UNIVERSITA' DEGLI STUDI DI
NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base
Corso di Laurea in Ingegneria Informatica

Elaborato d'esame

Reti di Calcolatori I

Raccolta e analisi di tracce di traffico di applicazioni mobili

Anno Accademico 2018/2019

Professore

Prof. Antonio Pescapè

Gruppo – **Gruppo28**

Pasquale Angelino
Matr. N46 003194

Giuseppe Capasso
Matr. N46 003195

Indice

Sommario

Nel presente elaborato verrà trattato un tema sempre più importante nella società contemporanea: la classificazione e analisi del traffico sulla rete Internet. In particolar modo verrà esposto in cosa consiste, quali sono gli oggetti della classificazione e quali sono i principali approcci alla classificazione del traffico come gli approcci port based, payload based e flow-feature-based. Sarà descritto nel dettaglio un progetto di cattura e analisi di traffico generato da applicazioni mobile quali Waze, Viber e One Football, descrivendo le fasi di cattura, avvenuta nel laboratorio di reti ARCLAB della Federico II, di classificazione attraverso TIE, di validazione attraverso uno scprit realizzato in python combinato con bash e infine di analisi attraverso considerazioni e osservazioni fatte in excel con il file di output.

Capitolo 1: Classificazione del traffico

La Traffic Classification (TC) è una pratica che ha come obiettivo quello di catturare e analizzare il traffico dati che attraversa una determinata rete, grazie ad essa è perciò possibile studiare e capire le dinamiche e la composizione dei flussi di informazioni scambiati da dispositivi su una rete per raggiungere obiettivi che riguardano principalmente:

- La gestione di una rete
- L'individuazione di traffico malevolo
- Ottenere informazioni utili per advertising, compagnie di assicurazioni e agenzie di sicurezza.

Essa consiste nell'associare flussi, pacchetti, biflussi o altri oggetti di classificazione alle applicazioni che li hanno generati.
Negli ultimi anni l'ascesa di smartphone ed altri dispositivi portatili ha cambiato notevolmente il traffico presente sulla rete con la nascita di numerosi servizi e contenuti presenti sulla rete stessa. Il traffico mobile risulta essere ricco di informazioni che possono essere utili per fini economici e di sicurezza, tuttavia questo solleva problematiche legate alla privacy e alla sicurezza.

1.1 Livelli di dettaglio della classificazione

La classificazione del traffico può essere eseguita su diversi livelli di dettaglio.

Una classificazione più generale è quella in base alla classe di traffico a cui un flusso di dati appartiene. Alcune classi di traffico sono:

- **Bulk:** trasferimenti ftp o mail via SMTP;
- **Interactive:** generato da applicazioni telnet, SSH;
- **Mail:** IMAP, SMTP, POP3.

Una classificazione più nel dettaglio consiste nel classificare il traffico a seconda della categoria a cui appartiene l'applicazione che l'ha generato un esempio di tali categorie possono essere:

- **Chat**
- **Streaming**
- **Web**
- **File sharing**

Procedendo più nel dettaglio possiamo scegliere di classificare il traffico in base all'applicazione o al servizio dell'applicazione che lo genera.
Quando cerchiamo di classificare il traffico di una singola applicazione si parla di identificazione.

1.2 Oggetti della classificazione

Gli oggetti di una classificazione possono essere:

- **Singolo pacchetto:** rappresenta il livello più basso di granularità nella classificazione;
- **Connessioni TCP:** insieme di pacchetti;
- **Flussi:** insieme di pacchetti relativi ad uno stesso servizio, una quintupla identifica il flusso ed è costituita da proto (protocollo TCP o UDP), IP_src (sorgente), IP_dst (destinatario), port_src, port_dest;
- **Flussi bidirezionali** (biflussi);
- **Host;**

- **Altri oggetti:** Burst, Service Burst.

1.3 Approcci alla classificazione

La classificazione può essere effettuata attraverso diversi approcci. Nel corso del tempo alcuni di essi sono diventati obsoleti (a causa dell'evoluzione dei servizi e dei protocolli) ed altri sono stati introdotti per avere un'accuratezza maggiore nella classificazione che i precedenti approcci non hanno. In particolar modo gli approcci più semplici, maggiormente utilizzati in passato sono:

- **Port based**
- **Payload based**

L'approccio **port based** consiste nel classificare il traffico attraverso un'associazione tra porto e applicazione, in particolare grazie all'assegnazione dei numeri di porto effettuata dallo IANA e alla conoscenza dei porti utilizzati comunemente dalle applicazioni è possibile risalire all'applicazione che ha generato il flusso conoscendo i numeri di porto del flusso stesso. Tale approccio risulta obsoleto in quanto oggi molte applicazioni non usano porti standard ma porti casuali (dinamici), altre invece mascherano il proprio traffico attraverso porti standard di altre applicazioni.

L'approccio **payload based**, invece, consiste nell'ispezionare il payload (contenuto informativo) di livello di trasporto per identificare determinate stringhe relative all'applicazione che lo ha generato effettuando un matching con un insieme di stringhe note. L'approccio payload based risulta obsoleto a causa dell'aumentare del traffico cifrato che rende inaffidabile la classificazione.

Attualmente si fa riferimento ad approcci più intelligenti detti f**low-feature-based**.
Questi approcci sono tipicamente basati su tecniche di Machine Learning e fanno riferimento a delle feature estratte dai flussi. Tali feature sono un insieme di caratteristiche distintive del traffico che permettono di addestrare dei classificatori Machine Learning per discriminare il traffico. In particolare, in questi approcci vi è una prima fase (**addestramento**) in cui il classificatore impara, attraverso l'analisi di molto traffico di cui conosce l'origine attraverso una ground truth, un insieme di informazioni prese da analisi effettuate su traffico noto. Una seconda fase consiste nel sottoporre il classificatore a traffico ignoto e in base alle informazioni contenute nella ground truth studiare il traffico classificato.
Le feature vengono estratte dall'intestazione dei pacchetti e ed elaborate per ottenere un insieme di statistiche riguardanti, ad esempio, la dimensione dei pacchetti e il tempo di inter arrivo.

Gli approcci basati sul ML si dividono in

- **Supervised Learning**
- **Unsupervised Learning**
- **Deep Learning**

Nei primi il classificatore impara attraverso traffico noto, nei secondi impara su traffico non conosciuto a monte e quindi sconosciuto. Attraverso questi approcci è possibile raggruppare flussi di dati appartenenti ad applicazioni con caratteristiche comuni.
Gli approcci Deep Learning sono approcci basati sul ML tuttavia permettono di superare alcune limitazioni degli algoritmi standard di ML che già stanno sopraggiungendo, permette di effettuare velocemente la classificazione online attraverso l'utilizzo di GPU. Gli algoritmi deep learning hanno un tempo di addestramento molto elevato che può essere fatto offline.

