

Progetto gestione di reti 2019/2020

Nicola Stano

September 22, 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. Per questo progetto mi sono posto il problema di monitorare la presenza in rete di devices in maniera passiva, evitando di creare traffico sulla rete, in modo da sapere quando e quanto un device è stato attivo in rete. Eventi come rilevamento di nuovi MAC, nuove attività da parte di devices ecc...(si veda sezione Notifiche) vengono notificati tramite Telegram.

2 Struttura Programma

Il tool è stato sviluppato in python 3.6 ed è composto di 2 comandi:

```
1 $ python main.py --help
2 Usage: main.py [OPTIONS] COMMAND [ARGS]...
3
4 Options:
5 --conf TEXT configuration file for influxdb, telegram. Default is
6 config.json
7
8 --help Show this message and exit.
9
10 Commands:
11 addname Adds a Mac -> Name association
12 netwatch Construct a presence table for devices in the local network,...
```

- addname

```
1 $ python main.py addname --help
2 [*] Reading config file config.json
3 Usage: main.py addname [OPTIONS] MAC NAME
4
5 Adds a Mac -> Name association
6
```

```
7 Options:
8 --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 Monitors devices presence in the local network, sending data to InfluxDB and
6 alerts via Telegram
7
8 Options:
9 --arp sniff only arp packets
10 -i TEXT net interface
11 --help Show this message and exit.
```

2.1 File di configurazione

Ho optato per un file di configurazione *config.json* per configurare InfluxDB, Telegram. La ragione principale è evitare di passare da riga di comando un numero eccessivo di parametri.

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

3 Installazione

Per installare le librerie necessarie:

```
pip3 install click scapy influxdb-client python-telegram-bot requests
```

Per configurare la sezione Telegram bisogna:

- creare un bot iniziando una nuova conversazione con BotFather tramite il comando in chat /newbot
- dopo aver scelto il nome del bot (supponiamo netwatchbot) , si riceverà un token del tipo:

110201543:AAHdqTcvCH1vGWJxfSeofSAs0K5PALDsaw
questo ci servirà successivamente

- ora si inizi una conversazione con il nuovo bot appena creato (supponiamo netwatchbot) scrivendo in chat qualsiasi messaggio
- si esegua lo script helper telegramconf.py con il token come argomento:

```
python3 telegramconf.py TOKEN
```

- ora il file di configurazione config.json nella sezione telegram ha tutti i campi configurati correttamente

Per info su InfluxDB si consulti la documentazione ufficiale. Per installare InfluxDB si veda
<https://docs.influxdata.com/influxdb/v2.0/get-started/#start-with-influxdb-oss>

Terminata l'installazione si consiglia di accedere alla sezione Data e creare un nuovo bucket e un nuovo token. Inserire nel file di configurazione nella sezione influx:

- bucket: nome del bucket creato
- org : organization name inserita durante l'installazione
- token : il token creato
- url : l'url che nel caso di installazione locale è già impostata a `http://localhost:9999`

4 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:

```
sudo python3 main.py netwatch
```

```

$ sudo /home/n/.virtualenvs/netwatch2/bin/python main.py --conf myconfig.json netwatch
[*] Reading config file myconfig.json
[*] Network interface = any filter = ether[0]&1=1
NAME          MAC          IP          LAST SEEN
undefined     54:b1:21:e6:ff:00 192.168.1.91 5d 21h 6m 7s ago
cell C        f0:c8:50:13:09:64 192.168.1.71 2d 18h 5m 26s ago
undefined     4c:cc:6a:cf:13:aa 192.168.1.72 1d 22h 39m 12s ago
laptop N      ec:0e:c4:2f:b7:57 192.168.1.230 0d 23h 9m 47s ago
cell F        e8:e8:b7:7d:3c:f3 192.168.1.46 0d 22h 29m 45s ago
cell T        ec:89:14:bb:86:a3 192.168.1.54 0d 20h 58m 38s ago
undefined     88:f5:6e:b6:1a:4a 192.168.1.41 0d 20h 58m 22s ago
undefined     d4:11:a3:eb:16:75 192.168.1.104 0d 18h 25m 49s ago
laptop G      94:e9:79:de:65:77 192.168.1.238 0d 16h 34m 21s ago
cell L        b8:94:36:1d:63:34 192.168.1.15 0d 2h 15m 49s ago
laptop M      3c:6a:a7:b9:4a:65 192.168.1.44 0d 1h 53m 10s ago
cell N        e8:5a:8b:c9:5e:7f 192.168.1.98 0d 0h 6m 27s ago
raspberrypi   b8:27:eb:f5:58:04 192.168.1.219 0d 0h 1m 24s ago
undefined     9e:97:26:bf:0b:5a 192.168.1.253 0d 0h 1m 19s ago
Router B      9c:97:26:bf:0b:5a 192.168.1.254 0d 0h 1m 19s ago
undefined     7c:2a:db:ed:d0:7d 192.168.1.73 0d 0h 0m 51s ago
cell B        1c:cc:d6:07:1a:6d 192.168.1.164 0d 0h 0m 34s ago
cell G        d0:16:b4:b4:f6:9d 192.168.1.180 0d 0h 0m 25s ago
PC desk       4c:cc:6a:cf:13:ab 192.168.1.72 0d 0h 0m 7s ago
cell M        d8:c7:71:8b:0d:55 192.168.1.204 0d 0h 0m 6s ago
gateway       98:0d:67:37:29:70 192.168.1.1 0d 0h 0m 6s ago

```

Se non viene specificata un'interfaccia di rete, lo sniffer si metterà in ascolto su tutte le interfacce (any). Il filtro di cattura standard è `ether[0]&1=1` per traffico broadcast e multicast, mentre con il flag `-arp` viene catturato solo traffico ARP. Per ogni pacchetto catturato si aggiorna la tabella persistente (sqlite3), si inviano eventuali notifiche tramite Telegram e viene scritto in InfluxDB un point (single data record con quattro componenti: measurement, tag set, field set, e timestamp).

