# Vulnerability documentation:

Java RMI uses a proxy-based architecture. In this architecture, a client interacts with a service through

a proxy, which is code downloaded from a directory and installed on the client’s machine. When critical systems are built using the proxy-based architecture, the security of these systems is a natural concern. The use of proxies introduces security vulnerabilities that need to be dealt with before people can rely on systems built using technologies such as Java RMI.

An attacker who controls the communication channels or the directory may compromise the confidentiality and integrity of the client and of the service.

The use of proxies causes several challenges in providing

security mechanisms for proxy-based systems. For example,

the goal to provide mutual authentication between

a client and a service is complicated by the presence of a

proxy. The client and the service communicate through the

proxy, which neither the client nor the service fully controls

and trusts. The proxy is generated by the service; thus

the client cannot trust the proxy more than it trusts the service,

even after the proxy has been authenticated. On the

other hand, the proxy is executed under the control of the

client; thus, the service cannot fully trusts the proxy either.

One approach that might seem plausible is to treat the proxy

as part of an insecure communication channel connecting

the client and the server, and to use standard authentication

techniques to provide mutual authentication between them.

**2. Vulnerability Analysis**

We now analyze the vulnerabilities of the proxy-based

architecture and derive three security goals. For concreteness,

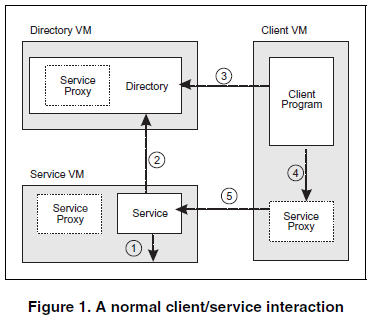
we use Java RMI in the presentation; however, the

same vulnerabilities also exist in other proxy-based distributed

systems such as Jini-based systems. The sequence

of interactions among the service, the client, and the proxy

is shown in Figure 1 and described below. In these interactions,

three virtual machines (VMs) are involved: the client VM, the service VM, and the directory VM.

1. The service makes a call to the RMI runtime to export

the service. In this process a proxy for the service is

created in the service VM.

2. The service makes a remote call to a directory (e.g.,

rmiregistry) to register its proxy. After this call, the directory

VM has the proxy object.

3. The client looks up the service and downloads the

proxy from the directory. After this step, the client VM

has the proxy.

4. Each time the client needs to use the service, it calls

the proxy locally as if it is the remote service.

5. The proxy communicates with the service to serve the

client’s request.

After a client has located and installed a proxy, the client

may reuse the proxy to make further requests by repeating steps 4 and 5 for as long as the proxy is considered valid by

the service.

**2.1. Vulnerabilities**

In our vulnerability analysis for proxy-based systems,

we consider three parties: the user, the service provider, and

the attacker. We assume that the user has full control over

the client VM and that the service provider has full control

over the service VM. An attacker, however, may control

the network and the directory VM. In other words, we

consider the directory as untrusted; a compromised directory

should not compromise the security of the client or the

service.

The risks to the client are:

**Client confidentiality** The client may transmit sensitive

information to the proxy when trying to use the service.

For example, if the service provides access to

the user’s investment accounts, the user wants to make

sure that his instruction is sent only to the service

provider, but not to anyone else.

**Client integrity** The user’s VM and local environment

may be damaged by the proxy.

Client confidentiality may be compromised by the attacker

in two ways. One, the attacker can eavesdrop on the communications

between the service and the proxy. Two, the attacker

may get the client to install and run a bogus proxy,

either by compromising the directory or by actively attacking

the communication channel between the directory VM

and the client VM or the channel between the directory VM

and the service VM. In this attack, client integrity may also

be compromised.

The risks to the service are:

**Service confidentiality** The service may send sensitive information

to the client through the proxy.

**Service integrity** The service needs to perform certain

functionalities on behalf of its client, and an attacker

who successfully impersonates a client may

result in damages.

Service confidentiality and integrity may be compromised

by an attacker who controls the communication between the

proxy and the service, by an attacker who can get the client

to accept a bogus proxy, and by an attacker who can impersonate

a client.

In this paper we do not consider risks related to availability,

as a denial-of-service attack can always be carried

out by an attacker who controls the communication channels

and/or the directory.

# References:

1. Ninghui Li; Mitchell, J.C.; Tong, D., "Securing Java RMI-based distributed applications," *Computer Security Applications Conference, 2004. 20th Annual* , vol., no., pp.262,271, 6-10 Dec. 2004  
   doi: 10.1109/CSAC.2004.34. URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1377233&isnumber=30059>
2. Second reference

# Keywords:

Java;client-server systems;message authentication;remote procedure calls;virtual machines;Java RMI-based distributed application;Java-based toolkit;distributed proxy-based computing;proxy-based architecture;security architecture;service register;signed authentication;Application software;Authentication;Communication system control;Computer architecture;Distributed computing;Information security;Java;Protocols;Virtual machining;Virtual manufacturing