Gli approcci per la classificazione del traffico possono essere combinati, si può: ad esempio utilizzare più classificatori che fanno rifermento ad approcci diversi oppure allo stesso approccio con delle variazioni, in tal caso si parla, appunto, di Multi-classificazione.

La classificazione può essere effettuata online, cioè in tempo reale questo richiede però un classificatore leggero e veloce che fornisca i risultati nell'ordine dei millisecondi. È basata sull'hardware e deve necessariamente fare riferimento ad un insieme di dati limitato da processare.

Una ground truth si realizza attraverso ispezioni dei payload, euristiche, ispezioni manuali e collaborazione tra utenti.

1.4 Classificazione del traffico mobile

Nel mondo del traffico Mobile la classificazione del traffico risulta complicata a causa dell'uso crescente di protocolli cifrati per la quale approcci più semplici, quali payload-based, risultano essere obsoleti e inefficaci. Bisogna perciò fare riferimento a classificatori basati su approcci di flow-feature-based. Inoltre, il traffico mobile risulta essere particolarmente eterogeneo e dinamico e perciò più difficile da classificare. L'aumento continuo delle app complica ulteriormente il lavoro del classificatore. Gli algoritmi di Deep Learning sono l'ultima tappa della ricerca sulla classificazione del traffico, in particolare quello mobile. Questi algoritmi permettono di superare le limitazioni degli algoritmi di Machine Learning standard per i classificatori quali tempistiche onerose, mancanza di automazione, obsolescenza dovuta al cambiamento continuo del traffico mobile. Gli approcci che fanno uso di tali algoritmi prevedono un ispezione e sintesi del contenuto informativo dei pacchetti ricavando in maniera efficiente le correlazioni tra pacchetti e flussi presenti nei dati, minimizzando lo sforzo di processing dei dati. Il calcolo parallelo (gpu-based) è in grado di ottenere risultati soddisfacenti nell'ordine dei millisecondi, permettendo quindi l'analisi del traffico in tempo reale e permettendo la gestione di reti con interventi tempestivi. Questi algoritmi risultano essere il futuro nel campo della classificazione del traffico mobile.

Nei prossimi capitoli tratteremo un progetto di raccolta e analisi di traffico di applicazioni mobili in particolare descriveremo le fasi di cattura, classificazione e validazione della classificazione prodotta.

Capitolo 2: Strumenti utilizzati

In questo capitolo, saranno descritti gli strumenti utilizzati in ciascuna fase del progetto, quali:

- cattura del traffico
- classificazione del traffico
- analisi

2.1 Strumentazione hardware

Per generare il traffico da analizzare sono stati utilizzati due dispositivi Xiaomi:

- Mi5 (smartphone)
- Nexus 7 (tablet)

I due dispositivi sono stati connessi ad Internet attraverso una connessione wireless e successivamente collegati ad un calcolatore dotato di sistema Unix mediante connessione dati (USB).
I due host sono stati precedentemente root-ati per consentire al sistema di monitorare un processo del sistema operativo (Android) al fine di ottenere tutte le system call effettuate per interfacciarsi con la rete.

Le applicazioni in esame sono state eseguite in maniera alternata su ciascun dispositivo.
La classificazione del traffico è stata eseguita sulle tracce generate Mi5, mentre quelle del tablet sono state tenute in caso di fallimento.

Per la progettazione e lo sviluppo di software atti ad automatizzare il processo di classificazione è stato utilizzato un notebook Asus con le seguenti specifiche:

- CPU i5-6198DU, 2.8 GHz
- SSD Samsung EVO 860 500 GB
- RAM 8GB
- OS Fedora 29 x64 (in esecuzione attraverso virtual box), host Windows 10 Home

2.2 Strumentazione software

Nella configurazione appena descritta, i due dispositivi sono stati utilizzati per generare traffico mobile attraverso delle applicazioni prescelte. In questo paragrafo saranno descritti gli strumenti utilizzati per catturare il traffico generato, classificarlo e per effettuare analisi.

2.2.1 Strumentazione software per la cattura del traffico

Su ciascun'end-system è stata avviata una sessione di tcpdump al fine di catturare tutti i pacchetti in transito sull'interfaccia di rete portandola in modalità promiscua.
Tcpdump è un tool sviluppato da Van Jacobson e Craig Leres e Steven McCanne nell'università di Berkley disponibili sulla maggior parte dei sistemi Unix-like.

2.2.2 Strumentazione software per la classificazione del traffico

La classificazione del traffico è stata fatta in base al concetto di biflusso che rappresenta un accettabile grado di granularità in confronto a quello del singolo pacchetto. Per classificare un biflusso è stato utilizzato **TIE**.
È un software sviluppato da Alberto Dainotti, Walter de Donato, Alessio Botta, Antonio Pescapè dell'università degli studi di Napoli Federico II che consente di effettuare una deep packet inspection attraverso il plugin **ndping_1.0,** non provvisto di interfaccia grafica.

Attraverso TIE, dato in ingresso una traccia di traffico, è stata ottenuta una tabella. Ciascuna riga di questa tabella contiene una classificazione inaffidabile della traccia di traffico. In particolare, si hanno i campi appartenenti alla quintupla utile ad identificare un biflusso con alcune informazioni ad esso collegate, quali valori di timestamp, pacchetti ricevuti ed inviati, tipo di applicazione e affidabilità.

Un altro passaggio per la classificazione del traffico è stato fatto attraverso l'uso di **tshark.**
Si tratta di un software compatibile con la maggior parte degli ambienti Unix-like non provvisto di interfaccia grafica che consente sia di catturare il traffico su un'interfaccia di rete, ma anche di analizzare una traccia di traffico attraverso una vasta gamma di filtri. È stato ideato e sviluppato da Gerald Combs. In questo progetto è stato utilizzato per applicare dei filtri al file di cattura per ottenere informazioni particolari quali quelle di risoluzioni DNS, SNI.
Questo software è disponibile anche con interfaccia grafica sotto il nome di **wireshark**.

