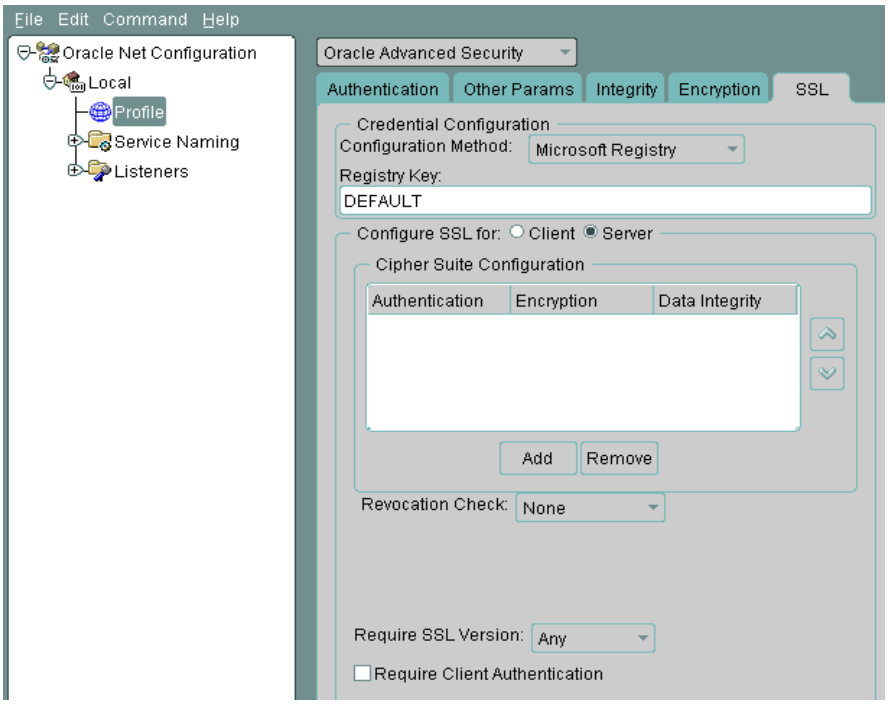


Figure 4.3I. Oracle Net Configuration showing the setting up of SSL with Oracle advanced Security option.

- DATA ENCRYPTION (DATA AT REST)

A master key is created for the Wallet that will enable open at anything when provided. This master key is then entered into the database with the "alter" command. Next, create a tablespace in the database and location of the Wallet's credentials, the size and the encryption algorithm type with the "using" command. After that a check to see if went successfully is done as seen in Figure 4.3J showing the list of tablespaces that are encrypted. Encryption of Columns is done also since whole database encryption is not supported.

```
SQL> desc v$encrypted_tablespaces
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------

 TS#                                                NUMBER
 ENCRYPTIONALG                                      VARCHAR2(7)
 ENCRYPTEDTS                                        VARCHAR2(3)
```

Figure 4.3J. List of encrypted tablespaces.

- SECURITY UPDATES INSTALLATIONS

Installation of updates, which are critical to Oracle database are always issued when there are any discovered vulnerabilities. Critical Patch Updates[6], as they are called, Security Alerts and Third Party Bulletins usually gives information about security vulnerabilities and if there are available fixes. This is available mostly for Oracle products' customers who are having contract that can grab these fixes when out.

- OS AUTHENTICATION BYPASS

This is when the Authentication to Oracle database is done on the Operating System level. Bypassing it will give straight access to the database, a matter of creating a new user in Windows, adding the user to a group, adding the user to OS database operators and administrator, and finally addingos_authen_prefix= OPS$ to the user's parameters and rebounce

[6]http://www.oracle.com/technetwork/topics/security/alerts-086861.html

the database. Figure 4.3K illustrates the sign-in process when the OS Authentication is bypassed. The database is authenticated using password credentials. Default settings will allow authentication simply by using the default *sqlplus /*.

```
C:\Windows\system32>sqlplus /

SQL*Plus: Release 11.2.0.1.0 Production on Sun May 18 14:38:38 2014

Copyright (c) 1982, 2010, Oracle.  All rights reserved.

ERROR:
ORA-01017: invalid username/password; logon denied


Enter user-name: jon
Enter password:

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL>
```

Figure 4.3K.The signing process when the OS Authentication is bypassed. The database is authenticated using password credentials.

