Enterprise JavaBeans (EJB) security

Nasser M. Abbasi

sometime in 2000 Compiled on September 8, 2023 at 12:12am

Contents

1 Introduction

This report describes the tests I made and the process of testing performed for the EJB security component of the NetDynamics server.

NetDynamics Application Server. 5.0.0.22 SP2. For Windows NT 4.0 + ND patch 40-REFRESH, and Solaris 5.6 + ND patch 40-REFRESH.

2 Overview of security testing

The container is responsible for enforcing the security rules. It will enforce that the EJB security methods in a bean at run-time are allowed to be called by the caller (bean client). This is done by checking for the security role and the principle of the context that the call is made under.

The principle is the name assigned to the SECUIRTY_PRINCIPLE by the caller.

The role is the name of the role (or group) that the principle belongs to.

The container will check that the bean method is allowed to be called by the principle making the call, by checking that the SECURITY_PRINCIPLE of the caller belongs to a role that the bean methods belongs to.

Roles are assigned to methods at design time of the bean, using the ND studio. The container will read this information from the deployment serialized file when it first loads the bean class. The mapping of the principle name to a role is already known to the server, this was done earlier by loading the file called ejbUsers.txt into the server.

A stateful session bean context is fixed when the bean is created (when the ejbCreate() method is called). It is not allowed to change the context of the stateful session bean after it was created.

Figure 1 illustrates some of the above discussion.

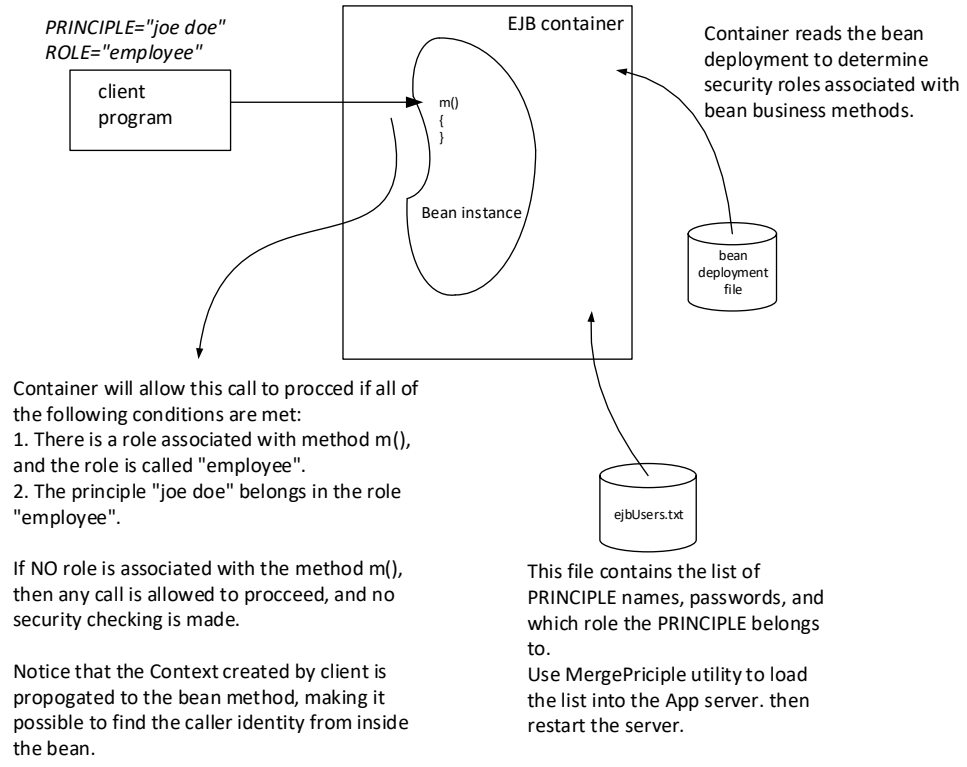


Figure 1: Overview of EJB security concepts. ejb_sec_1.vsd

3 Web project dataflow

For web projects testing, the client will access the bean using a web server.