Per automatizzare la classificazione del traffico è stato utilizzato **python3.**
È un linguaggio di programmazione interpretato basato sulla Oriented Object Programming largamente diffuso per effettuare data analysis su una grande quantità di informazioni. Si è scelto di utilizzare questo linguaggio per 2 motivi:

- **Calcolo parallelo**: al fine di sfruttare al massimo la potenza di calcolo dei dispositivi odierni si è deciso di utilizzare la classe Pool della libreria Multiprocessing per far eseguire più task completamente diversi a processi differenti. Python mette a disposizione metodi ottimizzati per eseguire le operazioni differenti in modo asincrono.
- **Flessibilità:** python consente di eseguire programmi esterni con una discreta facilità attraverso la libreria OS. Inoltre, è stata valutata anche la facilità per le operazioni di lettura e scrittura su vari file.

Combinato con python è stato utilizzato il linguaggio **bash**. È stata effettuata questa scelta per interfacciarsi in maniera più diretta con il sistema operativo utilizzato e per coordinare la gestione di files e directories e rendere gli script (presentati nel prossimo capitolo) parametrizzabili.

2.2.3 Strumentazione software per l'analisi del traffico

Per l'analisi della classificazione è stato utilizzato Microsoft Excel. È un software che consente di creare un foglio di calcolo sul quale eseguire delle analisi statistiche. Sarà utilizzato manualmente dopo aver effettuato la classificazione del traffico.

Capitolo 3: Cattura e analisi del traffico mobile

3.1 Cattura del traffico

La cattura del traffico è stata effettuata presso l'ARCLAB situato nel complesso della Federico II in via Claudio. La cattura è stata articolata in due sessioni diverse in ciascuna delle quali è stata prevista la a
Le applicazioni in esame sono:

- Waze (Sessione 1)
- One football (Sessione 1)
- Viber (Sessione 2)
- Slither.io (Sessione 2)

Per ogni applicazione sono state previste 6 catture, della durata di 5 minuti ciascuna, durante le quali è stato previsto un uso intensivo attraverso la sincronizzazione con servizi esterni (Google, Facebook) oltre a quello ordinario attivati con account creati ad hoc.

Prima di effettuare la cattura, sono stati creati due account con ciascuno dei quali è stato configurato un dispositivo. Una volta scaricate le applicazioni dal Play Store, si procede con il collegare al sistema Unix i due dispositivi.
L'uso delle applicazioni è stato eseguito mediante il seguente approccio: alle prime catture sono state dedicate le operazioni di sincronizzazione, mentre alle restanti è stato riservato un uso ordinario cercando (ove possibile) di usare funzioni di condivisione.

Dopo le operazioni di cattura, sono stati ricevuti 12 cartelle (una per ogni cattura del Mi5) contenenti:

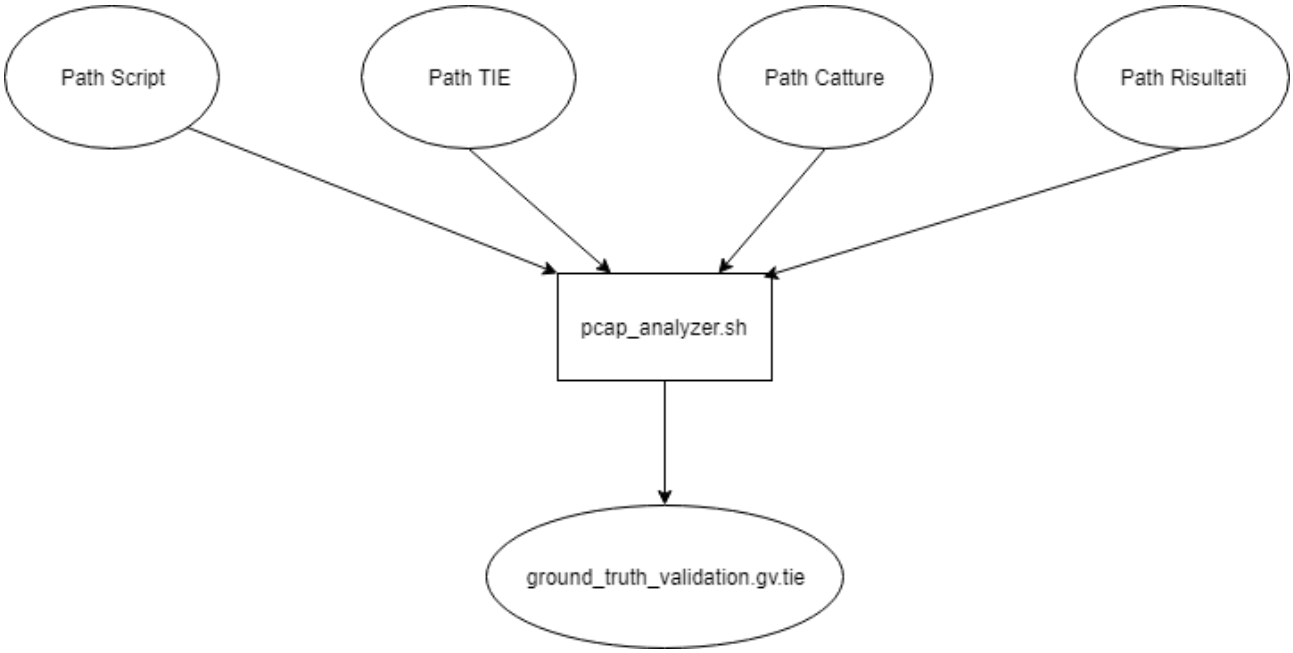
- **traffic. pcap**: file generato da tcpdump presente sul dispositivo contenente il traffico catturato
- **strace.log**: file generato dal sistema di cattura contenente le system call effettuate sul dispositivo durante la cattura.
- **packages.log**: file contenente una configurazione dei permessi per Android
- **logcat.log**: file contenente la storia dei servizi del sistema Android
- **events.log**: file di configurazione per le periferiche del dispositivo
- **pids.log**: file contenente un'associazione tra i PID dei processi e i nomi dei package di chi li gestisce

Ai fini della classificazione del traffico saranno utilizzati solamente traffic. pcap e strace.log.
Le cartelle ricevute contengono i files delle catture di Waze, One football e Viber.

L'approccio utilizzato per classificare il traffico è quello di rendere affidabile attraverso il confronto con il file strace.log la classificazione non affidabile eseguita da TIE.
Per ogni biflusso saranno aggiunti i campi package, risoluzione DNS, SNI e host HTTP.

3.2 Script per l'automazione delle analisi

Per automatizzare il processo di classificazione del traffico, è stato utilizzato un approccio gerarchico: uno script principale in bash (pcap_analyzer.sh) controlla l'esecuzione di 2 script in python (strace_filter.py e match_comunication.py), oltre ai quali lancia alcuni comandi per la gestione dei file e per l'esecuzione di tshark. In figura è mostrato uno schema a blocchi del funzionamento del sistema di script.



