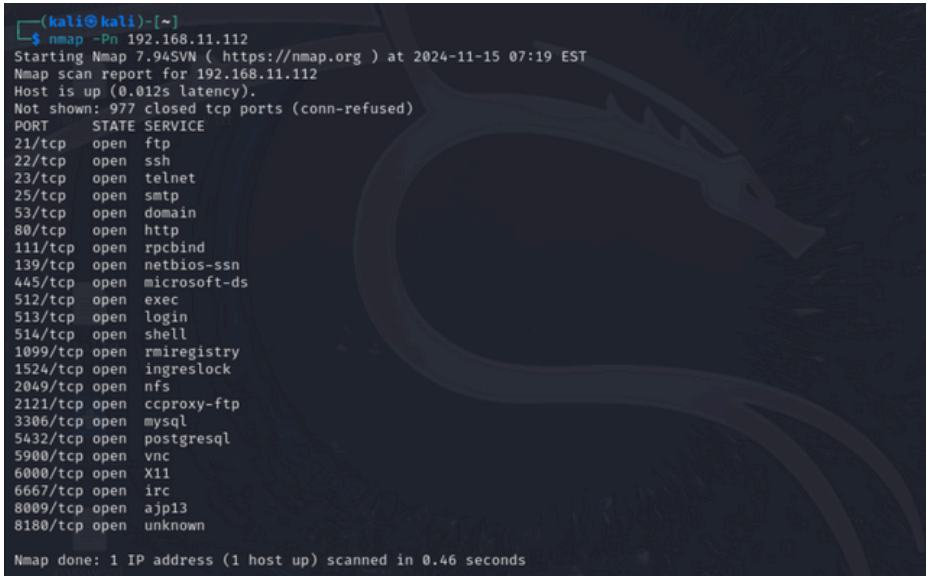
PROGETTO S7/L5

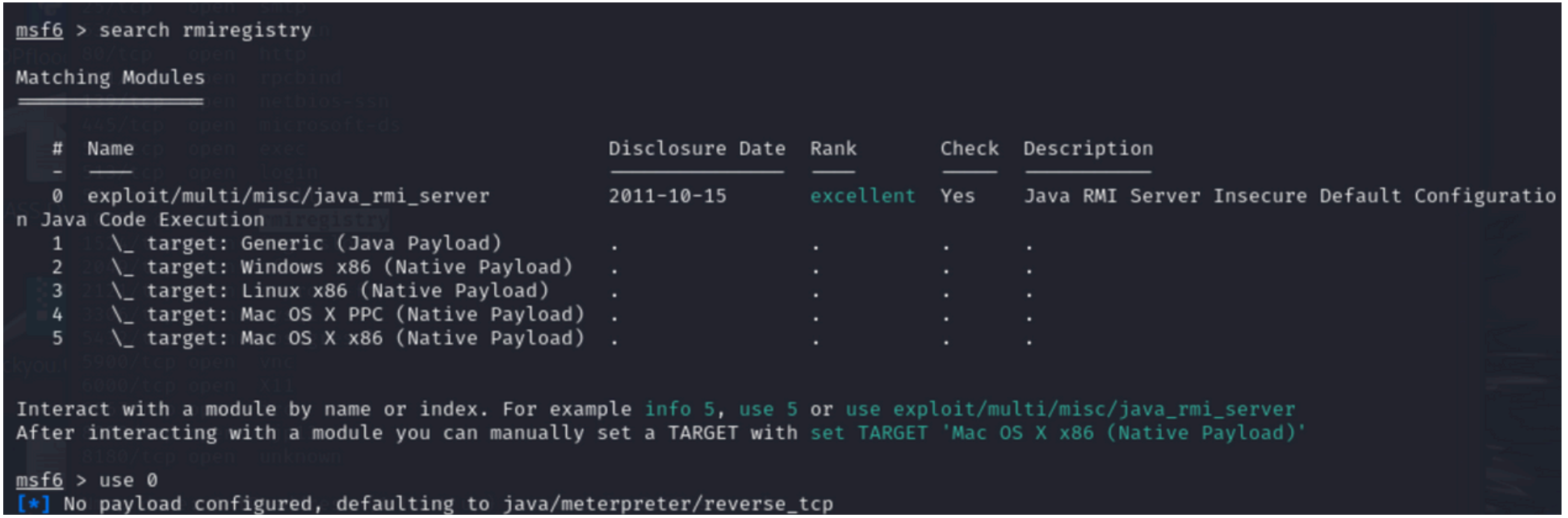
Configurazione ambiente virtuale

Per configurare l'ambiente virtuale è stato impostato nella macchina Kali l'indirizzo IP: 192.168.11.111 e in quella di Metasploitable l'indirizzo IP: 192.168.11.112, una volta messi in comunicazione siamo partiti con l'utilizzo di nmap:

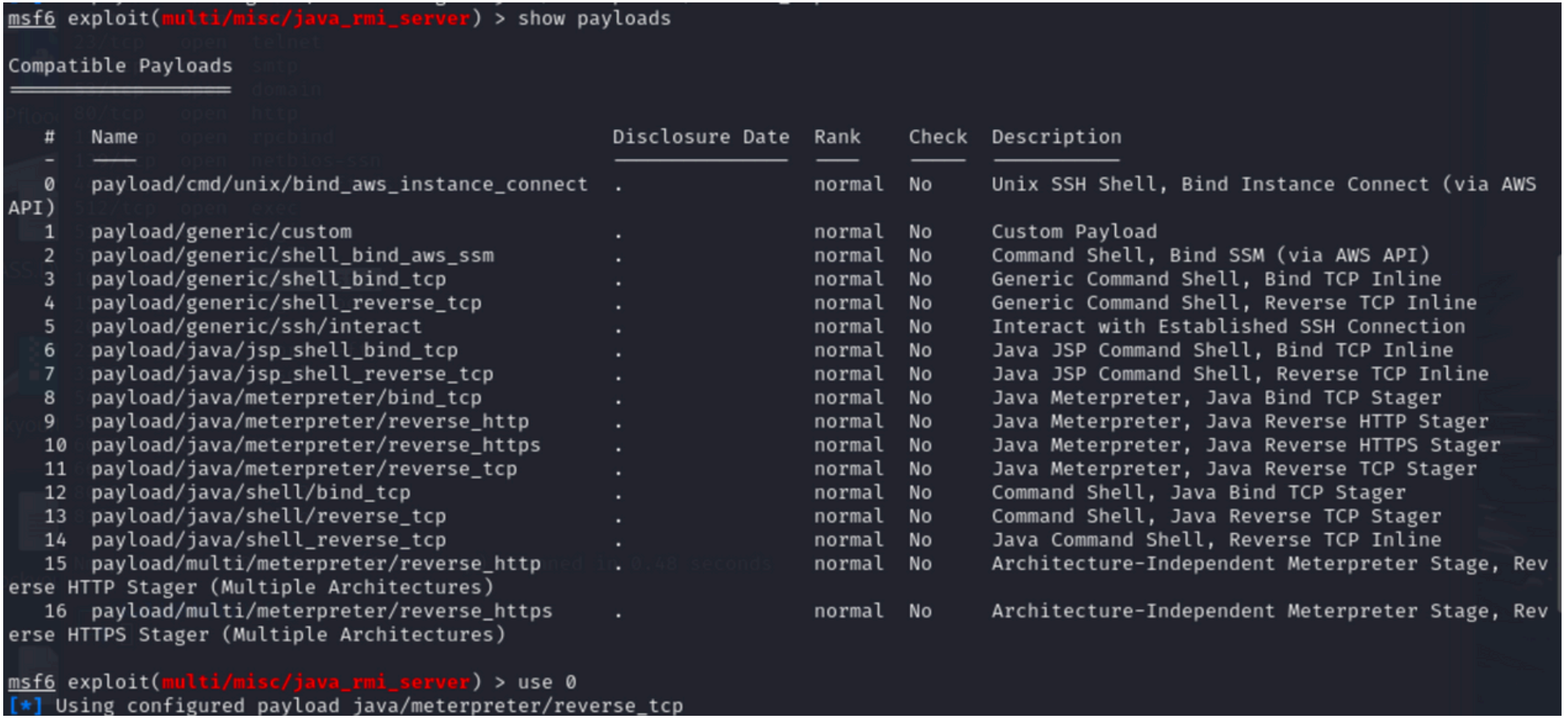


Nmap, strumento potente e versatile per la sicurezza delle reti e la gestione di sistemi. È ampiamente utilizzato per testare la sicurezza delle reti, identificare vulnerabilità e monitorare le configurazioni della rete. Con questo strumento abbiamo scansionato l'indirizzo IP target, della macchina di Metasploitable, tra le tante vulnerabilità prendiamo in considerazione il servizio della porta 1099 - Java RMI.

