ALMAMATER STUDIORUM - Università degli studi di Bologna
Campus di Cesena

Scuola di Ingegneria
Corso di laurea in Ingegneria e Scienze Informatiche

Sistema di monitoraggio rete layer 2 attraverso ARPwatch

Tesi di Laurea
in
System Integration

Laureando:
Michele Proscia

Relatore:
Prof. Vittorio Ghini

Indice

Capitolo 1: Introduzione

In un sistema altamente complesso come la rete universitaria è necessario monitorare costantemente i dispositivi connessi in modo da far fronte a possibili disservizi, attacchi o misconfiguration (configurazione errata).

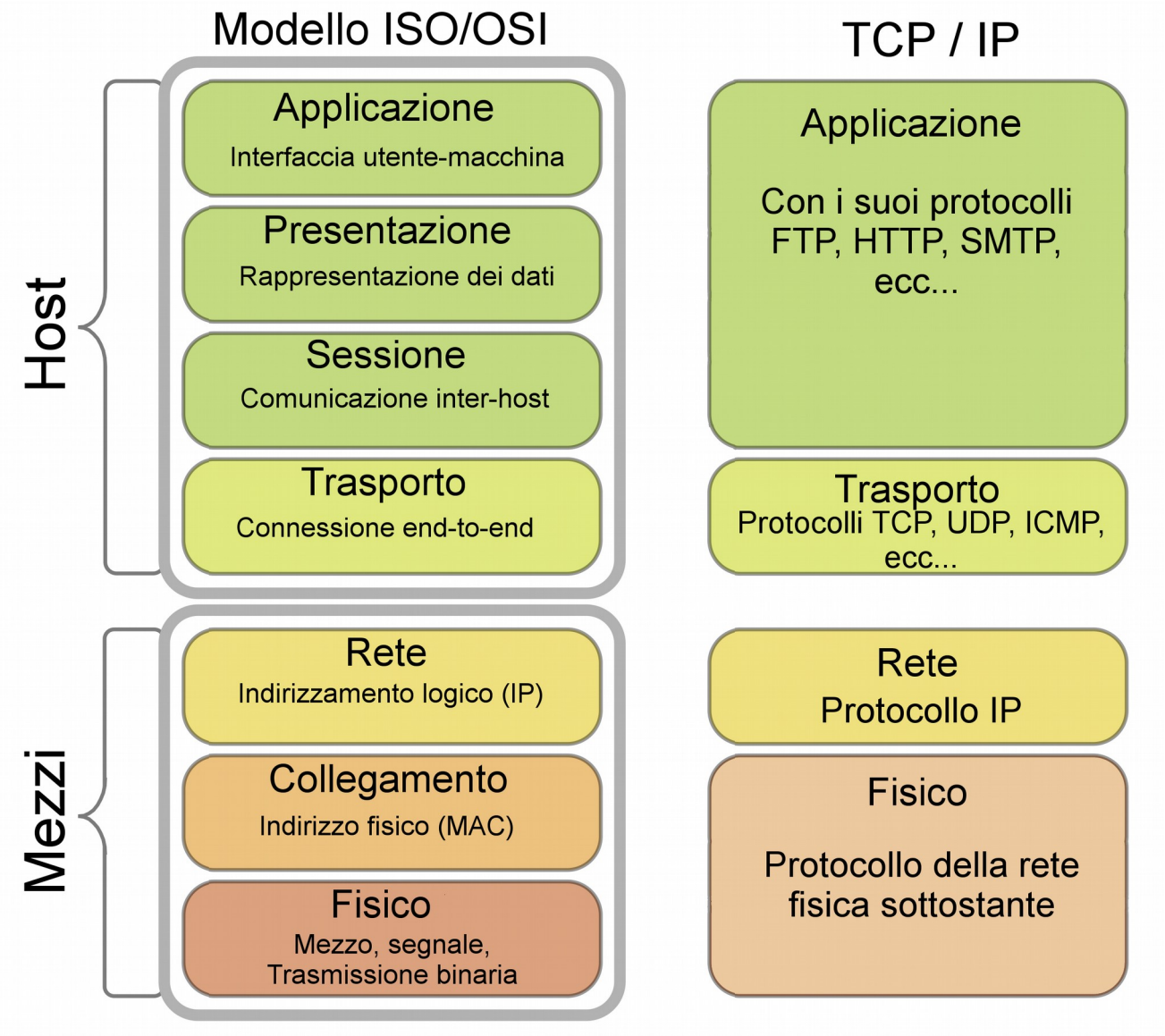
Questo progetto è nato per far fronte ad alcune anomalie molto comuni all'interno della facoltà, quali conflitti di IP o computer che prendono IP non registrati sul server DNS o non a loro assegnati; queste in particolare sono difficili da individuare perché sono molto volatili, se non avvengono nel momento in cui le si sta osservando è impossibile ricostruire le dinamiche della loro manifestazione. Per ricreare le dinamiche delle anomalie il sistema da me creato cattura i dati parziali provenienti da ARPwatch e IMC per catalogare tutti i dispositivi della rete LAN universitaria e storicizzarli all'interno di un database, in modo che siano facilmente consultabili tramite un portale web. Inoltre i dati raccolti facilitano piccoli lavori di manutenzione come controllare lo stato dei server DNS o conoscere gli indirizzi IP inutilizzati che possono essere ricollocati dove c'è più bisogno.

Il progetto è stato sviluppato in queste fasi:

- Individuazione dei dati da raccogliere
- studio del metodo di raccolta dei dati
- stesura degli script
- creazione del portale web di interrogazione
- messa in funzione del sistema
- manutenzione ed usi futuri

Capitolo 2: Stack ISO/OSI

Modello ISO/OSI



Il modello OSI (chiamato anche ISO/OSI) è uno standard di riferimento per la realizzazioni di reti aperte di computer. L'OSI viene rappresentato come una pila suddivisa in sette strati, per ognuno dei quali sono definiti i protocolli che realizzano le funzioni assegnate a ciascun livello:

- Livello 7: Applicazione

  Si occupa delle applicazioni connesse alla rete.

  Questo strato fornisce protocolli che operano a stretto contatto con le applicazioni che a loro volta permettono agli utenti di comunicare con la macchina.

- Livello 6: Presentazione

  Fornisce una rappresentazione standard dei dati per le applicazioni.

  Ha l'obbiettivo di trasformare i dati forniti dalle applicazioni in un formato standard e fornire alcuni servizi come la crittografia, la compressione e la riformattazione dei dati.

- Livello 5: Sessione

  Gestisce la sessione fra le applicazioni.

  Si occupa di gestire le modalità di dialogo (mono direzionale o bidirezionale) tra i dispositivi. In questo strato vengono implementate tecniche per la gestione di risorse condivise e la possibilità, tramite checkpoint, di riprendere una comunicazione interrotta prima del suo completamento. Questo livello controllare la comunicazione tra applicazioni, instaura, mantene ed abbatte connessioni (sessioni) tra applicazioni cooperanti.

- Livello 4: Trasporto

  Fornisce la connessione end-to-end con controllo della congestione.

  Si occupa di stabilire, mantenere e terminare una connessione garantendo il funzionamento ottimale della sottorete ed evita che arrivino troppi pacchetti allo stesso router per scongiurare le congestioni. Il livello 4 inoltre effettua la frammentazione dei dati provenienti dal livello superiore in "segmenti" per trasmetterli in modo efficiente attraverso il livello Rete.

- Livello 3: Rete

  Gestisce la connessione alla rete.

  Il livello 3 ha il compito di rendere i livelli superiori indipendenti dai mezzi di trasmissione dei dati definiti nei livelli sottostanti e occuparsi della consegna dei pacchetti. Si occupa di fare Routing, ossia scegliere il percorso ottimale per garantire la consegna dei dati e di convertire i pacchetti per il passaggio da un tipo di rete ad un altro, ossia:

  - tradurre gli indirizzi di rete
  - Se la nuova rete ha un MTU (Maximum Transmission Unit) differente valutare la necessità di frammentare i pacchetti
  - Valutare la necessità di gestire diversi protocolli attraverso l'utilizzo di getway

- Livello 2: Collegamento

  Provvede alla trasmissione dei dati sulla rete fisica.

  Questo livello si occupa di preparare i dati, suddivisi in frame, alla trasmissione su livello fisico (nella maggior parte dei casi il filo in rame), esso dovrà sincronizzare i dispositivi ed accertarsi che le trasmissioni avvengano senza errori. I frame non sono altro che i dati provenienti dallo strato superiore che vengono racchiusi in un nuovo pacchetto a cui viene aggiunto un nuovo header (testa) e tail (coda), i frame possono anche essere utilizzati per sequenze di controllo. Per controllare la presenza di errori ogni qual volta il destinatario riceve un pacchetto risponde al mittente inviando un pacchetto ACK (acknowledgement, conferma) contenette l'esito della transazione, se non è andata a buon fine o non e stato ricevuto nessun ACK il mittente re-invia il pacchetto. Nel caso in cui uno dei due dispositivi sia più veloce dell'altro, il

protocollo rallenta la velocità di trasmissione del più veloce per evitare perdite di pacchetti dovute alla congestione risultante da una connessione sbilanciata.
Il livello 2 è in oltre suddiviso in 2 sottolivelli:

- MAC (Medium Access Controll): Si occupa di controllare quali dispositivi possono accedere alla rete e se possono trasmettere dati.
- LLC (Logical Link Control): Incapsula i dati e controlla gli errori e la sincronizzazione dei frame.

- Livello 1: Fisico

  Definisce le caratteristiche fisiche della rete.

  Trasmette il flusso di dati attraverso un mezzo fisico (filo di rame, onde radio, fibra ottica, ecc…) occupandosi delle forme d'onda, dei voltaggi e delle frequenze. Nel livello 1 vengono specificati: il numero di bit in un collegamento, le tensioni per rappresentare i livelli logici, la durata in microsecondi del segnale che identifica i bit, la modulazione e la codifica, il tipo di trasmissione hal-duplex o full-duplex e la forma meccanica dei connettori.

ARP

L’Address Resolution Protocol è uno standard pubblicato nel 1982, nonostante la sua veneranda età è un protocollo ancora molto flessibile e di vitale importanza per le reti odierne di computer. Da un punto di vista puramente teorico, esistono diverse scuole di pensiero riguardo alla collocazione di ARP all’interno dello stack ISO/OSI:

- esso è a **livello 3** (Rete) in ragione del fatto che i suoi messaggi sono incapsulati nel payload di un pacchetto di livello 2 (e.g.: trame Ethernet);
- nello stack TCP/IP invece ARP è a **livello 2** (Collegamento), in ragione del fatto che fornisce servizi di risoluzione degli indirizzi fisici a partire da quelli logici di rete.

