**Relazione sull'Esercizio di Sfruttamento della Vulnerabilità Java RMI**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Obiettivo dell'esercizio**

L'obiettivo principale di questo esercizio è sfruttare una vulnerabilità presente nel servizio **Java RMI** esposto sulla **porta 1099** della macchina **Metasploitable** utilizzando **Metasploit**, al fine di ottenere una sessione remota **Meterpreter** sulla macchina vittima. Successivamente, è richiesto raccogliere due evidenze:

1. Configurazione di rete della macchina vittima.

2. Tabella di routing della macchina vittima.

**Requisiti**

* Configurazione delle macchine virtuali:
* Macchina attaccante (Kali Linux); IP = 192.168.11.111
* Macchina target (Metasploitable): IP = 192.168.11.112
* Modalità di connessione: entrambe le macchine configurate sulla stessa rete interna (rete privata), assicurando la comunicazione bidirezionale
* Influenza NAT/PAT

Quando due dispositivi devono comunicare in un contesto di test o attacco, è fondamentale che siano configurati all'interno della stessa rete locale privata per i seguenti motivi.

* Assenza di interferenze da NAT/PAT

NAT (Network Address Translation): Trasforma gli indirizzi IP privati in pubblici, complicando l'instradamento diretto dei pacchetti.

PAT (Port Address Translation: Effettua una mappatura di porte tra più dispositivi interni e un indirizzo pubblico. Questo può disturbare la ricezione di connessioni dirette richieste dai payload reverse shell, perché la macchina attaccante non può determinare con precisione quale porta/indirizzo corrisponde alla macchina vittima.

* Isolamento dell'ambiente

Configurare entrambe le macchine su una rete interna le protegge da accessi esterni e garantisce che le comunicazioni siano contenute, evitando effetti collaterali su altre reti.

**Exploit**

Un exploit è uno strumento o una procedura progettata per sfruttare una vulnerabilità di un sistema informatico, come errori di programmazione, configurazioni errate o meccanismi di sicurezza deboli.

* Descrizione dell'exploit utilizzato
* Modulo Metasploit

Il modulo utilizzato è `exploit/multi/misc/java\_rmi\_server`. Questo modulo sfrutta la vulnerabilità del protocollo Java RMI (Remote Method Invocation), che accetta input arbitrario non autenticato sulla porta 1099. La mancanza di validazione consente l'esecuzione di codice remoto.

* Payload utilizzato

Il payload configurato automaticamente è `java/meterpreter/reverse\_tcp`, che stabilisce una connessione inversa con la macchina attaccante per ottenere il controllo remoto.

* Dettaglio del processo

Il modulo invia una richiesta con un oggetto Java serializzato contenente il payload alla macchina vittima.

La vulnerabilità permette di caricare ed eseguire codice arbitrario.

Il payload, una volta eseguito, stabilisce una connessione TCP inversa verso la macchina attaccante.

**Protocollo Java RMI**

Il Remote Method Invocation (RMI) è un protocollo di comunicazione peer-to-peer progettato per consentire ad applicazioni Java distribuite di eseguire metodi su oggetti remoti come se fossero locali.

* Funzionamento
* Il client invia una richiesta al registro RMI per ottenere il riferimento all'oggetto remoto, il server ospita l'oggetto remoto ed esegue i metodi richiesti.
* Comunicazione p2p (peer to peer), è un modello di rete in cui due o più dispositivi (detti peer) condividono risorse e comunicano direttamente tra loro, senza la necessità di un server centrale che media o gestisce lo scambio di dati, ogni dispositivo è allo stesso tempo client e server; quindi, invia e riceve informazioni. Questo comportamento, nonostante le limitazioni del protocollo, aumenta la scalabilità in quanto, non essendoci un server centralizzato, l’aumento dei peer, non va a sovraccaricare i sistemi
* La comunicazione avviene tramite oggetti serializzati, che contengono i parametri di input e il risultato della funzione.

La serializzazione è il processo di conversione di oggetti Java in un formato binario che può essere trasmesso tramite rete o salvato su disco. La deserializzazione riconverte i dati in un oggetto.

* Vulnerabilità e problematiche
* Oggetti dannosi possono essere inviati al server, e durante la deserializzazione possono eseguire codice non autorizzato.
* Mancanza di autenticazione: di default, RMI non verifica l'identità dei client, consentendo l'accesso non autorizzato.
* Assenza di sanificazione dell'input: l'input inviato al server non viene controllato, aprendo la porta a payload dannosi.
* Compatibilità limitata: RMI funziona solo in ambienti Java, riducendo l'interoperabilità.