When a remote request is made to the node on the network, it contacts the Oracle*Net which decides if to trust it and map it to a database or not. Default setting of allowing remote connect to Oracle is reconfigured to require authentication. Figure 4.3L shows the system configuration with "*remote_os_authent*" set to "*FALSE*", thus not allowing authentication request to the database from remote location.

```
SQL> show parameter remote_os_authent;
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
remote_os_authent                    boolean     FALSE
SQL> show parameter os;
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_lost_write_protect                string      NONE
diagnostic_dest                      string      C:\APP\SE
optimizer_index_cost_adj             integer     100
os_authent_prefix                    string      OPS$
os_roles                             boolean     FALSE
remote_os_authent                    boolean     FALSE
remote_os_roles                      boolean     FALSE
timed_os_statistics                  integer     0
```

Figure 4.3L. System configuration showing "*remote_os_authent*" set to "*FALSE*", this will not allow authentication request to the database from remote location.

4.4 ActiveMQ

- CHANGE DEFAULT PORTS

ActiveMQ allows default ports to be changed by assigning a new port number to the *transportConnector*setting in the *activemq.xml* configuration file, save it and restart the broker. Figure 4.4A shows ActiveMQ using a non-default port at start-up indicated by the red rectangle

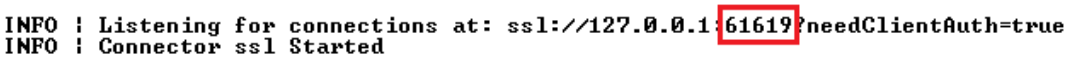


Figure 4.4A. Change to the connection port that ActiveMQ uses showing SSL connection running on a non-default port.

- SSL CONFIGURATION

Create a *Keystore* and *Truststore* to key the needed certificates. Generate a self-signed certificate for both the broker and the server; then export or copy them to the */conf/* folder and instruct the *transportconnector* in */activemq.xml* file to use *ssl* and *https* while requesting the client to authenticate. Figure 4.4B shows the red rectangles with *SSL* and *HTTPS* connections made successfully with authentication set to true. The transport connector for *HTTPS* runs on port 8443, while the *ssl* runs on port 61619 which it uses for web console.

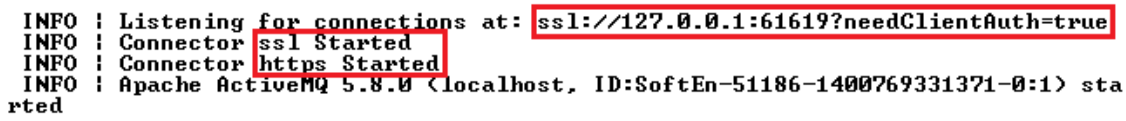


Figure 4.4B. SSL and HTTPS connections started successfully with clients required to authenticate before using.

- AUTHENTICATION AND AUTHORIZATION (SECURITY REALMS)

Authentication for client is configured using the BasicAuthenticator with HashLoginService while while encrypting the passwordin the configuration file with StandardPBEStringEncryptor. Figure 4.4C shows ActiveMQRealm requesting a client to present credentials for it to be authenticated before access the webconsole running on the localhost server on port 8161.

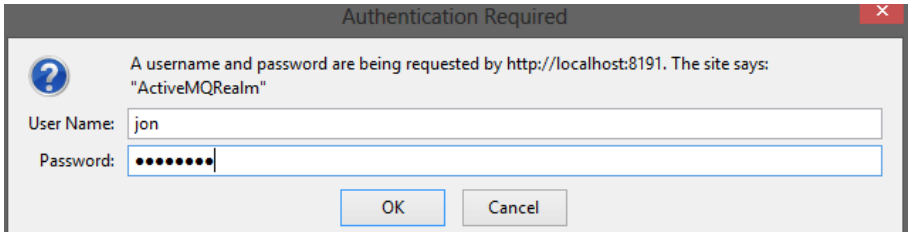


Figure 4.4C. A client being presented with to present ActiveMQRealm demanding credentials for it to be authenticated before access to the webconsole running on the localhost server on port 8161.