Per esprimere questa ambiguità, spesso si colloca ARP in un livello intermedio, indicato con L2.5, e più propriamente denominato Logical-Link Control (LLC)

Fonte: https://tools.ietf.org/html/rfc826

DNS

Il Domain Name System permette di risolvere gli indirizzi IP dei dispositivi di rete partendo da nomi, che solitamente sono più facili da ricordare che una serie di numeri. Supponiamo che un client voglia contattare `csi-syslog2018.campusfc.unibo.it`, non sapendo dove si trova invia una richiesta al server DNS, questo controlla la sua lista associata nomi-IP, fa un confronto e se ha esito positivo ci risponde con l’indirizzo IP (es. `137.204.10.2`) del destinatario che stavamo cercando, a questo punto il client inizia a comunicare direttamente con il nodo richiesto, nel caso in cui non ci sia nessuna corrispondenza il server ci risponde con l’errore 105 (ERR_NAME_NOT_RESOLVED).

DHCP

Dynamic Host Configuration Protocol è il protocollo che si occupa di comunicare ai dispositivi che fanno richiesta di accesso alla rete la corretta configurazione IP necessaria per stabilire una connessione con la rete locale. In una rete basata su Internet Protocol (IP) è importante che ad ogni dispositivo venga assegnato un indirizzo IP facente parte dell'insieme degli indirizzi ammessi all'intera sottorete e che sia univoco. In una rete molto ampia e popolosa sarebbe impossibile per un amministratore assegnare gli indirizzi IP a mano per ogni singolo computer, ancor di più se questi computer sono volatili e non sempre disponibili, per questo è possibile programmare il server DHCP in modo che assegni gli indirizzi in autonomia:

- Allocazione Dinamica: Viene impostato un intervallo di indirizzi IP che il server DHCP potrà assegnare ai dispositivi che ne fanno richiesta. Il server DHCP inoltre controlla ad ogni intervallo di tempo gli indirizzi per tenere traccia di quelli assegnati e liberi.
- Allocazione Automatica: Il funzionamento è identico a quella Dinamica, ma in questo caso il server DHCP conserva una tabella degli indirizzi IP assegnati e cerca di riassegnare sempre lo stesso indirizzo allo stesso dispositivo, ovviamente quando ciò è possibile.

Capitolo 3: Dispositivi e Strumenti

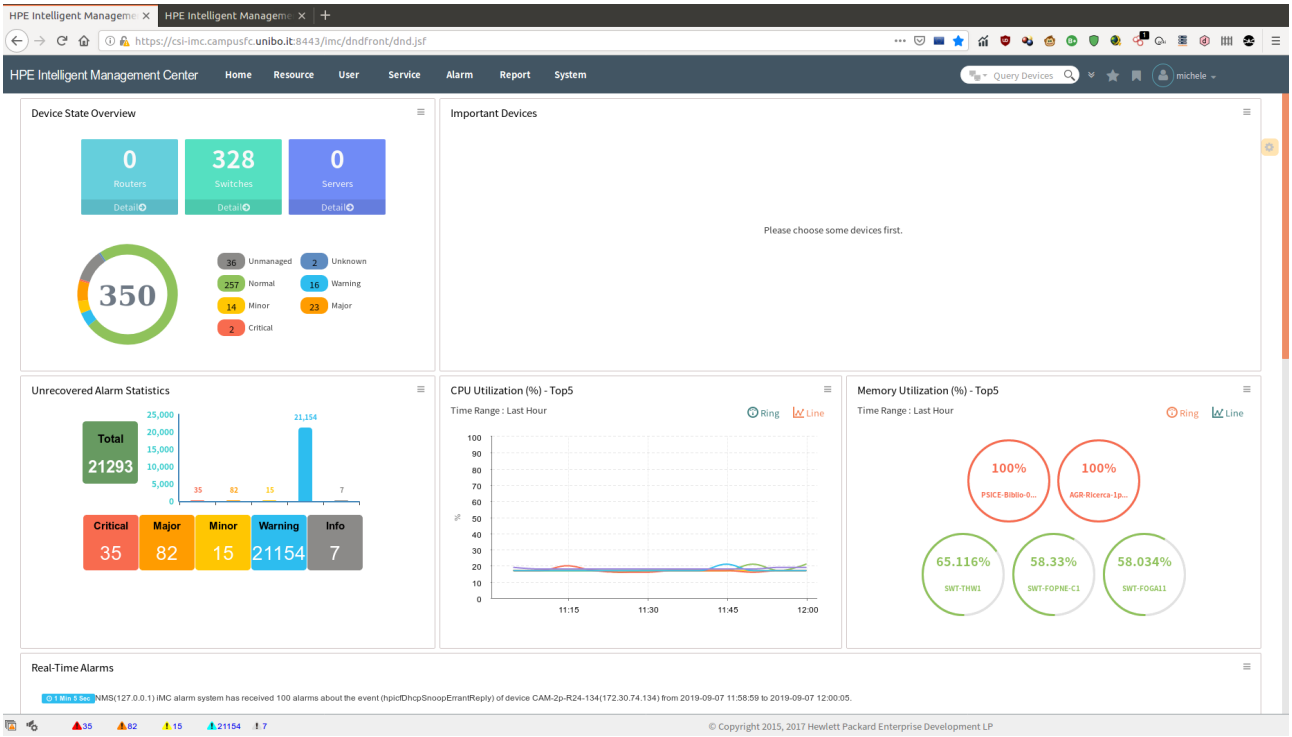
Switch e VLAN

Uno Switch è un apparato di rete che mette in comunicazione bidirezionale tutti i dispositivi a lui collegati. A differenza dell'HUB (che invia ogni pacchetto a tutti quanti indistintamente) lo Switch è in grado di memorizzare la porta a cui è collegato uno specifico dispositivo (tramite il suo indirizzo MAC) e di conseguenza direzionare i pacchetti verso la porta a cui è collegato il rispettivi destinatari, in questo modo riduce notevolmente la congestione dell'infrastruttura e limita le collisioni (i broadcast vengono comunque inviati a tutti). Nei moderni Switch è possibile effettuare alcune configurazioni, come la prioritizzazione dei pacchetti o la creazione di VLAN. Una VLAN (Virtual Local Area Network) è, come suggerisce il nome, una rete locale virtuale; ossia una rete separata da quella fisica ma che comunque risiede su di essa. Le VLAN sono molto utilizzata all'interno dell'università, permette di creare reti isolate tra di loro pur condividendo lo stesso supporto fisico. In questo modo ogni laboratorio ha la propria sotto-rete indipendente dalle altri e anche i dispositivi collegati all'ALMAWIFI si trovano in una VLAN; così è più facile tenere sotto controllo tutti i computer, arginare i malfunzionamenti e far fronte ad eventuali attacchi informatici. Oltretutto i docenti possono richiedere l'attivazione di VLAN a scopo didattico o per ricerca, magari temporanee, per i loro progetti (es. connessione

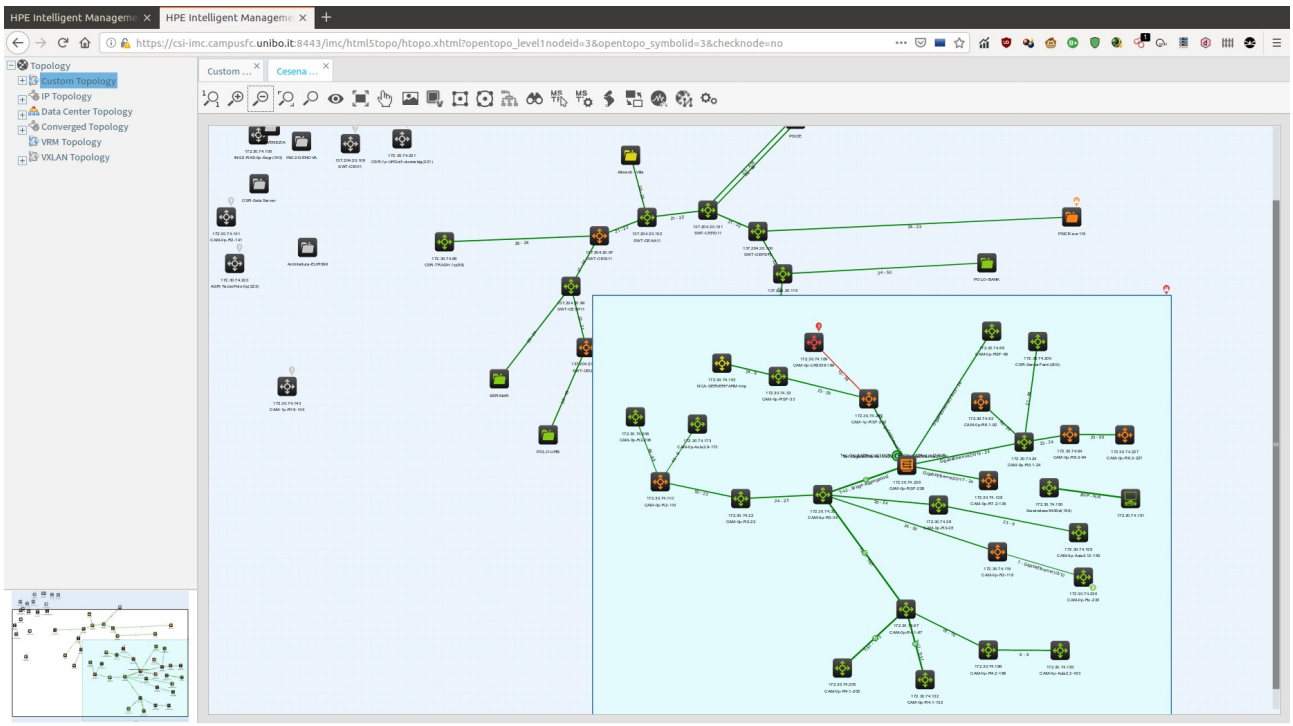
di schede raspberry o simili ad internet) e i tecnici possono crearle semplicemente riprogrammando gli Switch senza mettere mano alla rete fisica.

IMC

Per raccogliere tutti i dati si è pensato di interrogare il sistema di monitoraggio IMC (Intelligent Management Center) fornito da HP e già installato sulla rete locale dell'università. IMC è un potentissimo strumento in grado di monitorare molteplici aspetti di una rete, è in grado di rilevare svariati errori e generare allarmi specifici; può creare automaticamente mappe della rete dialogando direttamente con i dispositivi che la compongono per avere rapidamente uno schema della nostra infrastruttura.