The client sends a URL string which contains the name of the web plugin, and the name of the page to invoke. The webserver will then invoke the plugin executable and pass it the information. The plugin will then communicate with the ND plugin server, the ND plugin server then passes the data to a CP worker, which creates the bean using information from the web server environment variables that was passed to it from the web plugin via the ND plugin server.

The web plugin knows where the ND plugin server is by looking up its IP address from a file called plugin.nd.

The ND plugin Server then returns the web page back to the client via the web plugin. Additional requests/responses will follow this pattern. see figure **??** for illustration.

The following is an example of URL send from the client to the application server to bring up a web page:

```
http://localhost/SCRIPTS/nd_CGI_50.exe/securityTestsWeb/Stateless3Page?^start=&^uniqueValue=9
```

Figure 2 illustrates this.

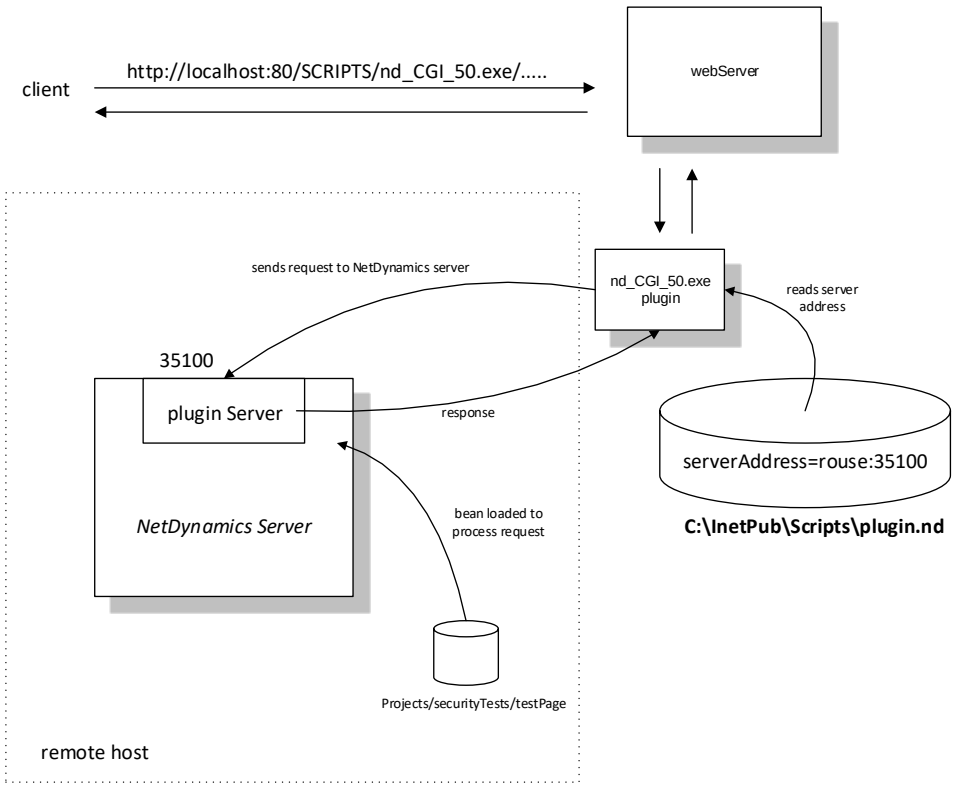


Figure 2: Accessing a bean using a web page. web_flow.vsd

4 Test setup and preparation

We first list the steps needed to run the tests from the standalone clients, then show more details on each step.

In General, these are the steps needed to run the tests

1. Install ND server.
2. Add roles and users to server.
3. Write the tests using ND studio. This steps includes adding roles to bean methods.
4. Use the console and set the needed parameters for the standalone tests.
5. Use the console to configure CP for servlets.
6. Update the license to increase users limit.
7. Update the test harness and run the tests
8. Run the servlet tests.
9. Run the web page project tests.
10. Investigate any failed tests. Report problems found.