- INSTALL SECURITY UPDATES

Security updates for ActiveMQ are done by installing only the core *.jar* components files in order to keep configuration settings intact. While for internal security updates, the *Keystore* and *Truststore* to be updated with the newer keys if the old ones have expired or there issome security breach somewhere[18].

4.5 Jetty Web Server

- CHANGE THE DEFAULT CONNECTION PORT

The default connection port Jetty uses for connect can be changed in any of these two ways: When starting Jetty and from the configuration file. Default port Jetty runs on is port 8080 and has dedicated port for configuration of SSL 443. Figure 4.5A shows Jetty connection running on port 8181with the help of a *jetty.port* module when starting Jetty server.

```
C:\jetty-9.1.3.v>java -jar start.jar jetty.port=8181
2014-05-20 13:29:00.516:INFO::main: Logging initialized @673ms
2014-05-20 13:29:00.970:INFO:oejs.Server:main: jetty-9.1.3.v20140225
2014-05-20 13:29:01.007:INFO:oejdp.ScanningAppProvider:main: Deployment monitor
[file:/C:/jetty-9.1.3.v/webapps/] at interval 1
2014-05-20 13:29:01.044:INFO:oejs.ServerConnector:main: Started ServerConnector@
55b3a564{HTTP/1.1}{0.0.0.0:8181}
2014-05-20 13:29:01.046:INFO:oejs.Server:main: Started @1206ms
```

Figure 4.5A. Changing the default port for Jetty connection using the *jetty.port* module when starting Jetty server.

- SSL CONFIGURATION

Default Jetty connects to the browser using the default HTTP protocol, which is old and not secure through port 8080. SSL configuration changes connections from HTTP to HTTPS through some configuration file tweaking. The keys and certificates are generated and signed, then later converted to a format, which Jetty understands - PKCS12. In Jetty, the pkcs12 certificate is added to the Jetty's keystore in the ~\etc\ folder where other configurations files are also located. The XML files configured for *jetty-https.xml*, *jetty-spdy.xml* and *jetty-ssl*. Passwords for the KeystorePassword, TruststorePassword, KeyManagerPassword and paths to the KeystorePath andTrustStorePassword are provided in the configuration file. Figure 4.5B shows Chrome and Firefox browsers running the server with SSL (https) protocol and the bottom shows Firefox browser warning against using that connection to the certificate presented.

Figure 4.5B. Chrome and Firefox browsers running the Jetty server with SSL (https) protocol as well as the error message.

- AUTHENTICATION AND AUTHORIZATION (SECURITY REALM)

Authentication to the webapp had to be put in place to protect it from access by anyone without due permission. For this to happen, the *test* application's XML file in the ~\webapp\ folder has to be configured to make *securityHandler* use the HashLoginService and BASICAuthentication as the Authenticator. Other files like the *realm.property*, *jetty.xml* connected to the realms in the ~\etc\ folder were also configured to match the rest configurations already done for the Authentication and Authorization. Figure 4.5C below is the Realm1 requesting the user to enter the necessary credentials for Authentication to gain access to the webapp.

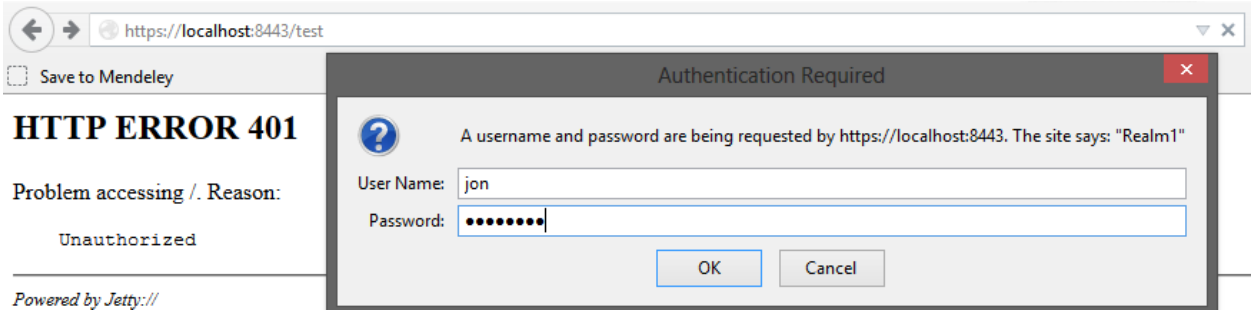


Figure 4.5C. Firefox browser running the a test webapp via a secure connection and the realm "Realm1" requesting for Authentication credentials for access to the webapp.