**Confronto Java RMI e FTP**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Architettura | Autenticazione | Comunicazione | Compatibilità | Sicurezza |
| Java RMI | Decentralizzata | Non richiesta | Diretta | Solo sistemi Java | Non criptato |
| FTP | Centralizzata | Richiesta | Mediata | Universale | Può essere criptato |

* Architettura

Java RMI è autonomo e non dipende da un’entità centrale, questo rende il sistema più saldo è più difficile da monitorare. FTP dipende da un server centrare, questo ne aumenta il controllo ma introduce il collo di bottiglia,

* Autenticazione

Di base Java RMI non richiede delle credenziali di autenticazione a differenza del protocollo FTP

* Comunicazione

La comunicazione dipende dall’architettura di base e dunque Java RMI prevede delle comunicazioni dirette tra i peer: Al contrario, con FTP tutti i dati sono mediati da un server centrale.

* Compatibilità

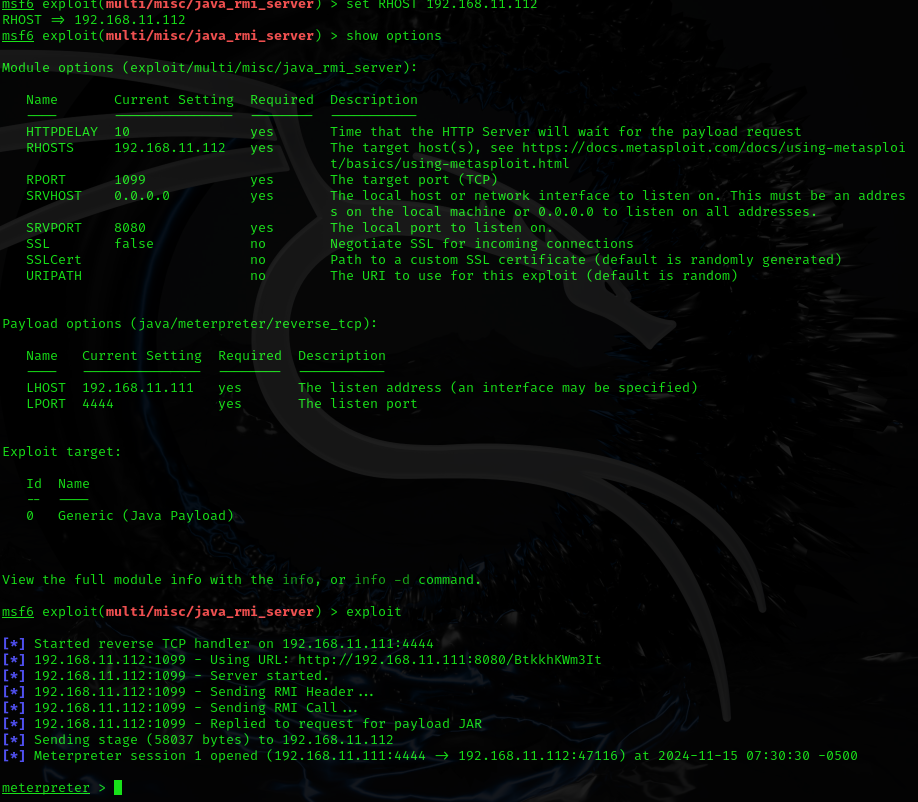
La compatibilità dei sistemi Java RMI è limitata ai sistemi Java, è un’imponente restrizione; a differenza di FTP che risulta essere universale

* Sicurezza

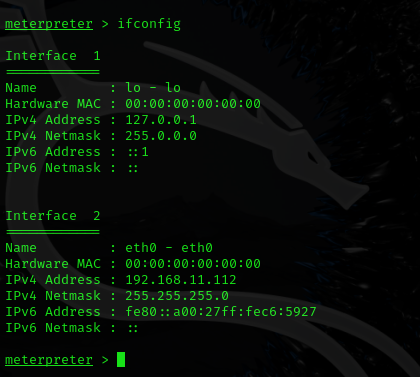
Java RMI risulta meno sicuro rispetto a FTP in quanto, oltre a non prevedere un’autenticazione, di base non è neanche un protocollo criptato, aumentando la manipolazione dei dati. Contrariamente, FTP può usufruire della criptazione nel trasferimento dei dati con FTPs

**Risultati**

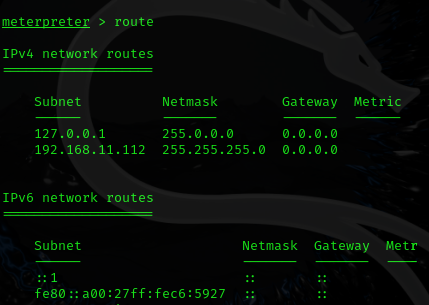
Sessione meterpreter



Configurazione di rete della macchina metasploitable



Routing table della macchina metasploitable



**Conclusioni**

Questo esercizio ha dimostrato la pericolosità di una configurazione non sicura del protocollo Java RMI. Approfondendo le caratteristiche di RMI, è stato possibile comprendere le sue limitazioni in termini di sicurezza e l'importanza di validare input e configurare l'autenticazione. Il confronto con FTP evidenzia come meccanismi di sicurezza basilari, come l'autenticazione, possano fare la differenza nella protezione di un sistema.