



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 – 16

Mariano Astarita
Matr. N46002698

Giuseppe Carrella
Matr. N46002595

Indice

Indice.....	II
Capitolo 1: Classificazione del traffico.....	3
1.1 Introduzione	3
1.1.1 Machine Learning	5
1.1.2 Deep Learning.....	6
Capitolo 2: Strumenti utilizzati	7
2.1 Strumenti per la classificazione del traffico.....	7
2.1.1 Traffic Identification Engine (TIE).....	7
2.1.2 Wireshark	8
2.1.3 tShark	10
Capitolo 3: Cattura e analisi del traffico mobile	11
3.1 Cattura del traffico	11
3.1.1 Dati e Metadati.....	11
3.2 Preparazione analisi	12
3.3 Script per l'automazione delle analisi	12
3.2.1 Script	12
Capitolo 4: Risultati sperimentali	14
4.1 Riepilogo dei risultati.....	14
4.1.1 File 1	14
4.1.2 File 2	14
4.1.3 File 3	15
4.1.4 File 4	15
4.1.5 File 5	15
4.1.6 File 6	15
4.1.7 File 7	16
4.1.8 File 8	16
4.1.9 File 9	16
4.1.10 File 10	16
4.1.11 File 11	17
4.1.12 File 12	17
4.1.13 File 13	17
4.1.14 File 14	17
4.1.15 File 15	17
4.1.16 File 16	17
4.1.17 File 17	18
4.1.18 File 18	18
Conclusioni	19
Considerazioni sui risultati.....	19

Capitolo 1: Classificazione del traffico

1.1 Introduzione

Oggi giorno, sono molti gli utenti in possesso di dispositivi che permettono l'accesso ad Internet, aumentando in maniera esponenziale il traffico di rete. Con il passare del tempo, gli Internet Service Provider (ISP), i quali consentono la connettività ad internet, hanno sollevato una serie di problematiche. Essi necessitano, ad esempio, di conoscere il tipo di traffico generato da un'applicazione, in modo tale da poter reagire ai vincoli di sicurezza e qualità stabiliti dall'utente. Un esempio si ha nel caso di offerte dei provider telefonici, quando viene permesso l'utilizzo gratuito in rete di una determinata applicazione. Per fare ciò, gli ISP hanno la necessità di classificare il traffico, così da essere certi della correlazione con l'applicazione di cui l'utente può usufruire con traffico illimitato. Questi problemi e tanti altri sono risolti dalla classificazione del traffico di rete, con il quale si cerca di controllare e gestire efficacemente le risorse nelle reti, nonché di migliorare l'affidabilità. La classificazione del traffico mobile fornisce informazioni utili in molti ambiti, ma riscontra delle evidenti problematiche di privacy (divulgazione di dati personali), "eavesdropper" e terze parti malevole.

Per la classificazione del traffico, possono essere utilizzati diversi approcci. Uno di questi consiste nell'identificare, mediante la porta sorgente e destinazione del livello trasporto, il flusso di traffico generato da un'applicazione. Si considerano, quindi, pacchetti consecutivi con la stessa 5-tupla (protocollo, source address, source port, destination address, destination port) come appartenenti allo stesso flusso.