Tutte queste funzioni sono interrogabili tramite REST API.



ARPwatch

ARPwatch è un programma open-source scritto in C, non possiede interfaccia grafica (GUI), è implementato per operare come servizio/demone Linux, non necessita di particolari permessi e non richiede molte risorse di calcolo per svolgere le sue mansioni. La funzione di ARPwatch è quella di monitorare tutte le richieste ARP ed avvisare il personale tramite mail o syslog in caso di anomalie, è in grado di controllare una sola interfaccia di rete alla volta, ma è possibile lanciare più istanze concorrenti per monitorarne di più (come nel nostro caso per monitorare tutte le VLAN).

ARPwatch è in grado di rilevare diverse anomalie, queste sono alcune delle più comuni in cui mi sono imbattuto durante lo sviluppo di questo progetto:

- **New Station:** La prima volta che ARPwatch incontra traffico ARP proveniente da un indirizzo MAC, lo notifica con questo evento. Durante i primi minuti di

esecuzione di ARPwatch, dopo un'installazione da zero, oppure in contesti intrinsecamente transitori, c'è da aspettarsi un gran numero di eventi di questo tipo.

- **New Activity:** Questo evento riguarda le coppie (MAC, IP) di cui si ha già memoria, ma di cui non si registra attività da almeno 6 mesi.
- **Flip Flop:** In questo caso ci troviamo difronte a due indirizzi MAC che si contendono lo stesso indirizzo IP e se lo rubano in continuazione (da qui il nome evocativo) è un chiaro segno della presenza di una mal configurazione di qualche host o di un qualche tipo di attacco malevolo.
- **Changed Ethernet Address:** Questo evento riguarda il caso in cui uno stesso IP ha cambiato proprietario (ossia indirizzo MAC): questo può indicare sia attività malevole/sospette, sia normali disguidi nella configurazione o attività legittime come l’operato di un servizio DHCP in una rete sottodimensionata.
- **Reused Old Ethernet Address:** Questo evento è simile al precedente (flip flop), da cui si differenzia solo perché la contesa dell'IP non avviene tra i 2 MAC più recenti, ma tra il 1° e il 3°(un host più vecchio del 2°).

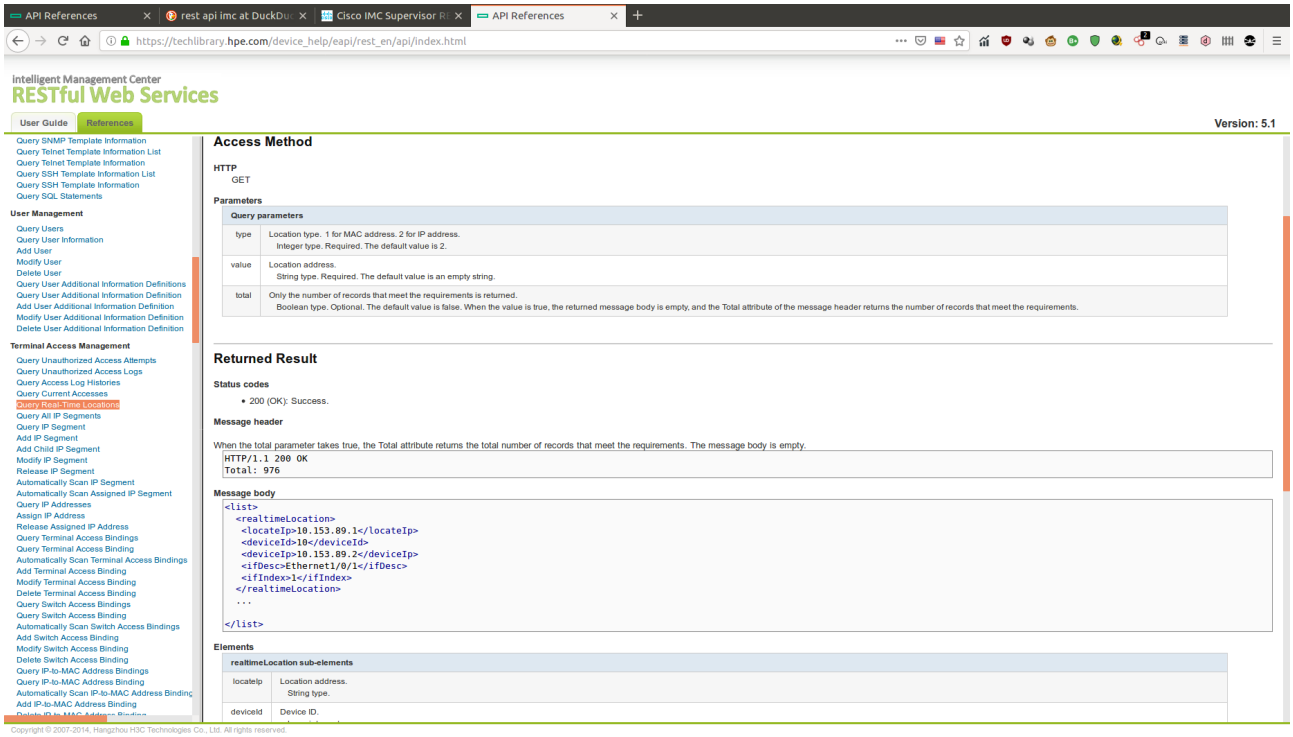
Manuale: https://linux.die.net/man/8/arpwatch

Come si può notare anche ARPwatch lavora contemporaneamente su due livelli, il 2 e il 3.

REST API

HP fornisce delle REST API per interrogare il loro sistema di controllo integrato IMC attraverso delle richieste http (GET, POST, PUT, ecc…), grazie a questo tipo di struttura è possibile sfruttare le API con qualsiasi linguaggio di programmazione (noi abbiamo usato PHP). A questo link possiamo notare la moltitudine di funzioni messe a disposizione da IMC in particolare in questo progetto abbiamo ampiamente utilizzato **Query Real-Time Locations**

http://h17007.www1.hpe.com/device_help/eapi/rest_en/api/index.html



È possibile ricevere i dati delle richieste come Json o XML.

Cron

È un programma presente nei sistemi Unix, composto da un demone (`crond`) che ogni minuto controlla un file (`crontab`) ed esegue i comandi in esso contenuti.
Ogni comando (chiamato `cronjob`) viene inserito riga per riga nel `crontab` ed ha una codifica speciale per indicare con quale frequenza deve essere eseguito.
È possibile specificare la frequenza di un cronjob tramite 5 campi:

m	Minuto (0 - 59)
h	Ora (0 - 23)
dom	Giorno del mese (1 - 31)
mon	Mese (1 - 12)
dow	Giorno della settimana (0 - 6)

Ogni campo può essere impostato con più valori, ma per questo ci servono dei parametri:

- La virgola “,” indica una lista di valori (es. 2,3,5)
- Il trattino “-” specifica un intervallo (es. 1-5 equivale a 1,2,3,4,5)
- L’asterisco “*” questo è come un jolly, indica tutti i valori possibili (es. un * nel campo dei minuti è come se indicasse)

Possiamo notare il `crontab` messo a punto per il nostro sistema:

```
#m   h dom mon dow  command
 *   * *   *   *    ls /tmp/arpw/ | wc -l > /var/www/html/network_dev/coda.txt
 @reboot            /root/vlanstart #attiva le vlan al'avvio
 */5 * *   *   *    /usr/bin/php -f /var/www/html/network_dev/script/ARP-IM...
```

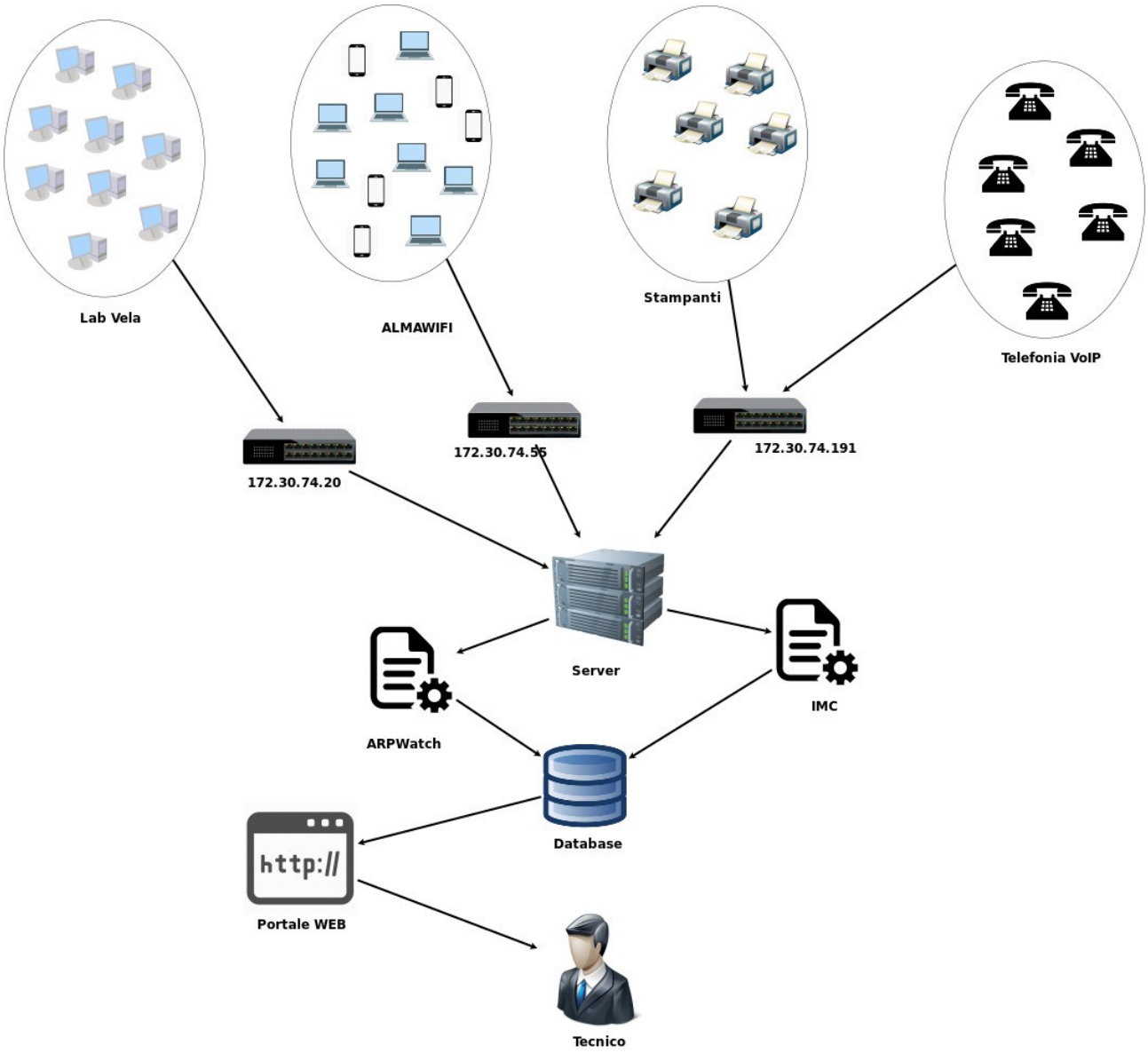
Il primo cronjob esegue ogni minuto, il secondo una volta ad ogni avvio del sistema e l’ultimo ogni 5 minuti.