1 Funzionamento generale script

Nei sotto-paragrafi seguenti sarà mostrato in dettaglio il funzionamento di tutti gli script, mentre il loro codice è fornito in file esterni.

Nella cartella risultati saranno presenti 12 (numero di tracce ricevute) cartelle contenenti:

- **Strace.log**: una copia del file strace.log
- **Class.tie**: risultato dell'applicazione TIE per la DPI
- **Ground_truth_validation.gv.tie**: risultato dello script

In ciascuna cartella sarà aggiunto una variante di Ground_truth_validation.gv.tie prodotta da fogli Excel per analisi.

Nei risultati allegati saranno presenti solo i due file ground_truth_validation entrambi manipolabili come file CSV.

Nei sotto-paragrafi seguenti sarà mostrato in dettaglio il funzionamento di tutti gli script, mentre il loro codice è fornito in file esterni omonomi.

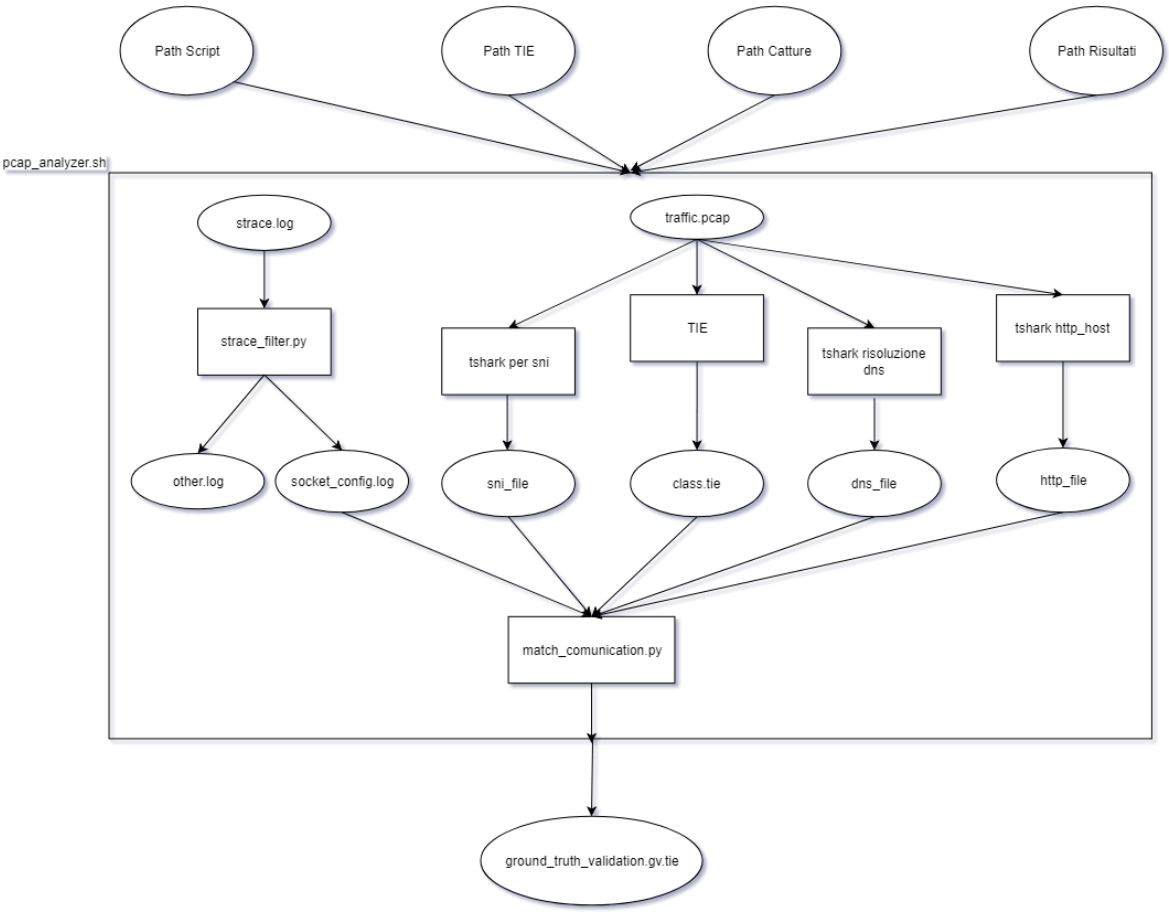
3.2.1 Pcap_analyzer.sh

Gestisce gli altri script in python, i file e le cartelle, l'esecuzione di TIE e automatizza l'esecuzione del processo per tutte le catture passate nel terzo parametro. Come si vede dalla figura in alto, l'utente lancerà lo script passando in ingresso 4 parametri:

- **path scripts:** percorso in cui si trovano gli script in python da lanciare

- **path TIE:** percorso in cui si trova l'eseguibile di TIE
- **path catture:** percorso in cui si trovano le cartelle contenenti le tracce di traffico
- **path risultati:** percorso in cui viene salvato il risultato. Sarà creata una cartella chiamata 'risultati'. Questa cartella conterrà il file ground_truth_validation.gv.tie.

