Summary of the security threats in Java RMI technology:

|  |  |  |  |
| --- | --- | --- | --- |
| Threat | Why it is a threat | Impact | Difficulty Level |
| Use of proxy | This proxy is essentially some java code via which a client interacts with a service. | An attacker may control the communication channels and may compromise the confidentiality and integrity of the client and of the service. | Medium |
| Object Serialization | A generated Stub created on the server-side by calling a static export Object method on UnicastRemoteObject class is a serialized object; this eventually facilitates a Java method call to be transmitted over a network. | The serialized object is written into byte stream and if the byte stream is readable then the values of the normally inaccessible fields can be read. Furthermore, it may also be possible to modify or forge the preserved values with the intention that when the object is deserialized, the values are corrupted. | Basic |
| Lack of type compatibility verification in RMI protocol implementation. | RMI implementation does not verify the type compatibility of a deserialized object with the input argument of a target method call. The RMI server reads and instantiates object provided as an argument to the call using RMIClassLoader. Even if the concerning object (i.e. object to read) is of an unknown class, an attempt will still be made to fetch the class data from the Codebase URL provided by the client. | This characteristic of the RMI implementation alone creates a possibility for remote loading and execution of user provided Java code. | Advanced |