- FORM LIMIT FOR WEBAPPS AND SERVER

The maximum size of Form Content the webapps and the server could or should take could be adjusted in the configuration files. The ~\etc\ folder contains the *jetty.xml* file in which these webapps and server parameter are changed. The MaxFormContentSize and MaxFormKeys attributes must be set to the wanted amount to maintain the security of the webapps and server as shown in Figure 4.5D.

```
<!-- Form Limit for webapp     -->
<Set name="maxFormContentSize">20000</Set>
<Set name="maxFormKeys">200</Set>

<!-- Form limit configuration for the server  -->
<Call name="setAttribute">
  <Arg>org.eclipse.jetty.server.Request.maxFormContentSize</Arg>
  <Arg>10000</Arg>
 </Call>
<Call name="setAttribute">
  <Arg>org.eclipse.jetty.server.Request.maxFormKeys</Arg>
  <Arg>200</Arg>
 </Call>
</Configure>
```

Figure 4.5D Form limits for *webapps* and server shows the attributes for *MaxFormContentSize* and *MaxFormKeys* changed according to desired security of both.

5 Test and verify the solution on MCS test environment

Crowd – Crowd server ran well during the test with minimal or no errors. I would still recommend a longer test period to verify that all implemented solutions would stand the test of time.

Mongodb- Mongodb has free and enterprise versions. The free version does not support SSL out of the box but could be configured to work with it by building Mongodb from source code with the help of some third party tools, as it was done in this project. Enterprise version comes with SSL support. I would highly recommend buying the enterprise version because of the reliable support it has and all other features bundled with it. During the testing period for SSL implementation, there was loss of communication due to a "could not negotiate SSL connection: EOF detected" after running the server for a while. I would recommend a longer test to identify the cause of this and find a fix.

Oracle – Oracle database ran well with the solutions implemented. Although the implemented solutions being tested and verified are sufficient to secure the system, further testing would be recommended to ensure stability and reliability for the solutions. Also a good recommendation would be to purchase an enterprise edition due to the security support it offers.

ActiveMQ–The test and verification on ActiveMQ was fine. It ran without errors for the short test period, longer test period is recommended. A stronger authenticator should be used as well.

Jetty web server - For testing purpose Basic authenticator was used, it would be better to use a stronger authenticator in a production environment. I would also recommend increasing the form limit for test purposes to simulate an event of DOS attacks. That should verify its threshold and get you prepared for a worst case scenario.

Conclusion

The aim of this work was to provide a better security solution for ESTCube-1 Mission Control System. Currently, none of the components that were tested are running on default setting configuration any longer, thereby making them more secure than they were. In long run, the methods and solutions applied in this thesis ensure the security of the satellite operations.

Based on the components, the vital results are as follows:

- Connections between Crowd server and client are encrypted using SSL encryption while running on a non-default port; access to vital resources has been restrained by enforcing Authentication and Authorization; and finally, it was made sure that the technologies used are up-to-date security wise.
- Mongodb server and clients were configured to run behind Firewall; the default connection port was changed; the server was restricted to accept communication from certain ip addresses, by *binding*; and Authentication and Authorization were enabled in the database; SSL was configured to secure the connection as well.
- To initiate the security in Oracle database, the default passwords were changed and inactive/unused accounts were closed; password parameters were changed from their default limits; network communication was encrypted using SSL certificates stored in the Oracle Wallet; OS authentication bypass was disabled and database authentication enabled.
- The ActiveMQ connection between client and broker was efficiently secured using SSL certificates; server connection port for listening and the port for client's web console was changed; as well as Authentication and Authorization for client was enabled.
- Jetty web server is now running on a non-default port; Jetty was configured to use SSL protocol to secure connection between server and webapp; security realm for Authentication and Authorization was configured; finally, Form Limit for both the server and webapp to minimize any chance of Denial of Service attacks was set.