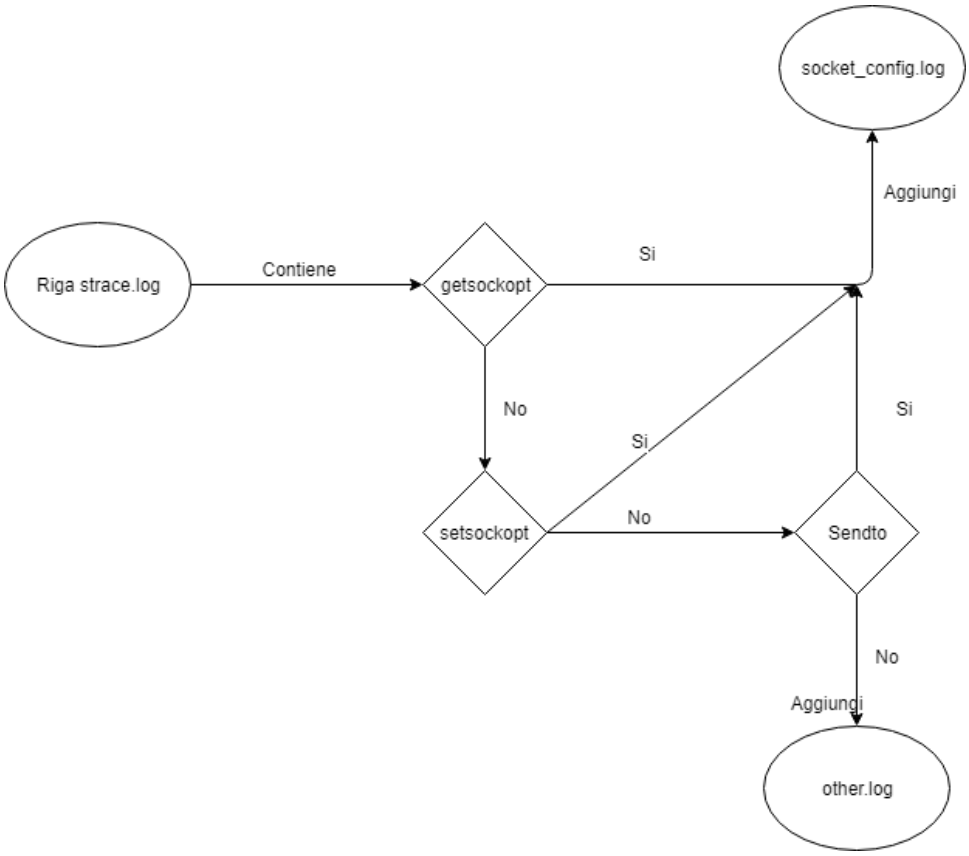
Lo script è articolato su un unico ciclo for mediante il quale saranno eseguite le medesime operazione per ogni elemento nella cartella delle catture. Nel costrutto iterativo, viene lanciato TIE con in ingresso il file traffic.pcap, un time-out di 300 secondi (durata di una cattura) per evitare di classificare più volte lo stesso biflusso e il plugin ndping1.0; la cartella viene salvata dove indicato nel quarto parametro e rinominato in accordo con la variabile 'i'. Successivamente, dal file traffic.pcap sono generati i file http_file, sni_file e dns_resolution attraverso tshark applicando dei filtri. Infine, prima di classificare il traffico viene applicato un filtro al file strace con lo script strace_filter.py. Dunque, viene lanciato lo script match_comunication.py che esegue la validazione.
Una volta terminata l'esecuzione vengono eliminati i file di appoggio e incrementata la variabile per la rinomina delle cartelle in cui salvare i risultati. Agli script python è passato in ingresso il path della cartella per cui si sta validando il traffico per facilitare le operazioni di lettura e scrittura di queste ed aumentare il grado di automazione del programma. Per snellire gli script successivi si è deciso di copiare nella cartella risultati il file strace.log, in modo tale da uniformare il parametro da passare in ingresso.



2 Funzionamento pcap_analyzer.sh

3.2.2 Strace_filter.py

Prende in ingresso il path della cartella out i-esima contenuta nella cartella dei risultati attraverso la libreria sys, prendendo l'attributo arg[1].
È uno script di preparazione per il file strace.log. Conoscendo l'interfaccia delle primitive per le socket, si è deciso, al fine di abbassare i tempi di esecuzione, di applicare un filtro al suddetto file per ridurne le dimensioni. In particolare, saranno eliminate tutte le righe che non contengono system call quali setsockopt, getsockopt e sendto. Studiando gli strace.log si è notato infatti che le socket sono configurate con delle apposite chiamate di configurazione, mentre gli elementi della comunicazione non sono mai inseriti direttamente nelle chiamate classiche quali connect, send, recv.
Di seguito è mostrato uno schema logico del suo funzionamento:



