Backfire-Monitor

Analisi e Filtraggio Tramite Tecniche di ARP-Spoofing

Gestione di Rete 2017/2018,
Samuele Sabella

Mentre le tecniche tradizionali di monitoraggio e filtraggio del traffico di rete prevedono l'adozione di più router e l'introduzione di devices che si frappongano tra questi, *Backfire-Monitor* consente ad un qualsiasi host di filtrare e analizzare i pacchetti in uscita dalla rete. Per far ciò viene sfruttata la mancanza di autenticazione nel protocollo ARP, utilizzato per la traduzione di indirizzi IP in indirizzi di livello Collegamento, per eseguire un attacco di ARP poisoning [1] e dirottare il traffico verso il dispositivo utilizzato per il monitoraggio. Tramite questo approccio è quindi possibile, anche se con alcune limitazioni, evitare l'acquisto di apparecchiature specifiche con costo molto elevato [2].

1 Guida all'utilizzo

Come prima funzionalità il programma permette di filtrare il traffico della rete tramite apposita politica di sicurezza, specificata dal file *"policy.txt"*, in cui ciascuna riga costituisce un filtro [1].
Tutto il traffico che non rispetti almeno uno dei filtri viene bloccato mentre i restanti pacchetti vengono analizzati e poi reinoltrati al router/switch. *Backfire-Monitor* rimane quindi trasparente all'utente finale, testimone solamente del rallentamento del traffico.
Un esempio di politica di sicurezza che blocchi tutto il traffico non diretto verso *Facebook* o *StackExchange* è il seguente:

```
dns
dst host facebook.com
dst host stackexchange.com
```

Per quanto riguarda il monitoraggio, l'implementazione attuale consente solo la scoperta degli host presenti nella rete e la percentuale di pacchetti che questi inviano [2], tuttavia è possibile registrare altre informazioni estendendo la funzione *Analyze* definita nel file *"analyzer.h"*. Questa viene chiamata ogniqualvolta un pacchetto ricevuto viene ammesso dalla politica di sicurezza e prende come argomenti il pacchetto catturato e la sua lunghezza in Byte.
Una volta analizzato, il pacchetto viene poi reinoltrato al router, con indirizzo sorgente impostato a *broadcast* così e da evitare che questo associ il MAC address dell'amministratore all'interfaccia tramite cui è collegato e per permettere così all'host

[1] Per dettagli sul formato dei filtri visitare https://www.tcpdump.org/manpages/pcap-filter.7.html
[2] Percentuale calcolata in base al numero totale di pacchetti inoltrati dal programma al router

sorgente di ricevere le risposte per i pacchetti inviati [3].
Il programma può lavorare in due modalità: *Poisoner*; *Gratuitous*. Mentre nella prima si utilizzano semplici pacchetti ARP, la secondo utilizza pacchetti ARP con formato *Gratuitous*[4]. Le due modalità si differenziano nella quantità di traffico generato dal programma e dalla velocità con cui le tabelle ARP degli host nella rete vengono avvelenate. Utilizzando la modalità di default *Poisoner* il traffico generato dall'applicazione sarà molto limitato ma si impiegherà più tempo a dirottare il traffico verso l'host che ha avviato il programma. La modalità *Gratuitous*, selezionabile tramite l'opzione *-g*, generarà invece una quantità molto elevata di traffico portando però più velocemente a convergenza [3].
Una volta compilato tramite il comando *make*, per essere avviato correttamente *Backfire-Monitor* ha bisogno delle seguenti informazioni:

- Indirizzo mac dell'host target dello spoofing, specificato dall'opzione obbligatoria *-a*.
- L'interfaccia di rete da utilizzare per catturare e inoltrare i pacchetti, specificata tramite l'opzione obbligatoria *-d*.
- L'indirizzo IP del host target, con valore di default *192.168.1.1*, configurabile tramite l'opzione *-i*.

