Progetto gestione di reti 2019/2020

Nicola Stano

August 21, 2020

1 Introduzione

Negli ultimi anni abbiamo assistito ad una rapida crescita di smart devices nelle abitazioni comuni: Chromecast, smartTV, telecamere di sicurezza, ecc... I router presenti nelle comuni abitazioni, solitamente forniti dall'ISP, sono tipicamente "locked-down" ovvero offrono poche, se non nulle, possibilità di configurazione e monitoraggio da parte dell'utente finale. Con questo progetto mi sono posto il problema di monitorare la presenza in rete di devices in maniera passiva, evitando di creare traffico sulla rete, ed eventualmente eseguire arp spoof in modo da poter sniffare il traffico del device vittima e, con tools come wireshark/nDPI, effettuare un'analisi del traffico.

2 Struttura Programma

Il tool è stato sviluppato in python 3.6 ed è composto di 3 comandi:

```
$ python main.py --help
Usage: main.py [OPTIONS] COMMAND [ARGS]...

Options:
--conf TEXT configuration file with influx options and devices names,
defaults config.json

--help Show this message and exit.

Commands:
addname Adds a Mac -> Name association to the config file provided...
netwatch Sniffs packets and constructs a presence table for each...
spoof Spoofs the target ip, disable ip forwarding with flag --deny
```

- addname

```
$ python main.py addname --help
[*] Reading config file config.json
Usage: main.py addname [OPTIONS] MAC NAME

Adds a Mac -> Name association to the config file provided (config.json
default)
```

```
7
8 Options:
9 --help Show this message and exit.
```

- netwatch

```
1 $ python main.py netwatch --help
2 [*] Reading config file config.json
3 Usage: main.py netwatch [OPTIONS]
4
5 Sniffs packets and constructs a presence table for each connected device
6 sending data to influxdb. When an unknown device (not present in config
7 file) is detected sends a notification (notify-send), and a message using
8 telegram bot (configured in config file)
9
10 Options:
11 --arp sniff only arp packets
12 -i TEXT net interface
13 --help Show this message and exit.
```

- spoof

```
1 python main.py spoof --help
2 [*] Reading config file config.json
3 Usage: main.py spoof [OPTIONS] VICTIMIP
4
5 Spoofs the target ip, disable ip forwarding with flag --deny
6
7 Options:
8 --deny
9 --help Show this message and exit.
```

2.1 File di configurazione

Ho optato per un file di configurazione *config.json* per configurare InfluxDB, Telegram e le associazioni (MAC,nome). La ragione principale è evitare di passare da riga di comando un numero eccessivo di parametri.

```
1  {
2   "devices": {
3    "11:11:11:11:11:11": "device-name-here"
4   },
5   "influx": {
6    "bucket": "",
7    "org": "",
8    "token": "",
9    "url": "http://localhost:9999"
10  },
11  "telegram": {
12   "token": "",
13   "chatid": ""
14  }
```

```
15 }
```

Per configurare la sezione Telegram bisogna creare un bot tramite BotFather e inserire il *token* nel file di configurazione. Per ottenere *chatid* si segua questa guida.

Per configurare InfluxDB si consulti la documentazione ufficiale.

3 Installazione

Per installare le librerie necessarie

pip install click scapy influxdb-client python-telegram-bot

4 Esempi d'esecuzione

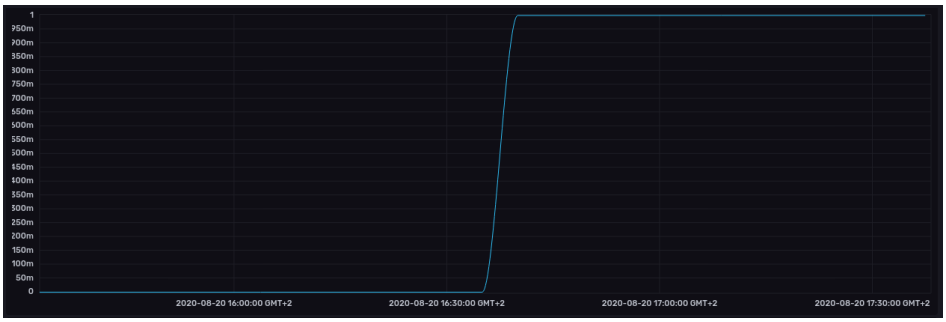
Dopo aver configurato correttamente InfluxDB e Telegram nel file di configurazione, lanciare il programma come superuser, in alternativa si possono aggiungere le capabilities CAP_NET_RAW+eip, CAP_NET_ADMIN+eip a python e tcpdump (usato da scapy).

Esempio comando netwatch:

```
→  netwatch sudo ~/venvs/netwatch/bin/python main.py --conf myconfig.json  netwatch
[*] Reading config file myconfig.json
[*] Network interface= using default  filter= arp or (udp port 5353)
[*] Influx health status OK
NAME            MAC               IP                      LAST SEEN

NAME            MAC               IP                      LAST SEEN

NAME            MAC               IP                      LAST SEEN

NAME            MAC               IP                      LAST SEEN

NAME            MAC               IP                      LAST SEEN
gateway         98:0d:67:37:29:70  192.168.1.1        0d  0h  0m  1s ago
laptop N        ec:0e:c4:2f:b7:57  192.168.1.230      0d  0h  0m  1s ago

NAME            MAC               IP                      LAST SEEN
gateway         98:0d:67:37:29:70  192.168.1.1        0d  0h  0m  6s ago
laptop N        ec:0e:c4:2f:b7:57  192.168.1.230      0d  0h  0m  6s ago
cell M          d8:c7:71:8b:0d:55  192.168.1.204      0d  0h  0m  3s ago

NAME            MAC               IP                      LAST SEEN
gateway         98:0d:67:37:29:70  192.168.1.1        0d  0h  0m 11s ago
laptop N        ec:0e:c4:2f:b7:57  192.168.1.230      0d  0h  0m 11s ago
cell M          d8:c7:71:8b:0d:55  192.168.1.204      0d  0h  0m  8s ago

NAME            MAC               IP                      LAST SEEN
gateway         98:0d:67:37:29:70  192.168.1.1        0d  0h  0m 16s ago
laptop N        ec:0e:c4:2f:b7:57  192.168.1.230      0d  0h  0m 16s ago
cell M          d8:c7:71:8b:0d:55  192.168.1.204      0d  0h  0m 13s ago
```

Esempio di cella con devices stacked e sum del numero di pacchetti sniffati in InfluxDB:

Esempio di cella con singolo device:



Esempio di messaggio inviato dal bot Telegram quando un device sconosciuto viene rilevato: