ABSTRACT

Java Remote Method Invocation (RMI) is a mechanism that allows one to invoke a method on an object that exists in another address space. The other address space could be on the same machine or a different one. The RMI mechanism is basically an object-oriented RPC mechanism. One of the most common problems one encounters with RMI is a failure due to security constraints. Thus, we propose a secured RMI architecture. By additional technologies: Kerberos and Java Authentication and authorization Service, one can greatly improve Java Remote Method Invocation security. These enhancements are called Secure RMI. In our proposed secure RMI architecture we use Kerberos as additional technology for enhancements of security in Java RMI.

Keywords: Framework, Remote Procedure Call (RPC), Java Remote Method Invocation (RMI), Kerberos, Java Authentication and Authorization.

INTRODUCTION

In this paper, we are developing a secure RMI framework. RMI applications are often comprised two separate programs: a server and a client. A typical server application creates some remote objects, makes references to them accessible and waits for clients to invoke methods on these remote objects. A typical client application gets a remote reference to one or more remote objects in the server and then invokes methods on them. RMI provides the mechanism by which the server and client communicate and pass information back and forth. Such an application sometimes referred to as a distributed object application. Java RMI is object oriented approach of Remote Procedure Call. RMI functionality comes in the package java.rmi.*. RMI is developed by Sun Microsystems. It uses object serialization.

Our proposed system is designed to provide a secure RMI connection between client application and server. The preliminary objectives for this purpose are as follows:

- SSL (Secure Socket Layer) is used as a secure transport level.
- Kerberos is used for authentication in RMI application.

Kerberos is an authentication service designed for use in a distributed environment. It makes use of a trusted third party authentication service that enables client and server to establish authenticated communication. RMI uses a standard mechanism (employed in RPC systems) for communicating with remote objects: stubs and skeletons:

- RMI uses stub skeleton objects to provide the connection between client and remote objects.
- A stub is a proxy for a remote object which is responsible for forwarding method invocation from the client to the server where the actual remote object implementation resides.
- A client’s reference to a remote object, therefore, is actually a reference to a local stub. the client has a local copy of the stub object.
- A skeleton is a server side object which contains a method that dispatches calls to the actual remote object implementation.
• A remote object has an associated local skeleton object to dispatch remote calls to it.

RELATED WORK

Java RMI tool is very popular for call object methods located remotely. Much of work in the field of RMI security comes from Sun Microsystems. For security enhancement and performance improvement of RMI, we can use number of encoding technique during object serialization [1]. Different open source web applications were developed using Java RMI tool. Communication integrity an confidentiality can be put on transport layer [2]. The role of RMI in distributed environment is also significant. The Distributed Embedded Object Model (DEOM), a dependable, distribution-transparent RMI model is develop to support the execution of distributed embedded system applications [3]. And The Distributed Real-Time Specification for Java (DRTSJ) includes the details of the Real-time RMI infrastructure and the approach to support distributed real-time scheduling [4].

Whereas, The standard Java API is constructed in a way that it will allow Java Reflection to be used remotely while avoiding the limitations encountered when using a purely Java solution.[5].On the surface, the RMI system is just another RPC mechanism, much like CORBA and DCOM. But on closer look, RMI represents a very different evolutionary progression, one that results in a system that differs not just in detail but in the very set of assumptions made about the distributed systems in which it operates. These differences lead to differences in the programming model, capabilities, and way the mechanisms interact with the code that implements and built the distributed systems. [6,7,8]

ARCHITECTURE

RMI provide connection between client and server. Simple RMI architecture does not provide better security between client and server. By using advance technologies like Kerberos and Java Authentication and Authorization can provide secure connection. Secure communication can satisfy integrity, confidentiality and authentication. Our RMI architecture uses standard technologies like RMI, Java Authentication and Authorization service and Generic Security Service.

Step for creating and execution of RMI application:

• Design and implement the components of your distributed application.
  • Define the remote interface(s)
  • Implement the remote object(s)
  • Implement the client(s)
• Compile sources and generate stubs (and skeletons)
• Make required class network accessible
• Run the application

The RMI architecture consists of three layers as can be seen below [11]:
• RMI System Layer
• Remote Reference Layer
• Transport Layer
There are some of disadvantage are available in the Simple RMI approach. Existing RMI framework design for client server connection quite powerful and comprehensive but does not provide enough security. In RMI applications objects sends over network are not encrypted. This approach has a number of serious disadvantages:

- It is much slower than direct RMI connection.
- It is synchronous by nature and does not allow the client to have flexible callbacks.
- Developers need to have their own format of request and data.
- It is hard to pass big bunches of data. (eg: for graphical applications)
- After all, it is still unsecured hence settings cannot be allowed.