Per sfruttare questo servizio vulnerabile viene utilizzato **Metasploit**, framework di test di penetrazione, utilizzato per sviluppare e eseguire exploit contro sistemi remoti. È uno strumento molto potente e versatile, utilizzato per identificare e sfruttare vulnerabilità nei sistemi.

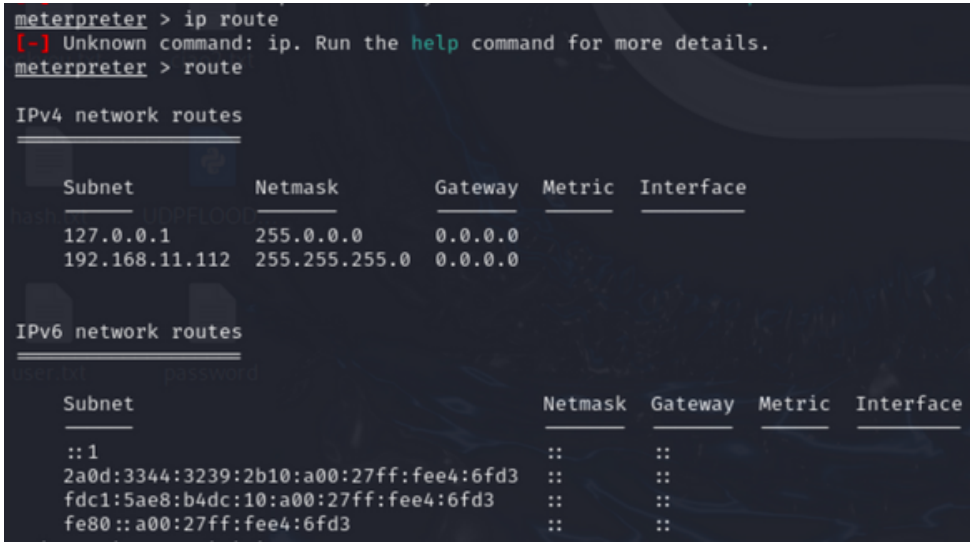


Prima di tutto si ricerca un **EXPLOIT** per il servizio in questione. Un exploit è un programma, un codice o una sequenza di comandi che sfrutta una vulnerabilità presente in un software, un sistema operativo o un'applicazione per ottenere risultati non autorizzati, a differenza del malware che è un codice malevolo che si va ad insinuare in un programma o codice vulnerabile.. Spesso, gli exploit sono proprio un mezzo per introdurre malware nei sistemi target.
Tra le varie funzionalità che hanno gli exploit sicuramente possiamo ottenere accesso a sistemi o reti protette, eseguire comandi o programmi non autorizzati su un sistema, rubare informazioni sensibili, stabilire un accesso persistente a un sistema compromesso.