Esempio di grafico di presenza in rete di un device:



5 Serie temporali

Ad ogni MAC-address viene associata una measurement in influxDB, che non è altro che un container per tags, campi e timestamp. Ogni punto della serie è composto dal campo `presence`, intero sempre uguale a 1. I tag `ip` e `name` contengono rispettivamente l'indirizzo IPv4 e il nome definito tramite il comando `addname`.

true dateTime:RFC3339	true dateTime:RFC3339	false dateTime:RFC3339	false long	true string	true string	true string	true string
_start	_stop	_time	_value	_field	_measurement	ip	name
2020-09-17T18:00:01.06518346Z	2020-09-18T18:00:01.06518346Z	2020-09-17T18:00:25.693Z	1	presence	94:e9:79:de:b5:77	192.168.1.238	Laptop G
2020-09-17T18:00:01.06518346Z	2020-09-18T18:00:01.06518346Z	2020-09-17T18:00:26.693Z	1	presence	94:e9:79:de:b5:77	192.168.1.238	Laptop G
2020-09-17T18:00:01.06518346Z	2020-09-18T18:00:01.06518346Z	2020-09-17T18:00:31.10Z	1	presence	94:e9:79:de:b5:77	192.168.1.238	Laptop G
2020-09-17T18:00:01.06518346Z	2020-09-18T18:00:01.06518346Z	2020-09-17T18:01:03.099Z	1	presence	94:e9:79:de:b5:77	192.168.1.238	Laptop G
2020-09-17T18:00:01.06518346Z	2020-09-18T18:00:01.06518346Z	2020-09-17T18:01:04.884Z	1	presence	94:e9:79:de:b5:77	192.168.1.238	Laptop G
2020-09-17T18:00:01.06518346Z	2020-09-18T18:00:01.06518346Z	2020-09-17T18:01:05.303Z	1	presence	94:e9:79:de:b5:77	192.168.1.238	Laptop G
2020-09-17T18:00:01.06518346Z	2020-09-18T18:00:01.06518346Z	2020-09-17T18:01:06.307Z	1	presence	94:e9:79:de:b5:77	192.168.1.238	Laptop G
2020-09-17T18:00:01.06518346Z	2020-09-18T18:00:01.06518346Z	2020-09-17T18:03:01.695Z	1	presence	94:e9:79:de:b5:77	192.168.1.238	Laptop G
2020-09-17T18:00:01.06518346Z	2020-09-18T18:00:01.06518346Z	2020-09-17T18:03:02.694Z	1	presence	94:e9:79:de:b5:77	192.168.1.238	Laptop G

6 Notifiche

Le notifiche tramite telegram sono le seguenti:

- Viene rilevato un nuovo MAC address

```
2020-09-12 19:33:53
NEW_MAC
MAC = 7c:2a:db:ed:d0:7d
IP = 192.168.1.73 19:33
```

- Nuova attività, ovvero il tempo trascorso dall'ultima rilevazione alla attuale è maggiore di delta_new_activity impostato a 6 ore, si può configurare nel file pktconsumer.py riga 12

```
2020-09-12 20:29:59
NEW_ACTIVITY
NAME = undefined
MAC = 88:f5:6e:b6:1a:4a
IP = 192.168.1.41 20:29
```

- Mismatch fra MAC address sorgente e campo hwsrc in pacchetto ARP

```
ETH_MISMATCH
NAME = PC desk
###[ Ethernet ]###
dst    = ff:ff:ff:ff:ff:ff
src    = 4c:cc:6a:cf:13:ab
type   = ARP
###[ ARP ]###
hwtype = 0x1
ptype  = IPv4
hwlen  = 6
plen   = 4
op     = who-has
hwsrc  = e8:e8:b7:7d:3c:f3
psrc   = 192.168.1.72
hwdst  = 00:00:00:00:00:00
pdst   = 192.168.1.33 18:35
```

- MAC address della trama Ethernet e/o il campo hwsrc del pacchetto ARP contengono un indirizzo di broadcast L2

```
ETH_BROADCAST
###[ Ethernet ]###
dst   = ff:ff:ff:ff:ff:ff
src   = ff:ff:ff:ff:ff:ff
type  = ARP
###[ ARP ]###
hwttype = 0x1
ptype  = IPv4
hwlen  = 6
plen   = 4
op      = who-has
hwsrc  = 4c:cc:6a:cf:13:ab
psrc   = 192.168.1.72
hwdst  = 00:00:00:00:00:00
pdst   = 192.168.1.1
```

18:52