5 EJB Notes

Netdynamics application server EJB service is an EJB container that meets 1.0 EJB specification. Some of the services mentioned in the specification are:

- Swap to/from passivation storage for non-active beans (for session beans). Currently, passivation storage in in-memory only.
- Persistence management for entity beans. (ND server currently does not implement entity beans).
- Provide Home object implementation creation.
- Provide JNDI service to allow clients to lookup home interface object implementation on the server.
- Creation and removal of bean instances.
- Support for transaction for business methods.
- Implementation of security related checking at run-time.
- generation of skeletons and stubs for remote method calls for the home interface and remote interface implementation (objects).

Diagram 3 Illustrates a high level view of relationship between the server, the container, and the EBJ instances.

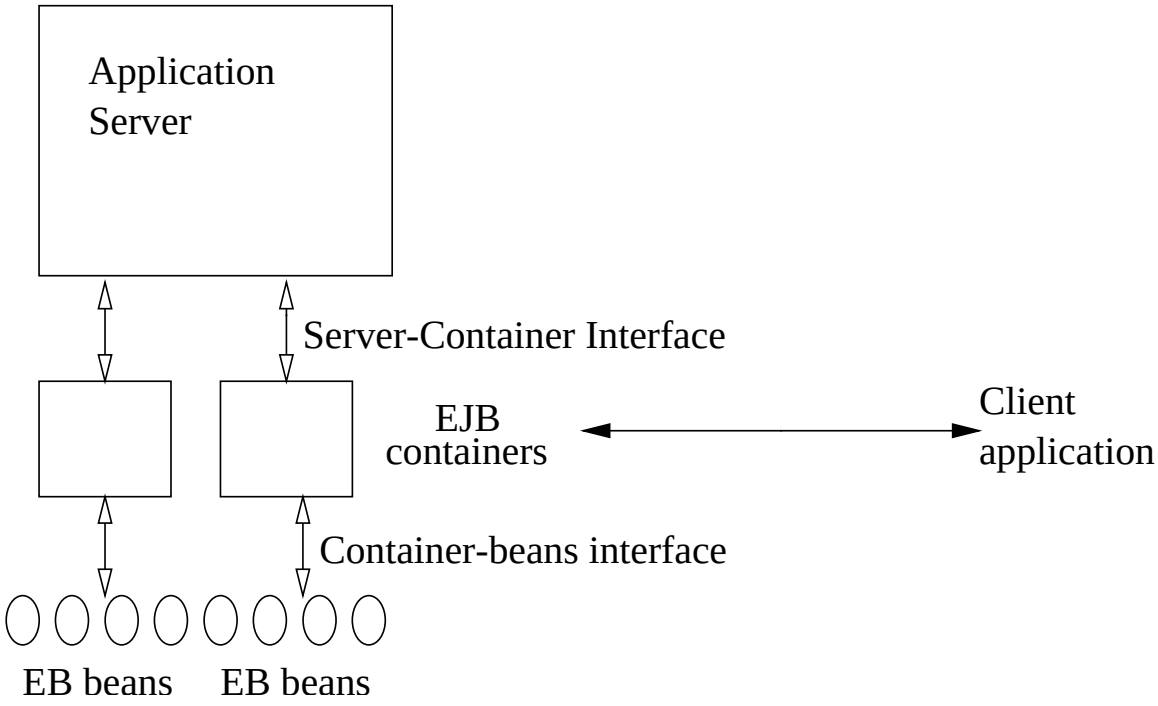


Figure 3: Server, Container and EJB instances. view1.fig

A client uses JNDI to locate a reference to the Home object. The Home object is the name given to implementation of the Home interface of the bean. Once the client has obtained this reference, it will call the create() method in the Home object. This causes the container to create the bean instance, and to allocate a remote interface object on the server side (called EJBObject), which it then return its reference back to the client. The client then uses the reference to the remote object to access the business methods inside the bean.

The client does not have a direct reference to the bean class instance itself, but to the implementation of its remote interface. The remote interface then calls the business methods in the bean on behave of the client.

So, the remote interface implementation, called EJBObject, is the one the calls the bean business methods. In addition, the EJBObjec communicate with the container to determine security and transaction context of the bean, since the container is the one who have knowledge of such information.

So, the EJBObject, the implementation of the remote interface, acts as a remote proxy for the bean instance.

The bean itself is not a network object, and can not be accessed directly over the network. Access to the bean must occur via its proxy, the EJBObject. Both the EJBObject, and the bean must reside in the same container.

Each bean instance, has a corresponding EJBObject instance. The EJBObject instance contains a reference to the bean instance, which the container initialized the EJBObject with when the bean was created.

The same Home interface can be used to create more instances of the bean (implying more instances of EJBObject).

Figure 4 Illustrates some of the above concepts.

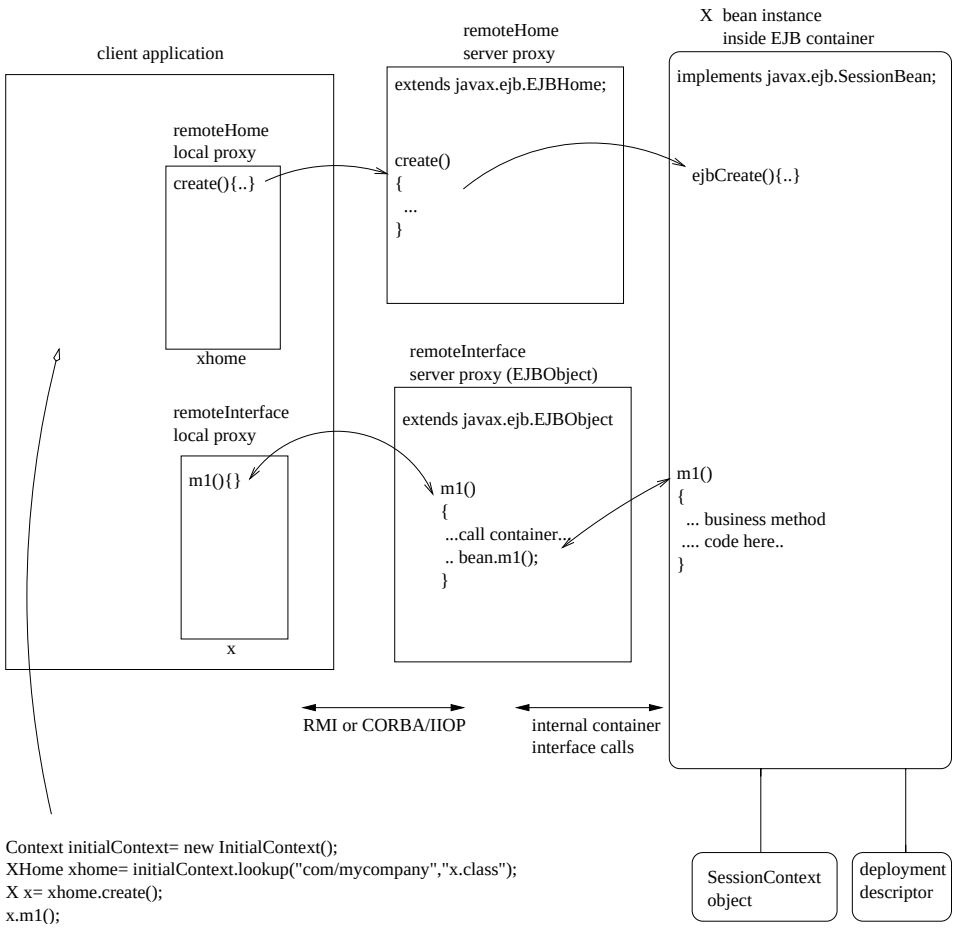


Figure 4: Container, Home, Remote, and bean relationship. view2.fig

The following are additional specific notes written down during EJB overview meeting given by ELain.

- Container or worker is itself a stateful type of instance.
- Container is a CORBA object.
- On the client side, a proxy is used to communicate with the server side implementation object.
- The client proxy contains Meta data that represent data that do not change (such as name of bean.) Call to obtain meta data does not need to be send to the server side.
- On the client side, a pool is created when the first call to initial context is made. This pool is used to store the local proxy objects. (Hash table indexed by container handler).

- Each bean instance is uniquely identified by its object ID, and session ID.
- All beans belonging to same session are located inside single container.
- Web projects are stateless.
- Standalone EJB sessions do not go throw CP.
- Web project EJB sessions are created by the CP.
- Need to worry about CP view of the session, and the EJB view of the session for web project.
- The studio automatically generates the EJBHome local proxy, EJBHome remote implementation, EJBObject local proxy, and EJBObject remote implementation classes.