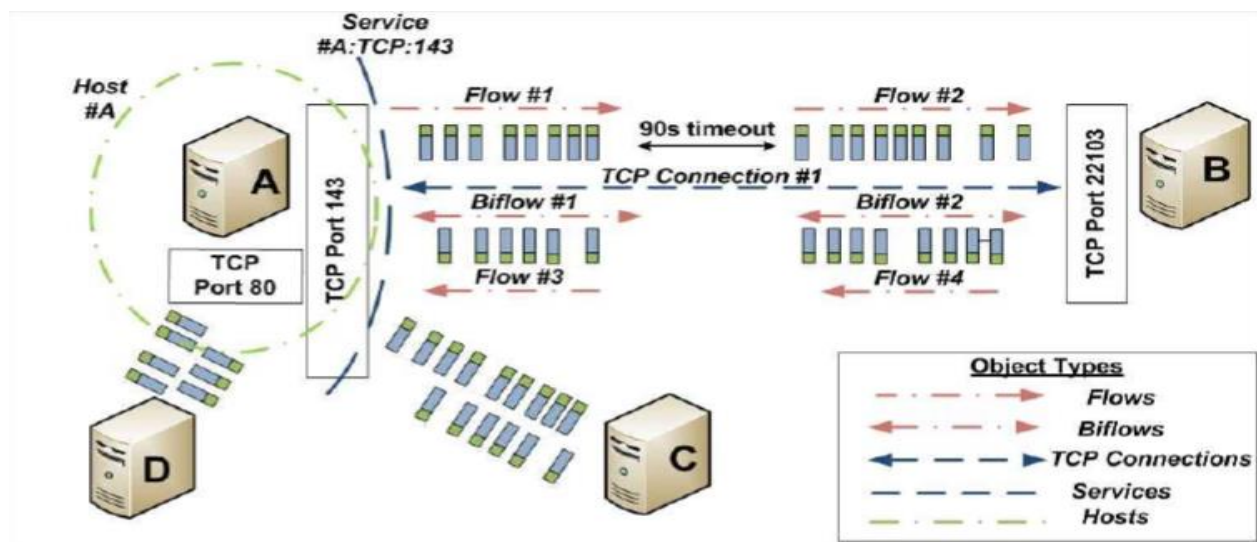
Una prima analisi viene fatta sulla cattura del traffico per costruire la ground truth, un traffico noto utilizzato per addestrare i classificatori, che consenta l'apprendimento di determinate regole. In questo modo, classificatore sarà in grado di riconoscere quale applicazione avrà generato un determinato flusso.

Classificare un singolo pacchetto, dal punto di vista operativo, è molto complicato. Per tale motivo si preferisce andare a classificare stream di pacchetti, ovvero interi flussi. Se si considera la direzione del flusso dal mittente al destinatario e viceversa, si parla di flusso bidirezionale. In questo caso la coppia indirizzo IP Sorgente-Porto Sorgente e indirizzo IP Destinatario-Porto Destinatario, sono interscambiabili. L'oggetto della classificazione sarà, quindi, un aggregato di traffico, al quale viene associata un'etichetta.

Nell'ambito della classificazione possiamo andare a definire la gradualità, il tipo e il livello di dettaglio del traffico che si andrà a considerare. Ciò è possibile attraverso la definizione delle classi. In particolare, possiamo avere:

- *Classi di traffico*, le quali riguardano una classificazione di più alto livello. Un esempio sono le classi a bulk, le quali trasportano una quantità elevata di traffico dati (backup notturni aziendali, ecc...).
- *Categorie di applicazioni*, attraverso le quali il traffico può essere classificato in base alla categoria di applicazione. Ad esempio, una categoria può essere l'insieme di tutte le applicazioni di messaggistica.
- *Applicazioni* (Facebook, Skype, ecc.).
- Un *servizio* all'interno di una applicazione.

Ci sono diversi approcci per effettuare la classificazione. Il port-based è un approccio che sfrutta la condizione per la quale un'applicazione invia traffico ad un'altra applicazione, la quale è in ascolto su un determinato porto. Quindi, conoscendo l'associazione univoca porto-applicazione, con opportuni tool, si può osservare il porto sorgente contenuto nel pacchetto e di conseguenza individuare quale applicazione lo ha generato. Ad esempio, siccome il traffico http è sempre diretto sulla porta 80, nel momento in cui si comprende che il porto destinazione è 80, si intuisce che il traffico è di tipo http. Al giorno d'oggi, diventa difficile avere un'associazione univoca, considerando che la maggior parte dei flussi generati è cifrata. Infatti, riferendoci all'esempio precedente, anche se il traffico è diretto verso la porta 80, non si ha la sicurezza che sia traffico http, anche se può apparire tale. In conclusione, possiamo dedurre che il classificatore port-based ha una scarsa accuratezza.



Un altro approccio utilizzato per la classificazione è il payload-based. Esso consiste nell'osservare direttamente il payload dei pacchetti, ovvero la sezione del pacchetto contenente le informazioni "sensibili". Difatti, questo approccio è basato sull'ispezione del payload di livello trasporto al fine di identificare stringhe relative al protocollo di livello applicazione (ed in generale all'applicazione) ed

eseguire un matching con un set di stringhe predefinite. Il problema che si riscontra in questo caso è il traffico che, al giorno d'oggi, è cifrato e può rendere tale approccio inefficiente.

1.1.1 Machine Learning

I problemi scontrati dai precedenti approcci, vengono risolti con un approccio flow-feature-based, con un algoritmo di Machine Learning (ML). Tale algoritmo consiste nell'estrarre delle feature, cioè delle caratteristiche distintive del traffico, le quali andranno ad addestrare il classificatore ML per permettere di discriminare il traffico.

Quindi, il Machine Learning effettua una classificazione in base a delle feature che, essendo il traffico cifrato, non sono estratte dal payload, ma dall'header contenente informazioni, in questo caso molto utili, riguardanti la dimensione del pacchetto e il tempo di arrivo del pacchetto.

I classificatori Machine Learning si dividono in due categorie:

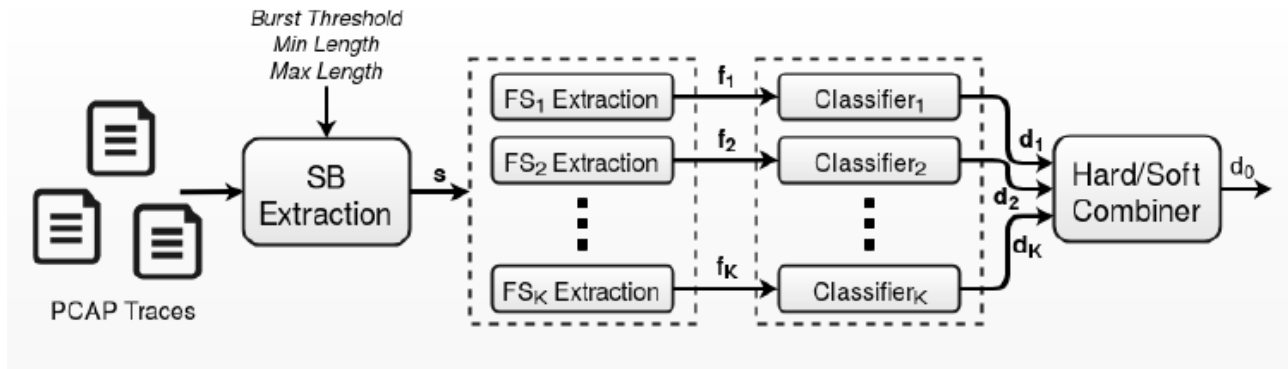
- *Learning Supervisionato*. Una prima fase di questo approccio consiste nell'apprendimento del traffico: il classificatore osserva una grande quantità di traffico in modo da essere in grado, in un secondo momento, di creare una ground truth. Al termine, il traffico ignoto, confrontato con la ground truth, sarà quindi riconosciuto e classificato.
- *Learning Non Supervisionato*. In questo caso i classificatori apprendono su traffico non noto cioè traffico non tabellato. Non sappiamo, quindi, a quale categoria esso appartenga. In ogni caso, a partire dai dati stessi, viene chiesto alla macchina di estrarre una regola che raggruppi i casi presentati, secondo le caratteristiche che sono state ricavate.

Possiamo avere diversi classificatori che operano in parallelo e, quindi, combinare i tre approcci sopra presentati. Il classificatore port-based, ad esempio, ha un'elevata accuratezza per quanto riguarda il traffico che utilizza porte standard, mentre uno payload-based possiede alte prestazioni, in termini di accuratezza, per quanto riguarda il traffico in chiaro. Infine, un classificatore ML garantisce un'elevata accuratezza sulla classificazione di traffico cifrato. L'unione di questi tre approcci, anche definita multi-classificazione, consente di ottenere un'accuratezza globale elevata. In questo caso, il responso finale generato dal classificatore, sopra descritto, è ottenuto tramite una votazione a maggioranza.

La classificazione può avvenire secondo due modalità: classificazione online e offline. La classificazione effettuata in modalità offline, non ha dei vincoli riguardo la velocità con cui viene effettuata, di conseguenza si ha un'accuratezza maggiore. Nel caso di una classificazione online, invece, il classificatore deve essere molto rapido, ovvero osservando una piccola quantità di traffico, si ha la necessità di conoscere l'appartenenza all'applicazione utilizzata. In queste situazioni è importate utilizzare una tecnica con time-out. Si impone un certo time-out allo scadere del quale viene aggregato il traffico per poi classificarlo. Un'apparente soluzione consiste nell'imporre un time-out molto grande, ma in questo caso il traffico classificato risulterà inutile, nonostante un'accuratezza molto elevata.