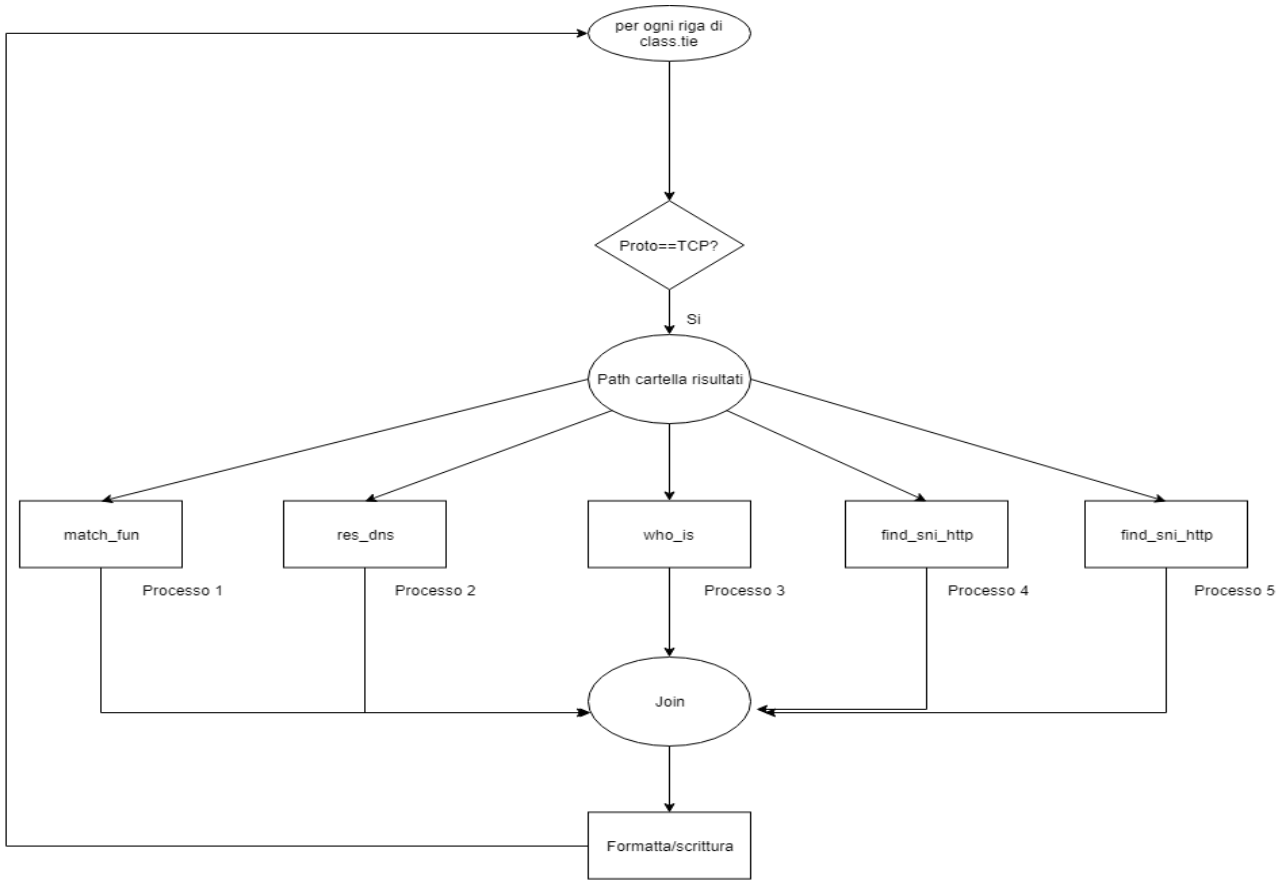
3 Funzionamento strace_filter.py

3.3.3 Match_comunication.py

Prende in ingresso il path della cartella out i-esima contenuta nella cartella dei risultati attraverso la libreria sys, prendendo l'attributo arg[1]. Questo script utilizza, come già accennato nel capitolo 2, utilizza la classe Pool della libreria Multiprocessing, per effettuare calcolo parallelo. Per evitare di analizzare traffico non direttamente generato dalle applicazioni come chiamate DNS, richieste ARP, ecc… si è deciso di analizzare solamente i biflussi basati su connessioni TCP (di qualsiasi versione). Nella prima parte del programma sono state definite 5 funzioni:

- **match_fun(path, ip_src, ip_dest, src_prt, dest_prt ,proto)**: ricava il package dal file socket_config.log
- **who_is(path, ip_dest**): ricava delle informazioni relative all'end-system raggiunto attraverso l'IP di destinazione
- **res_dns(lines, ip_dest)**: trova la richiesta DNS effettuata dal file dns_resolution
- **rev_dns(path, ip_dest):** esegue una richiesta inversa al DNS di Google (8.8.8.8), in caso who_is fallisca.
- **find_sni_http(path, ip_src, ip_dest, prt_src, prt_dst):** restituisce il campo SNI o quello http host a seconda del path passato in ingresso. È stata utilizzata la stessa funzione per aumentare l'efficienza del programma grazie al principio di località e perché le operazioni effettuate sono uguali.

Nella funzione principale, sono eseguite le funzioni sopra definite per ogni riga del file dato in uscita da TIE (class.tie) da cui sono estratti i campi IP sorgente, IP destinazione, porto sorgente porto destinazione e protocollo. In particolare, sono inizializzati 5 processi, ciascuno dei quali eseguirà una funzione per poi ricongiungersi agli altri con il metodo **join().** Ogni processo eseguirà una funzione grazie al metodo **apply_async()** con il quale viene specificato l'entry point, mentre il risultato viene assegnato ad una variabile dopo essere stato prelevato con il metodo **ris_p.get().**
Dopo che sono terminati i processi, i risultati sono stati formattati ed inseriti nel file di uscita **ground_truth_validation.gv.tie**. Di seguito sono riportati uno schema logico e codice dello script:



4 Funzionamento match_comunication.py

Capitolo 4: Risultati sperimentali

I file ottenuti con le procedure precedentemente descritte (ground_truth_validation.gv.tie) sono stati inseriti in fogli Excel per effettuare analisi statistiche sui flussi di traffico.

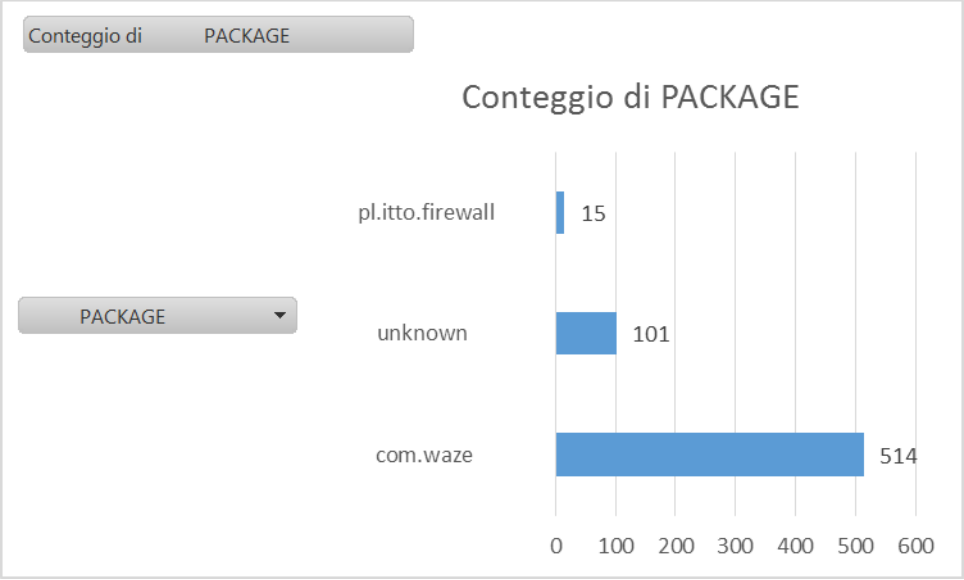
4.1 Analisi del traffico

I singoli file Excel per cattura contengono flussi classificati per intero (valori per tutti i campi e per intero) ed altri che presentano "unknown" o nessun valore nei vari campi. Sono stati evidenziati in verde i primi e in giallo i secondi.
Procederemo con l'analizzare i dati ottenuti nei file di output per le singole applicazioni.

4.2 Analisi del traffico di Waze

Abbiamo ottenuto un totale di 630 flussi di traffico nel catturare traffico dell'applicazione Waze. Osservando i dati relativi a tali flussi è possibile capire che i package utilizzati dall'applicazione risultano essere:

- **com.waze**: il package dell'applicazione stessa;
- **pl.itto.firewall**: il package dell'applicazione Ultra Firewall;
- **unknown**: sconosciuto.



Conteggio di package

Tra i flussi ottenuti troviamo alcuni legati a server di aziende, come Facebook, per la fornitura di third-party services. All'avvio dell'applicazione infatti, nelle varie catture effettuate (out2-out4), vi vi sono flussi con un server Facebook legati al servizio degli Open Graph.

APP_TYPE (TIE)	PACKAGE	WHOIS	DNS RESOLUTION
Facebook	com.waze	descr:Facebook	graph.facebook.com

Vi sono flussi legati alla fornitura di pubblicità da parte di Google come il flusso in figura:

PACKAGE	WHOIS	DNS RESOLUTION	SNI
com.waze	OrgName:GoogleLLC	ds-resources-legacy.waze.con	ads-resources-legacy.waze.com
com.waze	OrgName:GoogleLLC	ads-resources-legacy.waze.com	ads-resources-legacy.waze.com

Altri flussi sono generati dal servizio di oper graph di Facebook.

PACKAGE	WHOIS	DNS RESOLUTION	SNI
com.waze	descr:Facebook	graph.facebook.com	graph.facebook.com

Infine, vi sono flussi legati a servizi Google

PACKAGE	WHOIS	DNS RESOLUTION	SNI
com.waze	OrgName:GoogleLLC	gstatic.com	gstatic.com

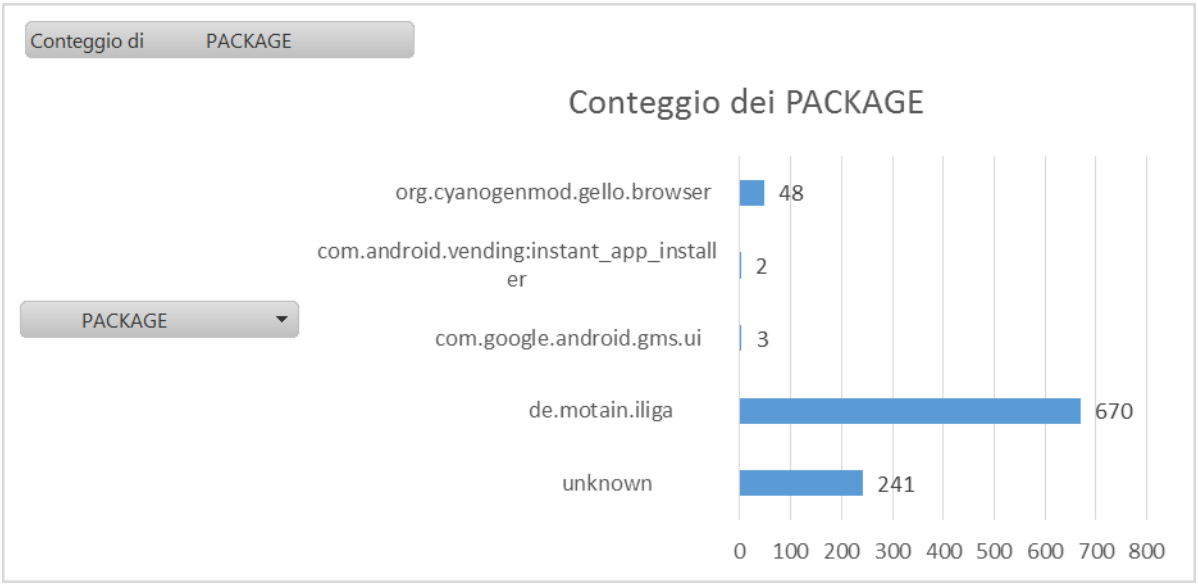
Alcune catture, in particolare la terza (cpt3), presentano numerosi valori "unknown" nel campo package che fanno riferimento a server quali *stats.cyanogenmod.org*, tali flussi sono dovuti alla comunicazione di statistiche Cyanogenmod. Riportiamo alcuni dei flussi di seguito:

PROTO	APP_TYPE (TIE)	PACKAGE	WHOIS	DNS RESOLUTION
TCP	Other TCP	unknown	OrgName:AmazonTechnologiesInc.	stats.cyanogenmod.org
TCP	Other TCP	unknown	OrgName:AmazonTechnologiesInc.	stats.cyanogenmod.org
TCP	Other TCP	unknown	OrgName:AmazonTechnologiesInc.	stats.cyanogenmod.org
TCP	Other TCP	unknown	OrgName:AmazonTechnologiesInc.	stats.cyanogenmod.org
TCP	Other TCP	unknown	OrgName:AmazonTechnologiesInc.	stats.cyanogenmod.org

4.3 Analisi del traffico di One Football

Abbiamo ottenuto un totale di 964 flussi di traffico nella cattura del traffico generato dall'applicazione One Football.
I package utilizzati dall'applicazione sono:

- **org.cyanogen.gello.browser**: package appartenente al browser Gello sviluppato da Cyanogen, l'utilizzo di tale browser è causato probabilmente dall'apertura dei vari link di notizie e contenuti presenti nell'applicazione.
- **com.android.vending:instant_appinstaller:** package android per l'installazione dell'applicazione;
- **com.google.android.gms.ui:** package di applicazioni Google e API che supportano la funzionalità sui vari dispositivi;
- **de.montain.iliga:** il package dell'applicazione stessa;
- **unknown:** sconosciuto.



Osserviamo che l'applicazione effettua numerose connessioni con aziende come Google, Facebook, Yahoo, ecc…

In particolare, alcuni di questi flussi sono relativi alla fornitura di pubblicità (ads) all'interno dell'applicazione come i flussi di seguito riportati:

PACKAGE	WHOIS	DNS RESOLUTION	SNI
unknown	OrgName:GoogleLLC	googleads.g.doubleclick.net	googleads.g.doubleclick.net
de.motain.iliga	descr:Yahoo!Europe	ads.flurry.com	ads.flurry.com
de.motain.iliga	OrgName:MoPub,Inc.	ads.mopub.com	ads.mopub.com

Vi sono flussi relativi ai servizi Facebook degli Open Graph:

PACKAGE	WHOIS	DNS RESOLUTION	SNI
de.motain.iliga	descr:Facebook	graph.facebook.com	graph.facebook.com

Flussi generati da servizi di Analytics:

PACKAGE	WHOIS	DNS RESOLUTION	SNI
de.motain.iliga	OrgName:AmazonTechnologiesInc.	analytics.localytics.com	analytics.localytics.com
de.motain.iliga	OrgName:GoogleLLC	www.google-analytics.com	None

Flussi relativi ad altri servizi di terze parti:

PACKAGE	WHOIS	DNS RESOLUTION	SNI
de.motain.iliga	OrgName:AmazonTechnologiesInc.	s3.amazonaws.com	s3.amazonaws.com
de.motain.iliga	org-name:AdjustGmbH	app.adjust.com	app.adjust.com

Flussi relativi ai contenuti multimediali e non presenti nella sezione notizie dell'applicazione:

PACKAGE	WHOIS	DNS RESOLUTION	SNI
de.motain.iliga	OrgName:TwitterInc.	mobile.twitter.com	mobile.twitter.com
de.motain.iliga	descr:Facebook	scontent-lhr3-1.cdninstagram.com	scontent-lhr3-1.cdninstagram.com
de.motain.iliga	ame:AmazonDataServicesIrelandLir	news.onefootball.com	news.onefootball.com
de.motain.iliga	OrgName:AmazonTechnologiesInc.	m.calcionapoli24.it	audit.quantcast.mgr.consensu.org
de.motain.iliga	OrgName:GoogleLLC	encrypted-tbn0.gstatic.com	encrypted-tbn0.gstatic.com