5.1 EJB classes

We now look at the classes involved in implementation of EJB. First, we look at the classes used to implement the client side of the Home and Remote interfaces.

Figure 5 shows the relationship.

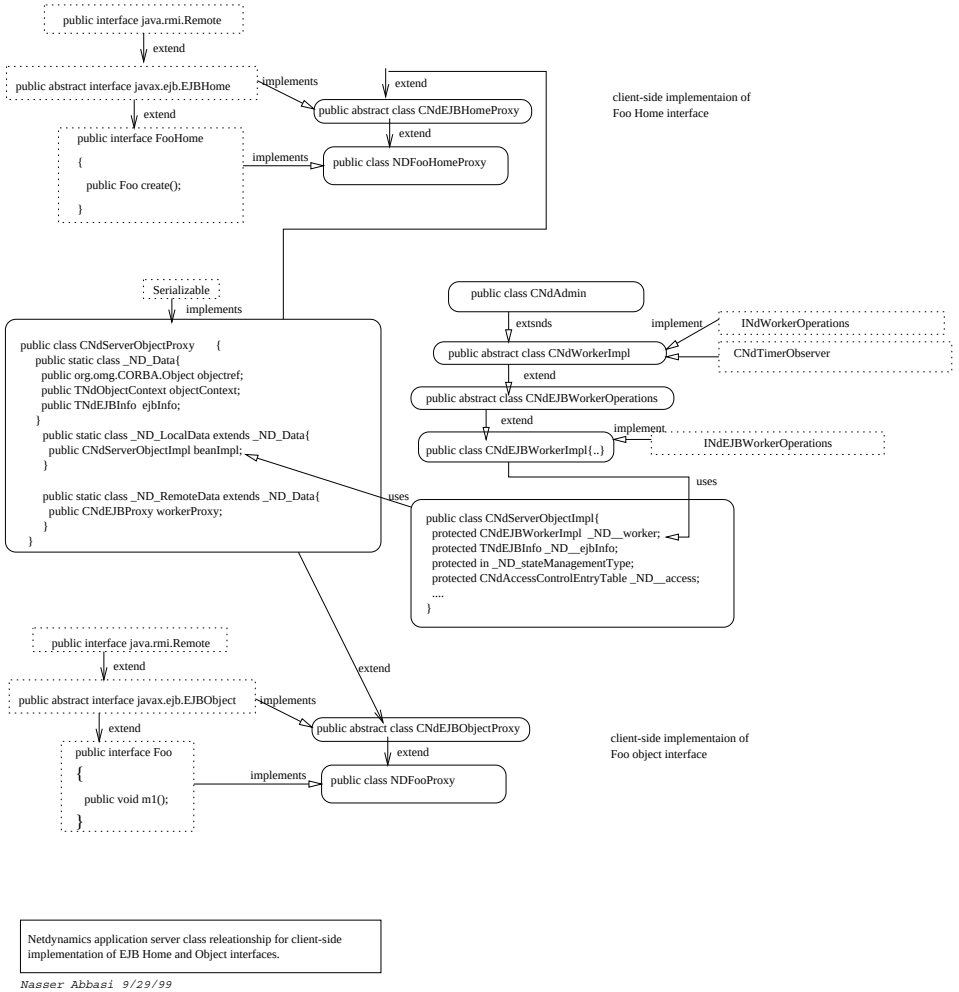


Figure 5: Client side EJB classes. ejb__client__side__classes.fig

5.2 J2EE servlet and JSP Functionality testing for NetDynamics 5.0 SP2

This is a report on how to setup and run the J2EE Compatibility Tests for servlet/JSP for ND 5.0 SP2.

