Università degli Studi di Camerino

Scuola di Scienze e Tecnologie
Corso di Laurea in Informatica

ARPWATCH DASHBOARD: command & control center for remote monitoring

Tesi di Laurea
in
Reti e Sicurezza Informatica

Laureando:
Giacomo Fagioli

Relatore:
Prof. Fausto Marcantoni

A Marghe,

unica donna capace di accendere i miei sensi ed il mio intelletto.
Moglie, amica, madre, compagna di viaggio.
Mia origine, mio orizzonte.

Indice generale

§ 1 - Introduzione

1.1 - Prefazione

Nonostante i protocolli di rete siano in continua evoluzione, con lo scopo dichiarato di migliorare sicurezza e prestazioni, e di allargare le funzionalità offerte, questa pressione evolutiva trova un naturale antagonista nella necessità di stabilità e retro-compatibilità derivante dal pre-esistente parco di hw e sw installato. Tale resistenza ai miglioranti non è uniformemente distribuita lungo i vari livelli dello *stack* di protocolli di rete, ma va crescendo man mano che scendiamo e ci avviciniamo ai livelli fisici (che per loro natura sono meno modificabili), e dove le interdipendenze tra protocolli aumentano. L'architettura stessa dello stack, i cui servizi di livello superiore sono implementati sulla base dei sottostanti, genera una maggiore rigidità alla base, dato che i livelli inferiori risultano maggiormente vincolati al *contratto di interfaccia* (Principio di sostituzione di *Liskov* [1]). Questo fenomeno favorisce l'obsolescenza, e trova il suo apice nei "piani bassi" dello stack, dove infatti capita spesso di trovare protocolli progettati più di 30 anni fa (e.g.: Ethernet IEEE-802.3 [2]), in contesti radicalmente diversi da quelli in cui oggi continuano ad essere utilizzati (onore al merito degli ingegneri che hanno progettato protocolli così flessibili!).
L'*Address Resolution Protocol* (ARP) è uno di questi "fossili viventi", che continua a sorreggere stoicamente le fondamenta delle moderne reti (specialmente quelle basate sui protocolli *Ethernet* e *IP*), mostrando però qualche "crepa", legata soprattutto alla natura non-autenticata dei suoi aggiornamenti [3].
Come ben sappiamo, le falle sono terreno fertile per chi desidera esplorare i limiti della tecnologia (i.e.: *hacker*), o per chi ne vuole sfruttare le debolezze a proprio vantaggio (i.e.: *cracker*): i limiti progettuali di ARP hanno consentito lo sviluppo di un'intera famiglia di *malware*, tra quello detto *Man-In-The-Middle* (MITM), caratterizzata -come in una partita a scacchi- dalla comune "apertura" con *ARP-poisoning* [3], che avremo modo di descrivere meglio in seguito.

La presente tesi prende le mosse da queste premesse, per analizzare i punti di debolezza del protocollo ARP e proporre una semplice soluzione sw, di facile utilizzo, orientata al monitoraggio distribuito degli eventi ARP, specialmente quelli associati ad attacchi basati sull'*exploit* delle falle strutturali di ARP. Oltre che ai principi-base della sicurezza informatica (capillarità, tempestività, ridondanza, etc.), il sw proposto è stato sviluppato pensando anche alla portabilità multi-piattaforma, e alla necessità di semplificare l'accesso al pannello di controllo (i.e.: da browser web).
Il nome del sw, *ArpWatch Dashboard*, richiama il noto sw *open-source* Arpwatch [4], cui si appoggia funzionalmente e di cui sfrutta le doti di rilevamento, ampliandole, colmandone le lacune, e rendendone la UI più accessibile.

Come nota di carattere professionale, è interessante evidenziare che le competenze impiegate nello sviluppo di questo sw possano definirsi a buon diritto "*full-stack*", attraversando tutti i campi dello spettro *IT/networking*, dal livello *data-link* (i.e.: L2), al livello applicativo (i.e.: UI), passando per *verbi* HTTP, e *rotte* IP.

1.2 - Sinossi

I capitoli di quest'opera sono così suddivisi:

- §1 - Introduzione: descrizione del pre-esistente contesto generale, in cui si inserisce il sw;
- §2 - Specifiche: descrizione delle richieste del committente;
- §3 - Architettura Generale/Distribuita: descrizione statica/strutturale del sw proposto come soluzione;
- §4 - Protocollo di Comunicazione: descrizione dinamica/comportamentale del sw;
- §5 - Manuale dell'Utente: descrizione operativa dei casi d'uso del sw;
- §6 - Strumenti e Metodi di Sviluppo: descrizione dell'ambiente di sviluppo utilizzato;
- §7 - Criticità e Limiti Operativi: descrizione delle difficoltà tecniche incontrate;
- §8 - Conclusioni e Sviluppi Futuri: obiettivi raggiunti e spazi di miglioramento;

1.3 - ARP

L'*Address Resolution Protocol* (ARP) è uno standard pubblico ufficiale, definito dall'RFC-826 [5] del 1982; per dare un senso a questa data, si consideri che in quell'anno [6]: il presidente dell'Unione Sovietica era Brezhnev, e il "computer per le masse" era il *Commodore 64*.
Da un punto di vista puramente teorico/accademico, esistono diverse scuole di pensiero riguardo alla collocazione di ARP: nello stack OSI esso è a livello 3 (*network layer*), in ragione del fatto che i suoi messaggi sono incapsulati nel *payload* di un pacchetto/PDU di livello 2 (e.g.: trame Ethernet); nello stack TCP/IP invece ARP è a livello 2 (*data-link*), in ragione del fatto che fornisce servizi di risoluzione degli indirizzi fisici a partire da quelli logici di rete [7]. Per esprimere questa ambiguità, spesso si colloca ARP in un livello intermedio, indicato con L2.5 [8], e più propriamente denominato *Logical-Link Control* (LLC), o IEEE-802.2, che è il *sub-layer* superiore (nel livello *data-link*), *medium-independent* e orientato al *multiplexing* verso i vari protocolli di L3.
Nota: LLC è da intendersi come distinto e contrapposto al *sub-layer* inferiore (nel livello *data-link*), detto *Media Access Control* (MAC), che invece è *medium-dependent*, e si declina nella moltitudine dei possibili mezzi trasmissivi, codifiche del segnale d'onda, e protocolli di accesso al mezzo, complessivamente rappresentati dalla cosiddetta scheda di rete.

Alla luce di quanto detto, ARP ha il ruolo di raccordo tra il protocollo L2 adottato dalla scheda di rete, e il protocollo L3 adottato dall'amministratore di rete: ciò significa che ARP è agnostico rispetto a tali scelte, e in teoria supporta qualsiasi combinazione (L2, L3), e.g.: (ATM, DECnet) [7]; di fatto però la combinazione più comune è (Ethernet IEEE-802.3, IPv4), e perciò -senza perdita di generalità- assumeremo tale caso come base di riferimento, nel prosieguo della trattazione.

Da una prospettiva funzionale ARP svolge il ruolo di risoluzione L3-to-L2, ossia, nel caso in esame, degli indirizzi IP (noti), negli indirizzi Ethernet, detti *MAC-address* (ignoti). Dopo che un host scopre le mappature dei MAC-address sugli IP-address (con la procedura di *neighbor-discovery* che illustreremo tra breve), queste vengono mantenute in una memoria temporanea, detta ***ARP-cache***, o *ARP-table* (una per ogni scheda di rete [9]), visualizzabile col comando:

```
gf@silenzio:~$ arp -a
```

i cui record hanno i seguenti campi:

{*IP-address, MAC-address, Time-to-Live*}

Nota: la chiave della tabella è l'IP-address (che quindi non può essere nullo), mentre è consentito che il campo MAC-address sia nullo, ossia *all-zero*, (temporaneamente, fino a che non viene risolto via ARP): questo stato è indicato con "*incomplete*" o "*invalid*".

Nota: in IPv6 il ruolo di ARP (e di ICMP) è assunto dal protocollo *Neighbor-Discovery Protocol* (NDP), come descritto dall'RFC-4861 [10]. La trattazione di NDP travalica gli scopi di questa tesi, tuttavia è interessante notare che, tra le migliorie introdotte, vi è l'estensione sicura SEND(=*Secure NDP*, RFC-3971, RFC-6494) che usa *CGA* (=*Cryptographically Generated Addresses*) e RPKI (=*Resource Public-Key Infrastructure*, RFC-6480, RFC-6487).

E.g.: Microsoft Windows 10 sample output (*-v* for verbose):

```
C:\Users\gf> arp -av
...

Interface: 0.0.0.0 --- 0xffffffff
  Internet Address      Physical Address      Type
  224.0.0.2             01-00-5e-00-00-02     static
  224.0.0.22            01-00-5e-00-00-16     static

Interface: 192.168.211.68 --- 0x3
  Internet Address      Physical Address      Type
  192.168.211.69        00-1e-52-81-24-eb     dynamic
  192.168.211.211       f8-1a-67-b4-a2-e2     dynamic
  192.168.211.244       f0-92-1c-9a-e6-af     dynamic
  192.168.211.255       ff-ff-ff-ff-ff-ff     static
  224.0.0.2             01-00-5e-00-00-02     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

Interface: 192.168.1.145 --- 0x6
  Internet Address      Physical Address      Type
  192.168.1.68          00-00-00-00-00-00     invalid
  192.168.1.69          00-1b-63-b8-94-71     dynamic
  192.168.1.254         00-04-56-f8-71-ee     dynamic
  192.168.1.255         ff-ff-ff-ff-ff-ff     static
  224.0.0.2             01-00-5e-00-00-02     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static
```

Nota: il *Time-to-Live* (TTL) è l'attesa dopo la quale, in assenza di aggiornamenti, la singola *entry* "scade" ed è rimossa automaticamente; tale ritardo è *OS-dependent*, *medium-dependent*, ed inoltre spesso incorpora anche una componente *random*, introdotta per de-sincronizzare gli *update* ARP ed evitare i conseguenti *ARP-storm* [11]:

- Microsoft Windows *default*: 2-10 min (*before-Vista*), o 15-45 sec (*after-Vista* [12]), visualizzabile con il comando [13] [14]:

```
C:\Users\gf> netsh interface ipv4 show interface 6
Interface Ethernet Parameters
----------------------------------------------
IfLuid                             : ethernet_32769
IfIndex                            : 6
State                              : connected
Metric                             : 10
Link MTU                           : 1500 bytes
Reachable Time                     : 38500 ms
Base Reachable Time                : 30000 ms
Retransmission Interval            : 1000 ms
DAD Transmits                      : 3
...
```

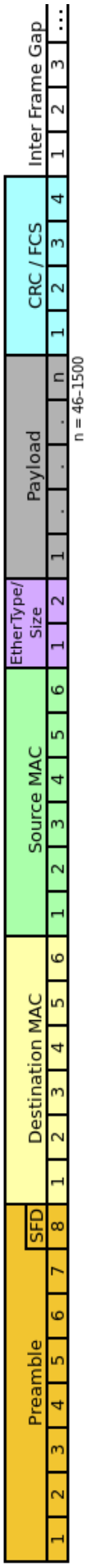
- Linux Ubuntu *default*: 60sec, visualizzabile con il comando [15]:

```
gf@silenzio:~$ cat /proc/sys/net/ipv4/neigh/default/gc_stale_time
60
```

Nota: l'*integer* restituito indica i secondi.

- Cisco IOS *default*: 4hr, visualizzabile con il comando [16] [11]:

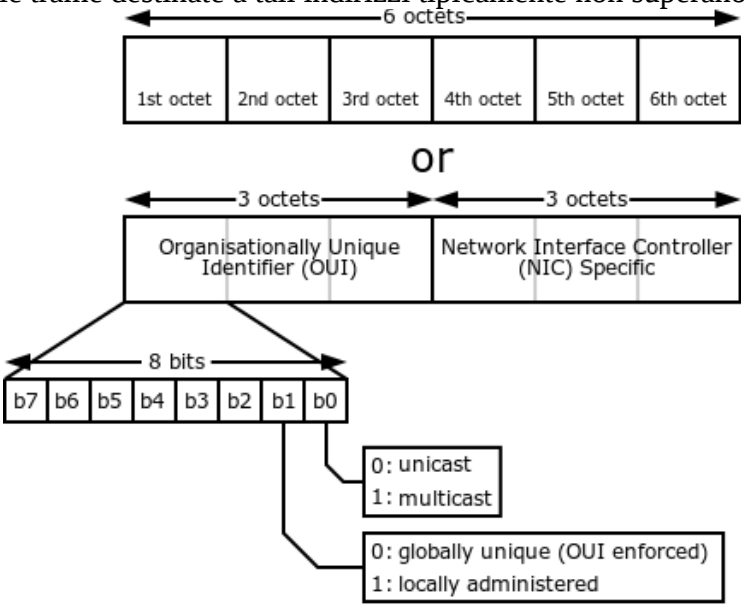
```
# show arp
Protocol  Address  Age(min)   Hardware Addr   Type    Interface
Internet  10.2.2.1      -     30e4.dbb7.7e02  ARPA    GigabitEthernet0/0/2
Internet  10.2.2.2    253     0004.c01d.7c1a  ARPA    GigabitEthernet0/0/2
```



Scendiamo quindi nel dettaglio del bit, ed analizziamo semantica e sintassi dei campi Ethernet che hanno un ruolo nel contesto ARP.
La generica struttura dalla trama Ethernet è illustrata qui a fianco, e prevede 6 campi di base [17]. Nota: Ethernet trasmette sul medium prima i Byte più significativi (ma, all'interno di ogni Byte, prima i bit meno significativi) [18].

La struttura e i contenuti dei campi della trama Ethernet, nelle sue mille varianti, sono oggetto di interi volumi di autori ben più attrezzati dello scrivente, e tuttavia - senza appesantire la trattazione al di là del necessario - sia sufficiente qui chiarire il ruolo dei seguenti campi Ethernet:

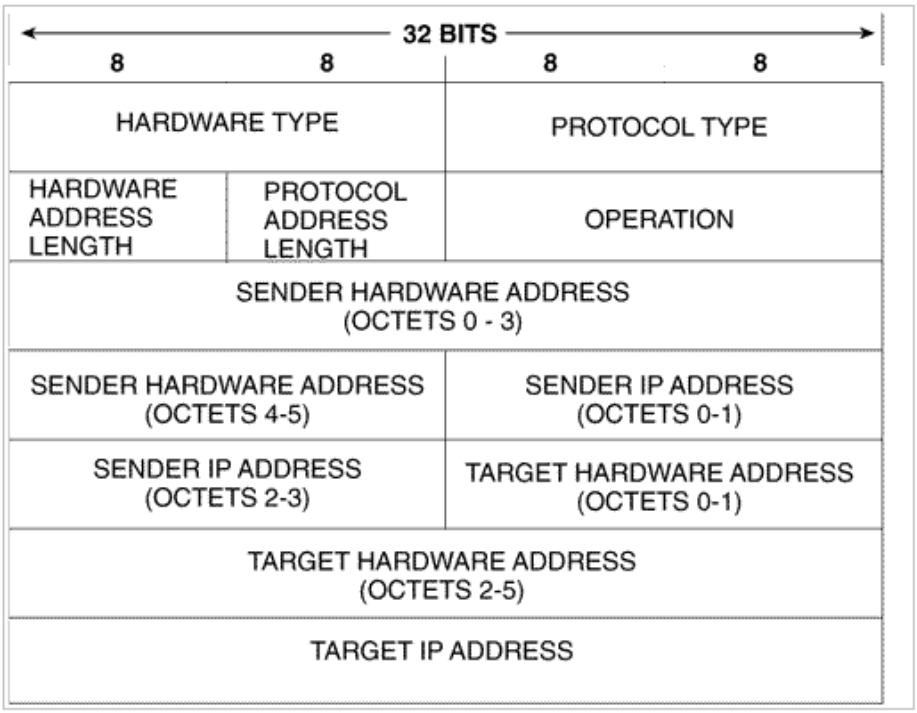
- ***MAC-address***: sono sempre composti da 6Byte (o 48bit); esistono 3 tipi di indirizzi MAC:
  - ***Unicast***: identifica al più 1 scheda di rete; è un indirizzo composto di 2 parti:
    - *Organizational-Unique Id (OUI)*, 3Byte alti: anche detti *Vendor-Id*, sono rilasciati dall'IEEE ai vari produttori, e sono globalmente univoci [19]; Nota: in realtà non tutti i bit dell'OUI sono utilizzabili per l'indirizzamento, poichè il bit meno significativo del Byte più significativo (che è il primo bit di indirizzo trasmesso sul medium) è riservato per indicare se quello che segue è un indirizzo di unicast (**bit_0=0**), o no (bit_0=1), come illustrato sotto [20].
    - *Network-Interface-Card Id (NIC-id)*, 3Byte bassi: sono rilasciati dal Vendor, senza garanzia di univocità locale. Nota: non tutte le schede di rete sono dotate di MAC-address, e.g.: alcuni *switch* e *hub* non li hanno, poichè tipicamente non sono nè *source* nè *destination* di trame (sono dispositivi "trasparenti" a L2).
  - *Multicast*: identifica più di 1 scheda di rete (ma non rileva, in questa sede).
  - ***Broadcast****=0x**FF:FF:FF-FF:FF:FF***: identifica tutte le interfacce di rete "in ascolto", connesse al medium (che infatti è detto "dominio di broadcast" di L2); le trame destinate a tali indirizzi tipicamente non superano i *router* L3;



Nota: il tipo di MAC-address utilizzabile nei campi di indirizzo della trama Ethernet (e, più in generale, in ogni campo di indirizzo MAC), vale il principio generale che il *source* MAC-address debba essere di unicast (poichè è sempre noto a chi genera la trama), mentre il *destination* MAC-address può essere di tutti i tipi (1 alla volta).

- ***EtherType***: identifica il protocollo di livello superiore (L>2) incapsulato nel payload L2 (questa informazione è necessaria al *de-multiplexing* in ricezione, verso più protocolli L>2); se il payload trasporta un messaggio ARP, il campo *EtherType=**0x0806*** (il Byte#1=*0x08* è il primo dei due ad essere trasmesso sul medium; segue il Byte#2=*0x06)*.

Il contenuto del payload della suddetta trama Ethernet è il pacchetto ARP, che ha questa struttura:



I pacchetti ARP hanno una struttura prefissata, intesa come successione predefinita di campi, ma -caso insolito nel mondo dei protocolli di basso livello- alcuni dei suoi campi (e.g.: *Hsrc*, *Psrc*, *Hdst*, *Pdst*) hanno lunghezze variabili, in base alla lunghezza degli indirizzi dei protocolli L2+L3 che ARP collega: nel caso (Ethernet+IP), il pacchetto ARP ha dimensione caratteristica **28 Byte**, così strutturati [5]:

- ***Htype*** (2 Byte): identifica il protocollo L2, e.g.: per Ethernet, *Htype=0x0001*;
- ***Ptype*** (2 Byte): identifica il protocollo L3, e.g.: per IPv4, *Ptype=0x0800* (stessa codifica del campo *EtherType* di Ethernet);
- *Hlen* (1 Byte, valore dipende da *Htype*): lunghezza in Byte degli indirizzi L2, e.g.: per i MAC-address, *Hlen*=6;
- *Plen* (1 Byte, valore dipende da *Ptype*): lunghezza in Byte degli indirizzi L3, e.g.: per gli IP-address, *Plen*=4;
- ***Oper*** (2 Byte): identifica l'operazione implementata, ossia il tipo di messaggio ARP:
  - ***Request*** (*Oper*= **1**): interroga gli altri nodi per conoscere quale di essi risponda ad un dato L3-addr; tipicamente è trasportata su un ***broadcast*** L2, e.g.: una trama Ethernet con *destination MAC-address=0xFF:FF:FF-FF:FF:FF;*
  - ***Reply*** (*Oper*= **2**): tipicamente risponde alla Request con un *unicast* L2, ma è possibile, accettabile, e a volte obbligatoria, una risposta in broadcast L2 [21];

- ***Hsrc*** (dimensione variabile secondo *Htype*, e.g.: se *Htype=0x0001*, allora *Hsrc* è di 6Byte): identifica un indirizzo L2 come source (unicast); Nota: questo è il campo che, nelle ARP-Reply, contiene la risposta alla *query* che ha originato la ARP-Request;
- *Psrc* (dimensione variabile secondo *Ptype*, e.g.: se *Ptype=0x0800*, allora *Psrc* è di 4Byte): identifica un indirizzo L3 come source.
- ***Hdst*** (dimensione variabile secondo *Htype*, e.g.: se *Htype=0x0001*, allora *Hdst* è di 6Byte): identifica un indirizzo L2 come destination; Nota: questo campo, nelle ARP-Request, è ignoto, ed è quindi riempito con un dati di *padding*;
- *Pdst* (dimensione variabile secondo *Ptype*, e.g.: se *Ptype=0x0800*, allora *Pdst* è di 4Byte): identifica un indirizzo L3 come destination;

Illustriamo ora come avviene la risoluzione ARP degli indirizzi L3-to-L2 nei casi notevoli:

Caso-base:

- Il classico *neighbor-discovery* ARP avviene così [22]:
  1. nel caso più semplice (e frequente) un host ha già in ARP-cache il MAC-address corrispondente all'IP-address da contattare, e perciò risolve localmente la query, senza generare traffico di *overhead* sul medium;
  2. quando invece un host deve popolare l'ARP-cache (e.g.: dopo *reboot*), la procedura si innesca al momento dell'invio del primo datagramma IP (L3) verso una nuova destinazione: esso andrebbe incapsulato nella trama Ethernet (L2) ma ciò risulta impossibile, poichè non si conosce ancora il valore da dare al suo destination MAC-address; in tal caso il driver della scheda scarta silenziosamente il datagramma IP (lasciando ai protocolli L>2 la decisione e l'onere di ri-trasmettere il pacchetto [5]), e inizia invece la procedura di risoluzione ARP, inviando in broadcast L2 la *ARP-Request* con il destination IP-address richiesto (il campo *Hdst*, ossia il destination MAC-address ignoto, è valorizzato tipicamente con lo stesso valore del destination MAC-address della trama, ossia *0xFF:FF:FF-FF:FF:FF* [23], ma ciò non è obbligatorio (l'RFC-826 lascia libertà in questo senso) ed esistono casi (e.g.: DHCP, Zeroconf, etc.) in cui tale campo va invece valorizzato con *0x**00:00:00-00:00:00*** [23] [24]: ciò ha storicamente portato ad considerare impropriamente tale indirizzo come sinonimo di broadcast L2 [25];
  3. in condizioni ideali, il broadcast L2 della ARP-Request è ricevuto da tutti gli host "in ascolto":
     1. il driver della scheda di rete passa il traffico ARP al modulo ARP, che innanzitutto aggiorna la ARP-cache locale (nel campo MAC-address) con le informazioni prese dal campo *Hsrc* della ARP-Request (solo per IP-address già in ARP-cache [5]); questo aggiornamento rinfresca il TTL della entry;
     2. se il campo *Pdst* corrisponde al proprio IP-address (sulla scheda di rete che ha ricevuto la ARP-request), allora il modulo ARP deve rispondere con una ARP-Reply, che viene forgiata completando i campi (implicitamente) incompleti (e.g.: *Hdst*) e invertendo i campi omologhi (src,dst) della ARP-Request; infine la ARP-Reply viene inviata in unicast L2 al richiedente;
  4. in condizioni ideali, il richiedente riceve la ARP-Reply e completa la propria ARP-cache con i dati forniti (rinfrescando anche il relativo TTL), e infine scarta il pacchetto ARP; se i protocolli di L>2 attiveranno la ri-trasmissione del datagramma (inizialmente scartato), essa ora troverà una pronta risoluzione dell'indirizzo L3-to-L2.

Casi speciali (ma *legit*):

- Il meccanismo di *L3 Address-Conflict-Detection*, o *Duplicate-Address-Detection* (DAD) usa speciali messaggi, detti ***ARP-probe***, in cui il campo *Psrc*=(src IP-address)= $[\mathbf{00...0}]_{BIN}$ (tipicamente inviato dagli host che hanno appena ricevuto un IP-address come *DHCP-lease* dal *DHCP-server*, per verificare, prima dell'utilizzo, se altri host usano tale IP-address, ed evitare così duplicati) [21] [24] [26];

- L'*ARP-**Announcement***, anche detto messaggio ***Gratuitous**-ARP* (***GARP***) [7], è la notifica non sollecitata dei propri indirizzi L2/L3, usata legittimamente in vari scenari:
  - in Zeroconf, per difendere il proprio *link-local IP-address* (in 169.254.0.0/16);
  - in contesti di *failover/HA-clusters/hot-standby*, per il *take-over* dell'IP-address del server guasto da parte del suo backup [23];

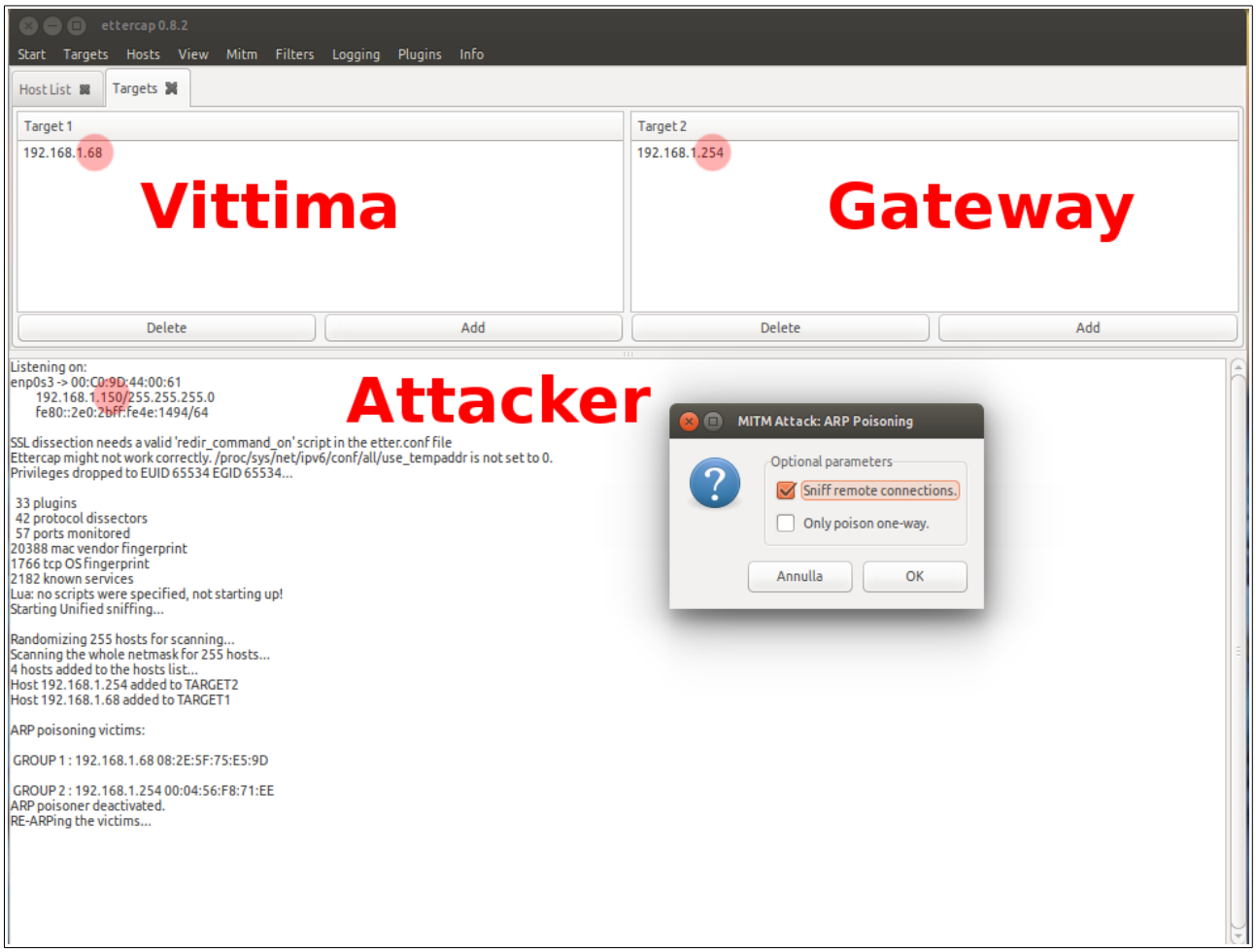
Non è consentito disattivare l'ascolto delle notifiche ARP, poichè il protocollo impone esplicitamente di accettare *tutto* il traffico ARP in arrivo:

- "*When an address resolution packet is received, the receiving Ethernet module gives the packet to the Address Resolution module [...]; if the pair <protocol-type, sender-protocol-address> is already in my translation table, **update the sender-hardware-address** field of the entry with the new information in the packet [...]*" [5];
- "*... any node receiving any ARP packet (Request or Reply) **MUST update** its local ARP cache with the Sender Protocol and Hardware Addresses in the ARP packet, if the receiving node has an entry for that IP address already in its ARP cache. This requirement in the ARP protocol applies even for ARP Request packets, and for ARP Reply packets that do not match any ARP Request transmitted by the receiving node*" [27].

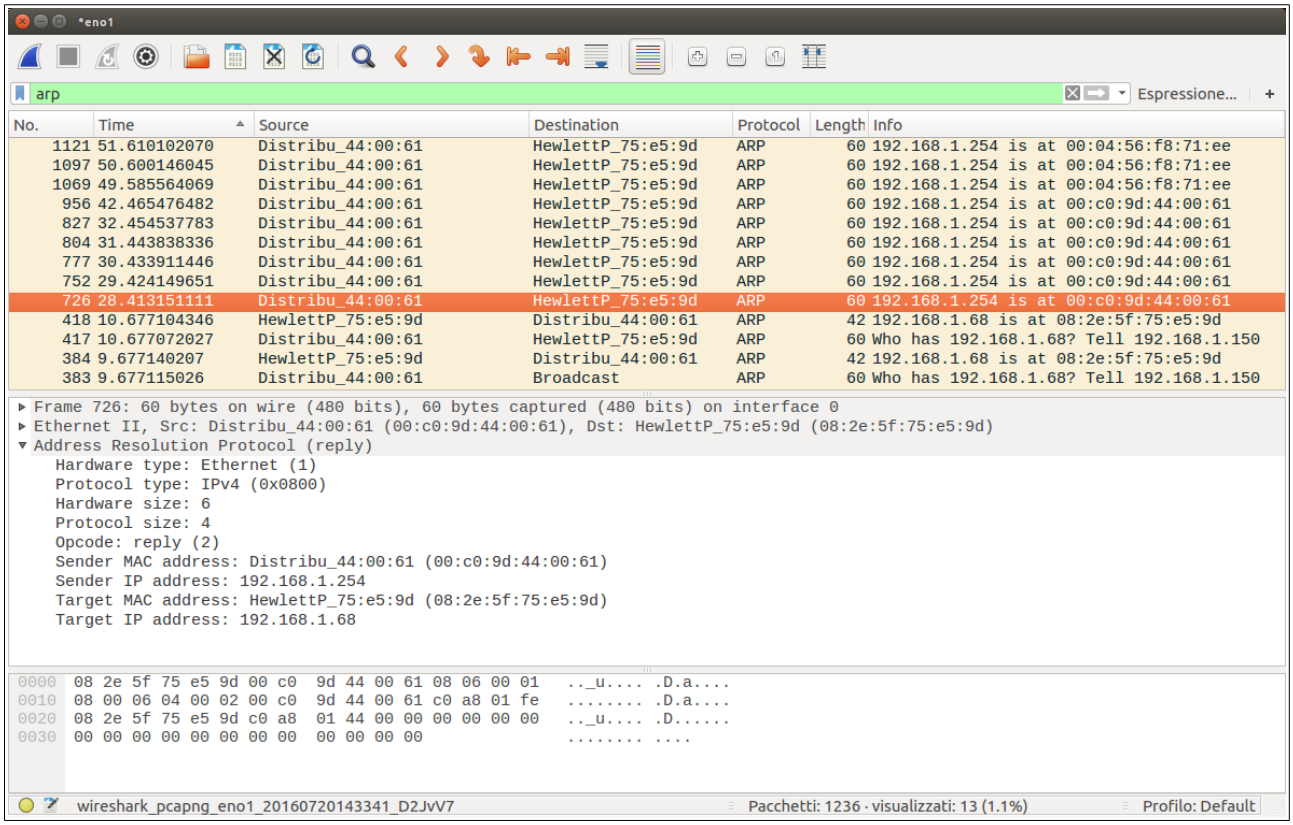
1.4 - ARP-Poisoning/-Spoofing

Poichè -come abbiamo visto- ARP non è autenticato, chiunque abbia accesso **fisico** allo *shared-medium* (e.g.: accesso allo switch Ethernet, o più semplicemente connessione WiFi) può forgiare e diffondere pacchetti ARP con informazioni arbitrarie; ovviamente ciò rende qualsiasi host dotato di ARP (e.g.: IPv4) passibile di attacchi malevoli, detti *spoofing*, o *poisoning*, che permettono ad un attaccante di modificare remotamente la ARP-cache di un host-vittima, per i più svariati scopi (per lo più, illeciti).

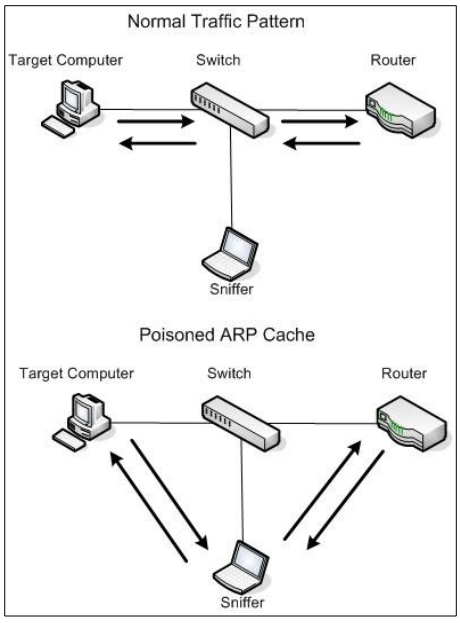
Lo strumento che utilizziamo per questo tipo di attività è un *packet-injector* (e.g.: ***Ettercap***):



Ecco di seguito il *packet-capture* di una semplice sessione di 2-way ARP-poisoning (2-way poichè "avveleniamo" contemporaneamente l'host-vittima e il suo default-gateway, sostituendoci alla rispettiva controparte), catturata tramite un *packet-sniffer* (i.e.: ***Wireshark***):



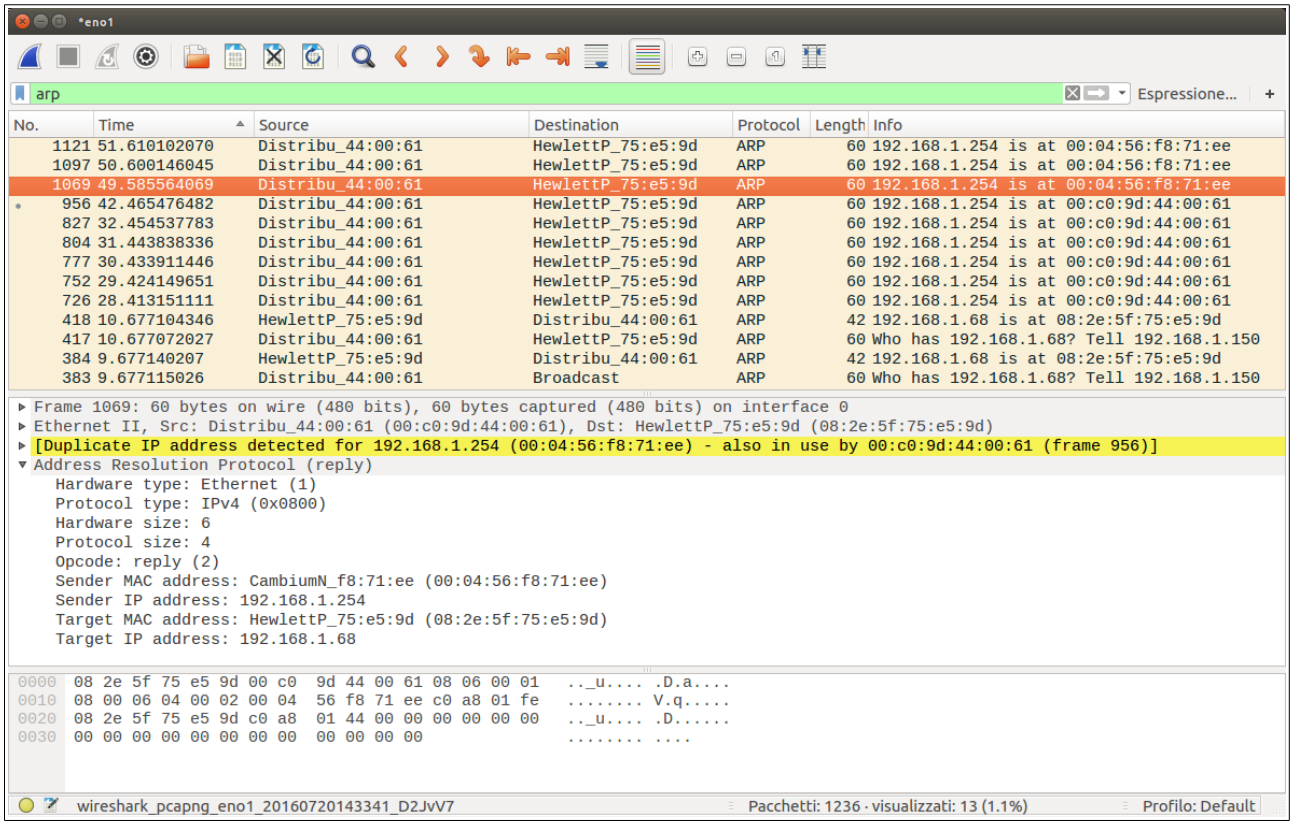
- inizialmente l'host-vittima (08:2e:5f:75:e5:9d, 192.168.1.68) partecipa ai normali scambi ARP-Request/Reply (t<28);
- all'istante (t=28.4..) evidenziato in figura, l'ARP-Poisoning inizia, utilizzando messaggi GARP (ARP-Reply non-sollecitati), che inducono la vittima ad aggiornare la ARP-cache sostituendo il gateway legit (00:04:56:f8:71:ee, 192.168.1.254) con quello dell'attaccante (00:c0:9d:44:00:61); contemporaneamente una simmetrica ARP-Reply (qui non rappresentata) è inviata anche al gateway, per sostituirsi alla vittima (nell'ARP-cache del gateway);
- da questo momento in poi, la vittima spedisce tutto il traffico (destinato al gateway) all'attaccante, che può farne ciò che vuole (e.g.: catturarlo, analizzarlo off-line, etc.) [3].



Nota: esistono molte **varianti** di ARP-poisoning, tra cui:

- *ARP-Request* fittizia, in ***unicast*** diretto all'unico host che sappiamo già potrà (e dovrà) rispondere [28];
- *ARP-Request* fittizia, con {Hdst=$[00...0]_{BIN}$, Pdst:= Psrc= (src-IP-address)};
- *ARP-Reply* fittizia, con {Hdst:= Hsrc, Pdst:= Psrc};

Al termine della sessione di ARP-poisoning, l'attaccante "rimette le cose al loro posto", per non lasciare tracce visibili (e.g.: evitando errori, disconnessioni, o malfunzionamenti palesi, sull'host-vittima):



- Il *de-poisoning* dell'ARP-cache avviene con una ARP-Reply (t=49.5..) funzionalmente simmetrica a quella iniziale, ossia tale da indurre la vittima a ripristinare l'associazione del default-gateway legittimo (00:04:56:f8:71:ee, 192.168.1.254);
- fatto questo, l'attaccante può allontanarsi senza lasciare tracce (a patto che nessuno abbia monitorato il traffico ARP ...).

1.5 - Attacchi MitM

Naturalmente l'ARP-poisoning non è fine a se stesso, ma è finalizzato a dirottare (cfr: *hijacking*) il traffico destinato ad altri, verso nuove destinazioni, tipicamente sotto il controllo dell'attaccante. Allo stesso tempo però l'attaccante deve nascondere la propria intrusione agli occhi della vittima, che deve continuare le proprie attività in rete senza malfunzionamenti: ciò tipicamente si ottiene ruotando/girando tutto il traffico della vittima (giunto all'attaccante), verso il gateway originale, il quale provvederà ad inoltrarlo alle destinazioni richieste dalla vittima (le quali poi risponderanno, passando attraverso gateway ed attaccante, fino a raggiungere la vittima stessa).

La sequenza di passaggi iniziale:

Vittima ⇄ Gateway ⇄ Remote-Peers

diventa quindi:

Vittima ⇄ (**Attaccante**) ⇄ Gateway ⇄ Remote-Peers

Da questa topologia prende il nome la famiglia di attacchi che ne deriva, detti appunto *Man-In-The-Middle* (MitM), in cui l'attaccante funge da *last-hop gateway* verso la vittima, sostituendosi al router *legit*.

Lo strumento-base per questo tipo di attacchi è semplicemente il *kernel Linux*, nella misura in cui possiamo tramutarlo in un router IP: se questa opzione è stata abilitata al momento della compilazione del kernel Linux, per attivarla è sufficiente 1 comando (da utente *root*):

```
gf@silenzio:~# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Fatto questo (e iniziato l'ARP-poisoning come descritto nel precedente capitolo), stiamo già portando un attacco MitM, ma non ne stiamo ancora traendo alcun vantaggio. Ciò che manca è la capacità di ispezione del traffico passante: per fare ciò esistono moltissimi strumenti, e.g.: dal semplice *tcpdump* (per l'analisi off-line del traffico), ai più comuni ***driftnet***, *urlsnarf, dsniff, msgsnarf* (per la visualizzazione in real-time del contenuto di immagini, URL, query DNS, o *instant-messaging* della vittima) [29].
Fin qui abbiamo solo visto come ascoltare (in *read-only mode*) il traffico altrui, ma nulla (a parte la Legge e la Morale) ci impedisce, a questo punto, di inserirci attivamente nelle comunicazioni della vittima, alterandole, ad esempio aggiungendo (o togliendo) a piacere.

Ancora una volta: esiste ampia letteratura di pubblico dominio, riguardo ai modi in cui è possibile sfruttare la posizione privilegiata di gateway per carpire illecitamente le informazioni riservate immerse nel traffico passante (anche qualora fossero crittografate, nel caso disponessimo di sufficiente potenza di calcolo e tempo), ma in questa sede ci interessa di più trattare delle contromisure.

1.6 - ArpWatch

In contesti IPv4, e soprattutto dove il mezzo fisico è intrinsecamente insicuro (e.g.: WiFi e LAN pubbliche di *cyber-cafè*, biblioteche, università, aeroporti, ...), l'unica contromisura agli attacchi di ARP-poisoning (ad eccezione del *mapping* statico degli indirizzi L3-to-L2, poco praticabile) è il monitoraggio passivo della rete (per rilevare nuovi MAC-address, e/o il cambiamento delle associazioni IP-address/MAC-address, specialmente dei gateway). Uno dei sw più noti e longevi [30] in questo ruolo di rilevamento delle intrusioni è *Arpwatch*: un progetto *open-source* (scritto in C) del *Lawrence Berkeley National Laboratory* (LBNL), un ente governativo statunitense [31] per la ricerca scientifica e tecnologica.

Il funzionamento di arpwatch è semplice: esso implementa un servizio/demone Linux, leggero, che gira in background, senza GUI, senza permessi amministrativi, e rileva tutte le condizioni anomale del traffico ARP su una data interfaccia di rete, segnalandole via email e/o *syslog* all'amministratore di sistema.
Nota: ogni istanza di arpwatch ascolta il traffico su 1 sola interfaccia; tuttavia è possibile lanciare più istanze concorrenti, su più interfacce di rete [32].

La riga di comando di arpwatch dispone delle seguenti opzioni [33]:

```
gf@silenzio:~# arpwatch [-dN] [-f datafile] [ -i interface ] [-n net[/width]]
[-r file] [-s sendmail_path] [-p] [-a] [-m addr] [-u username] [-R seconds] [-
Q] [-z ignorenet/ignoremask]
```

il cui utilizzo è chiaramente spiegato nel manuale di sistema ("*man arpwatch*"), e di cui non serve fare la copia qui. Le opzioni di cui abbiamo avuto bisogno, e di cui invece ha senso riportare il manuale, sono:

- *[-**d**]*: enable **debugging**; this also inhibits forking into the background and emailing the reports; instead, they are sent to *stderr*;
- *[-N]*: disables reporting any **bogons** (source ip-address not local to the local subnet);
- *[-**i** interface]*: to override the default **interface**;

I file di arpwatch con cui l'amministratore di sistema deve interagire sono i seguenti:

Path-name:	Esempi [34]:

/etc/arpwatch.conf	`# /etc/arpwatch.conf: Debian-specific way to watch multiple i/faces`
	`# Format of this configuration file is:`
	`#<dev1> <arpwatch options for dev1>`
	`#<dev2> <arpwatch options for dev2>`
	`# ...`
	`# global options (for all interfaces) in: /etc/default/arpwatch`
	`# For example:`
	`#eth0  -m root`
	`#eth1  -m root`
	`eth0 -a -n 192.168.0.0/24 -m admin@mydomain.local`

/etc/default/arpwatch	`# Global options for arpwatch(8). Default for all interfaces.`
	`# Debian: don't report bogons, don't use PROMISC.`
	`#ARGS="-N -p"`
	`# it restarts in 2 min, if net-interface went down`
	`ARGS="-R 120"`
	`# Debian: run as 'arpwatch' user.  Empty this to run as root.`
	`RUNAS="arpwatch"`

/etc/init.d/arpwatch	`#!/bin/sh`
	`# start-up daemon-launching script`
	`...`

/usr/sbin/arpwatch	`ELF executable file (main logic)`

/usr/share/arpwatch/ethercodes.dat	`# MAC-address OUI database:`
	`0:0:0  XEROX CORPORATION`
	`...`
	`e4:1f:13  IBM`
	`e8:a4:c1  Deep Sea Electronics PLC`
	`e8:b:13  Akib Systems Taiwan, INC`
	`ec:30:91  Cisco Systems`
	`ec:6c:9f  Chengdu Volans Technology CO.,LTD`
	`f0:bc:c8  MaxID (Pty) Ltd`
	`f0:de:71  Shanghai EDO Technologies Co.,Ltd.`
	`f4:ac:c1  Cisco Systems`

/var/lib/arpwatch/arp.dat
or [35]
arp.dat-
or
arp.dat.new
	`# local history of known MAC-addresses w/ timestamps:`
	`...`
	`08:00:27:fe:29:96  192.168.1.254  1468778956  CPE  eno1`
	`00:13:9b:ee:b9:6f  192.168.1.254  1468773022  CPE  eno1`
	`08:2e:5f:75:e5:9d  192.168.1.68  1468787779  eno1`
	`e8:ce:06:6e:ae:61  192.168.1.119  1467656702  eno1`
	`08:00:27:fe:29:96  192.168.1.119  1467656686  eno1`
	`08:00:27:d1:1d:1d  192.168.1.119  1465846229  eno1`
	`08:00:27:90:19:f1  192.168.1.119  1465841750  vbox  eno1`
	`08:00:27:3d:55:ce  192.168.1.119  1465841054  vbox  eno1`
	`08:00:27:bd:df:d5  192.168.1.119  1465681308  eno1`
	`...`

Gli eventi di rete che scatenano risposte, notifiche o messaggi di arpwatch sono una decina [33], ma vale la pena passarli in rassegna (nei seguenti paragrafi), poichè essi rappresenteranno la chiave di lettura della UI del sw sviluppato (non a caso denominato *ArpWatch Dashboard*).

1.6.1 - Bogon

Arpwatch rileva la configurazione IP della scheda di rete (su cui è configurato per il monitoring), ne deduce la *IP-network*, e quindi chiama "traffico *bogon*" ciò che fa riferimento ad altre IP-network. Purtroppo questo tipo di segnalazione produce spesso falsi-positivi (ossia rumore), poichè ricadono nel caso "bogon" anche molti degli indirizzi IP utilizzati in contesti *legit*, e.g.:

- 0.0.0.0 usato nel *DHCP-Discover* [36];
- 169.254.0.0/16 usato in Zeroconf [37]);
- *IP-misconfiguration*: in una rete (e.g.: accademica) popolata da molti utenti giovani e alle prime armi con le reti, può capitare (spesso) che la configurazione del proprio laptop sia inesatta, e anche questo può generare falsi-allarmi in arpwatch;
- *Multi-route scenario*: nei laptop capita spesso di avere contemporaneamente più di un'interfaccia di rete attiva (e.g.: rete cablata, e WiFi); in tali scenari la *routing-table* dell'OS assegna una diversa metrica/precedenza/costo alle varie rotte, premiando quelle cablate, che così finiscono per trasportare anche traffico "alieno" (e.g.: della rete WiFi);

Sta quindi all'amministratore di rete decidere quale soglia di rumore è tollerabile, attivando o disattivando la relativa flag [-N].

E.g.: Windows

```
C:\Users\gf>route print -4
===========================================================================
Interface List
  2...68 94 23 0b d4 d5 ......Microsoft Wi-Fi Direct Virtual Adapter
  6...08 2e 5f 75 e5 9d ......Realtek PCIe GBE Family Controller
  3...68 94 23 0b d4 d3 ......Ralink RT3290 802.11bgn Wi-Fi Adapter
  9...68 94 23 0b d4 d4 ......Bluetooth Device (Personal Area Network)
  1...........................Software Loopback Interface 1
 10...00 00 00 00 00 00 00 e0 Microsoft ISATAP Adapter
 12...00 00 00 00 00 00 00 e0 Microsoft Teredo Tunneling Adapter
  8...00 00 00 00 00 00 00 e0 Microsoft ISATAP Adapter #2
===========================================================================

IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0    192.168.1.254    192.168.1.145     10
          0.0.0.0          0.0.0.0  192.168.211.211   192.168.211.68     25
        127.0.0.0        255.0.0.0         On-link         127.0.0.1    306
        127.0.0.1  255.255.255.255         On-link         127.0.0.1    306
  127.255.255.255  255.255.255.255         On-link         127.0.0.1    306
      192.168.1.0    255.255.255.0         On-link     192.168.1.145    266
    192.168.1.145  255.255.255.255         On-link     192.168.1.145    266
    192.168.1.255  255.255.255.255         On-link     192.168.1.145    266
    192.168.211.0    255.255.255.0         On-link    192.168.211.68    281
   192.168.211.68  255.255.255.255         On-link    192.168.211.68    281
  192.168.211.255  255.255.255.255         On-link    192.168.211.68    281
        224.0.0.0        240.0.0.0         On-link         127.0.0.1    306
        224.0.0.0        240.0.0.0         On-link     192.168.1.145    266
        224.0.0.0        240.0.0.0         On-link    192.168.211.68    281
  255.255.255.255  255.255.255.255         On-link         127.0.0.1    306
  255.255.255.255  255.255.255.255         On-link     192.168.1.145    266
  255.255.255.255  255.255.255.255         On-link    192.168.211.68    281
===========================================================================
...
```

E.g.: MacOSX

```
poesia:~ gf$ netstat -nr

Routing tables

Internet:
Destination        Gateway            Flags        Refs      Use   Netif Expire
default            192.168.1.254      UGSc           30     3310     en0
default            192.168.211.211    UGScI           1        0     en1
127                127.0.0.1          UCS             0        0     lo0
127.0.0.1          127.0.0.1          UH              2     1637     lo0
169.254            link#4             UCS             0        0     en0
169.254            link#5             UCSI            0        0     en1
192.168.1          link#4             UCS             2        0     en0
192.168.1.68       8:2e:5f:75:e5:9d   UHLWIi          2   945738     en0   1068
192.168.1.69/32    link#4             UCS             0        0     en0
192.168.1.254/32   link#4             UCS             1        0     en0
192.168.1.254      0:4:56:f8:71:ee    UHLWIir        33     5348     en0   1195
192.168.1.255      ff:ff:ff:ff:ff:ff  UHLWbI          0        2     en0
192.168.211        link#5             UCS             0        0     en1
192.168.211.69/32  link#5             UCS             1        0     en1
192.168.211.69     0:1e:52:81:24:eb   UHLWI           0        2     lo0
192.168.211.211/32 link#5             UCS             1        0     en1
192.168.211.211    f8:1a:67:b4:a2:e2  UHLWIir         3      122     en1   1115
...
```

1.6.2 - New Station

La prima volta che arpwatch incontra traffico ARP proveniente da un MAC-address, lo notifica con questo evento. La base di riferimento per questo tipo di verifica è il file */var/lib/arpwatch/arp.dat* o uno degli altri file (*arp.dat-*; *arp.dat.new*; ... [35]) creati nella stessa directory (nel caso di più istanze concorrenti di arpwatch). Durante i primi minuti di esecuzione di arpwatch dopo un'installazione da zero, oppure in contesti intrinsecamente transitori (e.g.: WiFi in aeroporto), c'è da aspettarsi un gran numero di eventi di questo tipo.

1.6.3 - New Activity

Questo evento riguarda le coppie (MAC-address, IP-address) di cui si ha già memoria, ma di cui non si registra attività da almeno 6 mesi.

1.6.4 - Changed Ethernet Address

Questo evento riguarda il caso in cui uno stesso IP-address ha cambiato proprietario (ossia MAC-address): questo può indicare sia attività malevole/sospette (e.g.: *macchanger-gtk*), sia normali *misconfiguration*, sia pure attività *legit* (e.g.: normali avvicendamenti del *lease* DHCP, in particolare con un *pool* di indirizzi sotto-dimensionato rispetto alla finestra di utilizzo).

1.6.5 - Flip Flop

Questo evento è simile al precedente (*changed ethernet address*), da cui si differenzia solo per il fatto di essere ripetuto a breve distanza di tempo con i ruoli invertiti tra i 2 MAC-address in conflitto (da cui il nome, evocativo di un IP-address conteso tra 2 MAC-address, che se lo rubano a vicenda). E' uno dei segnali più chiari di attività *malicious* sulla rete.

1.6.6 - Reused Old Ethernet Address

Questo evento è simile al precedente (*flip flop*), da cui si differenzia solo perchè la contesa dell'IP-address non avviene tra i 2 MAC-address più recenti, ma tra il 1° e il 3°(=un host più vecchio del 2°).

1.6.7 - Ethernet Broadcast

Questo evento segnala i casi in cui il campo di source MAC-address della trama Ethernet e/o il campo *Hsrc* del pacchetto ARP contengono un indirizzo di broadcast L2 (MAC-address con tutti-1, o tutti-0 [33]), il che naturalmente è sempre sospetto, dato che:

- il MAC-address locale è sempre noto al mittente (ma si è scelto deliberatamente di nasconderlo);
- uno dei possibili intenti è quello di generare traffico (a scopo di *flooding/DoS/ARP-storm* [38]) confidando sul fatto che le ARP-Reply verranno erroneamente inviate "in unicast" verso l'indirizzo di broadcast L2;

Curiosamente il manuale di arpwatch riporta 2 volte, e in modo leggermente diverso, questo tipo di segnalazione, come se ve ne fossero 2 distinte ma denominate allo stesso modo; purtroppo non ho trovano nulla che possa confermare o smentire qualsiasi ipotesi a riguardo (da una veloce analisi dei sorgenti C [31], credo che l'ipotesi più probabile sia quella di un refuso nella documentazione).

1.6.8 - IP Broadcast

Questo evento segnala i casi in cui l'IP-address dell'host (di origine?) è di broadcast L3. Anche qui, come nel caso precedente (ma a L3), sembra emergere l'intento di generare risposte multiple.

1.6.9 - Ethernet Mismatch

Questo evento rileva una mancata corrispondenza tra il source MAC-address della trama Ethernet, e l'omologo campo *Hsrc* del pacchetto ARP. Un'attività che genera questo evento è il cambio di MAC-address locale (e.g.: con *macchanger-gtk*) durante una sessione di *arping*.

1.6.10 - Suppressed DECnet Flip Flop

Questo evento rileva un flip flop riconducibile ad un modello di schede di rete note per avere problemi di funzionamento.

A titolo meramente esemplificativo, e senza presunzione di completezza, si riportano qui di seguito alcuni estratti dei 2 distinti mezzi con cui gli eventi arpwatch vengono inviati all'attenzione dell'amministratore di sistema: syslog, e email.

Syslog:

```
...
May 26 18:52:49 silenzio arpwatch: listening on wlo1
May 26 18:53:53 silenzio arpwatch: chdir(/var/lib/arpwatch): Permission denied
May 26 18:53:53 silenzio arpwatch: (using current working directory)
May 26 18:53:53 silenzio arpwatch: pcap open wlo1: wlo1: You don't have permission to capture on that device
(socket: Operation not permitted)
May 26 18:56:15 silenzio arpwatch: new station 192.168.211.102 00:1e:52:81:24:eb wlo1
May 26 19:01:38 silenzio arpwatch: new station 192.168.211.103 00:1e:52:81:24:eb wlo1
May 26 19:04:54 silenzio arpwatch: new station 192.168.211.104 00:1e:52:81:24:eb wlo1
May 26 19:09:06 silenzio arpwatch: new station 192.168.211.69 00:1e:52:81:24:eb wlo1
May 26 19:09:06 silenzio arpwatch: bogon 0.0.0.0 00:1e:52:81:24:eb wlo1
May 26 19:09:07 silenzio arpwatch: bogon 0.0.0.0 00:1e:52:81:24:eb wlo1
May 26 19:11:10 silenzio arpwatch: message repeated 3 times: [ bogon 0.0.0.0 00:1e:52:81:24:eb wlo1]
May 26 19:11:11 silenzio arpwatch: new station 192.168.211.109 00:1e:52:81:24:eb wlo1
May 26 19:11:11 silenzio postfix/pickup[1276]: 517231C2DCA: uid=0 from=<root>May 26 19:11:11 silenzio
postfix/cleanup[5233]: 517231C2DCA: message≠id=<20160526171111.517231C2DCA@silenzio>
May 26 19:11:11 silenzio postfix/qmgr[2560]: 517231C2DCA: from=<root@silenzio>,size=557, nrcpt=1 (queue active)
May 26 19:11:11 silenzio postfix/local[5235]: 517231C2DCA: to=<root@silenzio>, orig_to=<root>, relay=local,
delay=0.06, delays=0.04/0.01/0/0.01, dsn=2.0.0, status=sent (delivered to mailbox)
May 26 19:11:11 silenzio postfix/qmgr[2560]: 517231C2DCA: removed
May 26 19:11:19 silenzio arpwatch: bogon 0.0.0.0 00:1e:52:81:24:eb wlo1
May 26 19:12:26 silenzio arpwatch: message repeated 5 times: [ bogon 0.0.0.0 00:1e:52:81:24:eb wlo1]
May 26 19:12:53 silenzio 00:00:00:00:00
May 26 19:13:06 silenzio arpwatch: bogon 0.0.0.0 00:1e:52:81:24:eb wlo1
May 26 19:13:07 silenzio arpwatch: message repeated 2 times: [ bogon 0.0.0.0 00:1e:52:81:24:eb wlo1]
May 26 19:15:39 silenzio arpwatch: bogon 0.0.0.0 00:1e:52:81:24:eb wlo1
May 26 19:15:40 silenzio arpwatch: message repeated 2 times: [ bogon 0.0.0.0 00:1e:52:81:24:eb wlo1]
May 26 19:15:42 silenzio arpwatch: new station 192.168.211.244 f0:92:1c:9a:e6:af wlo1
May 26 19:28:09 silenzio arpwatch: new station 192.168.211.100 cc:f3:a5:5f:c6:ec wlo1
May 26 22:07:02 silenzio arpwatch: new station 192.168.211.101 00:bb:3a:63:67:db wlo1
May 26 22:38:15 silenzio arpwatch: exiting
...
```

Email:

```
Date: Sat, 11 Jun 2016 23:50:33 +0200 (CEST)
From: Arpwatch silenzio <arpwatch@silenzio>
To: root@silenzio
Subject: changed ethernet address (192.168.1.119) eno1
             hostname: <unknown>
           ip address: 192.168.1.119
            interface: eno1
    ethernet address: 08:00:27:fe:29:96
     ethernet vendor: CADMUS COMPUTER SYSTEMS
old ethernet address: 08:00:27:bd:df:d5
 old ethernet vendor: CADMUS COMPUTER SYSTEMS
           timestamp: Saturday, June 11, 2016 23:50:32 +0200
  previous timestamp: Saturday, June 11, 2016 23:41:48 +0200
               delta: 8 minutes
```

§ 2 - Specifiche

Chiarito il contesto generale in cui si inserisce questo progetto, illustriamo ora le richieste del Committente (il Relatore della presente tesi), al fine di documentare il processo di trasformazione di tali richieste (formulate in modo discorsivo e con i margini di approssimazione tipici della normale comunicazione verbale), in una solida base di partenza condivisa, completa, coerente, e compatibile con le risorse in campo. Un vero e proprio *design by contract*.

A tale scopo, in ossequio ai principi di ***Ingegneria del Software***, abbiamo compilato insieme una griglia che riassumesse i 2 "tagli" ortogonali (ossia indipendenti) che volevamo dare ai *requirements*, ossia:

- Specifiche **Funzionali** -vs- Specifiche **Non-Funzionali**: questa suddivisione è guidata dal dominio in cui le specifiche si muovono, e lo partiziona in:
  - L'ambito funzionale risponde alle domande: *cosa deve fare il sw? come deve rispondere agli input per produrre i suoi output?*
  - L'ambito non-funzionale risponde alle domande: *che proprietà deve avere il sw? in quale contesto operativo si deve inserire? quali sono le forze progettuali in campo?* [39]

- Metrica delle **priorità**: questa suddivisione è guidata dalla percezione soggettiva che il Committente ha dell'impatto che un'eventuale insuccesso locale (alla singola richiesta) avrà sul successo globale del progetto; in altre parole: quanto ritiene importante un requisito all'interno del progetto? Per consentire una formalizzazione delle priorità si è utilizzato lo standard di classificazione delle specifiche denominato ***MoSCoW*** [40], che prevede 4 livelli (*Must-Should-Could-Won't)*, ma soprattutto distingue i requisiti assolutamente inderogabili (pena il fallimento del progetto), da ciò che invece è opzionale e perciò rappresenta lo spazio di manovra per aumentare la soddisfazione del Committente (senza però compromettere il progetto, in caso di assenza).

2.1 - Specifiche Funzionali

Ad un primo livello di approssimazione, la richiesta fondamentale era quella di un sistema distribuito che permettesse il *monitoring* centralizzato di una pluralità di sonde/agent (distribuiti in varie LAN) che osservassero il traffico ARP e riportassero le attività sospette, le quali fossero poi navigabili/filtrabili da un pannello di controllo visuale (GUI), di facile accesso.

Già da questa formulazione è possibile individuare **3 entità/ruoli** distinti, che diventeranno altrettanti moduli sw (in ossequio alle prescrizioni di alta **coesione** funzionale [39]):

- **Agent**: è l'elemento periferico, replicabile a piacere, nel sistema distribuito;
- **Server**: è l'elemento centrale, unico, cui afferiscono tutte le informazioni degli agent, e in cui viene concentrata la business-logic applicativa (in modo assimilabile ad un Controller+Model, nel famoso pattern architetturale MVC [41]);
- **Dashboard**: è il modulo che dà una View al server, per completare così il trittico MVC.

Le specifiche di tali entità sono espresse qui di seguito, tramite i rispettivi *requirement*, marcati con le diverse priorità MoSCoW:

Entità:	Requisiti **Funzionali**	MoSCoW
Agent	deve rilevare i pacchetti ARP associati agli attacchi di ARP-spoofing;	M
	deve segnalare gli eventi positivi al server remoto;	M
	deve segnalare periodicamente il proprio buono stato di salute (keepalive);	M
	deve avere una configurazione locale costituita da un semplice file di testo (nel classico formato: 1 param per line; param=value; line-comments iniziano con #);	M
	dovrebbe segnalare gli eventi positivi in modo immediato e async (no timer);	S
Server	deve ricevere le segnalazioni degli Agent (inviate ad una porta TCP nota);	M
	deve salvare le segnalazioni su uno storage persistente (db);	M
	deve consentire la consultazione remota (via web) della lista delle segnalazioni;	M
	dovrebbe permettere vari livelli di dettaglio, selezionabili dal Viewer;	S
	dovrebbe filtrare le segnalazioni, riducendo il rumore, ed evidenziando le segnalazioni significative (e.g.: missed keepalives, flip-flop ...);	S
	potrebbe avere un file di configurazione, con cui specificare le credenziali;	C

Entità:	Requisiti **Funzionali**	MoSCoW
Dashboard	deve mostrare la lista degli agent e degli eventi ARP rilevanti	M
	deve permettere di ridurre la complessità e il rumore, attraverso ordinamenti e semplici filtri che riducano l'affollamento della UI;	M

2.2 - Specifiche Non-Funzionali

Le *forze progettuali* di un progetto di Ingegneria del sw sono tutti quei vincoli che non riguardano le funzionalità del sw, nel senso matematico di funzione (con un input ed un output), ma invece regolano il contesto in cui il sw opera: progettazione, esecuzione, manutenibilità, portabilità, performance, qualità, stabilità, ambiente/*environment*, infrastruttura/*framework*, etc. [39].

Ecco di seguito i punti salienti emersi in fase preliminare:

Entità:	Requisiti **Non-Funzionali**	MoSCoW
Agent	deve avere un file di configurazione locale, con cui specificare il server remoto;	M
	deve essere sviluppato in un linguaggio moderno, object-oriented (e.g.: Python, Java), facilmente portabile (e.g.: VM), e multipiattaforma in grado di girare su OS Windows (7) e Linux (Ubuntu);	M
	deve ri-utilizzare un motore di rilevamento ARP [30] di terzi, stabile, collaudato, documentato, e gratuito (in particolare sono stati raccomandati: arpwatch, e Xarp [42])	M
	dovrebbe avere un impatto minimo su performance di sistema;	S
	potrebbe avere un installer automatico;	C
	non utilizzerà la modalità promiscua, ma ascolterà solo il normale traffico ARP, su 1 interfaccia di rete;	W
Server	dovrebbe avere un impatto ragionevole su performance di sistema;	C
	potrebbe evitare il riempimento del disco, eliminando i log più vecchi;	C
Dashboard	deve produrre viste/UI in HTML5 standard, con JS e CSS3; visualizzabile dai maggiori browser;	M
	connettendosi al server in HTTP su una porta TCP custom (specificabile nell'URL);	M
	non ha caratteristiche di persistenza;	W

È interessante notare che nel corso dello sviluppo è emersa più chiaramente un'incompatibilità, inizialmente nascosta, tra il requisito di portabilità multi-piattaforma e il requisito di utilizzo di un motore di rilevamento scelto tra i 2 raccomandati: in particolare è emerso un netto divario tra la ricchezza di documentazione relativa ad arpwatch (open-source), e la scarsità relativa ad Xarp

(proprietario), senza parlare del fatto che quest'ultimo sw ha una licenza d'uso *shareware* (*free* ma non *open-source*), con funzionalità limitate (non nel tempo), e non permette alcuna indagine (e.g.: *reverse-engineering*), né il *bundle* con sw di terze parti.

Tutto ciò ne ha definitivamente decretato l'esclusione, a favore di arpwatch, che però attualmente gira "solo" su Linux.

Riportiamo di seguito la licenza d'uso di Xarp, estratta dal pacchetto/installer (.deb) per Linux Ubuntu, evidenziandone i passaggi critici:

Copyright XArp 2002-2011 Christoph Mayer
End user license agreement

This End User License Agreement (EULA) is a legal agreement between you (an end user: either an individual or a single legal entity) and Christoph Mayer, for the XArp software product, including any associated media, printed materials and electronic documentation (the "Software"). This EULA also applies to any updates, add-on components, patches and/or supplements for the XArp software product. By installing or otherwise using the Software you agree to the terms of this EULA. If you do not agree to the terms of this EULA, do not install or use the Software. Instead, you should remove the Software from your computer. The Software is owned by Christoph Mayer and is protected by intellectual property laws and treaties. The Software is licensed, not sold.

** SHAREWARE EVALUATION*
** RETAIL PURCHASE*
** GRANT OF LICENSE*
** SUPPORT*
** LIABILITY LIMITATION*

SHAREWARE EVALUATION

*The Software may be obtained as **functional-limited** software under the Shareware Evaluation terms of this EULA. These terms permit an unlimited shareware evaluation period. The purpose of the shareware evaluation period is to encourage you and others to try the functiona-limited Software before you buy it. The Shareware Evaluation terms do not apply to Retail copies of the Software (see below).*

To unlock full functionality of the Software you must purchase a Registration Key. If you decide for any reason that you do not want to purchase a Registration Key, you are free to use the functional-limited software.

Purchasing a Registration Key for the Software entitles you to use the unlimited functionality of the program. You must enter this Registration Key into the Software's license panel.

RETAIL PURCHASE

The Software may also be obtained through retail purchase. Retail copies of the Software include a Registration Key, whether obtained as an individual product or bundled with another product. You must enter this Registration Key into the Software's license panel.

GRANT OF LICENSE

The purchase price entitles you to a single Registration Key for the Software. Upon purchase of a Registration Key you are granted a non-exclusive, non-transferable license to use the Software according to the terms of the EULA. Prior to purchase of a Registration Key, you may use the software only as provided by the Shareware Evaluation terms of this EULA.

If you have licensed the Software for your personal use, you may install and multiple copies using a single Registration Key on computers you use at your residence. As a personal user, you may use the same Registration Key for additional copies of the Software on portable computers that are used by you and your immediate family.

*You may copy and redistribute unregistered Shareware Evaluation copies of the Software as long as you do not charge a fee for it; **do not modify the software, documentation or license statement in any way** and **do not bundle** it with another system.*

***You may not make the Software available to third parties**, with or without a fee, in connection with a service bureau, application service provider, or similar service, nor permit anyone else to do so. You may not rent, lease or lend registered copies of the Software to third parties. You may not redistribute your registration key.*

*You agree **not to take steps such as altering, decompiling, reverse assembling or reverse compiling** to derive the source code or "look and feel" equivalent(s) of the software*

SUPPORT

Registered users receive free support per Email. No other kind of support is available.

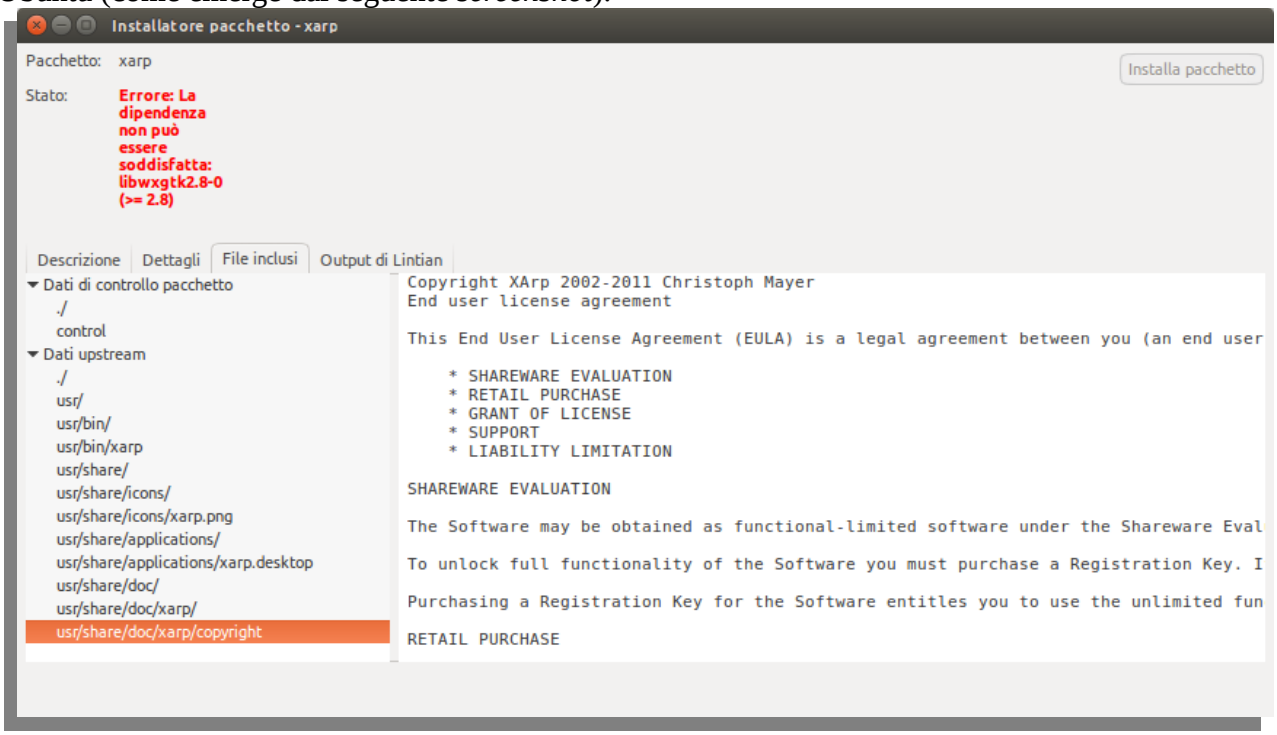
LIABILITY LIMITATION

IN NO EVENT WILL WE BE LIABLE FOR ANY DAMAGES IN EXCESS OF ANY PAYMENTS WE HAVE RECEIVED FROM YOU, EVEN IF WE HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES.

Christoph Mayer
www.chrismc.de

Nota: è interessante notare che anche solo l'operazione di *evidenziare*(=cambiare) parti della licenza d'uso, costituisca essa stessa una violazione dei termini di licenza. Chiediamo venia sin d'ora all'Autore, per questa violazione, fatta solo a scopi didattici.

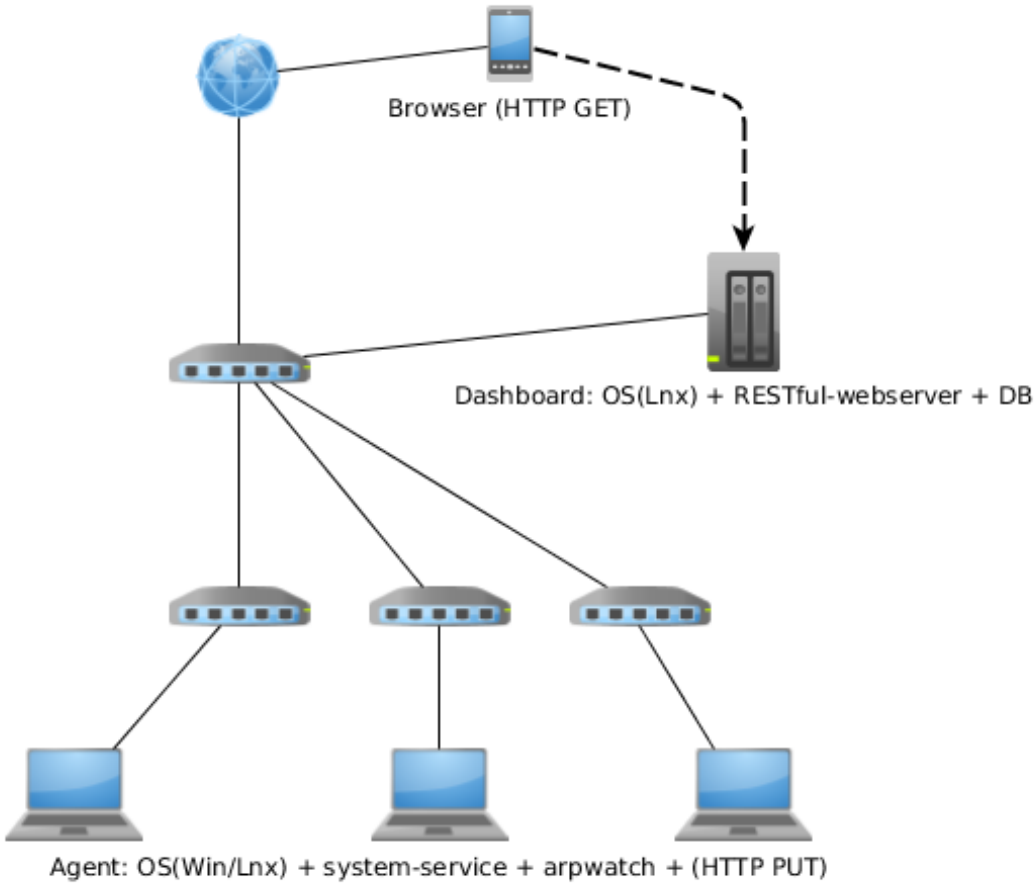
Nota: in aggiunta a quanto già detto rispetto alla licenza d'uso, Xarp presentava anche diverse criticità operative fin dall'installazione, per l'assenza delle librerie/dipendenze dal *repository* di Ubuntu (come emerge dal seguente *screenshot*).



§ 3 - Architettura Generale/Distribuita

La soluzione proposta (ai problemi descritti in specifica) è un sistema distribuito, sviluppato in **Java** (business-logic) e **HTML5/JavaScript** (front-end UI), per la massima portabilità e cross-compatibilità. Il sw è composto da 2-3 moduli, il più possibile *self-contained*: pochi file (*.jar*), eseguibili e compressi, che, oltre ad ottimizzare lo spazio-disco, nascondono la complessità dell'organizzazione del file-system progettuale, mostrandolo all'utente in modo semplice, maneggevole, atomico, e portabile (non necessitano di installazione, e possono essere spostati a piacere dell'utente). Nota: ogni eseguibile è accompagnato dal proprio file di configurazione (testo ASCII, <1KByte) contenente le variabili *custom* dell'utente (e.g.: URL, porte TCP, password).

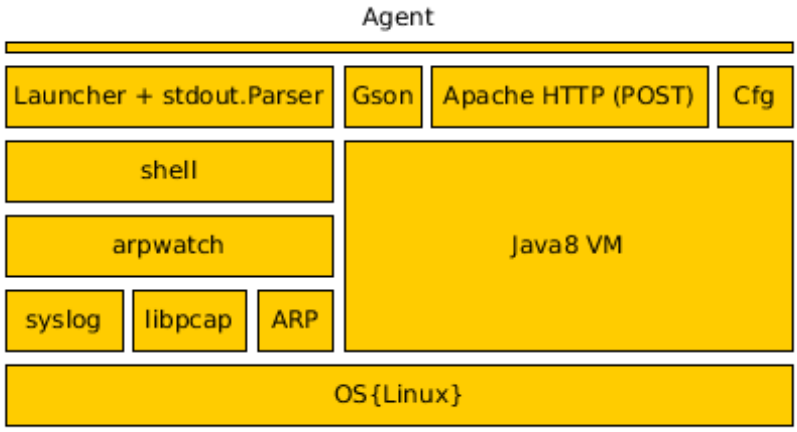
Come abbiamo già accennato, i moduli sw principali della soluzione proposta ricalcano fedelmente i ruoli delle entità emerse in fase di specifica: l'idea progettuale portante è quella di avere 3 blocchi funzionali distribuiti (agent, server, dashboard), il più possibile autonomi (in ossequio ai precetti di ***loose-coupling*** dei moduli sw [39]), ma comunque in grado di collaborare efficientemente in rete.

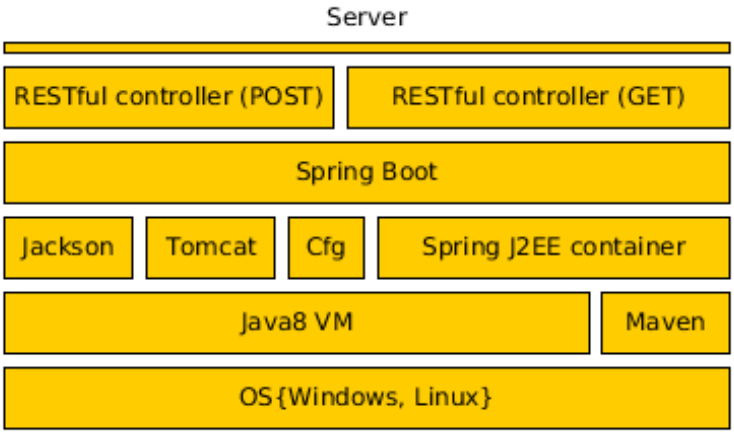


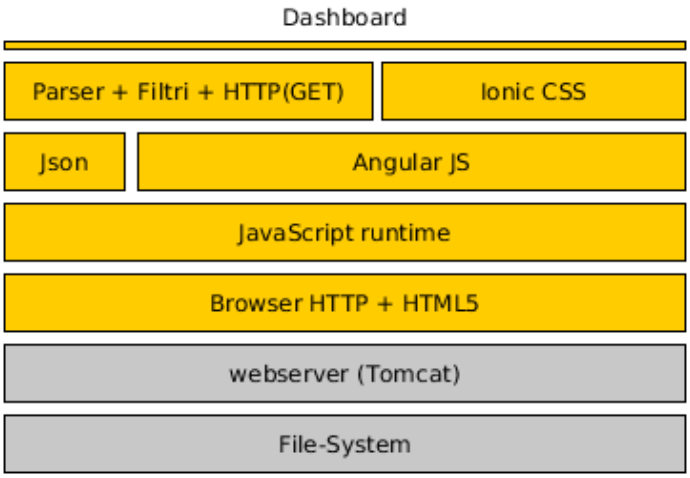
Nota: sebbene da una prospettiva progettuale il server e la dashboard siano entità completamente distinte ed autonome (tanto da essere sviluppate in linguaggi diversi), di fatto esse risiedono fisicamente nello stesso hw (server centrale), e ciò ha contribuito a semplificare l'aspetto esterno del sw, riducendo gli ***artefatti***/file finali da 3 a 2 file (***arp-agent.jar***, ***arp-server.jar***).
Entrambi questi file sono eseguibili, e si avviano da riga di comando (o con *double-click* da GUI):

```
gf@silenzio:~$ java -jar <modulo>.jar
```

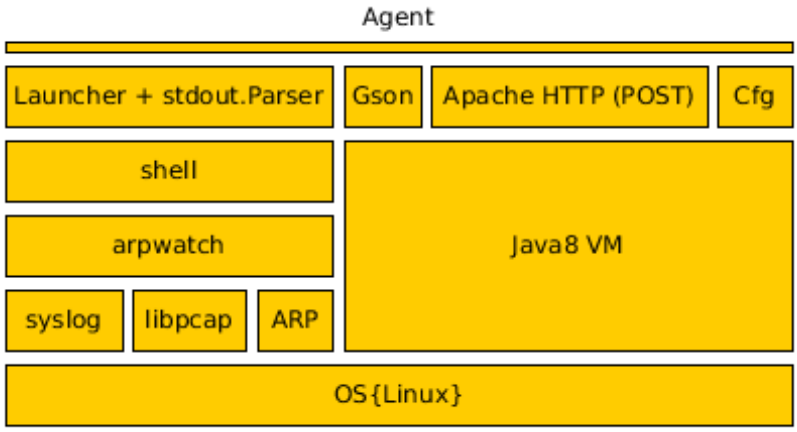
L’organizzazione generale dei 3 moduli può essere espressa in modo sintetico tramite questi grafici stratificati, dove la sovrapposizione dei livelli sta a significare, come di consueto, la **dipendenza** da **componenti**/librerie di terzi (i.e.: il livello superiore utilizza i servizi offerti dal livello inferiore):







3.1 - Architettura dell'Agent



Pre-requisiti:

- network link (route to server)
- Arpwatch (Linux, root)
- NTP (time-sync)
- Java8 VM

Lo strato più alto del grafico dei componenti dell'agent rappresenta il mio contributo, ossia l'ombrello di *business-logic* che tiene insieme tutte le librerie di terze parti (open-source), e le fa collaborare al fine di implementare un agent leggero, che essenzialmente svolga 3 task concorrenti:

- controllo continuo della bontà dell'entry ARP del *default-gateway* nell'**ARP-cache locale**: un sotto-processo gira indefinitamente in attesa che il MAC-address del gateway cambi nell'ARP-cache locale (forte indizio di ARP-poisoning);
- invio periodico di ***keepalive***, per segnalare al server il buono stato di salute dell'agent;
- (opzionale) avvio di **arpwatch** in un sotto-processo (tramite shell), catturando i suoi *stdout+stderr*, e girando tutte le segnalazioni (in formato Json) verso il server centrale.

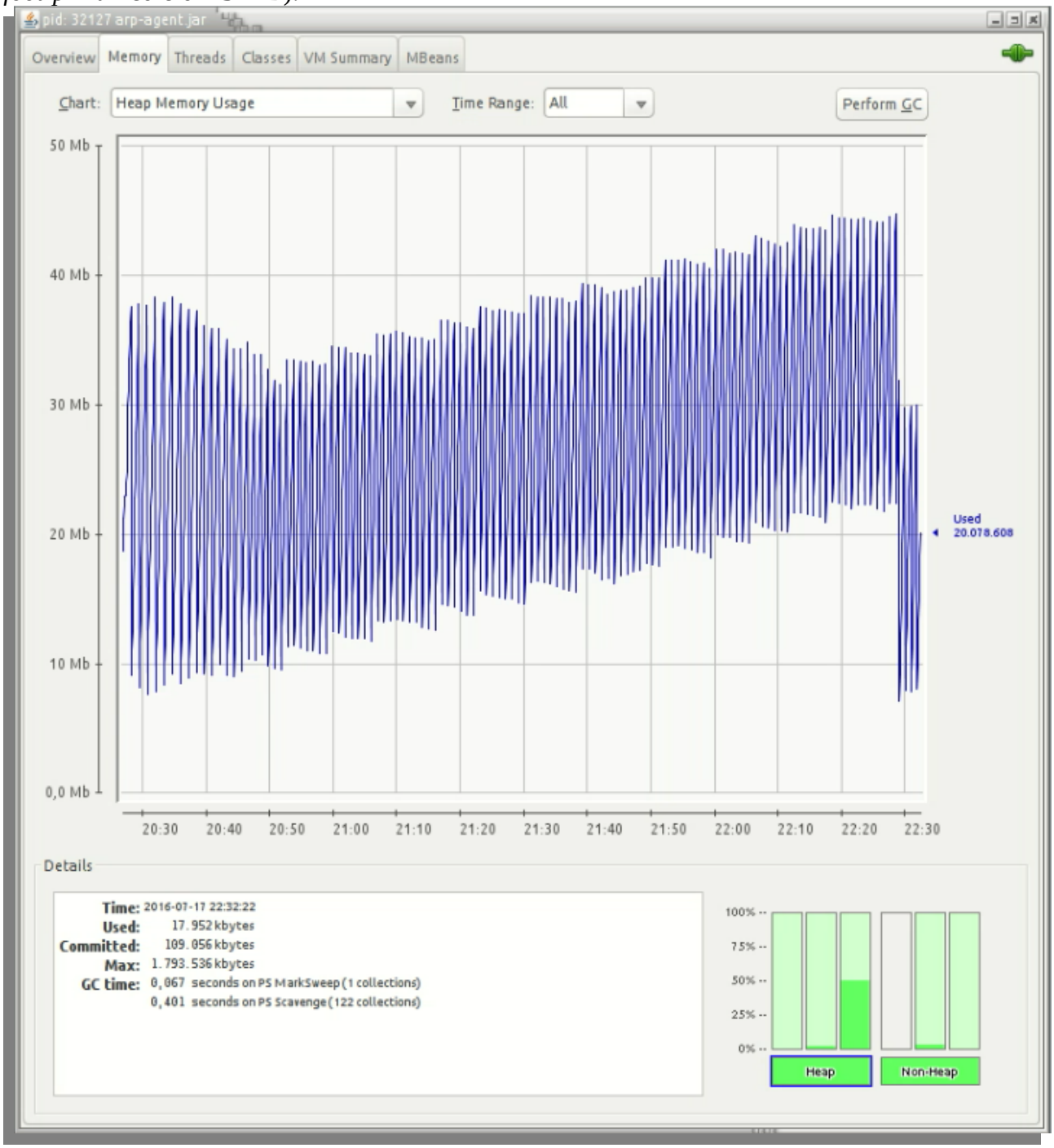
Nota: tutti gli invii sono singole sessioni HTTP (**POST**) ad un singolo REST-ful service remoto.

Sebbene l'agent possa anche essere avviato in *background* (nel solito modo Un*X, ossia terminando la *command-line* con "*&*") per poi scomparire alla vista, tuttavia si consiglia di lasciare aperta la VTY/TTY, in caso si voglia verificare l'attività e lo *stdout*.
Ciò che appare tipicamente in *console* è rappresentato di seguito:

```
gf@silenzio:~$ java -jar arp-agent.jar
Using configuration: config.properties; OS: LINUX; Fri Jul 22 00:27:44 CEST 2016
i/f: eno1; MAC: 08-2E-5F-75-E5-9D; IP: 192.168.1.68;
Stopping arpwatch ... done
Starting local monitor ... done
Starting arpwatch ... done
Starting AW listener ... done
Sending: 2016-07-22 00:27:46 ; Keepalive from: silenzio; AWalive=TRUE; pendingMsgs=0
Sending: 2016-07-22 00:27:57 ; Keepalive from: silenzio; AWalive=TRUE; pendingMsgs=0
Sending: 2016-07-22 00:28:08 ; Keepalive from: silenzio; AWalive=TRUE; pendingMsgs=0
Sending: 2016-07-22 00:28:19 ; Keepalive from: silenzio; AWalive=TRUE; pendingMsgs=0
Sending: 2016-07-22 00:28:30 ; Keepalive from: silenzio; AWalive=TRUE; pendingMsgs=0
Flushing tmp for timeout ...
Sending: 2016-07-22 00:28:36 ; From: arpwatch (Arpwatch silenzio)
To: root
Subject: flip flop (192.168.1.150) eno1
           hostname: <unknown>
         ip address: 192.168.1.150
          interface: eno1
```

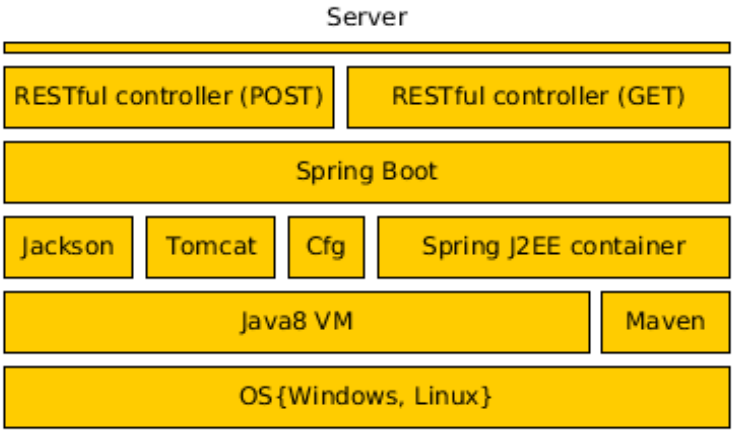
```
    ethernet address: 08:00:27:fe:29:96
     ethernet vendor: CADMUS COMPUTER SYSTEMS
old ethernet address: 00:12:fe:ce:69:3e
 old ethernet vendor: Lenovo Mobile Communication Technology Ltd.
           timestamp: Friday, July 22, 2016 0:28:33 +0200
  previous timestamp: Sunday, July 17, 2016 22:27:34 +0200
               delta: 4 days

Sending: 2016-07-22 00:28:41 ; Keepalive from: silenzio; AWalive=TRUE; pendingMsgs=0
Sending: 2016-07-22 00:28:52 ; Keepalive from: silenzio; AWalive=TRUE; pendingMsgs=0
Sending: 2016-07-22 00:29:03 ; Keepalive from: silenzio; AWalive=TRUE; pendingMsgs=0
...
```

Ecco di seguito l'andamento generale della memoria allocata dalla JVM al processo agent (*memory foot-print* medio di **25 MB**):



Nota: l'attività del GC (*Garbage Colletor*) della JVM in corrispondenza dello "scalino" nel grafo.

3.2 - Architettura del Server



Pre-requisiti:

- network-link (route to agents, and to viewing hosts)
- NTP (time-sync)
- Java8 VM

Lo strato più alto del grafico dei componenti del server rappresenta ancora l'ombrello di *business-logic* che tiene insieme tutte le librerie di terze parti (open-source), e le fa collaborare. Questa volta l'architettura è più strutturata, in ragione dei molteplici ruoli che il server ricopre, ossia:

- REST-ful (ossia *stateless* [43]) web-service per l'invio (HTTP-**POST**) degli eventi ARP da parte degli agenti remoti;
- **cache** interna (volatile, in RAM) dello stato degli *agent* remoti registrati;
- REST-ful web-service per le varie *query* (HTTP-**GET**/Json) da parte delle dashboard, visualizzate nei vari dispositivi remoti.

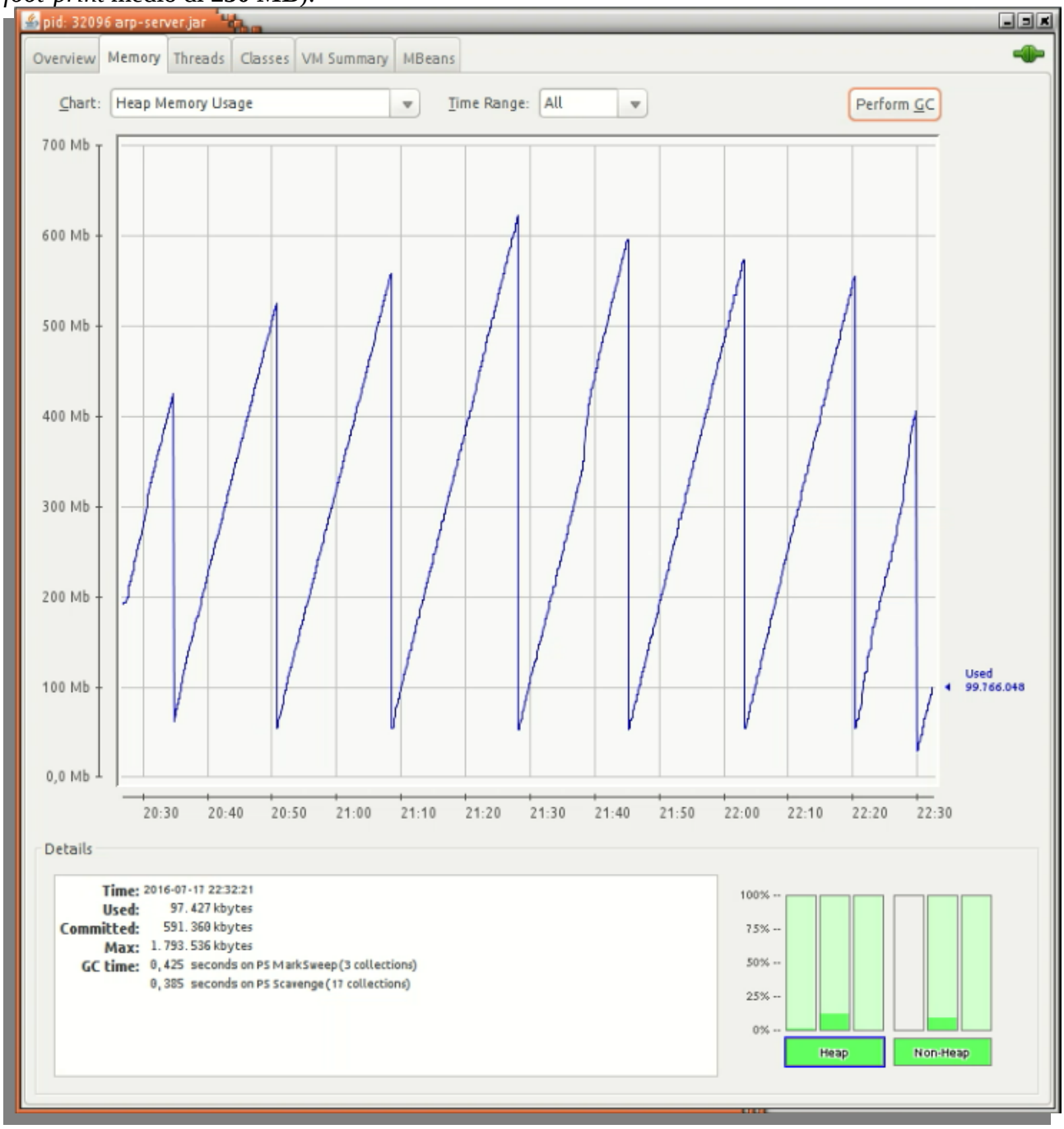
Anche il server può essere avviato in *background* (terminando la *command-line* con "&") per poi scomparire alla vista, ma si consiglia invece di lasciare aperta la VTY/TTY, per verificare l'attività e lo *stdout*. Ciò che appare tipicamente in *console* è rappresentato di seguito:

```
gf@silenzio:~$ java -jar arp-server.jar
-------- Start-up message in server console ---------

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::             (v1.4.0.M3)

2016-07-22 00:27:33.526  INFO 3784 --- [           main]
org.gfcyb.arp.server.ArpServer           : Starting ArpServer v0.0.1-SNAPSHOT on silenzio with PID
3784 (started by gf in /home/gf/Pubblici/20160717.3_minor.fixes/server)
2016-07-22 00:27:33.532  INFO 3784 --- [           main]
org.gfcyb.arp.server.ArpServer           : No active profile set, falling back to default
profiles: default
2016-07-22 00:27:34.187  INFO 3784 --- [           main]
ationConfigEmbeddedWebApplicationContext : Refreshing
org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext@5b37e0d2:
startup date [Fri Jul 22 00:27:34 CEST 2016]; root of context hierarchy
2016-07-22 00:27:36.778  INFO 3784 --- [           main]
s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat initialized with port(s): 6969 (http)
2016-07-22 00:27:36.801  INFO 3784 --- [           main]
o.apache.catalina.core.StandardService   : Starting service Tomcat
```

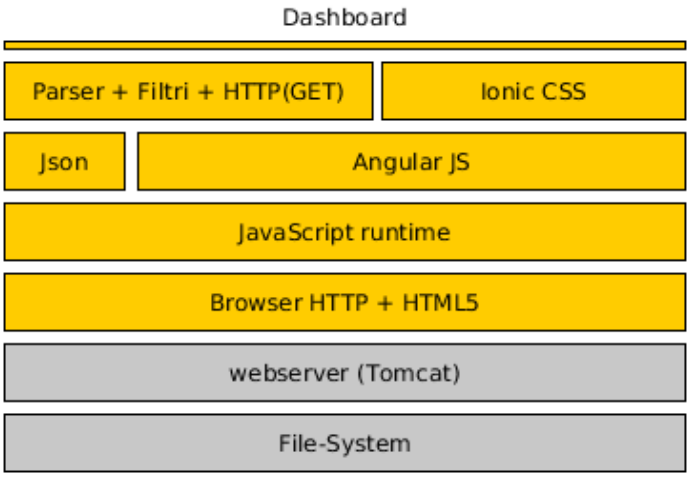
```
2016-07-22 00:27:36.803  INFO 3784 --- [           main]
org.apache.catalina.core.StandardEngine  : Starting Servlet Engine: Apache Tomcat/8.0.33
2016-07-22 00:27:36.945  INFO 3784 --- [ost-startStop-1] o.a.c.c.C.[Tomcat].[localhost].
[/]       : Initializing Spring embedded WebApplicationContext
2016-07-22 00:27:36.945  INFO 3784 --- [ost-startStop-1]
o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed in
2764 ms
2016-07-22 00:27:37.643  INFO 3784 --- [ost-startStop-1]
o.s.b.w.servlet.ServletRegistrationBean  : Mapping servlet: 'dispatcherServlet' to [/]
2016-07-22 00:27:37.651  INFO 3784 --- [ost-startStop-1]
o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'metricFilter' to: [/*]
2016-07-22 00:27:37.651  INFO 3784 --- [ost-startStop-1]
o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'characterEncodingFilter' to: [/*]
2016-07-22 00:27:37.652  INFO 3784 --- [ost-startStop-1]
o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'hiddenHttpMethodFilter' to: [/*]
2016-07-22 00:27:37.652  INFO 3784 --- [ost-startStop-1]
o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'httpPutFormContentFilter' to: [/*]
2016-07-22 00:27:37.652  INFO 3784 --- [ost-startStop-1]
o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'requestContextFilter' to: [/*]
2016-07-22 00:27:37.652  INFO 3784 --- [ost-startStop-1]
o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'webRequestLoggingFilter' to: [/*]
2016-07-22 00:27:37.652  INFO 3784 --- [ost-startStop-1]
o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'applicationContextIdFilter' to: [/*]
2016-07-22 00:27:38.106  INFO 3784 --- [           main]
s.w.s.m.m.a.RequestMappingHandlerAdapter : Looking for @ControllerAdvice:
org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext@5b37e0d2:
startup date [Fri Jul 22 00:27:34 CEST 2016]; root of context hierarchy
2016-07-22 00:27:38.239  INFO 3784 --- [           main]
s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/agents]}" onto public
org.springframework.http.ResponseEntity<java.util.List<org.gfcyb.arp.server.pojo.RemoteAgent>>
org.gfcyb.arp.server.MyRestController.allAsList()
2016-07-22 00:27:38.241  INFO 3784 --- [           main]
s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/agent/{agentId}]}" onto public
org.springframework.http.ResponseEntity<java.util.List<org.gfcyb.arp.server.pojo.Event>>
org.gfcyb.arp.server.MyRestController.agentMessages(java.lang.String)
2016-07-22 00:27:38.241  INFO 3784 --- [           main]
s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/msg/{agentId}/{msgId}]}" onto public
org.springframework.http.ResponseEntity<org.gfcyb.arp.server.pojo.Event>
org.gfcyb.arp.server.MyRestController.msgDetails(java.lang.String,int)
2016-07-22 00:27:38.242  INFO 3784 --- [           main]
s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/postEvent],methods=[POST]}" onto public
org.springframework.http.ResponseEntity<org.gfcyb.arp.server.pojo.Event>
org.gfcyb.arp.server.MyRestController.postEvent(org.gfcyb.arp.server.pojo.Event)
2016-07-22 00:27:38.245  INFO 3784 --- [           main]
s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error],produces=[text/html]}" onto public
org.springframework.web.servlet.ModelAndView
org.springframework.boot.autoconfigure.web.BasicErrorController.errorHtml(javax.servlet.http.HttpS
ervletRequest,javax.servlet.http.HttpServletResponse)
2016-07-22 00:27:38.246  INFO 3784 --- [           main]
s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error]}" onto public
org.springframework.http.ResponseEntity<java.util.Map<java.lang.String, java.lang.Object>>
org.springframework.boot.autoconfigure.web.BasicErrorController.error(javax.servlet.http.HttpServl
etRequest)
2016-07-22 00:27:38.255  INFO 3784 --- [           main]
o.s.w.s.c.a.WebMvcConfigurerAdapter      : Adding welcome page: class path resource
[static/index.html]
2016-07-22 00:27:38.274  INFO 3784 --- [           main]
o.s.w.s.handler.SimpleUrlHandlerMapping  : Root mapping to handler of type [class
org.springframework.web.servlet.mvc.ParameterizableViewController]
2016-07-22 00:27:38.294  INFO 3784 --- [           main]
o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/webjars/**] onto handler of type
[class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2016-07-22 00:27:38.295  INFO 3784 --- [           main]
o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/**] onto handler of type [class
org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2016-07-22 00:27:38.396  INFO 3784 --- [           main]
o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/**/favicon.ico] onto handler of type
[class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2016-07-22 00:27:38.593  INFO 3784 --- [           main]
roperties$SimpleAuthenticationProperties :

Using default password for shell access: 116a3bd6-5c2b-411c-bd98-18a493a26521

...

2016-07-22 00:27:41.410  INFO 3784 --- [           main]
```

```
s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 6969 (http)
2016-07-22 00:27:41.417  INFO 3784 --- [           main]
org.gfcyb.arp.server.ArpServer           : Started ArpServer in 8.395 seconds (JVM running for
9.049)
-------- Application lauched ---------
2016-07-22 00:27:46.725  INFO 3784 --- [nio-6969-exec-1] o.a.c.c.C.[Tomcat].[localhost].
[/]       : Initializing Spring FrameworkServlet 'dispatcherServlet'
2016-07-22 00:27:46.725  INFO 3784 --- [nio-6969-exec-1]
o.s.web.servlet.DispatcherServlet        : FrameworkServlet 'dispatcherServlet': initialization
started
2016-07-22 00:27:46.744  INFO 3784 --- [nio-6969-exec-1]
o.s.web.servlet.DispatcherServlet        : FrameworkServlet 'dispatcherServlet': initialization
completed in 19 ms
Remote event from: silenzio; 2016-07-22 00:27:46 ; Keepalive from: silenzio; AWalive=TRUE;
pendingMsgs=0
Remote event from: silenzio; 2016-07-22 00:27:57 ; Keepalive from: silenzio; AWalive=TRUE;
pendingMsgs=0
Remote event from: silenzio; 2016-07-22 00:28:08 ; Keepalive from: silenzio; AWalive=TRUE;
pendingMsgs=0
Remote event from: silenzio; 2016-07-22 00:28:19 ; Keepalive from: silenzio; AWalive=TRUE;
pendingMsgs=0
Remote event from: silenzio; 2016-07-22 00:28:30 ; Keepalive from: silenzio; AWalive=TRUE;
pendingMsgs=0
Remote event from: silenzio; 2016-07-22 00:28:36 ; From: arpwatch (Arpwatch silenzio)
To: root
Subject: flip flop (192.168.1.150) eno1
            hostname: <unknown>
          ip address: 192.168.1.150
           interface: eno1
    ethernet address: 08:00:27:fe:29:96
     ethernet vendor: CADMUS COMPUTER SYSTEMS
old ethernet address: 00:12:fe:ce:69:3e
 old ethernet vendor: Lenovo Mobile Communication Technology Ltd.
           timestamp: Friday, July 22, 2016 0:28:33 +0200
  previous timestamp: Sunday, July 17, 2016 22:27:34 +0200
               delta: 4 days

Remote event from: silenzio; 2016-07-22 00:28:41 ; Keepalive from: silenzio; AWalive=TRUE;
pendingMsgs=0
Remote event from: silenzio; 2016-07-22 00:28:52 ; Keepalive from: silenzio; AWalive=TRUE;
pendingMsgs=0
Remote event from: silenzio; 2016-07-22 00:29:03 ; Keepalive from: silenzio; AWalive=TRUE;
pendingMsgs=0
...
```

Ecco di seguito l'andamento generale della memoria allocata dalla JVM al processo server (*memory foot-print* medio di **250 MB**):



3.3 - Architettura della Dashboard (UI)



Pre-requisiti:

- network-link (route to server)
- browser HTML5
- VGA-level screen-size (modern mobile-phones OK)

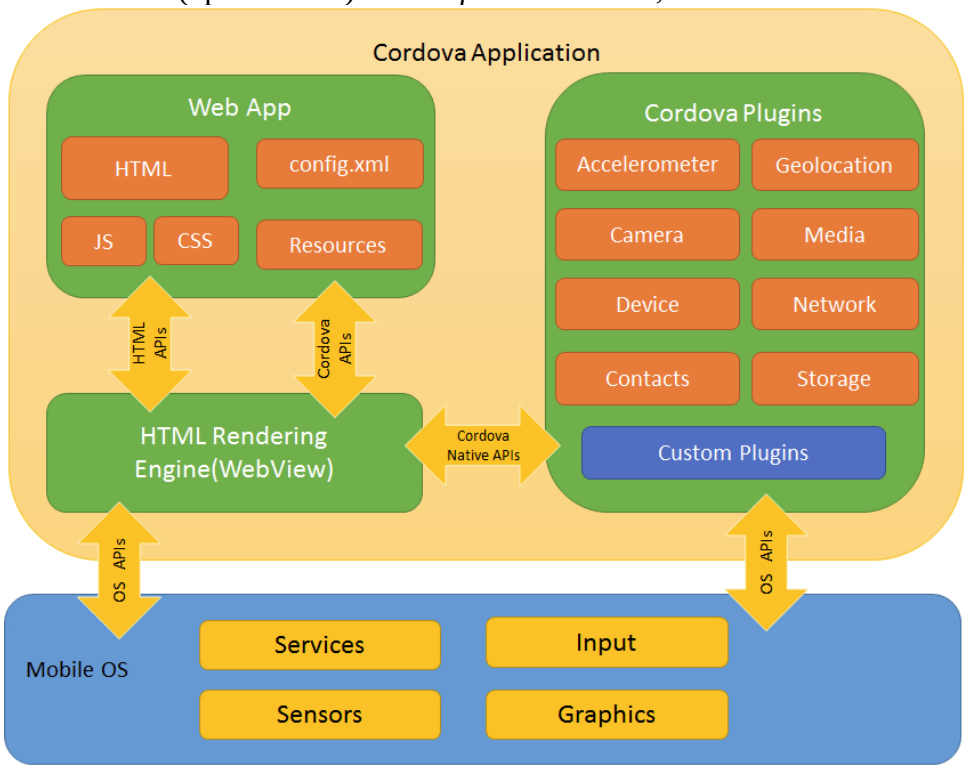
La duplice natura (programmatica e statica) del servlet-container/webserver Tomcat qui è tornata particolarmente utile, e la sua scelta si è rivelata fortunata:

- da una parte esso provvede alla gestione dei vari **REST**-ful web-service, risolvendo i vari (URL+HTTP-*verbs*) sui corrispondenti metodi dichiarati/mappati come *handler*;
- dall'altra, più banalmente, esso permette di pubblicare contenuti HTML5 **statici** (nel senso che non sono creati al-volo, programmaticamente, ma attingono al file-system).

Sfruttando questa dualità, si è potuto pubblicare la nostra ***web-app*** [44], basata sulle librerie Ionic/AngularJS, e in grado di far dialogare il browser dell'utente (e.g.: anche da smartphone) con l'application-server REST-ful, di fargli scaricare (=HTTP-GET) gli appropriati oggetti (e/o array) **Json**, ed infine di generarne il rendering nella GUI web-based.

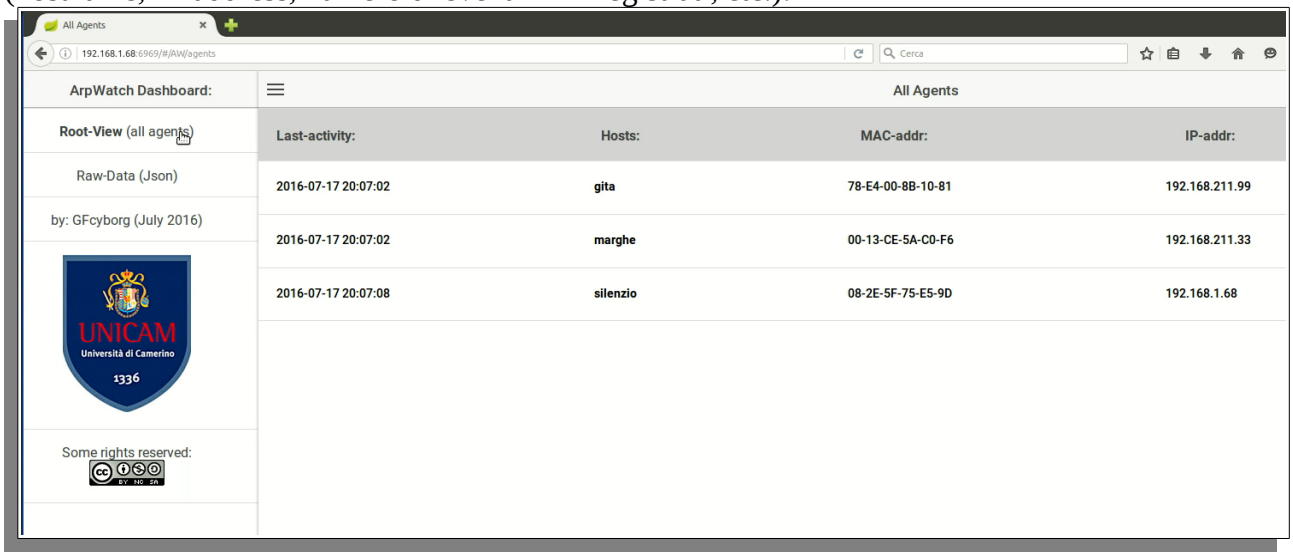
Il vantaggio di questo approccio naturalmente è di spostare il carico computazionale lato-client (grazie a Javascript), sollevando il server dall'onere di tracciare lo stato delle varie dashboard (virtualmente illimitate), ed anzi traendo vantaggio dal ***caching*** fornito da Ionic (e possibile solo grazie alla natura *stateless* dei servizi REST-ful).

Nota: come ulteriore vantaggio, si consideri anche la possibilità -ove richiesto- di generare applicazioni *mobile* (per Android e Apple iOS), in modo molto semplice, grazie al *packaging* automatico offerto dai sw (open-source) come *Apache Cordova*, basati sulle API ***WebView*** [45]:



L'aspetto della GUI/dashboard, visualizzata all'interno di un qualsiasi browser *HTML5-enabled*, è il seguente (suddiviso in 3 viste, gararchicamente organizzate):

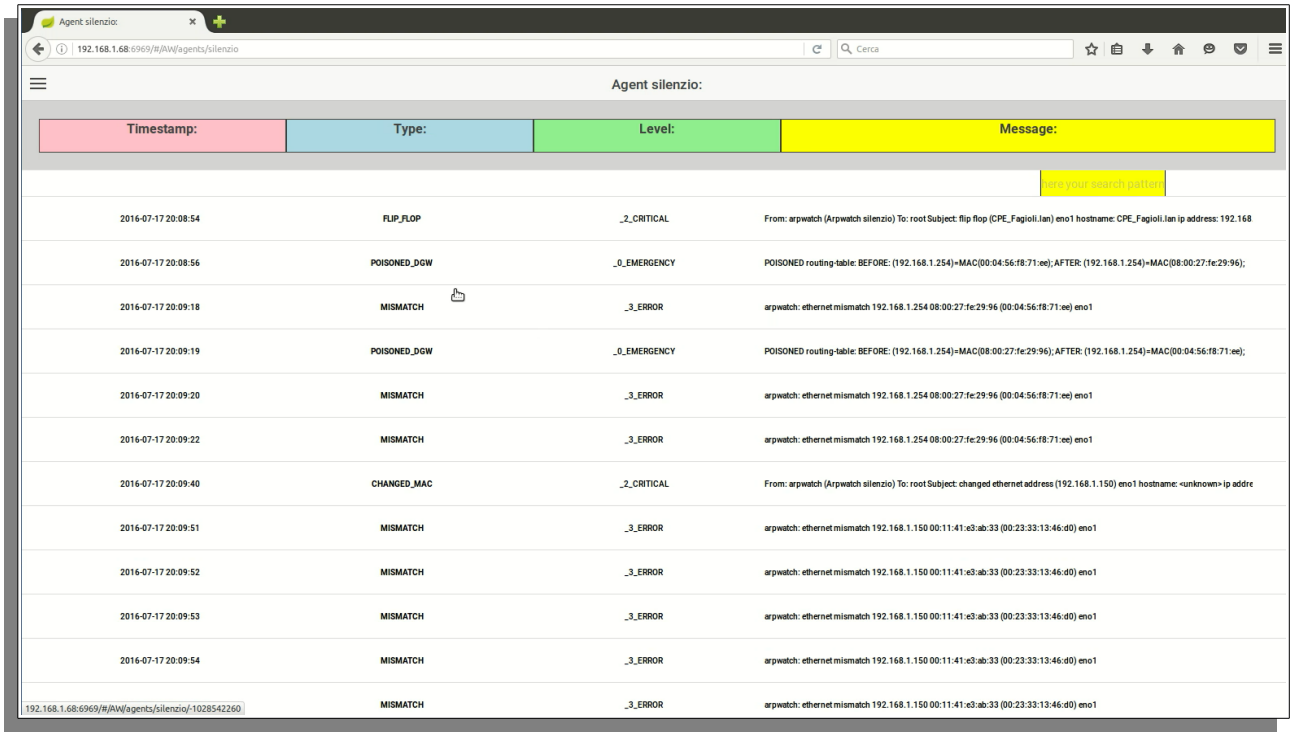
Root-View: è il punto di ingresso della web-app, e mostra la lista di tutti gli agent registrati presso il server centrale, con alcune informazioni sintetiche che aiutano ad individuarlo velocemente (hostname, IP-address, numero di eventi ARP registrati, etc.):



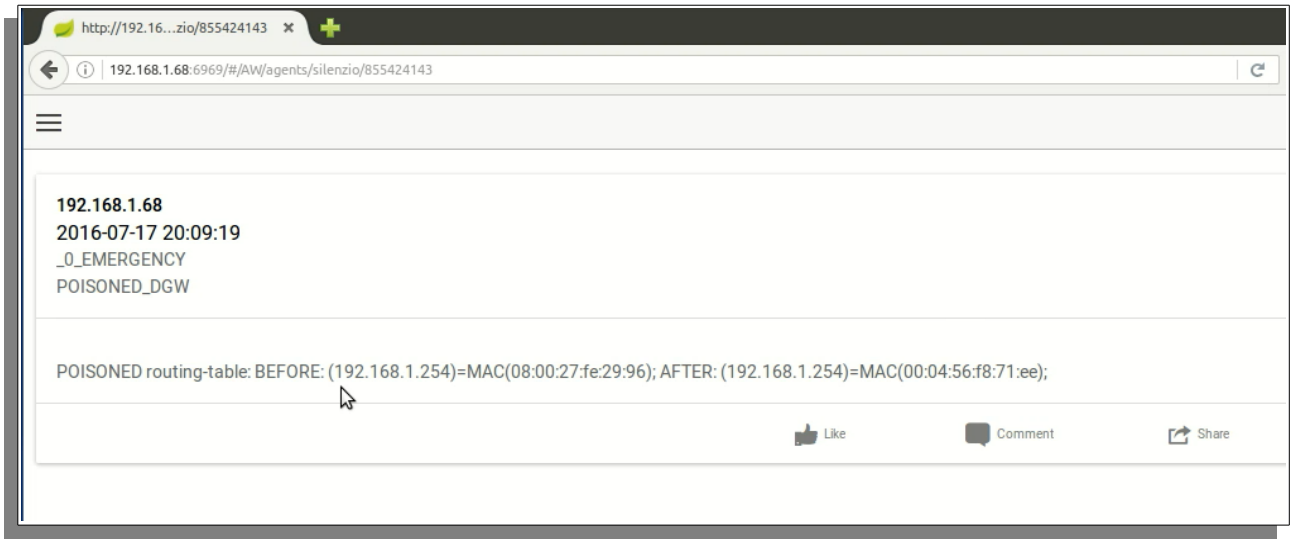
Cliccando su uno degli agent, si passa alla seconda vista, che ne offre il dettaglio in termini di una **lista di eventi ARP** (generalmente molto lunga, a regime): per questo motivo sono presenti alcune *facilities* che permettono una fruizione semplificata, per abbattere il rumore:

- **Ordinamento** diretto/inverso: attivato cliccando (più volte) sull'intestazione colorata della colonna da ordinare;
- **Ricerca** per parola-chiave: il motore Javascript filtra immediatamente il campo *Message*, in modo molto *responsivo*, alla ricerca del testo digitato nell'input-box;

Nota: le 2 funzionalità possono anche lavorare insieme (e.g.: è possibile ordinare il risultato di una ricerca, e viceversa).



Infine, cliccando su uno qualsiasi dei messaggi, si accede alla terza vista, che esprime il massimo del **dettaglio** offerto dal sw, ossia il **testo completo** del messaggio, insieme ai suoi ***meta-dati*** (*timestamp*, livello di rilevanza, etc.):



§ 4 - Protocollo di Comunicazione

Al fine di abbattere l'*overhead* di traffico sul medium di rete, tutte le comunicazioni sono ridotte a semplici oggetti **Json** (testo UTF-8 [46]): ciò che cambia tra i casi Agent/Server e Dashboard/Server è solo il senso di percorrenza di questi dati (naturalmente essi partono dagli agent, arrivano al server, che poi li gira a tutte le dashboard attive).

Il formato della *master-list* (Json) illustra (e contiene) anche quello degli oggetti gerarchicamente inferiori, e perciò vale la pena riportarlo qui, evidenziando le **strutture-dati** fondamentali:

```
[
   {
      "agentId":"silenzio",
      "mac":"08-2E-5F-75-E5-9D",
      "ip":"192.168.1.68",
      "lastActivity":"2016-07-22 02:36:37",
      "events":[
         {
            "msgId":-1114436362,
            "agentId":"silenzio",
            "mac":"08-2E-5F-75-E5-9D",
            "ip":"192.168.1.68",
            "timestamp":"2016-07-22 02:35:06",
            "origMsg":"POISONED routing-table: \nBEFORE:
(192.168.1.254)=MAC(00:04:56:f8:71:ee); \nAFTER: (192.168.1.254)=MAC(08:00:27:fe:29:96); \n",
            "lvl":"_0_EMERGENCY",
            "type":"POISONED_DGW"
         },
         {
            "msgId":-798858561,
            "agentId":"silenzio",
            "mac":"08-2E-5F-75-E5-9D",
            "ip":"192.168.1.68",
            "timestamp":"2016-07-22 02:35:14",
            "origMsg":"arpwatch: ethernet mismatch 192.168.1.254 08:00:27:fe:29:96
(00:04:56:f8:71:ee) eno1\n",
            "lvl":"_3_ERROR",
            "type":"MISMATCH"
         },
         ...,
         {
            "msgId":-148628990,
            "agentId":"silenzio",
            "mac":"08-2E-5F-75-E5-9D",
            "ip":"192.168.1.68",
            "timestamp":"2016-07-22 02:36:18",
            "origMsg":"From: arpwatch (Arpwatch silenzio)\nTo: root\nSubject: changed ethernet
address (192.168.1.150) eno1\n            hostname: <unknown>\n          ip address:
192.168.1.150\n           interface: eno1\n    ethernet address: 00:15:1f:72:13:8a\n     ethernet
vendor: Multivision Intelligent Surveillance (Hong Kong) Ltd\nold ethernet address:
08:00:27:fe:29:96\n old ethernet vendor: CADMUS COMPUTER SYSTEMS\n           timestamp: Friday,
July 22, 2016 2:36:16 +0200\n  previous timestamp: Friday, July 22, 2016 2:34:31 +0200\n
delta: 1 minute\n",
            "lvl":"_2_CRITICAL",
            "type":"CHANGED_MAC"
         },
         ...,
         {
            "msgId":-577935275,
            "agentId":"silenzio",
            "mac":"08-2E-5F-75-E5-9D",
            "ip":"192.168.1.68",
            "timestamp":"2016-07-22 02:36:30",
            "origMsg":"arpwatch: ethernet mismatch 192.168.1.150 00:1f:06:04:6c:f6
(00:15:1f:72:13:8a) eno1\n",
            "lvl":"_3_ERROR",
            "type":"MISMATCH"
         }
```