Infine, legati al funzionamento e configurazione dell'applicazione:

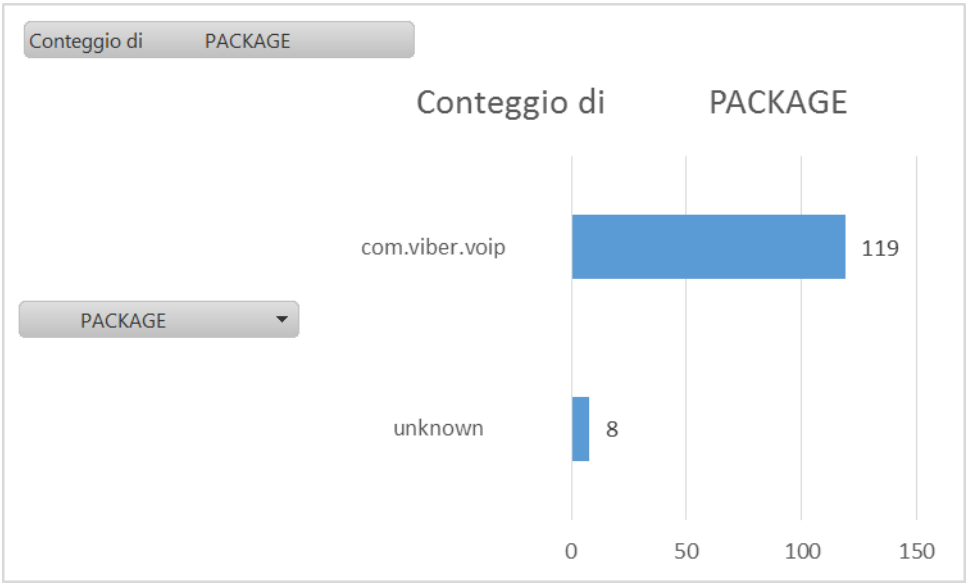
PACKAGE	WHOIS	DNS RESOLUTION	SNI
de.motain.iliga	descr:AkamaiTechnologies	config.onefootball.com	image-service.onefootball.com

Anche in questa cattura vi sono flussi che presentano stats.cyanogenmod.org nel campo DNS RESOLUTION.

4.4 Analisi del traffico di Viber

Catturando il traffico di Viber abbiamo ottenuto 127 flussi generati dall'applicazione.
I package utilizzati sono:

- **com.viber.voip:** il package dell'applicazione;
- **unknown:** sconosciuti.



I flussi ottenuti sono relativi a diversi servizi, ne individuiamo infatti varie tipologie, elencheremo di seguito i flussi relativi a categorie di servizi:

Flussi relativi alla fornitura di pubblicità all'interno dell'applicazione

PACKAGE	WHOIS	DNS RESOLUTION	SNI
com.viber.voip	OrgName:AmazonDataServicesIrelandLimited	ads.viber.com	ads.viber.com
com.viber.voip	descr:FRA1IBNet-HostIPs	mediation.adnxs.com	fra1-mobile.adnxs.com
com.viber.voip	OrgName:GoogleLLC	googleads.g.doubleclick.net	googleads.g.doubleclick.net

Flussi generati da servizi di web analysis e analytics:

PACKAGE	WHOIS	DNS RESOLUTION	SNI
com.viber.voip	OrgName:AmazonTechnologiesInc.	teutorigos-cat.com	teutorigos-cat.com
com.viber.voip	OrgName:GoogleLLC	www.google-analytics.com	www.google-analytics.com

Flussi relativi a servizi di terze parti:

PACKAGE	WHOIS	DNS RESOLUTION	SNI
com.viber.voip	descr:IPv4addressblocknotmanagedbytheRIPENCC	venetia.iad.appboy.com	venetia.iad.appboy.com

com.viber.voip	descr:LeasewebDeutschlandGmbH	app.adjust.com	app.adjust.com

Flussi generati dai contenuti multimediali presenti nell'applicazione:

PACKAGE	WHOIS	DNS RESOLUTION	SNI
com.viber.voip	descr:SuperNetworks.r.o.	bots.gameeapp.com	www.gamee.com

Flussi legati a Facebook, in particolar modo alla funzionalità di login all'applicazione attraverso Facebook:

PACKAGE	WHOIS	DNS RESOLUTION	SNI
com.viber.voip	descr:Facebook	staticxx.facebook.com	staticxx.facebook.com
com.viber.voip	descr:Facebook	connect.facebook.net	connect.facebook.net
com.viber.voip	descr:Facebook	sdk.accountkit.com	sdk.accountkit.com

Flussi generati da servizi Twitter:

PACKAGE	WHOIS	DNS RESOLUTION	SNI
com.viber.voip	OrgName:TwitterInc.	syndication.twitter.com	syndication.twitter.com

Effettuando delle ricerche in rete si è appreso che syndacation.twitter.com è un server a cui fa affidamento un malware che funge da dirottatore del traffico. Sostanzialmente esso ha la possibilità di controllare il browser modificando pagine iniziali (aggiungendo quelle di terze parti) aprire tabs e gestire pop-up.

Conclusioni

I risultati proposti da questo elaborato fanno capire come sia in continua evoluzione il traffico dati presente su Internet e quanto sia importante analizzarlo. I dati analizzati sono stati generati da applicazioni usate quotidianamente, ma fa capire quanto al giorno d'oggi sia diffusa la comunicazione macchina-macchina in contrapposizione a quella uomo-macchina. Data la grande quantità di applicazioni che lavorano sulla rete è importante capire il tipo di traffico che queste generano ai fini della gestione e della sicurezza della rete stessa. Ciò può essere fatto, come si accennava nel capitolo 1, solo attraverso meccanismi di apprendimento basato sull'analisi delle intestazioni dei pacchetti a causa dell'enorme quantità di traffico cifrato.

Bibliografia

[1] Dainotti, Alberto, Antonio Pescapè, and Kimberly C. Claffy. "Issues and future directions in traffic classification." *IEEE network* 26.1 (2012).

[2] James Kurose - Keith Ross, Reti di calcolatori e Internet un approccio top-down, Pearson, 2013.

[3] S. Seneviratne, H. Kolamunna, A. Seneviratne. “A Measurement Study of Tracking in Paid Mobile Applications”. In Proceedings of the 8th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec), New York City, USA. June, 2015.

[4] Malwarebytes, https://forums.malwarebytes.com/topic/216607-syndicationtwitter/,

10/12/2018