Il problema principale della classificazione ML, è l'eventuale numero eccessivo di applicazioni utilizzate sul dispositivo. Dato che le applicazioni generanti traffico possono essere eccessive, addestrare una ground truth manualmente, per un numero molto elevato di applicazioni, diventa molto complicato. Esistono tuttavia tecniche semiautomatiche per la generazione della ground truth, come quella che è stata utilizzata per lo sviluppo di questo elaborato. In questo caso, infatti, il

traffico è generato manualmente, ma in maniera automatica si riesce a correlare ad ogni connessione l'applicazione.



1.1.2 Deep Learning

Il ML presenta delle forti limitazioni. La classificazione ML non può essere automatizzata perché la fase di feature extraction necessita sempre la conoscenza del dominio del problema e quindi un intervento dell'operatore di rete, il quale conosce le feature più utili per la classificazione del traffico. Il ML ha una classificazione lenta, quindi tali tecniche di ML divengono in breve tempo obsolete perché necessitano di un aggiornamento di features costante. Ad esempio, se una determinata azienda aggiorna la corrispettiva applicazione ogni mese, si avrà la necessità di cambiare la ground truth e ciò significherebbe effettuare la reingegnerizzazione molto frequentemente. È ovvio che tale condizione risulta inconveniente.

Le limitazioni del ML sono superate dalle tecniche di classificazione basate sul Deep Learning (DL). Il DL risulta molto utile per la classificazione online, ma possiede un tempo di addestramento molto elevato. Per cui, si è giunti alla risoluzione del problema addestrando il classificatore offline. Il DL elimina sia la fase di estrazione delle features che la fase consistente nell'individuazione di quelle più rilevanti per la classificazione del traffico, riducendo la necessità di una conoscenza approfondita del dominio. Infatti, nel caso di personale inesperto il problema è risolto dall'algoritmo stesso di Deep Learning. In realtà, un riconoscimento delle feature sicuramente più utile è quello effettuato dal retista ma, anche in questo caso, si avrà il vantaggio di non dover stabilire poi quali siano le feature più interessanti per andare a classificare il traffico.

Capitolo 2: Strumenti utilizzati

2.1 Strumenti per la classificazione del traffico

2.1.1 Traffic Identification Engine (TIE)

TIE, Traffic Identification Engine, è un tool che consente di analizzare e classificare il traffico di rete effettuando una comparazione tra diversi approcci di classificazione e offrendone, quindi, una rappresentazione unificata dei risultati.

Possiede tre modi di operare: offline, realtime e ciclico. La modalità di funzionamento offline è molto utilizzata quando non ci sono limitazioni temporali o quando le tecniche di classificazione richiedono che il flusso di traffico sia osservato per tutta la durata della sua vita. Per quanto riguarda la modalità realtime, invece, la classificazione avviene online appena si ha a disposizione un numero sufficiente di informazioni. Infine, la modalità ciclica consiste nel classificare il flusso di traffico ad intervalli regolari di tempo, i cui risultati sono suddivisi in file di output corrispondenti agli intervalli.

TIE è scritto in linguaggio C, il software è costituito da un solo eseguibile ed un set di plugin di classificazione, caricati dinamicamente a runtime se le features che essi richiedono sono disponibili. Inoltre, vi sono una serie di utility che consentono di processare, in una fase successiva, i file di output.

