

Homework 1

Implementazione di una botnet in Python

Stefano Russo (1864451)

A.A. 2022-2023

1 Il progetto

Questo progetto si focalizza sull'implementazione di una botnet, un sistema distribuito composto da agenti di attacco chiamati bot e un centro di comando e controllo (C&C). Lo scopo principale di questo sistema è consentire all'attaccante di controllare e coordinare le attività dei bot, che sono installati su nodi compromessi.

Il bot è un componente eseguibile su Linux e funziona come un agente passivo che ascolta su un canale HTTP. Il principale limite di questo approccio è che il computer infetto deve essere esposto direttamente su Internet, poiché il bot agisce contemporaneamente come client e server.

2 Implementazione

Per la realizzazione del progetto, è stato utilizzato il linguaggio Python per implementare le componenti server e client della botnet. Sono state sfruttate diverse librerie e framework per semplificare lo sviluppo.

2.1 Librerie e Framework utilizzati

- **Flask**: framework HTTP scelto per la sua semplicità di creazione di nuovi endpoint e gestione delle richieste. Utilizzato sia nel C&C che nel bot.
- **Cmd**: libreria per creare una CLI interattiva, consentendo l'invio di comandi da parte del botmaster.
- **sqlite3**: libreria utilizzata nel C&C per salvare i dati dei bot.
- **requests**: libreria popolare per l'invio di richieste HTTP.
- **psutil**: libreria per raccogliere informazioni sui processi e l'utilizzo delle risorse di sistema.
- **getmac**: libreria necessaria per ottenere il MAC Address corretto.
- **pyinstaller**: utilizzato per creare l'eseguibile contenente tutte le dipendenze e l'interprete Python.

2.2 Il C&C

Il C&C è composto da tre componenti: il database SQLite, la command shell e il server Flask.

2.2.1 Database SQLite

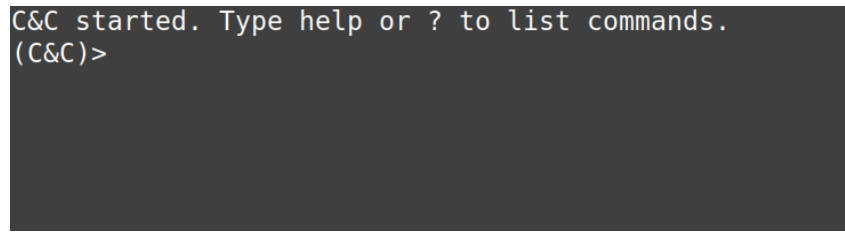
Semplice database disk-based, viene utilizzato per salvare i dati dei bot come MAC Address, Indirizzo IP, porta di ascolto, e informazioni sulle task eseguite.

2.2.2 Server Flask

Il server Flask nel C&C gestisce un singolo endpoint, `/heartbeat`.

Quando un bot viene avviato, invia una richiesta all'endpoint contenente il MAC Address per l'identificazione e la porta su cui è in ascolto. L'endpoint registra il bot, oppure cambia il suo stato in "online" se è già presente nel database. Periodicamente il C&C controlla che i bot abbiano inviato un heartbeat negli ultimi due minuti. Se non viene ricevuto alcun heartbeat per un certo periodo di tempo, il C&C imposta lo stato del bot su "offline".

2.2.3 Command Shell



```
C&C started. Type help or ? to list commands.
(C&C)>
```

Figure 1: Shell

La shell dei comandi è implementata utilizzando la libreria standard `Cmd` di Python e consente al botmaster di eseguire comandi per gestire la botnet. I comandi inviano richieste HTTP agli endpoint del bot per fargli eseguire determinate azioni, e impostano lo stato del bot su "running" per la durata della richiesta.

2.2.4 Comandi

- **help** [comando]
Mostra la lista dei comandi o fornisce una breve descrizione e sintassi di un comando specifico.
- **listbots** [all, online, available] (default=all)
Invia la lista di tutti i bot con relativo stato, ultima task eseguita, target e timestamp. È possibile specificare un argomento per visualizzare solo una sottosezione dei bot: all, online, available.
- **systeminfo** [MAC Address]
Restituisce le informazioni di sistema di tutti i bot o di un singolo bot specificando il MAC Address come argomento.
- **ddos** <HTTP Address> [seconds] (default=10)
Invia ai bot disponibili una richiesta contenente l'indirizzo IP da attaccare e la durata dell'attacco in secondi. Vengono creati thread separati per eseguire l'attacco contemporaneamente su ogni bot.
- **clear**
Pulisce la CLI

3 Bot

Il bot è composto da un singolo componente, il server Flask. All'avvio, il bot determina una porta libera e avvia il server Flask su quella porta. Successivamente, stabilisce una connessione con il C&C inviando le informazioni relative al proprio MAC Address e alla porta su cui è in ascolto. Dopo aver chiuso la connessione, il bot rimane in ascolto sulla porta scelta.

3.1 Endpoints

3.1.1 GET /systeminfo

Quando l'endpoint riceve una richiesta GET dal C&C, restituisce una serie di informazioni sull'hardware e il software presenti sulla macchina.

3.1.2 POST /ddos

Quando l'endpoint riceve una richiesta POST contenente l'URL da attaccare e la durata dell'attacco, invia un flood di richieste GET per la durata specificata.