Laravel

Laravel è un framework opensource scritto in PHP, è molto versatile e ricco di librerie che velocizzano l'implementazione dei siti internet; in particolare per questo progetto è stata utilizzata la versione 5.6.39.
È stato scelto questo framework perché di tipo MVC e quindi si avvicina alla più moderna programmazione orientata agli oggetti e anche perché è sempre più utilizzato in ambito aziendale.
Laravel utilizza dei costrutti in un suo linguaggio che rende il codice più leggibile, il server web converte questo codice nei classici tag HTML e PHP, che vengono poi letti dai browser e interpretati come una normalissima pagina web; in questo modo otteniamo un codice più compatto e di rapida comprensione mantenendo tutte le funzionalità del linguaggio.

Installazione: https://laravel.com/docs/5.6/installation

Capitolo 4: Obbiettivi della Tesi



L’obbiettivo di questo progetto è quello di riunire i due sistemi di monitoraggio già in uso in un unico sistema, in modo che sia più facile tenere traccia dei malfunzionamenti e garantire una più rapida risoluzione degli stessi.

Il server è la macchina fisica da noi utilizzata e comunica, attraverso i vari switch, con tutte le VLAN della rete; su di esso sono installati gli script e il server web che ospita il portale di interrogazione. Gli script interrogando ARPwatch e IMC

popoleranno il database che potrà essere consultato dal personale tecnico, previo login, per la risoluzione dei malfunzionamenti.

Il sistema monitora 180 switch che ospitano 22 delle VLAN presenti in tutto il polo di Cesena; all'attivo il database conta più di 13.000 record. Il sistema è attivo da circa due mesi (tralasciando i mesi estivi per la minore affluenza), in questo periodo sono stati registrati 3.780 dispositivi unici e circa 10.000 segnalazioni di ARPwatch (non c'è da spaventarsi, la maggior parte di queste non sono segnalazioni gravi o che richiedono un intervento immediato). In futuro i dispositivi aumenteranno, molto lentamente ma aumenteranno, e di conseguenza anche le segnalazioni ma in numero più elevato.

Capitolo 5: Progettazione ed Implementazione

Il progetto è stato sviluppato in ambiente Linux (Ubuntu 18.04) con l'ausilio dell'editor Atom.

Individuazione dei dati

In questa fase ho fatto alcune ricerche per identificare i dati più utilizzati dai tecnici, le richieste più di frequente fatte ad IMC e come disporre la visualizzazione tabella del database. Dopo un'accurata analisi le informazioni scelte per strutturare il database sono state:

- Data e Ora
- IP
- MAC
- IP dello switch
- porta dello switch
- VLAN
- nome DNS
- nome DNS incrociato
- nome DNS active directory
- info

I campi **Data**, **IP** e **MAC** vengono forniti direttamente da ARPwatch e servono per identificare quando è stato rilevata un'anomalia e a chi; **VLAN**, l'**IP** e la **porta dello switch** servono per identificare con precisione dove si trova il dispositivo all'interno dello stabile e di quale sotto-rete fa parte; le richieste dei **nomi di dominio** invece servono per verificare lo stato della tabelle dei due server DNS in uso sulla rete

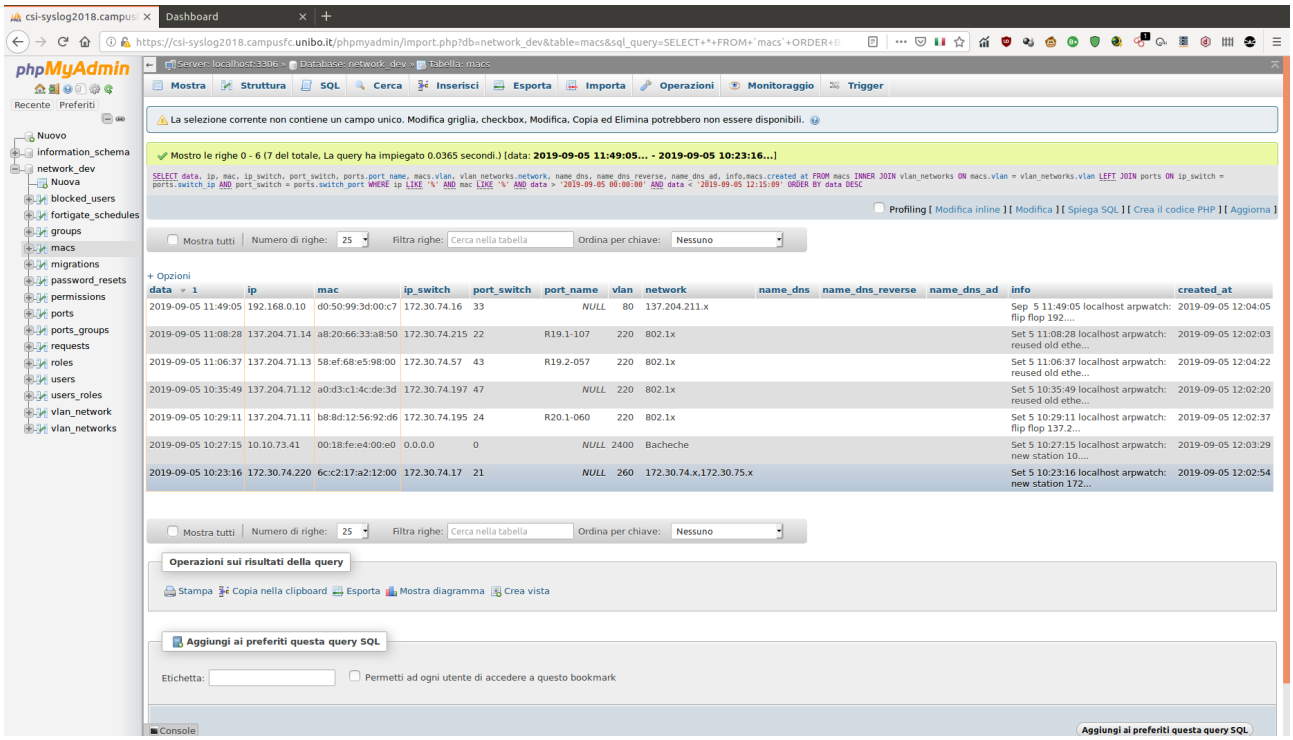
dell'università; in fine la colonna **Info** contiene tutti i dati grezzi ricevuti dalle varie interrogazioni, in questo modo è possibile monitorare il corretto operato del sistema di filtraggio ed eventualmente usare i dati grezzi in altri applicativi per scopi futuri.

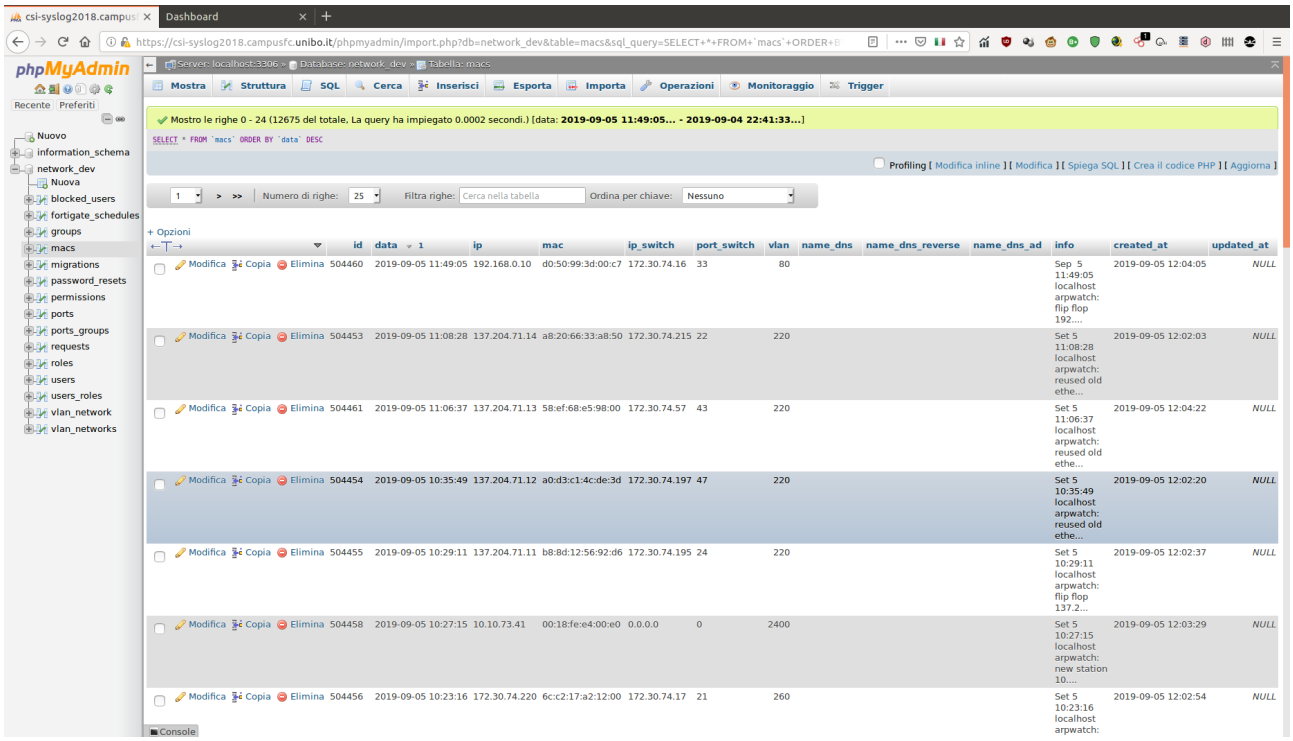
Query SQL creazione:

```
CREATE TABLE macs (
     id int,
     data datatype,
     ip varchar(255),
     mac varchar(255),
     ip_switch varchar(255)s,
     port_switch varchar(255),
     vlan varchar(255),
     name_dns varchar(255),
     name_dns_reverse varchar(255),
     name_dns_ad varchar(255),
     Info varchar(1024)
);
```

Esempio query di interrogazione:

```
SELECT data, ip, mac, ip_switch, port_switch, ports.port_name,
macs.vlan, vlan_networks.network, name_dns, name_dns_reverse,
name_dns_ad, info, macs.created_at
FROM macs
INNER JOIN vlan_networks ON macs.vlan = vlan_networks.vlan
LEFT JOIN ports ON ip_switch=ports.switch_ip
AND port_switch=ports.switch_port
WHERE ip LIKE "%" AND mac LIKE "%"
AND data > '2019-04-01 00:00:00'
AND data < '2019-05-01 00:00:00'
ORDER BY `data` DESC
```


csi-syslog2018.campus
Dashboard
https://csi-syslog2018.campusfc.unibo.it/phpmyadmin/import.php?db=network_dev&table=macs&sql_query=SELECT+*+FROM+`macs`+ORDER+B
phpMyAdmin
Recente Preferiti
Nuovo
information_schema
network_dev
Nuova
blocked_users
fortigate_schedules
groups
macs
migrations
password_resets
permissions
ports
ports_groups
requests
roles
users
users_roles
vlan_network
vlan_networks
Server: localhost:3306 » Database: network_dev » Tabella: macs
Mostra
Struttura
SQL
Cerca
Inserisci
Esporta
Importa
Operazioni
Monitoraggio
Trigger
La selezione corrente non contiene un campo unico. Modifica griglia, checkbox, Modifica, Copia ed Elimina potrebbero non essere disponibili.
Mostro le righe 0 - 6 (7 del totale, La query ha impiegato 0.0365 secondi.) [data: 2019-09-05 11:49:05... - 2019-09-05 10:23:16...]
SELECT data, ip, mac, ip_switch, port_switch, ports.port_name, macs.vlan, vlan_networks.network, name_dns, name_dns_reverse, name_dns_ad, info,macs.created_at FROM macs INNER JOIN vlan_networks ON macs.vlan = vlan_networks.vlan LEFT JOIN ports ON ip_switch = ports.switch_ip AND port_switch = ports.switch_port WHERE ip LIKE '%' AND mac LIKE '%' AND data > '2019-09-05 00:00:00' AND data < '2019-09-05 12:15:09' ORDER BY data DESC
Profiling [Modifica inline] [Modifica] [Spiega SQL] [Crea il codice PHP] [Aggiorna]
Mostra tutti
Numero di righe: 25
Filtra righe: Cerca nella tabella
Ordina per chiave: Nessuno
+ Opzioni
data 1 ip mac ip_switch port_switch port_name vlan network name_dns name_dns_reverse name_dns_ad info created_at
2019-09-05 11:49:05 192.168.0.10 d0:50:99:3d:00:c7 172.30.74.16 33 NULL 80 137.204.211.x Sep 5 11:49:05 localhost arpwatch: flip flop 192.... 2019-09-05 12:04:05
2019-09-05 11:08:28 137.204.71.14 a8:20:66:33:a8:50 172.30.74.215 22 R19.1-107 220 802.1x Set 5 11:08:28 localhost arpwatch: reused old ethe... 2019-09-05 12:02:03
2019-09-05 11:06:37 137.204.71.13 58:ef:68:e5:98:00 172.30.74.57 43 R19.2-057 220 802.1x Set 5 11:06:37 localhost arpwatch: reused old ethe... 2019-09-05 12:04:22
2019-09-05 10:35:49 137.204.71.12 a0:d3:c1:4c:de:3d 172.30.74.197 47 NULL 220 802.1x Set 5 10:35:49 localhost arpwatch: reused old ethe... 2019-09-05 12:02:20
2019-09-05 10:29:11 137.204.71.11 b8:8d:12:56:92:d6 172.30.74.195 24 R20.1-060 220 802.1x Set 5 10:29:11 localhost arpwatch: flip flop 137.2... 2019-09-05 12:02:37
2019-09-05 10:27:15 10.10.73.41 00:18:fe:e4:00:e0 0.0.0.0 0 NULL 2400 Bacheche Set 5 10:27:15 localhost arpwatch: new station 10.... 2019-09-05 12:03:29
2019-09-05 10:23:16 172.30.74.220 6c:c2:17:a2:12:00 172.30.74.17 21 NULL 260 172.30.74.x,172.30.75.x Set 5 10:23:16 localhost arpwatch: new station 172... 2019-09-05 12:02:54
Operazioni sui risultati della query
Stampa
Copia nella clipboard
Esporta
Mostra diagramma
Crea vista
Aggiungi ai preferiti questa query SQL
Etichetta:
Permetti ad ogni utente di accedere a questo bookmark
Console
Aggiungi ai preferiti questa query SQL


csi-syslog2018.campus
Dashboard
https://csi-syslog2018.campusfc.unibo.it/phpmyadmin/import.php?db=network_dev&table=macs&sql_query=SELECT+*+FROM+`macs`+ORDER+B
phpMyAdmin
Recente Preferiti
Nuovo
information_schema
network_dev
Nuova
blocked_users
fortigate_schedules
groups
macs
migrations
password_resets
permissions
ports
ports_groups
requests
roles
users
users_roles
vlan_network
vlan_networks
Server: localhost:3306 » Database: network_dev » Tabella: macs
Mostra
Struttura
SQL
Cerca
Inserisci
Esporta
Importa
Operazioni
Monitoraggio
Trigger
Mostro le righe 0 - 24 (12675 del totale, La query ha impiegato 0.0002 secondi.) [data: 2019-09-05 11:49:05... - 2019-09-04 22:41:33...]
SELECT * FROM `macs` ORDER BY `data` DESC
Profiling [Modifica inline] [Modifica] [Spiega SQL] [Crea il codice PHP] [Aggiorna]
1
> >>
Numero di righe: 25
Filtra righe: Cerca nella tabella
Ordina per chiave: Nessuno
+ Opzioni
id data 1 ip mac ip_switch port_switch vlan name_dns name_dns_reverse name_dns_ad info created_at updated_at
Modifica Copia Elimina 504460 2019-09-05 11:49:05 192.168.0.10 d0:50:99:3d:00:c7 172.30.74.16 33 80 Sep 5 11:49:05 localhost arpwatch: flip flop 192.... 2019-09-05 12:04:05 NULL
Modifica Copia Elimina 504453 2019-09-05 11:08:28 137.204.71.14 a8:20:66:33:a8:50 172.30.74.215 22 220 Set 5 11:08:28 localhost arpwatch: reused old ethe... 2019-09-05 12:02:03 NULL
Modifica Copia Elimina 504461 2019-09-05 11:06:37 137.204.71.13 58:ef:68:e5:98:00 172.30.74.57 43 220 Set 5 11:06:37 localhost arpwatch: reused old ethe... 2019-09-05 12:04:22 NULL
Modifica Copia Elimina 504454 2019-09-05 10:35:49 137.204.71.12 a0:d3:c1:4c:de:3d 172.30.74.197 47 220 Set 5 10:35:49 localhost arpwatch: reused old ethe... 2019-09-05 12:02:20 NULL
Modifica Copia Elimina 504455 2019-09-05 10:29:11 137.204.71.11 b8:8d:12:56:92:d6 172.30.74.195 24 220 Set 5 10:29:11 localhost arpwatch: flip flop 137.2... 2019-09-05 12:02:37 NULL
Modifica Copia Elimina 504458 2019-09-05 10:27:15 10.10.73.41 00:18:fe:e4:00:e0 0.0.0.0 0 2400 Set 5 10:27:15 localhost arpwatch: new station 10.... 2019-09-05 12:03:29 NULL
Modifica Copia Elimina 504456 2019-09-05 10:23:16 172.30.74.220 6c:c2:17:a2:12:00 172.30.74.17 21 260 Set 5 10:23:16 localhost arpwatch: 2019-09-05 12:02:54 NULL
Console

Raccolta dei dati

I tecnici fino ad ora dovevano prima controllare i dati di ARPwatch, identificare l'anomalia e a mano interrogare più volte IMC (non è possibile avere tutti i dati con una singola interrogazione) attraverso il suo portale web; tutto con una cospicua perdita di tempo e senza la possibilità di avere uno storico, poiché IMC fornice solo dati dello stato attuale della rete.

Grazie ad IMC possiamo sapere l'indirizzo IP, l'indirizzo MAC, la porta e l'indirizzo dello switch, ma solo se il dispositivo in questione è acceso, se è spento IMC non può dirci nulla; per questo salviamo tutto sul database.

cURL (che sta per "see URL") è un programma a linea di comando che si occupa ti trasferire dati da o verso un server, è compatibile con svariati protocolli (http, https, sftp, telnet, ecc…). È possibile usare cURL per scaricare file, pagine web o addirittura interi siti internet.
Fonte: https://linux.die.net/man/1/curl

Wget è simile a cURL, ma è molto più semplice e povero di funzioni;
Fonte: https://linux.die.net/man/1/wget

Per quanto ci abbia provato non mi è stato possibile utilizzare `cURL` (avrei preferito in quanto è più funzionale), in quanto ci sono stati problemi legati al certificato SSL del server su cui è installato IMC. D'altro canto wget ha funzionato fin da subito, sia per quanto riguarda l'autenticazione che per il certificato SSL.

Esempio richiesta IMC:

```
wget --http-user=$user --http-password=$password --no-check-
certificate -O /tmp/out.xml https://csi-
imc.campusfc.unibo.it:8443/imcrs/res/access/realtimeLocate?
type=1&value=$mac
```

L’unico disguido di `wget` è che scarica l’output in un file e successivamente deve essere letto e gestito opportunamente per estrarre i tag XML dei dati ricercati.

ARPwatch oltre a generare eventi salva uno storico dei dati che intercetta all’interno della cartella `/var/lib/arpwatch/`, ogni file porta il nome della scheda di rete osservata ed estensione `.dat`. Ogni file è così strutturato:

```
Indirizzo MAC       Indirizzo IP        Data            Interfaccia di rete
d0:50:99:11:70:d8   255.255.255.255     1555332200          ens33.120
78:ac:c0:9d:6c:ee   169.254.75.116      1555171270          ens33.120
...
```

Ora, prima che questo progetto nascesse ARPwatch era già in funzione da diversi anni all’interno della rete universitaria, quindi è stato possibile recuperare i dati da questo storico e importarlo nel database (ovviamente non è stato possibile riempire tutte le colonne del database in quanto IMC non conserva dati storici).