Future recommendations

The work was done in MCS test environment for a limited amount of time and achieved these results. I would recommend a longer period for Testing and Verification to ensure that there are

less errors. When the results from the longer Testing and Verification proves to be with minimal errors, then the same security solutions should be implemented in the live environment of ESTCube-1 Mission Control System.

References

[1] B. S. Hermansen, “A Satellite Mission Control System,” 2006.

[2] B. Snyder, D. Bosanac, and R. Davies, *ActiveMQ in Action*. Manning Publications Co., 2011.

[3] Jetty, “Jetty: The Definitive Reference.” [Online]. Available: http://www.eclipse.org/jetty/documentation/current/index.html.

[4] Eelco Plugge, Peter Membrey, and Tim Hawkins, “The Definitive Guide to MongoDB.” Apress, 2010.

[5] L. Okman, N. Gal-Oz, Y. Gonen, E. Gudes, and J. Abramov, “Security Issues in NoSQL Databases,” *2011IEEE 10th Int. Conf. Trust. Secur. Priv. Comput. Commun.*, pp. 541–547, Nov. 2011.

[6] M. D. Project, “MongoDB Security Guide,” 2014.

[7] R. Kothuri, A. Godfrind, and E. Beinat, *Pro Oracle Spatial for Oracle Database 11g*. Apress, 2007.

[8] R. Management and R. Series, “Security , Audit and Control Features Oracle Database.”

[9] A. Neagu, *Oracle 11g Anti-hacker ’ s Cookbook*. Packt Publishing Ltd., 2012.

[10] R. Ben Natan and P. Finnigan, *HOWTO Secure and Audit Oracle 10g and 11g*. Auerbach Publications, 2009.

[11] C. Five, “Threat Advisory : Atlassian Crowd,” no. June, pp. 1–12, 2013.

[12] Gracion software, “What is LDAP.” .

[13] TheCarlHall, “LDAP Authentication & Authorization Dissected and Digested.” .

[14] Atlassian’s Crowd, “Changing the Port that Crowd uses.” .

[15] Atlassian’s Crowd, “Integrating Crowd with Spring Security.” 2013.

[16] “ActiveMQ Security Guide,” 2012. [Online]. Available: http://fusesource.com/docs/esbent/7.1/amq_security/amq_security.html#Auth. [Accessed: 21-Apr-2014].

[17] oracle-base.com, “Basic Security Measures for Oracle.” [Online]. Available: http://www.oracle-base.com/articles/misc/basic-security-measures-for-oracle.php.

[18] The OpenSSL Project, “Heartbleed Bug.” [Online]. Available: http://heartbleed.com/. [Accessed: 22-Apr-2014].

[19] Webtide, “Why Choose Jetty? :: Jetty vs. Tomcat: A Comparative Analysis.” [Online]. Available: https://www.webtide.com/choose/jetty.jsp. [Accessed: 22-Apr-2014].

[20] “JAAS.” [Online]. Available: http://docs.codehaus.org/display/JETTY/JAAS.

[21] “10gen-MongoDB_Operations_Best_Practices.pdf.” [Online]. Available: http://info.mongodb.com/rs/mongodb/images/10gen-MongoDB_Operations_Best_Practices.pdf. [Accessed: 23-Apr-2014].

[22] A. M. W. Paper, “MongoDB Operations Best Practices,” no. April, 2014.