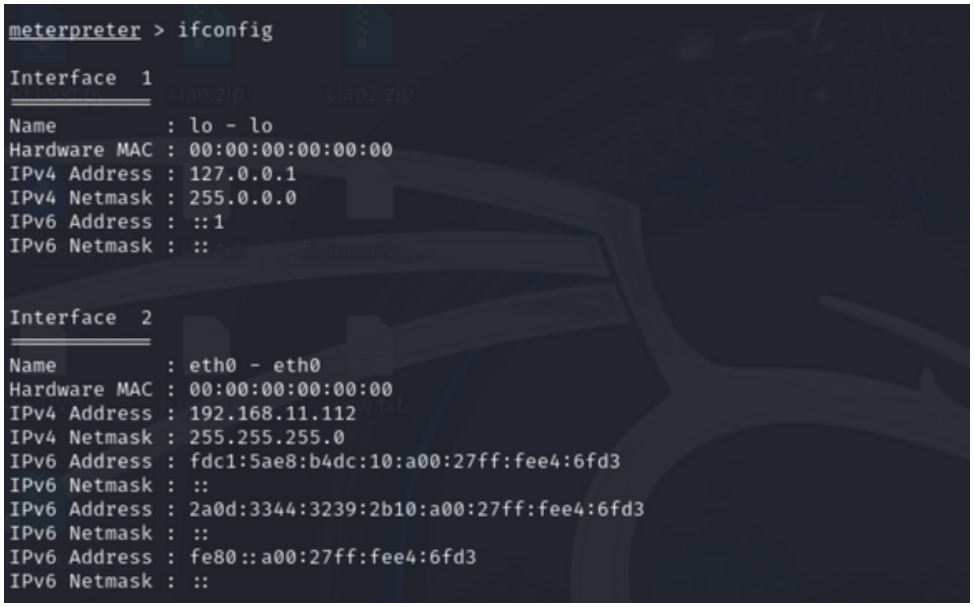


```
msf6 exploit(multi/misc/java_rmi_server) > show payloads

Compatible Payloads
===================

   #   Name                                              Disclosure Date  Rank    Check  Description
   -   ----                                              ---------------  ----    -----  -----------
   0   payload/cmd/unix/bind_aws_instance_connect        .                normal  No     Unix SSH Shell, Bind Instance Connect (via AWS
API)
   1   payload/generic/custom                            .                normal  No     Custom Payload
   2   payload/generic/shell_bind_aws_ssm                .                normal  No     Command Shell, Bind SSM (via AWS API)
   3   payload/generic/shell_bind_tcp                    .                normal  No     Generic Command Shell, Bind TCP Inline
   4   payload/generic/shell_reverse_tcp                 .                normal  No     Generic Command Shell, Reverse TCP Inline
   5   payload/generic/ssh/interact                      .                normal  No     Interact with Established SSH Connection
   6   payload/java/jsp_shell_bind_tcp                   .                normal  No     Java JSP Command Shell, Bind TCP Inline
   7   payload/java/jsp_shell_reverse_tcp                .                normal  No     Java JSP Command Shell, Reverse TCP Inline
   8   payload/java/meterpreter/bind_tcp                 .                normal  No     Java Meterpreter, Java Bind TCP Stager
   9   payload/java/meterpreter/reverse_http             .                normal  No     Java Meterpreter, Java Reverse HTTP Stager
   10  payload/java/meterpreter/reverse_https            .                normal  No     Java Meterpreter, Java Reverse HTTPS Stager
   11  payload/java/meterpreter/reverse_tcp              .                normal  No     Java Meterpreter, Java Reverse TCP Stager
   12  payload/java/shell/bind_tcp                       .                normal  No     Command Shell, Java Bind TCP Stager
   13  payload/java/shell/reverse_tcp                    .                normal  No     Command Shell, Java Reverse TCP Stager
   14  payload/java/shell_reverse_tcp                    .                normal  No     Java Command Shell, Reverse TCP Inline
   15  payload/multi/meterpreter/reverse_http            .                normal  No     Architecture-Independent Meterpreter Stage, Rev
erse HTTP Stager (Multiple Architectures)
   16  payload/multi/meterpreter/reverse_https           .                normal  No     Architecture-Independent Meterpreter Stage, Rev
erse HTTPS Stager (Multiple Architectures)

msf6 exploit(multi/misc/java_rmi_server) > use 0
[*] Using configured payload java/meterpreter/reverse_tcp
```

Una volta scelto l'exploit dobbiamo selezionare il **PAYLOAD** da utilizzare, che è una parte di un attacco informatico che viene consegnata a un sistema target, una volta che un exploit ha avuto successo. Quindi il payload è il codice o il programma che viene eseguito dopo che una vulnerabilità è stata sfruttata, e ha come obiettivo eseguire azioni dannose o non autorizzate sul sistema compromesso.



L’obbiettivo dell’esercitazione era quello di ottenere, la **configurazione di rete** della macchina vittima, che fornisce dettagli come gli indirizzi IP, le subnet mask e le informazioni sui gateway. e informazioni sulla **tabella di routing** del target, che contiene le informazioni necessarie per instradare i pacchetti di dati verso le destinazioni appropriate.



Conclusione

In questo esercizio, abbiamo utilizzato Metasploit per sfruttare una vulnerabilità nel servizio Java RMI sulla porta 1099 della macchina Metasploitable.
Attraverso l'ottenimento di una sessione Meterpreter, abbiamo avuto accesso alla macchina remota, consentendoci di raccogliere informazioni preziose sulla configurazione di rete e sulla tabella di routing.