A better solution towards securing RMI, is by means of supporting authenticated and encrypted transport so that a network attacker cannot alter data of communication, can be achieved by RMI using Kerberos.

**PROPOSED ARCHITECTURE**

This RMI framework is using Kerberos authentication service, developed as a part of Project Athena at MIT. Kerberos provide centralized private key third party authentication in a distributed network. It have an authentication server (AS), AS provide a non-corruptible authentication credential (Ticket Granting Ticket TGT). And it have Ticket Granting Server (TGS).

Steps used in Kerberos for requesting any service:

- Obtain ticket granting ticket from AS (once per session).
- Obtain Service Granting Ticket from TGT (for each distinct service required)
- Client/Server exchange to obtain service (on every service request).
All the stages of our framework are elucidating below:

**STEP 1.0:** Remote object and its stub and skeleton are created.
**STEP 1.1:** Stub is registered with directory server.
**STEP 2.1:** User logs on to workstation and request service on the remote host (server).
**STEP 2.2:** Authentication server verifies user’s access rights in database creates Ticket Granting Ticket and session key. Result are encrypted using key derived from user’s password.
**STEP 3.1:** Workstation prompts user for password and uses password to decrypt, incoming message, then sends ticket and authenticator that contains user’s name, network address and time to TGS.
**STEP 3.2:** TGS decrypts ticket and authenticator verifies request, then creates ticket for request server.
**STEP 4.1:** Workstation sends ticket and authenticator to server.
**STEP 4.2:** Server verifies that ticket and authenticator match then grants to service.
**STEP 5.1:** Stub of remote object is looked up in the directory server.
**STEP 5.2:** If the stub of required service is already registered in the directory server then stub of that service is returned otherwise error message is displayed that service is not found.
**STEP 6.0:** Remote method is invoked on the remote stub by the RMI client assuming it remote object.
**STEP 6.1:** A connection is established with the skeleton; parameters are serialized and transmitted over the network.
**STEP 6.2:** Parameters are de-serialized and method is invoked on the actual object.
**STEP 6.3:** Method call is completed and result is returned to the skeleton.
**STEP 6.4:** Result is serialized and transmitted over the network to the stub and connection is closed.

**STEP 6.5:** Result is de-serialized and returned to the RMI Client.

---

**IMPLEMENTATION VIA EXAMPLE**

**STEP OF COMPILATION AND EXECUTION**

There are five steps for compiling and running the RMI application:

- Compiling the remote interface, client and server classes. Command is used for compiling the java files,
  Javac –d . FileName.java;
- Create class for stub and skeleton. The “rmic” tool is used to create stub and skeleton for remote object.
  rmic –d . ClassNameOfRemoteObject;
- Create an object of RMI server and register it with directory server. The “java.rmi.naming” class provide static method to register and lookup remote object with directory server.

In server side coding, for bind or register of remote method, two methods are used.

---

**Fig. (c): Flow diagram of general RMI application.**
a) public static void bind(String name, Remote object) throws BindingException;
b) public static void rebind(String name, Remote object);

In the client side for lookup one method is used
a) public static lookup(String name);

- Start directory server and run registration code. In case of RMI a tool name “rmiregistry” is provided with jdk that works as a directory server for RMI server and client.
- Create a class for RMI client and execute it.

CONCLUSION

In this paper, we have proposed a new Secure RMI framework with additional Technologies. Simple RMI framework is insecure for transferring data from one computer or host to remote host. But with the help of Secure RMI framework we can transfer important data in encrypted form. This proposed framework gives enough security for transferring data over Internet or Intranet. RMI is heavily used in business processes, therefore, security is more important factor in Java RMI.

REFERENCES
1. Hei Ning Helen Ma and Lan Yang “Improvement of Object Serialization in Java Remote Method Invocation” 2006
4. Luiz F. Bacellar and Bhargav P. Upender “A Dependable Distribution-Transparent Remote Method Invocation Model for Object-Oriented Distributed Embedded Computer Systems”
24. Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional, 1995