Script

Il cuore di tutto il sistema è composto da questi 2 script:

- `ARPwCreator.php` che si occupa di intercettare gli eventi di ARPwatch
- `ARP-IMCwatch.php` si occupa di elaborare le richieste e popolare il database

Ogni qual volta ARPwatch rileva un evento richiama `ARPwCreator.php`, questo script si occupa di intercettare l'evento e di scriverlo in un file all'interno della directory `/tmp/arpw/` dando come nome al file data e orario a cui è stato rilevato.

```
<?php
$cartella = "/tmp/arpw/";
$estensione = ".arpw";

if (!file_exists($cartella)) { //se la cartella non esiste la creo
  mkdir($cartella, 0777, true);
}

$f = fgets(STDIN);
if (strpos($f, 'reaper') == false) { //prendo in considerazione solo le
segnalazioni che non contengono la parola "reaper"
  $tmp = explode (" arpwatch: ", $f); //splitto la segnalazione
  $tmp2 = explode (" ", $tmp[0]);
  $time = $tmp2[sizeof($tmp2)-2]; //ed estraggo l'ora a cui è avvenuta
  $datat = date('Y-m-d', time()) . "_" . $time; //aggiungo la data odierna per
creare un DATETIME

  $nome=$datat; //do il DATETIME come nome al file

  $i = 1;
  while(file_exists($cartella . $nome . $estensione))
  {
    //a volte capita che arrivino più segnalazione nello stesso secondo
    //questo ciclo aggiunge un numero incrementale per evitare di
sovrascrivere i file
    $nome = (string)$datat . "-" . $i;
    $i++;
  }
  //Ora è tutto corretto, posso scrivere il file
  $myfile = file_put_contents($cartella . $nome . $estensione, $f);
}
exit(0);
?>
```

Lo scopo dello script `ARP-IMCwatch.php` è invece quello di visitare la cartella `/tmp/arpw/` ed elaborare man mano in ordine tutti i file presenti.

```php
<?php
/*
Autore: Michele Proscia
Data: 2019/03/29
*/

//questa funzione csi occupa di creare una connessione al DB ed eseguire la query di inserimento
function DBinsert($_servername, $_username, $_password, $_dbname, $_data, $_ip, $_mac, $_ip_switch, $_port_switch, $_vlan, $_name_DNS, $_name_DNS_reverse, $_name_DNS_ad, $_info) {
  //Creo la connessione
  $conn = mysqli_connect($_servername, $_username, $_password, $_dbname);
  //Testo la connessione
  if (!$conn) {
    die("Connessione al DB fallita :-( " . mysqli_connect_error() . "\n");
  } else {
    echo "Connesso con successo al DB :-)\n";
  }
  //questa è la query
  $sql = "INSERT INTO macs (data, ip, mac, ip_switch, port_switch, vlan, name_dns, name_dns_reverse, name_dns_ad, info)
  VALUES ('$_data', '$_ip', '$_mac', '$_ip_switch', '$_port_switch', '$_vlan', '$_name_DNS', '$_name_DNS_reverse', '$_name_DNS_ad', '$_info')";
  //eseguo e testo l'esecuzione della query
  if ($conn->query($sql) === TRUE) {
    echo "Record inserito con sucesso\n";
  } else {
    echo "Errore: " . $sql . " \n" . $conn->error . "\n";
  }
  //chiudo la connessione al DB
  echo "Connessione al DB chiusa\n";
  $conn->close();
}

//Funzione per rimuovere un file dato il suo percorso
function rmFile($dir){
  exec ('rm ' . $dir);
  echo "rm $dir\n";
}

date_default_timezone_set('Europe/Rome');
//credenziali database
$DBservername = "localhost";
$DBusername = "network_dev";
$DBpassword = "network_passwd";
$DBname = "network_dev";
```

```
$directory = "/tmp/arpw/";
$lockfile = "/tmp/ARP-IMCwatch.lock";
$dir = new DirectoryIterator($directory);
$file = array();

$maxlife = 5; // minuti di vita massimi dei file

// credenziali IMC sola lettura
$user = "michele";
$password = "arianna";

if (!file_exists($lockfile)) {
  // creo il file di blocco per evitare aperture anomale di thread
  $lock = fopen($lockfile, "w");
  fclose($lock);

  // scorro tutti i file del percorso indicato
  foreach ($dir as $fileinfo) {
    $flag = 0;
    $cor=0;
    // se il file ha estensione .arpw allora lo importo
    if (pathinfo($fileinfo, PATHINFO_EXTENSION) === "arpw") {
      echo "Trovato: " . $directory . $fileinfo . "\n";

      array_push($file, "$fileinfo");
      $lines = file($directory . $fileinfo);

      // leggo la prima riga
      $line = fopen($directory . $fileinfo, 'r');
      $ARPstring = fgets($line);
      fclose($line);

        // se non contiene la parola "reaper" allora entro
        if (strpos($ARPstring, 'reaper') == false) {
          // splitto la stringa e cerco il tipo di messaggio
          $tmp1 = explode (" arpwatch: ", $ARPstring);
          //splitto il messaggio e controllo che vada tutto bene
          if (sizeof($tmp1) < 2) {
            echo "Parametro ARP non valido #1\n";
            rmFile($directory . $fileinfo);
            break;
          }

          $pieces = explode (" ", $tmp1[1]);

          if (sizeof($pieces) < 2) {
            echo "Parametro ARP non valido #2\n";
            rmFile($directory . $fileinfo);
            break;
          }
```

```
$tmp2 = explode (" ", $tmp1[0]);

if (sizeof($tmp2) < 2) {
  echo "Parametro ARP non valido #3\n";
  rmFile($directory . $fileinfo);
  break;
}

$hour = $tmp2[sizeof($tmp2)-2];

$tmp = explode(".", $pieces[sizeof($pieces)-1]);

if (sizeof($tmp1) < 2) {
  echo "Parametro ARP non valido #4\n";
  rmFile($directory . $fileinfo);
  break;
}

$vlan = (int)$tmp[1];
//in base al tipo di messaggio estraggo IP e MAC
if (strpos($ARPstring, 'arpwatch: new activity') !== false) {
  $ip = $pieces[2];
  $mac = $pieces[3];
} elseif (strpos($ARPstring, 'arpwatch: new station') !== false) {
  $ip = $pieces[2];
  $mac = $pieces[3];
} elseif (strpos($ARPstring, 'arpwatch: flip flop') !== false) {
  $ip = $pieces[2];
  $mac = $pieces[3];
} elseif (strpos($ARPstring, 'arpwatch: changed ethernet address') !== false) {
  $ip = $pieces[3];
  $mac = $pieces[4];
} elseif (strpos($ARPstring, 'arpwatch: ethernet broadcast') !== false) {
  $ip = $pieces[2];
  $mac = $pieces[3];
} elseif (strpos($ARPstring, 'arpwatch: ip broadcast') !== false) {
  $ip = $pieces[2];
  $mac = $pieces[3];
} elseif (strpos($ARPstring, 'arpwatch: bogon') !== false) {
  $ip = $pieces[1];
  $mac = $pieces[2];
} elseif (strpos($ARPstring, 'arpwatch: ethernet broadcast') !== false) {
  $ip = $pieces[2];
  $mac = $pieces[3];
} elseif (strpos($ARPstring, 'arpwatch: ethernet mismatch') !== false) {
  $ip = $pieces[2];
  $mac = $pieces[3];
} elseif (strpos($ARPstring, 'arpwatch: reused old ethernet address') !== false) {
  $ip = $pieces[4];
  $mac = $pieces[5];
} elseif (strpos($ARPstring, 'arpwatch: suppressed DECnet flip flop') !== false) {
  $ip = $pieces[4];
  $mac = $pieces[5];
} else {
  echo "Nessuna corrispondenza trovata\n";
  $cor++;
}
```

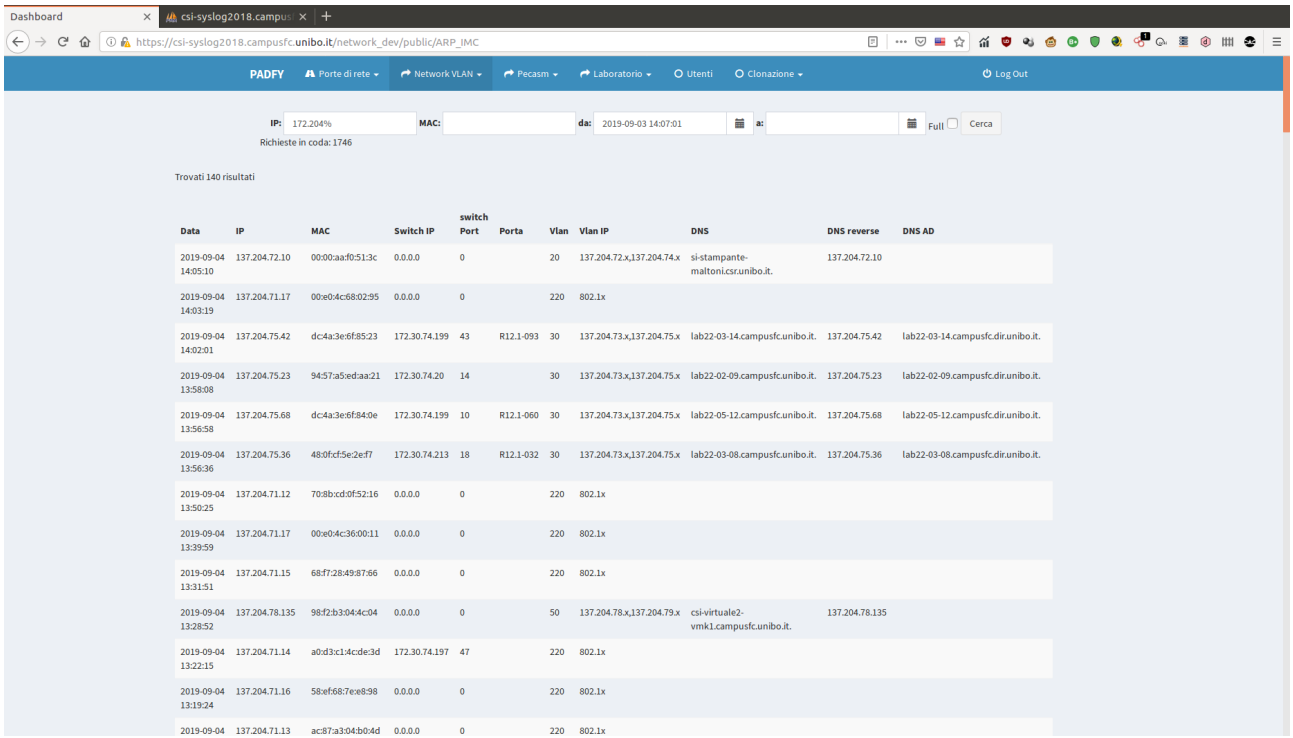
```
  //se ho trovato una corrispondenza allora entro
  if ($cor == 0) {
  $dt = explode("_", $fileinfo);
  $data = $dt[0] . " " . $hour;

  $ip_switch = null;
  $port_switch = null;

  // Riceco i dati di IMC tramite le sua API
  exec ('wget --http-user=' . $user . ' --http-password=' . $password . ' --no-
check-certificate -O /tmp/out.xml "https://csi-
imc.campusfc.unibo.it:8443/imcrs/res/access/realtimeLocate?type=1&value=' . $mac .
'" > /dev/null 2>&1');
  $xml = simplexml_load_file("/tmp/out.xml");
  //estraggo i dati dal XML
  $ip_switch = $xml->realtimeLocation->deviceIp;
  $port_switch = $xml->realtimeLocation->ifIndex;

  $pxml = print_r($xml, true);
  rmFile("/tmp/out.xml");

  // La stringa info contiene tutti i dati grezzi presi dai vari sistemi,
  // possono essere utili in caso di problemi
  $info = $ARPstring . "#" . $pxml;
  //Eseguo le richieste ai server DNS
  $name_DNS = exec("nslookup " . $ip . " 137.204.78.3 | grep name | awk -F'name = '
'{ print $2 } '");
  $name_DNS_reverse = exec("nslookup " . $name_DNS . " 137.204.78.3 | grep
'Address: ' | awk -F': ' '{ print $2 } '");
  $name_DNS_ad = exec("nslookup " . $ip . " 137.204.78.17 | grep name | awk -F'name
= ' '{ print $2 } '");
  //calcolo il tempo di vita del file
  $lifet = date_diff(date_create($data), date_create(date('Y-m-d H:i:s')));
  $lifetime = (int)$lifet->h * 60 + (int)$lifet->i;

  if ($ip_switch != null) { //se la richiesta IMC ha avuto successo salvo tutto
    $flag++;
    try {
      DBinsert($DBservername, $DBusername, $DBpassword, $DBname, $data, $ip, $mac,
$ip_switch, $port_switch, $vlan, $name_DNS, $name_DNS_reverse, $name_DNS_ad,
$info);
      } catch (\Exception $e) {
         $flag = 0;
         echo "Record non aggiunto " . "$e" . "\n";
      }
    }
```

```
            //Se il tempo è oltre al massimo stabilito salvo quello che ho
            if ( $ip_switch == null && ((int)$lifetime > $maxlife || $lifetime == null
            )) {
               $flag++;

               try {
                 DBinsert($DBservername, $DBusername, $DBpassword, $DBname, $data,
                 $ip, $mac, "0.0.0.0", "0", $vlan, $name_DNS, $name_DNS_reverse,
                 $name_DNS_ad, $info);
               } catch (\Exception $e) {
                 $flag = 0;
                 echo "Record non aggiunto " . "$e" . "\n";
               }
             }
           } else {
             $flag++;
           }

         }
       if ((int)$flag > 0) {
         rmFile($directory . $fileinfo);
       }
     }
   }
   rmFile($lockfile); //non ci sono più file in coda, elimino il lock e termino
}

exit(0);

?>
```

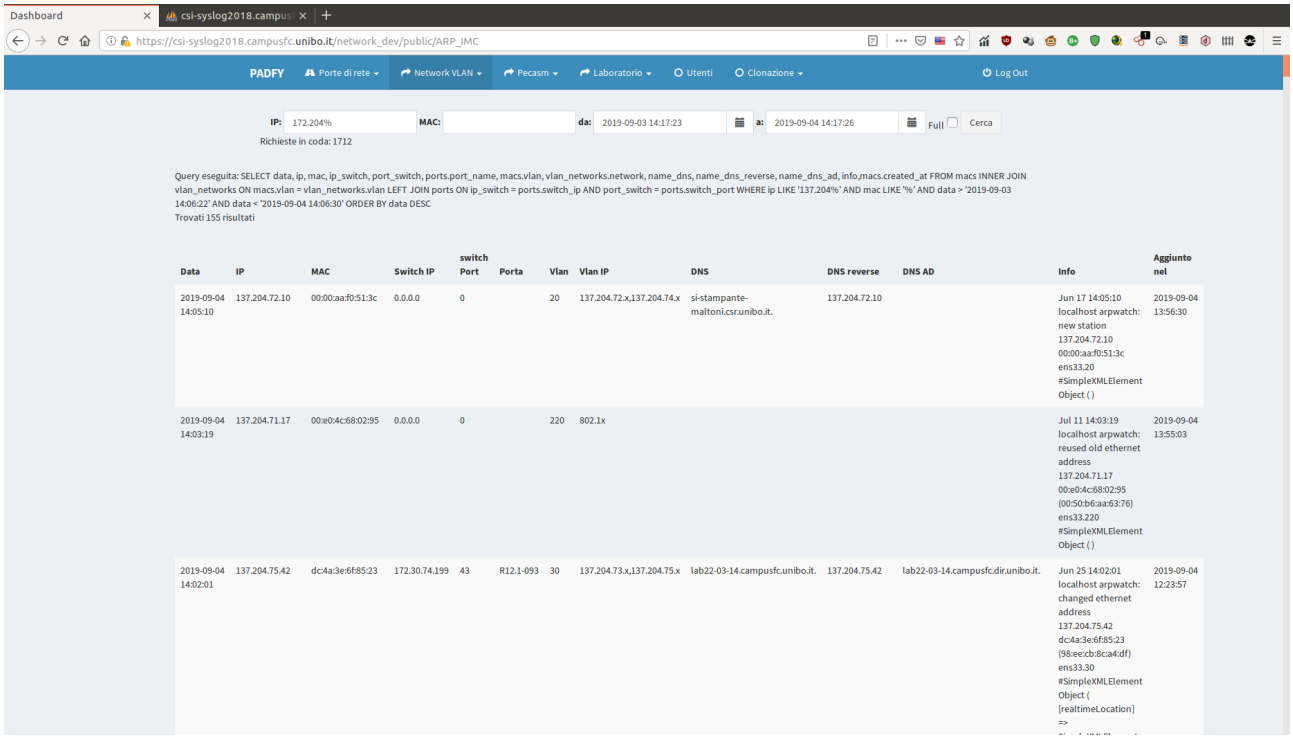
Il lavoro combinato dei due script fa si che si crei una una coda delle richieste di ARPwatch (strutturata in tanti singoli file all'interno della cartella `/tmp/arpw/`) per dare tutto il tempo necessario allo script `ARP-IMCwatch` di eseguire tutte le richieste ad IMC ed ai server DNS (esegue anche più tentativi nel caso di un iniziale fallimento), così si evita di congestionare il server.

Portale di interrogazione

Per poter accedere ai dati del database è stato creato un portale web in Laravel, presenta una form disposta in un unica riga in alto in cui è possibile specificare dei filtri per: indirizzo IP, indirizzo MAC e data ora di inizio e fine. Subito sotto è possibile vedere il numero degli eventi in coda che devono essere ancora processati dallo script `ARP-IMCwatch` (ossia il numero di file presenti in `/tmp/arpw/`) in questo modo possiamo tenere monitorato l'andamento degli script. Una volta lanciata una ricerca appare il numero totale di voci trovate subito sopra la tabella dei risultati. Nei capi indirizzo IP e MAC è possibile sfruttare i parametri disponibili nel linguaggio SQL (per esempio **%** o *****) per effettuare ricerche ancor più particolari e precise.



È presente in oltre una check box che se attivata mostra alcuni dati aggiuntivi: la colonna info coi dati grezzi presi da ARPwatch e IMC, la colonna che riporta la data e l'ora dell'inserimento in tabella (la colonna Data riporta la data della rilevazione dell'evento) e la query SQL eseguita in base ai filtri impostati.



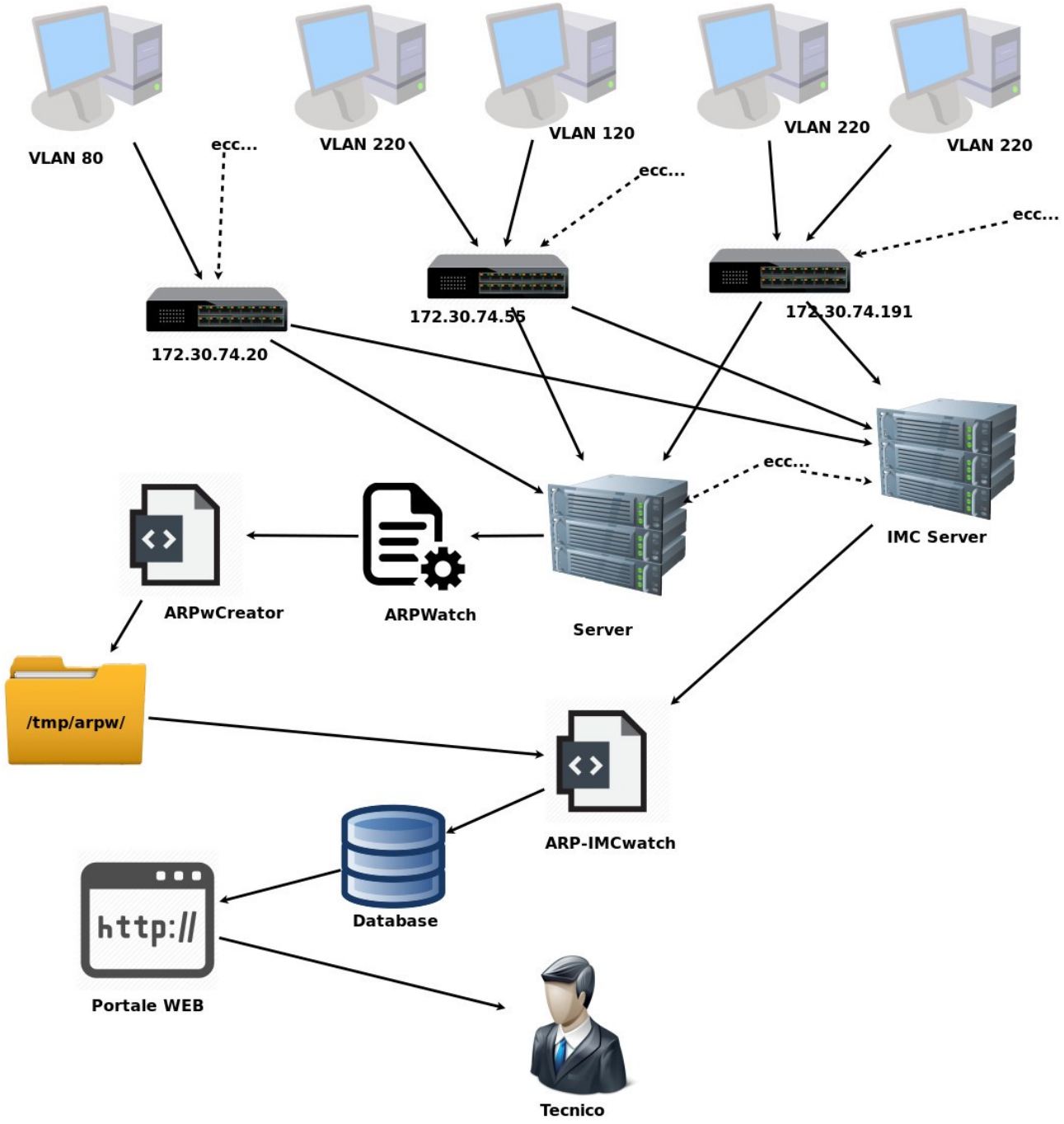
Capitolo 6: Deployment

Messa in funzione

Il sistema è stato installato su di una macchina virtuale Ubuntu server, in modo da poter lavorare in tranquillità (anche da casa attraverso VPN) e disporre anche dei permessi di Amministratore; la macchina è già fornita di alcuni strumenti necessari quali:

- PHP e alcune librerie
- wget
- ARPwatch adeguatamente configurato per interfacciarsi su tutte le VLAN da tenere sotto osservazione.
- Laravel

ARPwatch necessita di essere lanciato dall'utente Root per catturare i pacchetti delle VLAN, ma gli altri script non necessitano di permessi così alti, per tanto è possibile creare un utente limitato a cui affidare l'esecuzione del sistema.



Il server (chiamiamolo principale) è la macchina fisica da noi utilizzata e comunica, attraverso i vari switch, con tutte le VLAN della rete; su di esso sono installati ARPwatch, Laravel e i due script di interrogazione, IMC invece si trova in un server esterno ma sempre raggiungibile attraverso la rete.

Il file `/etc/rsyslog.conf` del server principale è stato modificato in modo tale da lanciare lo script `ARPwCreator.php` ad ogni evento di ARPwatch

```
#  /etc/rsyslog.conf     Configuration file for rsyslog.
#
#         For more information see
#         /usr/share/doc/rsyslog-doc/html/rsyslog_conf.html
[…]

if $rawmsg contains 'arpwatch'          then
action(type="omprog" binary="/usr/bin/php -f
/var/www/html/network_dev/script/ARPwCreator.php"
template="RSYSLOG_TraditionalFileFormat")

[...]
```

che a sua volta riporta gli eventi suddivisi in file all'interno della cartella `/tmp/arpw`. Ongi 5 minuti `cron` lancia lo script `ARP-IMCwatch.php` che prende i dati di ARPwatch presenti in `/tmp/arpw/`, esegue le interrogazioni al server IMC ed aggiunge i record sul database. In caso di necessità un tecnico, in possesso delle credenziali, può accedere al contenuto del database tramite il portale web e sempre attraverso quest'ultimo filtrare i dati in base alle sue necessità.

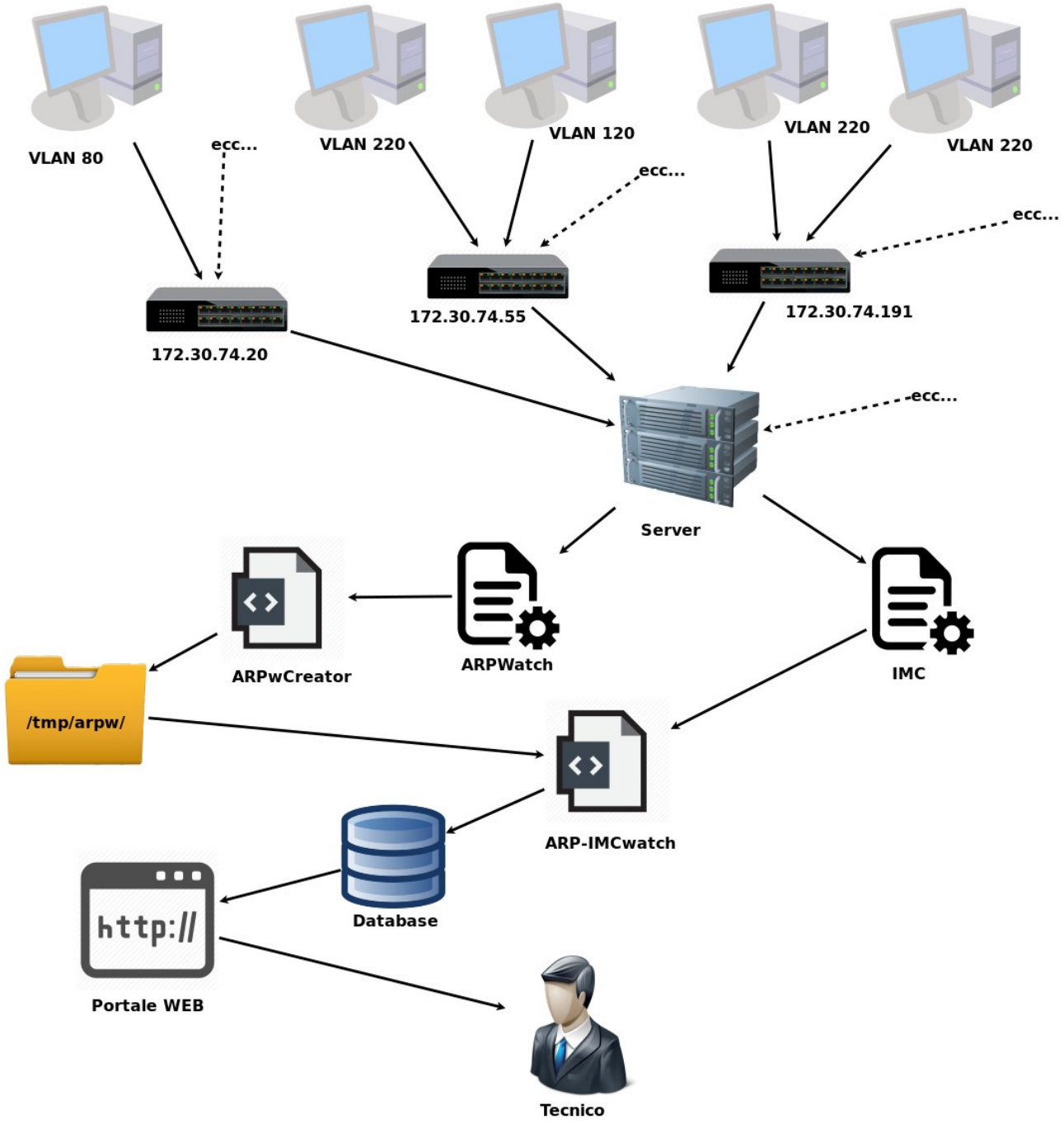
Manutenzione

Durante la prima settimana di test del sistema ci siamo accorti che la macchina virtuale andava in crash dopo qualche giorno di monitoraggio; abbiamo scoperto che `crontab` apriva istanze multiple dello script `ARP-IMCwatch.php` e questo non era previsto. Se lo script non termina la sua esecuzione entro i 5 min `crontab` apriva un'altra istanza concorrente, accedendo agli stessi file i processi si bloccano e continuando così sino a che la macchina, stremata, doveva essere resettata manualmente. Per ovviare a questo problema è stato creato un file `lock` che impedisce l'apertura di altri thread, se il file esiste lo script si chiude, così anche se `crontab` cerca di aprirne più di uno `cronjob` questi si chiuderanno immediatamente.

Lo script in questo caso ha ancora margine di miglioramento, non che ce ne sia un urgente bisogno, non ci sono mole di nuovi dispositivi così ingenti in poco tempo, ma è possibile rivederne il funzionamento per renderlo multi-thread in futuro.

Capitolo 7: Conclusioni

Questo progetto si è posto l'obbiettivo di creare un sistema in grado di integrare i servizi ARPwatch e IMC già presenti in università e renderli facilmente fruibili al personale tecnico.



Prima della messa in funzione sono state fatte alcune prove per verificare il corretto comportamento dei comandi eseguiti dal codice PHP e del corretto accesso al database; una volta appurate queste cose è stato avviata una versione semplificata del sistema. Per testarne la reattività, le prestazioni e il comportamento in relazione a svariati problemi (per esempio: server DNS non risponde o server IMC non produce risultati) sono stati simulati dei malfunzionamenti mirati atti allo scopo. Risolti alcuni problemi di crash improvvisi che interrompevano l'esecuzione dello script `ARP-IMCwatch` che di conseguenza si bloccava e causava un accumulo anomalo di file nella coda. Il sistema è in esecuzione da Luglio 2019 e sta tuttora monitorando la rete.

In futuro si potrebbero inglobare le reti del polo di Forlì (arrivando ad un totale di 320 switch monitorati) così da tenere sotto controllo altri dispositivi e migliorare il lavoro del personale tecnico.

Il sistema creato sarà un ottimo strumento per far fronte a svariati problemi conosciuti e imprevisti nella gestione della rete universitaria, la possibilità di ricostruire gli eventi passati/sfuggiti grazie allo storico presente sul database sarà sicuramente d'aiuto ai tecnici nella risoluzione tempestiva dei malfunzionamenti, riducendo drasticamente i tempi di disservizio. Tutti gli script utilizzati sono ben commentati per rendere semplice una loro futura modifica. In oltre è possibile rendere il sistema ancor più efficiente abbracciando il pensiero multi-thread rendendo gli script ancora più reattivi nel caso si vengano a creare situazioni di congestione, per esempio nell'aggiunta di altre VLAN che a loro volta portano ulteriori dispositivi da monitorare.

Ringraziamenti:

Ringrazio la mia famiglia che mi ha sempre sostenuto durante questo periodo della mia vita e senza di loro non ce l’avrei mai fatta, grazie.

Un grande grazie ad Arianna, che mi ha supportato e sopportato con amore nei momenti di crisi, ti amo!

Un grazie anche a tutti gli amici che ho incontrato lungo il percorso, agli aiuti dati e ricevuti, con cui ho instaurato una solida e duratura amicizia.