Nell'esempio sotto riportato il programma è stato avviato da una macchina con sistema operativo *MacOS* per analizzare il traffico di una rete domestica con indirizzo MAC del router *10:13:31:c3:b2:2* e con modalità *Gratuitous* attiva.

```
$ sudo ./main.o -d en0 -a 10:13:31:c3:b2:2 -g

  Tot packets: 196
  Device:      MAC/D0:81:7A:C0:3F:F4   IP/192.168.1.156   %(0.9592)
  Device:      MAC/E8:B4:C8:FB:7D:BD   IP/192.168.1.106   %(0.0357)
```

2 Implementazione

Per l'implementazione sono state utilizze due librerie: *"libpcap"* per la cattura e l'invio dei pacchetti; la libreria di terze parti *"uthash"* per memorizzare e gestire le informazioni degli host il cui traffico viene dirottato.
Il diagramma di stato sotto riportato mostra il funzionamento generale del programma.

[3] Diciamo che il programma converge nel momento in cui le tabelle ARP dei dispositivi connessi alla rete sono state *"avvelenate"*

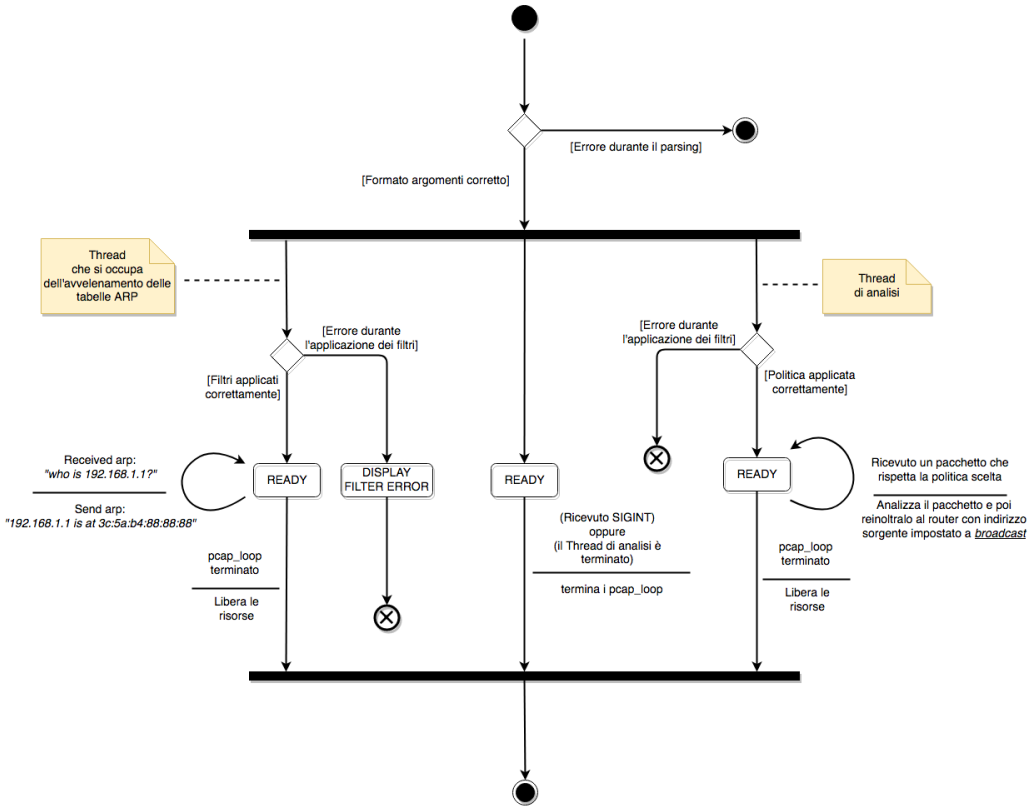

Fig.2

L'applicazione sfrutta tre thread, due dei quali (rispettivamente agli estremi della figura) eseguono la funzione *"pcapLoop"* definita nel file *"main.c"*. Questa, tramite la funzione *pcap_open_live()*, apre un handler sul device specificato dall'utente e vi applica il filtro passato come argomento. Al thread viene inoltre fornita una callback da passare come argomento alla funzione *pcap_loop*. I due thread creati differiscono solo per la callback e per il filtro da applicare all'handler creato.

References

[1] Ryan Spangler. Packet sniffing on layer 2 switched local area networks. 2003.

[2] Bitbridge: How does firewalla intercept traffic. https://www.firewalla.com/bitbridge-firewalla-intercept-traffic/.

[3] Behrouz A. Forouzan e Firouz Mosharraf. Computer networking: a top-down approach. page 316, 2000.

[4] Ed Harmoush. Gratuitous arp. 2017. http://www.practicalnetworking.net/series/arp/gratuitous-arp/.