3.2 Heartbeat

Il bot invia periodicamente una richiesta all'endpoint /heartbeat del C&C per segnalare che è ancora online.

4 Esempi di funzionamento

```
31.191.114.69 - - [04/Jun/2023 10:31:19] "GET / HTTP/1.1" 200 -
31.191.114.69 - - [04/Jun/2023 10:31:19] "GET / HTTP/1.1" 200 -
31.191.114.69 - - [04/Jun/2023 10:31:19] "GET / HTTP/1.1" 200 -
31.191.114.69 - - [04/Jun/2023 10:31:19] "GET / HTTP/1.1" 200 -
31.191.114.69 - - [04/Jun/2023 10:31:19] "GET / HTTP/1.1" 200 -
31.191.114.69 - - [04/Jun/2023 10:31:19] "GET / HTTP/1.1" 200 -
31.191.114.69 - - [04/Jun/2023 10:31:19] "GET / HTTP/1.1" 200 -
31.191.114.69 - - [04/Jun/2023 10:31:19] "GET / HTTP/1.1" 200 -
31.191.114.69 - - [04/Jun/2023 10:31:19] "GET / HTTP/1.1" 200 -
31.191.114.69 - - [04/Jun/2023 10:31:19] "GET / HTTP/1.1" 200 -
31.191.114.69 - - [04/Jun/2023 10:31:19] "GET / HTTP/1.1" 200 -
31.191.114.69 - - [04/Jun/2023 10:31:19] "GET / HTTP/1.1" 200 -
31.191.114.69 - - [04/Jun/2023 10:31:19] "GET / HTTP/1.1" 200 -
31.191.114.69 - - [04/Jun/2023 10:31:19] "GET / HTTP/1.1" 200 -
31.191.114.69 - - [04/Jun/2023 10:31:19] "GET / HTTP/1.1" 200 -
```

Figure 2: Esecuzione del comando "ddos http://37.187.89.187:8000/ 60"

```
(C&C)> listbots
[08:00:27:6f:a8:64]:
    Status: running
    Listening Address: http://10.0.2.8:57643
    Last Executed Task: ddos
    Last Executed Task Timestamp: 2023-06-04 10:30:19
    Target: http://37.187.89.187:8000/
[08:00:27:fe:25:df]:
    Status: running
    Listening Address: http://10.0.2.9:54553
    Last Executed Task: ddos
    Last Executed Task Timestamp: 2023-06-04 10:30:19
    Target: http://37.187.89.187:8000/
(C&C)> █
```

Figure 3: Esecuzione del comando "listbots" mentre la task DDOS è in corso

```
(C&C)> systeminfo
[08:00:27:6f:a8:64]
    Hostname: zombie2
    Listening Address: http://10.0.2.8:57643
    Platform: Linux 5.15.0-73-generic #80-Ubuntu SMP Mon May 15 15:18:26 UTC 2023
    Architecture: x86_64
    Processor: AMD Ryzen 5 4600H with Radeon Graphics
    RAM: 2 GB
[08:00:27:fe:25:df]
    Hostname: zombie1
    Listening Address: http://10.0.2.9:54553
    Platform: Linux 5.15.0-73-generic #80-Ubuntu SMP Mon May 15 15:18:26 UTC 2023
    Architecture: x86_64
    Processor: AMD Ryzen 5 4600H with Radeon Graphics
    RAM: 2 GB
(C&C)> █
```

Figure 4: Esecuzione del comando "systeminfo" senza argomenti aggiuntivi.

5 Utilizzo delle risorse visualizzato con htop (VM con Ubuntu Server 22.04.2, 1 core e 2 GB di ram)

```
CPU[ 0.0%] Tasks: 24, 27 thr; 1 running
Mem[|||||] 182M/1.93G Load average: 0.44 0.14 0.05
Swp[ 0K/2.00G] Uptime: 00:00:39
```

Figure 5: Mentre il bot è spento

```
CPU[ 0.0%] Tasks: 27, 28 thr; 1 running
Mem[|||||] 189M/1.93G Load average: 0.08 0.09 0.04
Swp[ 0K/2.00G] Uptime: 00:02:19
```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
998	zombie1	20	0	3048	1024	908	S	0.0	0.1	0:00.08	./botnet-client
999	zombie1	20	0	113M	33932	11272	S	0.0	1.7	0:00.32	./botnet-client
1000	zombie1	20	0	113M	33932	11272	S	0.0	1.7	0:00.00	./botnet-client

Figure 6: Mentre il bot è in ascolto (idle)

```
CPU[|||] 6.0%] Tasks: 27, 28 thr; 1 running
Mem[|||||] 190M/1.93G Load average: 0.01 0.06 0.03
Swp[ 0K/2.00G] Uptime: 00:04:01
```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
998	zombie1	20	0	3048	1024	908	S	0.0	0.1	0:00.08	./botnet-client
999	zombie1	20	0	113M	33932	11272	S	3.3	1.7	0:01.70	./botnet-client
1000	zombie1	20	0	113M	33932	11272	S	4.0	1.7	0:01.38	./botnet-client

Figure 7: Mentre la task DDOS è in corso (L'utilizzo della CPU oscilla tra ~2% e ~6%)