```
      ]
   },
   {
      "agentId":"vbx",
      "mac":"08-00-27-FE-29-96",
      "ip":"192.168.1.150",
      "lastActivity":"2016-07-22 02:35:24",
      "events":[
         {
            "msgId":129547518,
            "agentId":"vbx",
            "mac":"08-00-27-FE-29-96",
            "ip":"192.168.1.150",
            "timestamp":"2016-07-22 02:35:05",
            "origMsg":"arpwatch: reused old ethernet address 192.168.1.254 08:00:27:fe:29:96
(00:04:56:f8:71:ee) enp0s3\narpwatch: reused old ethernet address 192.168.1.68 08:00:27:fe:29:96
(08:2e:5f:75:e5:9d) enp0s3\n",
            "lvl":"_2_CRITICAL",
            "type":"REUSED_MAC"
         },
         {
            "msgId":-1662015708,
            "agentId":"vbx",
            "mac":"08-00-27-FE-29-96",
            "ip":"192.168.1.150",
            "timestamp":"2016-07-22 02:35:09",
            "origMsg":"From: arpwatch (Arpwatch vbx)\nTo: root\nSubject: flip flop
(CPE_Fagioli.lan) enp0s3\n            hostname: CPE_Fagioli.lan\n          ip address:
192.168.1.254\n           interface: enp0s3\n    ethernet address: 00:04:56:f8:71:ee\n
ethernet vendor: Motorola PTP Inc\nold ethernet address: 08:00:27:fe:29:96\n old ethernet vendor:
CADMUS COMPUTER SYSTEMS\n           timestamp: Friday, July 22, 2016 2:35:07 +0200\n  previous
timestamp: Friday, July 22, 2016 2:35:07 +0200\n              delta: 0 seconds\n",
            "lvl":"_2_CRITICAL",
            "type":"FLIP_FLOP"
         },
         ...,
         {
            "msgId":-872302403,
            "agentId":"vbx",
            "mac":"08-00-27-FE-29-96",
            "ip":"192.168.1.150",
            "timestamp":"2016-07-22 02:35:16",
            "origMsg":"arpwatch: ethernet mismatch 192.168.1.68 08:00:27:fe:29:96
(08:2e:5f:75:e5:9d) enp0s3\n",
            "lvl":"_3_ERROR",
            "type":"MISMATCH"
         }
      ]
   }
]
```


È interessante notare come esista un perfetto parallelismo tra l'organizzazione dei dati e quella delle viste; entrambe sono su **3 livelli**:

- *root-level*: corrisponde ad un ***array*** di agenti, ognuno dei quali è un oggetto Json {} contenente alcuni campi, tra cui i più importanti sono: la chiave univoca (***agentId***), e il sub-array (*events*);
  - *mid-level*: corrisponde al ***sub-array*** (***events***), in cui sono elencati i messaggi;
    - *master-detail*: corrisponde al più innestato degli oggetti Json, ossia (***origMsg***);

§ 5 - Manuale dell'Utente

Prima di entrare nei dettagli tecnici, è bene dire che l'utente-tipo cui si indirizza questo sw non è l'utente casalingo, ma un amministratore di sistemi, possibilmente con qualche competenza di reti: questa premessa non intende scoraggiare nessuno, dato che l'utilizzo del sw è banale, ma in fase di *setup* possono emergere problemi che necessitano di capacità di analisi, *troubleshooting*, e responsabilità. Per giustificare questa affermazione, basti pensare che i problemi (che potrebbero impedire il corretto funzionamento/utilizzo del sw) possono essere, solo per fare qualche esempio:

- errato IP-address del server;
- assenza di rotte IP tra agent e server, o tra dashboard e server (e.g.: NAT);
- errata denominazione dell'interfaccia di rete in uso (e.g.: *eno0, en0, eth0,* ...) [47];
- non-possesso dei privilegi di root (*agent only*);
- assenza permessi di esecuzione dei jar-file;
- assenza dei pre-requisiti (e.g.: arpwatch, Java<8);
- ...

Come già accennato, l'intero sistema si basa su 2 jar-file eseguibili e compressi (*arp-agent.jar* e *arp-server.jar,* che però possono essere rinominati con qualsiasi nome, a patto di preservarne l'estensione).
Al loro fianco, nella stessa directory del rispettivo jar-file, vanno copiati i file di configurazione:

- ***config.properties***: (obbligatorio) configurazione dell'agent;
- *application.properties*: (opzionale) configurazione del server (default TCP-port= **6969**).

Al momento della **prima esecuzione**, questa è la configurazione minimale (*config.properties*) che permette l'avvio dell'agent:

Linee (*ending with* <LF+CR>):	Note:
localpwd=my_secret_root_pwd	password in chiaro (quella con cui l'utente corrente esegue il comando ***sudo*** per ricevere i privilegi di *root*; l'utente deve essere già nel gruppo *sudoers*).
server=http://myserveraddress:6969	IP-address o FQDN del server, risolvibile dal DNS configurato sull'agent.
interface=myiface0	interfaccia di rete su cui vogliamo che arpwatch ascolti il traffico ARP.
agentid=	questo campo ***non*** deve essere compilato: in tal modo si forza l'***hashing*** della password.

Al primo avvio (se i dati della configurazione di base sono corretti), l'agent esegue alcuni controlli, e **corregge** la configurazione (che deve essere *scrivibile* dall'utente corrente), producendo questa nuova configurazione:

Linee (*ending with* <LF+CR>):	Note:
`# fixed config` `#Fri Jul 22 03:54:49 CEST 2016`	comments with: timestamp of first run
`localpwd=`**`H%x;$2.gçk\=£`**	hashed password
`agentid=`**`myhostname`**	self-detected from host runtime
`os=``linux`	self-detected from host runtime
`keepalive``=30`	default keepalive delay=30 (sec)
`log=``arp-agent.log`	default log-file= *arp-agent.log*
`server=http://myserveraddress:6969`	unchanged
`interface=myiface0`	unchanged

A partire da questa nuova configurazione corretta, possiamo apportare modifiche a tutti i campi, tranne quello della password e dell'*agentid* (salvo che non diventi necessario, a causa del cambio di password o di hostname: in tal caso bisogna ripetere la procedura dall'inizio).

Un altro esempio di configurazione (più completa) è:

Linee (*ending with* <LF+CR>):	Note:
`localpwd=Hx..k\=`	
`keepalive=11`	
`lauchAWbyscript=false`	should be always false
`agentid=silenzio`	
`os=linux`	
`server=http://192.168.1.68:6969`	
`log=arp-agent.log`	
`interface=eno1`	
`extendMonitorToLAN=true`	to use arpwatch too
`showBogons=true`	to show bogon ARP events

Vediamo infine come si presenta il file di configurazione del **server**, *application.properties* (opzionale, e comunque molto semplice):

Linee (*ending with* <LF+CR>):	Note:
server.port=12345	l'unica opzione configurabile è la porta TCP su cui il server ascolta le chiamate HTTP (GET e POST); il default è 6969.

Nota: le porte assegnabili devono essere non-privilegiate (>1024), se vogliamo poter lanciare il server senza i privilegi di root. [48] [49]

5.1 - Pre-requisiti

Vale la pena, a questo punto, spendere qualche parola sul perchè dei pre-requisiti:

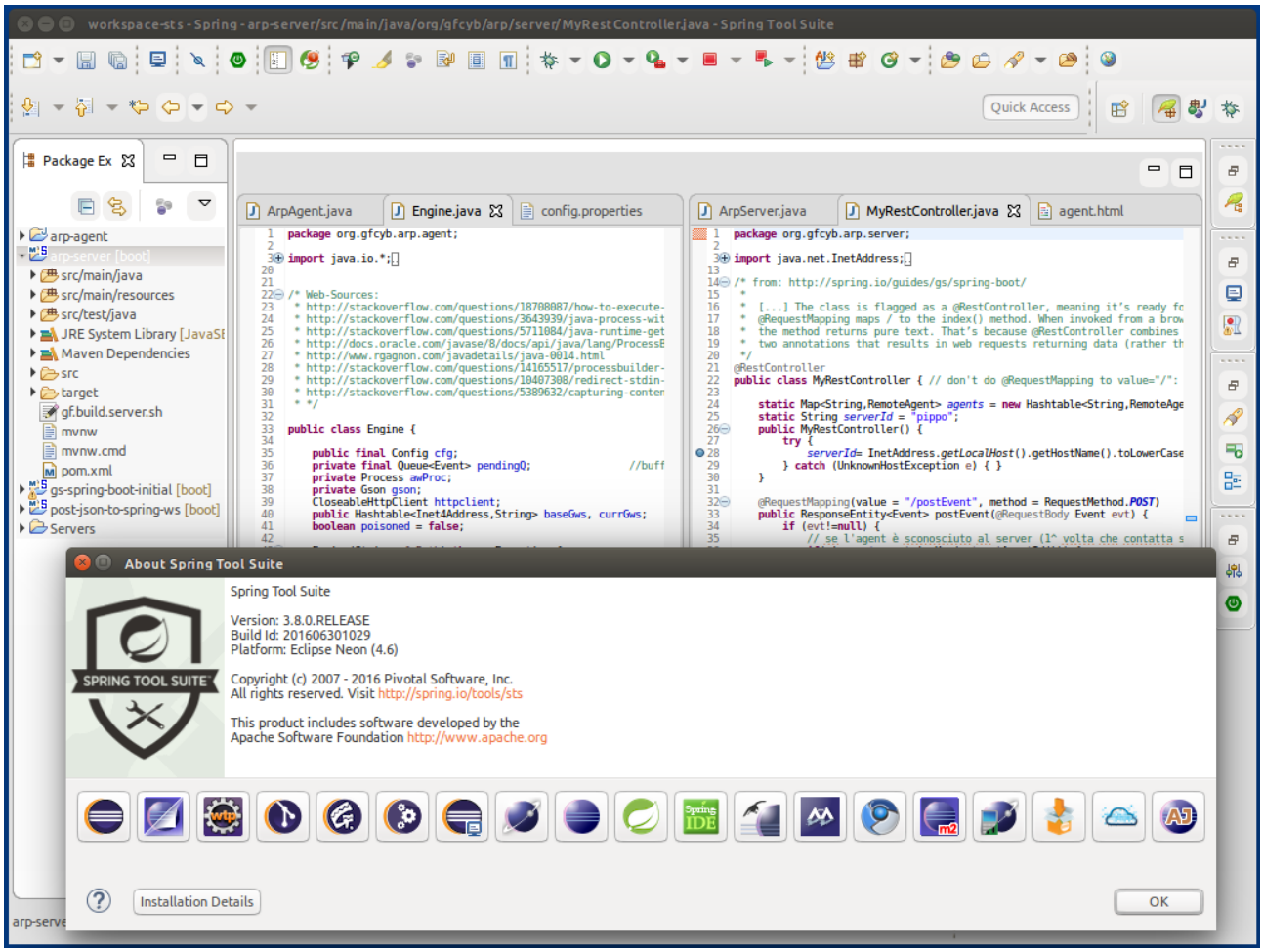
- Agent:
  - network link (route to server): l'agent deve "vedere" il server, cui deve trasferire gli eventi ARP sotto forma di Json, tramite una HTTP-POST. Non ha bisogno invece di "vedere" l'host che visualizza la dashboard, con cui non scambia direttamente alcuna comunicazione;
  - Arpwatch (Linux, **root**): nonostante arpwatch sia in grado di girare su Linux normalmente come demone senza permessi di root, nel nostro caso dobbiamo lanciarlo con tali privilegi, poichè la modalità in cui lo utilizziamo non è quella standard (demone), bensì in modalità di debug (flag *-d*), in ragione del fatto che vogliamo catturare gli stream di stdout e stderr (invece che fare *polling* periodico sul *syslog* o sulla *mailbox* di sistema), per avere un feedback *immediato* delle notifiche ARP (l'attaccante potrebbe contare proprio sul nostro ritardo di rilevamento/notifica, per inserirsi in modalità MitM, ed abbattere il traffico uscente diretto al nostro server);
  - NTP (time-sync): affinchè i timestamp degli eventi ARP abbiano un significato, è necessario che l'orologio di sistema sia ben configurato e sincronizzato: il Network Time Protocol fornisce proprio tale funzionalità (spesso integrata direttamente nell'OS, in modalità NTP-client);
  - Java8 VM: la scelta della versione di Java può sembrare eccessiva, data la semplicità delle operazioni implementate (che in effetti sarebbero potute essere agevolmente implementate anche in Java 7); tuttavia la scelta è ricaduta sulla 8, in ragione del recente ingresso della versione 7 nello stato di *EoL* (End-of-Life: fine degli aggiornamenti di sicurezza da parte di Oracle) [50].

- Server:
  - network-link (route to agents, and to viewing hosts): poichè il server è il centro-stella del nostro sistema distribuito, esso deve "vedere" contemporaneamente tutte le altre componenti, agent e utenti mobili (e infatti tipicamente sarà esposto/pubblicato sulla rete pubblica, o in DMZ, per consentire l'accesso alla dashboard, da smartphone);
  - NTP (time-sync): come sopra;
  - Java8 VM: come sopra;

- Dashboard:
  - network-link (route to server): l'utente mobile che voglia accedere alla dashboard deve "vedere" solo il server; non ha bisogno di vedere gli agent, con cui non scambia alcuna comunicazione;
  - browser **HTML5**: la UI della dashboard non è un semplice sito web statico, ma una vera e propria web-app, basata sulle API di AngularJS e Ionic, e in grado di svolgere compiti complessi come l'attivazione di sessioni HTTP-GET diverse da quelle utilizzate per il download dei file (.html, .js, .css, ...), e.g.: orientate allo scambio di dati Json;
  - VGA-level *screen-size*: la grande quantità di dati visualizzati dalla dashboard, specialmente nella vista della *message-list*, richiede uno schermo sufficientemente largo da evitare il sovraffollamento/sovrapposizione dei dati visualizzati; un normale smartphone (ragionevolmente moderno) è sufficiente allo scopo.

§ 6 - Strumenti e Metodi di Sviluppo

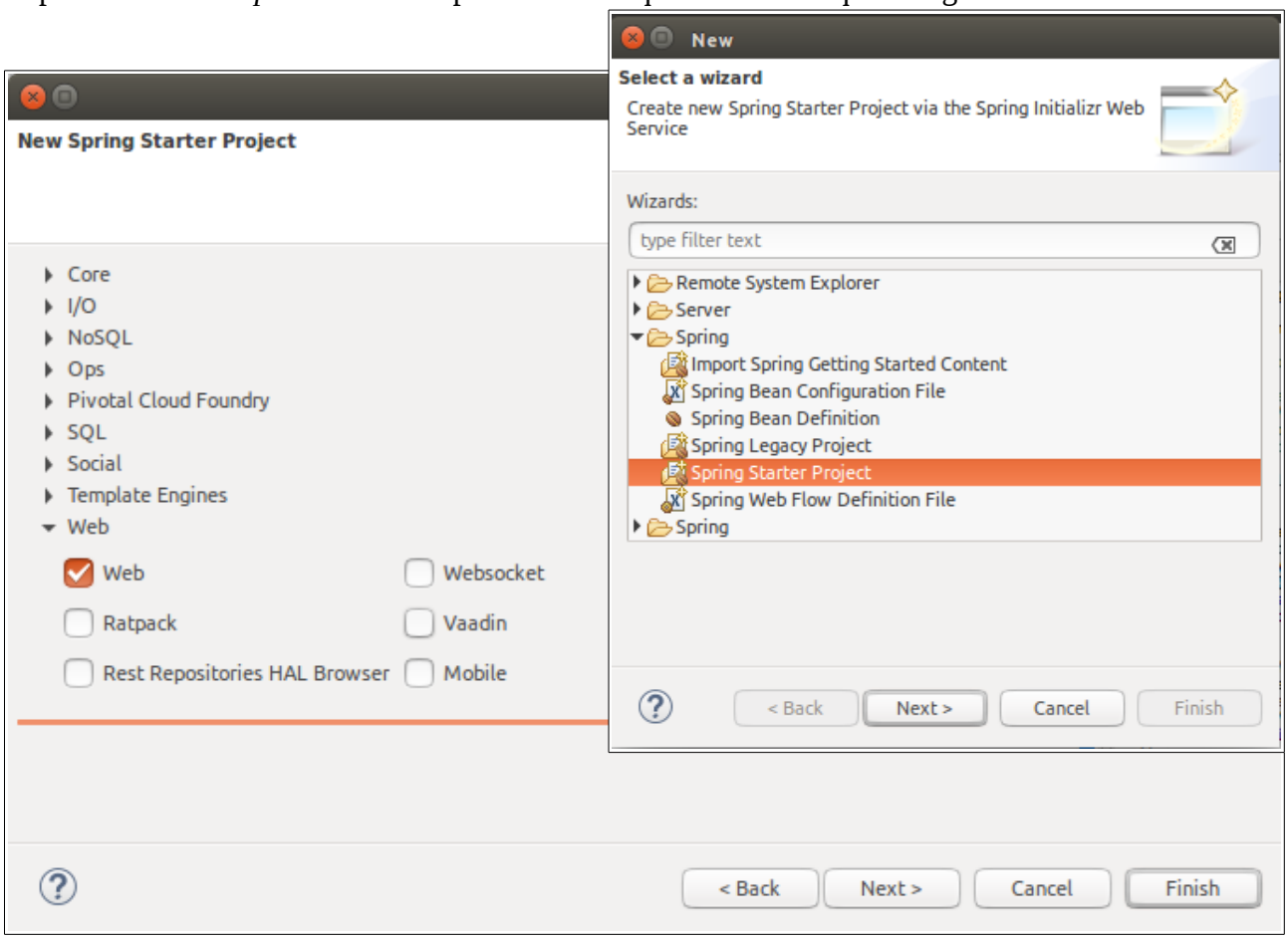
Tutte i moduli del sw sono stati creati su OS Linux Ubuntu 16.04, e su hw HP Envy Dv7 e Apple iMac 7.1.
Lo strumento principale con cui ho sviluppato questo sw è l'IDE ***Eclipse***, nella versione customizzata e ottimizzata per il framework *Spring*, chiamata ***Spring Tool Suite*** (STS), nella versione **3.8.0**. [51]



I motivi principali per cui ho scelto questo strumento sono:

- la mia pregressa conoscenza di *Eclipse*;
- l'alta integrazione tra Java ed *Eclipse*;
- la disponibilità (in STS) di moltissimi ***template***/*stub* (scaricabili in pochi click), che rappresentano -per uno sviluppatore- l'equivalente dei *semilavorati* del manufatturiero: di comprovata solidità, e pronti ad essere rifiniti in poco tempo;
- la disponibilità (in STS) di documentazione di buona qualità (***tutorial*** sui REST service [52] [53]);
- la possibilità di utilizzare STS nel mio OS preferito (GNU **Linux** Debian/Ubuntu);
- gli apprezzamenti di miei stimati colleghi, che avevano già lavorato con STS.

In particolare il *template* utilizzato per il server è quello indicato qui di seguito:



Le librerie utilizzate sono:

- Apache: commons-httpclient-3.1
- Google: gson-2.3.1
- arpwatch-2.1a15
- Spring-Boot 1.4.0 (*embedding* Tomcat 8.0, Jackson)
- Ionic 1.3.1 (*embedding* AngularJS 1.3)

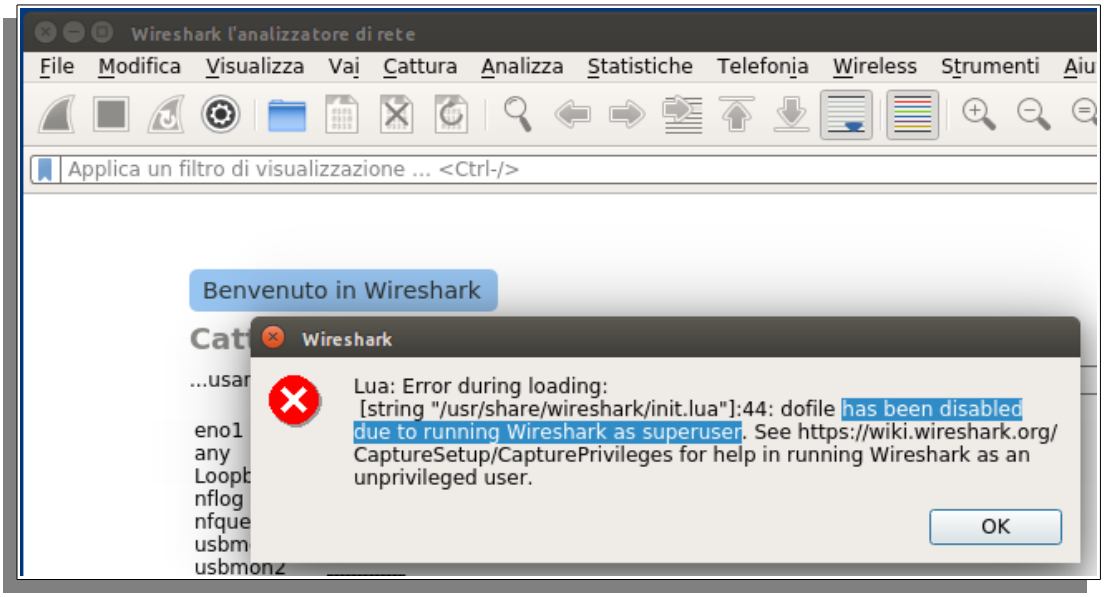
La lavorazione ha impiegato, complessivamente, circa 1 mese-uomo.

I vincoli temporali strettissimi non hanno consentito di stendere le *Test-Unit* prescritte dalle *best-practice* di Ingegneria del sw.

§ 7 - Criticità e Limiti Operativi

Come già accennato, il progetto ha dovuto trovare un compromesso riguardo alla portabilità del modulo agent su OS Windows, al fine di poter utilizzare arpwatch. Tuttavia il sw è stato sviluppato con il pensiero rivolto alla possibilità futura di espanderne l'utilizzo agli OS Microsoft (eventualmente rinunciando ad arpwatch, ed utilizzando solo le funzionalità di monitoring locale della ARP-cache/routing-table, che sono comunque disponibili su qualsiasi OS): i controlli sul tipo di OS sono già stati inseriti nel codice, e devono solo essere completati con nuovi casi (cfr: l'opzione "*os*" nel file di configurazione dell'agent).

Un altro limite operativo è ereditato da arpwatch, per il fatto che esso richiede privilegi di root per essere lanciato in modalità di debug: probabilmente è possibile aggirare questo problema utilizzando la modalità di esecuzione SUID, ma ho scelto di non intraprendere questa strada poichè rischiosa (soprattutto in relazione agli stretti vincoli temporali): dalla mia esperienza personale in ambito sistemistico ho tratto la lezione che spesso i sw orientati alla sicurezza (come sicuramante possiamo classificare arpwatch), tendono a comportarsi in modo imprevedibile se scavalchiamo/aggiriamo i requisiti sui privilegi, e.g.: alcuni sw eseguono il controllo a *runtime*, e si rifiutano di "girare" se trovano che i requisiti sono stati aggirati, altri invece falliscono silenziosamente, rendendo il debug molto difficile, e.g.:



Infine forse rileva qui ricordare che, nei test di laboratorio svolti durante lo sviluppo, abbiamo rilevato (con sorpresa) che arpwatch, posto fuori dalla direttrice d'attacco di ARP-poisoning (e.g.: su un host non interessato dall'attacco), non rilevava alcun traffico ARP sospetto [28]. Per questo motivo è stato necessario integrare le capacità di rilevamento di arpwatch con un semplice (ma efficace) sistema di ***monitoring locale*** della ARP-cache (*arp -an*) + routing-table (*route -n*), tale che rilevi modifiche del MAC-address del gateway della default-route (0.0.0.0).
Naturalmente questo sistema è artigianale, e non scala bene al crescere degli host (poichè ha complessità *O(n)*, mentre arpwatch era *O(1)*), tuttavia bisogna considerare che in realtà non ci interessa monitorare *tutti* gli host, ma solo quelli rilevanti in termini di risorse e/o privilegi, e quindi, in tale scenario, si mitiga (pur rimanendo) la necessità di replicare l'installazione del sw su più host.

§ 8 - Conclusioni e Sviluppi Futuri

La soluzione sw sviluppata è semplice e robusta: si basa sul *deploy* di un server centrale (al cui *setup* basta un solo *jar-file*, eseguibile su sistemi Linux e Windows), e di *n* agent periferici (anch'essi basati su jar-file e *configuration-file*). Entrambi i moduli server e agent sono ragionevolmente leggeri (27.3 e 3.5 MB, compressi) e in grado di girare anche su hw datati (purchè con JVM recenti).

Dalle prove di laboratorio svolte, il sistema rileva correttamente tutti gli attacchi di ARP-poisoning, raggiungendo così il suo scopo principale. Inoltre la dashboard di controllo permette a più utenti contemporanei di monitorare lo stato di salute di intere reti, con pochi click, e da qualsiasi dispositivo dotato di un browser (e.g.: *smartphone*): in fondo la cosiddetta ***sicurezza*** informatica passa anche per l'***usabilità*** dei suoi strumenti (e.g.: facilità del controllo).

Il sistema è naturalmente espandibile e migliorabile in molte direzioni: innanzitutto adattando l'agent ad OS Windows, e in secondo luogo -a mio giudizio- introducendo la possibilità di modificare da remoto (e.g.: dalla dashboard) la configurazione degli agent. Gli stretti vincoli temporali non hanno consentito lo sviluppo di tali *feature*, che tuttavia potranno essere implementate da chi mi seguirà in questa sede, utilizzando il *repository* di GitHub [54] che contiene i sorgenti condivisi (***open-source***).

Bibliografia

[1]: Wikipedia, Liskov substitution principle, , https://en.wikipedia.org/wiki/Liskov_substitution_principle

[2]: Wikipedia, IEEE 802.3, , https://en.wikipedia.org/wiki/IEEE_802.3

[3]: TechGenix Ltd., Understanding Man-in-the-Middle Attacks – ARP Cache Poisoning (Part 1), , http://www.windowsecurity.com/articles-tutorials/authentication_and_encryption/Understanding-Man-in-the-Middle-Attacks-ARP-Part1.html

[4]: Wikipedia, Arpwatch, , https://en.wikipedia.org/wiki/Arpwatch

[5]: IETF, RFC-826: An Ethernet Address Resolution Protocol, November 1982, https://tools.ietf.org/html/rfc826

[6]: Wikipedia, 1982 year events, , https://en.wikipedia.org/wiki/1982#Events

[7]: Wikipedia, Address Resolution Protocol, , https://en.wikipedia.org/wiki/Address_Resolution_Protocol#Operating_scope

[8]: Cisco Learning Network, ARP and ICMP Which layers??, , https://learningnetwork.cisco.com/thread/36117

[9]: IETF, RFC-1180: A TCP/IP Tutorial, , https://tools.ietf.org/html/rfc1180#page-11

[10]: IETF, RFC-4861: Neighbor Discovery for IP version 6 (IPv6), , http://www.ietf.org/rfc/rfc4861.txt

[11]: Cisco, ARP FAQ: Why are some dynamic ARP entries still present in the ARP table after the respective ARP timeout has expired?, , http://www.cisco.com/c/en/us/support/docs/ip/address-resolution-protocol-arp/117398-qanda-arp-timeout-00.html

[12]: Erikr, ARP CACHE TIMEOUT CHANGED IN WINDOWS VISTA AND 2008, , http://blogs.msmvps.com/erikr/2008/09/13/arp-cache-timeout-changed-in-windows-vista-and-2008/

[13]: Microsoft, Description of Address Resolution Protocol (ARP) caching behavior in Windows Vista TCP/IP implementations, , https://support.microsoft.com/en-us/kb/949589

[14]: Microsoft, ARP Cache, , https://technet.microsoft.com/en-us/library/cc958841.aspx

[15]: Ubuntu, How long is an ARP entry cached for?, , http://askubuntu.com/questions/516567/how-long-is-an-arp-entry-cached-for

[16]: Cisco, ARP cache timeout on Cisco routers, , https://supportforums.cisco.com/discussion/11416946/arp-cache-timeout-cisco-routers

[17]: Wikipedia, IEEE-802.1Q: VLAN, , https://en.wikipedia.org/wiki/IEEE_802.1Q

[18]: Wikipedia, Ethernet frame, , https://en.wikipedia.org/wiki/Ethernet_frame#Structure

[19]: IEEE, MAC-address OUI list, , http://standards-oui.ieee.org/oui.txt

[20]: Wikipedia, MAC-address, , https://en.wikipedia.org/wiki/MAC_address

[21]: IETF, RFC-5227: IPv4 Address Conflict Detection, , https://tools.ietf.org/html/rfc5227#section-1.2.1

[22]: IETF, RFC-1180: A TCP/IP Tutorial, , https://tools.ietf.org/html/rfc1180#page-9

[23]: Wireshark.org, Gratuitous ARP, , https://wiki.wireshark.org/Gratuitous_ARP

[24]: IETF, RFC-3927: Dynamic Configuration of IPv4 Link-Local Addresses, , https://tools.ietf.org/html/rfc3927#page-12

[25]: Cisco, ARP destination: 00:00:00_00:00:00 (00:00:00:00:00:00), , https://supportforums.cisco.com/discussion/11157626/arp-destination-0000000000-000000000000

[26]: IETF, RFC-2131: Dynamic Host Configuration Protocol, , https://tools.ietf.org/html/rfc2131#page-35

[27]: IETF, RFC-5944: IP Mobility Support for IPv4, Revised, , https://tools.ietf.org/html/rfc5944#section-4.6

[28]: Bryan Burns, Dave Killion, et al., Security Power Tools, , ISBN-978-0-596-00963-2

[29]: GEEKBLOGTV, Arp Cache Poisoning - Man In The Middle Attack, , https://www.youtube.com/watch?v=fjgWZyk4kGw

[30]: Wikipedia, ARP spoofing, , https://en.wikipedia.org/wiki/ARP_spoofing

[31]: Lawrence Berkeley National Laboratory, LBNL's Network Research Group, , http://ee.lbl.gov/

[32]: Everything about nothing, arpwatch on multiple interfaces, , http://sgros.blogspot.it/2012/01/arpwatch-on-multiple-interfaces.html

[33]: Linux OS, Command-line manual (man), ,

[34]: Debianizzati.org, Monitorare l'attività ARP con Arpwatch, , http://guide.debianizzati.org/index.php/Monitorare_l'attivit%C3%A0_ARP_con_Arpwatch

[35]: Fedora Project, arpwatch-2.1a15-35.fc23 RPM for ppc64le, , ftp://ftp.rpmfind.net/linux/RPM/fedora/updates/23/ppc64le/a/arpwatch-2.1a15-35.fc23.ppc64le.html

[36]: RedHat, [arpwatch] Gives 'bogon' message for 0.0.0.0 IP, , https://bugzilla.redhat.com/show_bug.cgi?id=78728

[37]: RedHat, apple mac mini floods arpwatch, , https://bugzilla.redhat.com/show_bug.cgi?id=244606

[38]: S.Vidya, R.Bhaskaran, ARP Storm Detection and Prevention Measures, , http://ijcsi.org/papers/IJCSI-8-2-456-460.pdf

[39]: Roger S. Pressman, Principi di Ingegneria del software, gennaio 2008, ISBN-9788838664182

[40]: Wikipedia, MoSCoW method, , https://en.wikipedia.org/wiki/MoSCoW_method

[41]: Wikipedia, Model–view–controller, , https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

[42]: Christoph Mayer, XArp – Advanced ARP Spoofing Detection, , http://www.xarp.net/

[43]: Oracle, What Are RESTful Web Services?, , http://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html

[44]: Wikipedia, Web-app, , https://en.wikipedia.org/wiki/Web_application

[45]: Apache, Cordova Overview, , https://cordova.apache.org/docs/en/latest/guide/overview/index.html

[46]: IETF, RFC-7159: The JavaScript Object Notation (JSON) Data Interchange Format, , https://tools.ietf.org/html/rfc7159

[47]: Ubuntu, Network interface name changes after update to 15.10 - udev changes, , http://askubuntu.com/questions/689070/network-interface-name-changes-after-update-to-15-10-udev-changes

[48]: W3C, Priviliged ports, , https://www.w3.org/Daemon/User/Installation/PrivilegedPorts.html

[49]: Wikipedia, List of TCP and UDP port numbers, , https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers#Well-known_ports

[50]: Oracle, Oracle Java SE Support Roadmap, , http://www.oracle.com/technetwork/java/eol-135779.html

[51]: Spring Framework, Spring Tool Suite, , https://spring.io/tools

[52]: Spring Framework, Building a RESTful Web Service, , https://spring.io/guides/gs/rest-service/

[53]: Spring Framework, Building an Application with Spring Boot, , https://spring.io/guides/gs/spring-boot/

[54]: GFcyborg, Arpwatch-Dashboard, , https://github.com/GFcyborg/arpwatch-dashboard

Ringraziamenti:

Un "Grazie!" gigante va alla mia famiglia, che mi ha supportato e sopportato nei lunghi giorni (e notti) di preparazione e studio. Marghe, Emma, Leo e Pietro: vi voglio bene!

Grazie anche a Mamma, che mi ha insegnato a ridere della vita e delle persone troppo serie.

Grazie a Babbo, che mi ha insegnato che anche gli Ingegneri sanno piangere. Mi manchi, Babbo!

Grazie a Chiara, che ha attraversato con me il deserto, e ne è uscita viva.

Grazie a Roberto, che mi ha consigliato con generosità e curiosità.

Grazie ai miei suoceri, Sandrina e Gino, perchè ci sono sempre, quando ne abbiamo bisogno.