Appendix

I. Security solutions for ActiveMQ

ActiveMQ	Pros	Cons
Authentication	Simplicity, customizable, plug-in support, good for testing environment, comes default, LDAP support, fastest security solution	Password stored in plain-text, not safe for large systems or production environment,
Authorization	Easy-of-use configuration file, JAAS support, LDAP authorization support, message-level authorization safer than operation-level authorization, independent of transport protocol ()	Configuration files can be easily accessed
LDAP authentication	SSL support, secure, flexible, ease-of-use, plug-ins support, also does authorization	Credential can be accessed from directory
Build custom security plug-ins	Flexible, secure, might be cost friendly, encryption can be added	Can be complex depending on the code, time consuming
Certificate-based security	Commercial CA are trusted, secure, versatile, fast Private CA is cheaper or free, OpenSSL support, no PKI needed, flexible	Commercial CA are costly, 4 years life span Private CA cannot be trusted by other application/OS, limited to max 1 year use, may be slower, use low hash and cipher technology, not trusted by all browsers, no support for advanced PKI function

II. Security solutions for Jetty Web Server

Jetty Web Server	Pros	Cons
SSL configuration	Trust worthy, integrity, confidentiality, encrypts data, authenticates target's identity	SSL certificate expires after some time, requires payment, installation difficulties, performance issues, absolute security not guaranteed (OpenSSL's "Heartbleed bug" [18]
Authorization and authorization	Multiple plug-ins support, encrypts data, flexible, safe, secure when great plug-ins are implemented, small memory footprint, stability, scalability, standard and community support [19] LPDAP support	Slow page load time, pressure on server,

Jetty Web Server	Pros	Cons
Webapp form limit configuration	Customization form, flexible, Prevent DOS attacks	Use vast amount of memory and CPU, DOS attacks point if not limit not set
Secure password obfuscation	Protects password from viewing, confidentiality, can be easily copied to another location when needed	Result is irreversible, changing algorithm is a problem when changes are to be carried out
JAAS configuration	Flexibility, LDAP support [20] , secure	Credential can be stored locally,
Spnego configuration	Supports Internet Explorer, Firefox, Chrome browsers, single password, secure, LDAP support, flexibility [3]	No remote connection access, works on MS supported browsers only, requires Active Directory, requires vast amount of knowledge to configure and test, setup can easily fail,

III. Security solutions for Mongodb

Mongodb	Pros	Cons
SSL configuration	Secure connections, encrypts data, can use any valid certificate, supports many programming language drivers, [21]	Does not ship with SSL, SSL certificate expires after some time, requires payment, installation difficulties, performance issues, absolute security not guaranteed[18]
Access control	Restrict unauthorized access, control over the database	No default authentication and authorization in place,
Kerberos authentication for Mongodb	Industry standard-good, Windows and Linux support, support for many language drivers, secure,	No support for older versions,
Network security	Secure communication, same procedures in Windows and Linux, filter connection to database, REST interface not secure-disabled [6]	Not straight-forward configuration procedures, no authentication for REST interface ,
Data encryption	Sensitive data encrypted, support for 3rd party encryption solutions,	Not whole database can be encrypted, costly,
Audit system activities	Many viewing options, gives knowledge of all operations and activities [22]	No syslog support for Windows, Mongodb Enterprise only,

IV. Security solutions for Oracle database

Oracle database	Pros	Cons
Enhance security settings	Safe practice, initial security in place,	Might be bad for some components that needs other components to function fully
Security updates installation	Makes both the database and OS secure,	Some fixes are slow to happen,
Data encryption	Many encryptions supported, data is secure,	Time consuming, only available for some editions, not a lot of options available, only file-level encryption available,
OS authentication bypass	Fast, time efficient,	Not safe for authentication, easy to get access to,
Security configuration enhancement	Safer database, control over activities, remove default settings, better password usage	Might be frustrating for users,
Network security	Secure the port for connection, control the listener and tnsname configuration files,	Access to the configuration files can be easy,

V. Security solutions for Crowd.

Crowd	Pros	Cons
Authentication and authorization	LDAP support, not too complicated,	Stored credentials locally,
Change Crowd's ports	Makes alternative ports unknown, gives protection,	
Spring security configuration	Customizable, secure	Can work over unprotected network
Change all default passwords	Safer	Default passwords can be guessed easily
Application updates installation	Addresses old security issues, easy to apply, increase performance	Might make changes in configuration

VI. License

Non-exclusive licence to reproduce thesis and make thesis public

I, **Ejiroghene John Okugbeni,**

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:
   1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and
   1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

of my thesis

Security Implementation of Mission Control System for ESTCube-1 Satellite,

supervised by Urmas Kvell,

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, **28.05.2014**