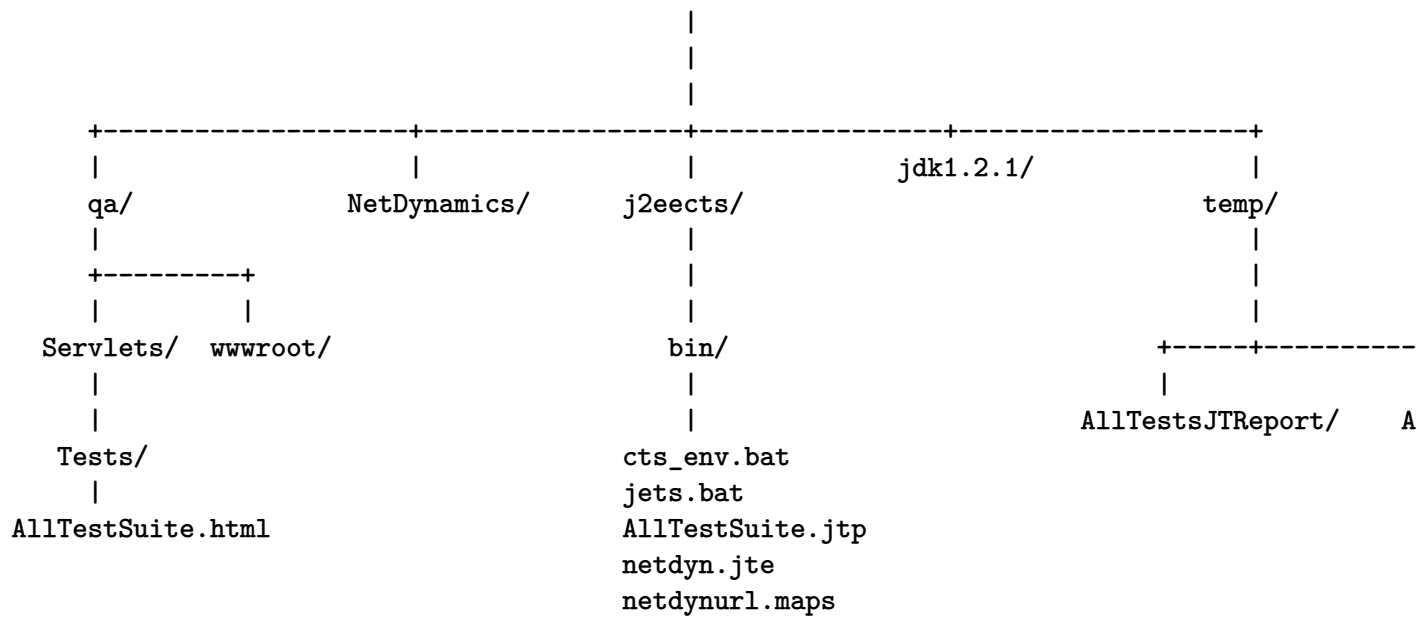
Directory structure

To make it easier to setup the test environment, I'll first show a diagram of how the directory structure should look like before running the tests.

Assume that you will install everything on the C: drive. (On UNIX, simply replace the C: in the diagram with the root where you will installed the software.

In This diagram, I'll also show the files that you need to edit/configure before running the tests.

```
some root (example C:\ or /)
                |
```



To set up the above directory, download the J2EE CTS and install it. Look for the file `j2eects.tar.gz` in the J2EE related web page. extract the content of the above tar file to create the /j2eects directory. Look at the CTS user guide for any special configuration before starting going further.

The latest qa directory shown above can be obtained from CVS (when it is on-line :)

Install JDK 1.2.1 (Make sure it is JDK 1.2.1 since that is the one j2EE CTS supports.

6 References

1. Sun 1.0 EJB specification document by Vlada Matena and Mark Hapner.
2. Developer's Guide to understanding EJB, by Chris Crenshaw, Nova labs.
3. Notes taken during Elain group meeting on EJB overview.
4. EJB white paper, by Anne Thomas, December 1988. Patricia Seybold Group.
5. Servlet Service Design report. By John Ellithorpe. April 14, 1999.
6. Release notes for NetDynamics 5.0 service pack 2. Servlets and JSP section.
7. Smita Srivastava report on EJB security test plan and setup.
8. Sun EJB 1.0 specification, Support for security section, pages 113,114.
9. Sun EJB 1.1 specification. chapter 15, Security management, pages 223-242.