TIE processa i pacchetti in 5 passi, di cui due variano a seconda del fatto che ci si trovi nella fase di addestramento o in quella di classificazione:

- *Packet Filter*: le frame di livello collegamento sono catturate, o lette da file, e filtrate sulla base di regole.
- *Session Builder*: il traffico di rete è organizzato in sessioni. In questa fase distinguiamo tra flussi, flussi bi-direzionali e host. Si tiene traccia delle varie sessioni tramite una chained hash table e, inoltre, è incluso un garbage collector che si occupa di rilasciare periodicamente le risorse e terminare le sessioni.
- *Feature Extractor*: per ciascuna sessione sono fornite features di base, più delle features che possono essere estratte on-demand se richieste.
- *Decision Combiner*: questa fase si presenta quando TIE è utilizzato in fase di classificazione. Il DC è utilizzato per classificare ciascuna sessione combinando differenti algoritmi di classificazione e, quindi, i loro risultati. Il risultato ottenuto, in termini di affidabilità globale, è valutato con un livello di confidenza che va da 0 a 100.
- *Output Generator*: questa fase si presenta quando TIE è utilizzato nella classificazione e rappresenta l'ultimo passo. In questo passo è generato un file di output che contiene

informazioni riguardo le sessioni processate e la loro classificazione. I campi contenuti in questo file sono i seguenti:

1. id: identificatore di flusso;
 2. src_ip: indirizzo IP sorgente;
 3. dst_ip: indirizzo IP destinazione;
 4. proto: protocollo (il numero 6 indica TCP, il numero 17 indica UDP);
 5. sport: porto sorgente;
 6. dport: porto destinazione;
 7. dwpkts: numero di pacchetti downstream;
 8. uppkts: numero di pacchetti upstream;
 9. dwbytes: numero di byte downstream;
 10. upbytes: numero di byte upstream;
 11. t_start: tempo d' inizio della sessione;
 12. t_last: tempo di fine della sessione;
 13. app_id: l'ID dell'applicazione è risultata dall'applicazione;
 14. sub_id: il sotto ID dell'applicazione è risultato dall'applicazione;
 15. app_details: dettagli applicazione;
 16. confidence: livello di confidenza del processo di classificazione.
- *Pre-classifier*: questo passo si ha quando TIE è utilizzato nella fase di addestramento del classificatore. Sono prelevate le etichette associate a ciascuna sessione a partire dal file della ground-truth.
 - *Trainer*: questo passo si ha quando TIE è utilizzato nella fase di addestramento. Sono invocate le funzionalità implementate da ciascun plugin in modo da collezionare le informazioni necessarie su ciascun pacchetto e avviare l'addestramento al termine dell'esecuzione di TIE.

2.1.2 Wireshark

Wireshark è un network analyzer, cioè un software che permette l'analisi dei pacchetti che transitano in rete, con lo scopo di analizzarli. Esso è fornito da una serie di tool che rendono piacevole e più dinamica l'analisi. Usare un analizzatore di rete è il modo migliore per capire cosa c'è che non va nella rete. Permette di capire approfonditamente quali e quanti pacchetti ci sono. Inoltre fornisce molti utili strumenti per individuare i fenomeni che disturbano la nostra navigazione. Wireshark riesce a "comprendere" la struttura di diversi protocolli di rete, è in grado di individuare eventuali incapsulamenti, riconosce i singoli campi e permette di interpretarne il significato. Ha diverse caratteristiche e funzioni, tra le quali è possibile analizzare dati acquisiti in tempo reale su una rete attiva e analizzare dati salvati precedentemente su file di cattura. I dati catturati su file possono essere facilmente modificati, convertiti o filtrati. È possibile filtrare i dati da visualizzare e utilizzare filtri di visualizzazione per colorare o evidenziare selettivamente le informazioni sommarie sui pacchetti. Inoltre, è possibile scomporre e analizzare molti protocolli di comunicazione.

Wireshark offre anche una comoda finestra con tutti i possibili filtri da applicare al dump. L'interfaccia grafica è costituita da una barra dei filtri, nella quale vengono inseriti quelli desiderati. Una schermata di esempio è riportata nella figura sottostante:

traffic.pcap						
File Modifica Visualizza Vai Cattura Analizza Statistiche Telefonja Wireless Strumenti Aiuto						
Applica un filtro di visualizzazione ... <Ctrl-/> Espressione...						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.20.105	52.35.0.252	TCP	74	32835 → 443 [SYN] Seq=0 Win=65535 L
2	3.996001	192.168.20.105	52.35.0.252	TCP	74	[TCP Retransmission] 32835 → 443 [S
3	12.010006	192.168.20.105	52.35.0.252	TCP	74	[TCP Retransmission] 32835 → 443 [S
4	27.323015	XiaomiCo_18:cc:bf	RealtekS_53:44:58	ARP	60	Who has 192.168.20.254? Tell 192.16
5	27.323036	RealtekS_53:44:58	XiaomiCo_18:cc:bf	ARP	42	192.168.20.254 is at 00:e0:4c:53:44
6	43.901021	192.168.20.105	8.8.8.8	DNS	86	Standard query 0x5113 A android.cli
7	43.936676	8.8.8.8	192.168.20.105	DNS	222	Standard query response 0x5113 A an
8	44.829020	192.168.20.105	216.58.205.46	QUIC	1392	Client Hello, PKN: 1, CID: 18147491
9	44.840016	192.168.20.105	216.58.205.46	TCP	74	48109 → 443 [SYN] Seq=0 Win=65535 L
10	44.855181	216.58.205.46	192.168.20.105	TCP	74	443 → 48109 [SYN, ACK] Seq=0 Ack=1
11	44.857018	192.168.20.105	216.58.205.46	TCP	66	48109 → 443 [ACK] Seq=1 Ack=1 Win=8
12	44.863175	216.58.205.46	192.168.20.105	QUIC	1392	Rejection, PKN: 1, CID: 18147491578
13	44.863190	216.58.205.46	192.168.20.105	QUIC	1392	Payload (Encrypted), PKN: 2, CID: 1
14	44.866017	192.168.20.105	216.58.205.46	TLSv1.2	283	Client Hello
Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)						
Ethernet II, Src: XiaomiCo_18:cc:bf (38:a4:ed:18:cc:bf), Dst: RealtekS_53:44:58 (00:e0:4c:53:44:58)						
Internet Protocol Version 4, Src: 192.168.20.105, Dst: 52.35.0.252						
Transmission Control Protocol, Src Port: 32835, Dst Port: 443, Seq: 0, Len: 0						

Notiamo che Wireshark permette di visualizzare informazioni di Transmission Control Protocol, formato da Src Port e Dst Port.

Inoltre, possiamo applicare filtri su specifici indirizzi IP Sorgenti e Destinatari:

Esempio: `ip.src == 192.168.20.105`

Il comando appena riportato, filtra tutte le richieste dall'indirizzo ip sorgente 192.168.20.105.

Esempio: `ip.dst == 216.58.205.46`

Tale comando, invece, filtra tutte le richieste dirette all'ip destinazione 216.58.205.46.

Inserendo nelle regole di filtraggio gli operatori logici, possiamo combinare diversi comandi. Ad esempio:

traffic.pcap						
File Modifica Visualizza Vai Cattura Analizza Statistiche Telefonja Wireless Strumenti Aiuto						
ip.src == 192.168.20.105 and ip.dst == 216.58.205.46 Espressione...						
No.	Time	Source	Destination	Protocol	Length	Info
8	44.829020	192.168.20.105	216.58.205.46	QUIC	1392	Client Hello, PKN: 1, CID: 18147491578658637633
9	44.840016	192.168.20.105	216.58.205.46	TCP	74	48109 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
11	44.857018	192.168.20.105	216.58.205.46	TCP	66	48109 → 443 [ACK] Seq=1 Ack=1 Win=87808 Len=0 TSval=7232032 T...
14	44.866017	192.168.20.105	216.58.205.46	TLSv1.2	283	Client Hello
15	44.883017	192.168.20.105	216.58.205.46	QUIC	70	Payload (Encrypted), PKN: 2, CID: 18147491578658637633
20	44.906014	192.168.20.105	216.58.205.46	TCP	66	48109 → 443 [ACK] Seq=218 Ack=1419 Win=90624 Len=0 TSval=7232...
21	44.908013	192.168.20.105	216.58.205.46	TCP	66	48109 → 443 [ACK] Seq=218 Ack=2837 Win=93440 Len=0 TSval=7232...
22	44.909017	192.168.20.105	216.58.205.46	TCP	66	48109 → 443 [ACK] Seq=218 Ack=3586 Win=96256 Len=0 TSval=7232...
23	44.919018	192.168.20.105	216.58.205.46	TLSv1.2	159	Client Key Exchange, Change Cipher Spec, Encrypted Handshake ...
26	44.945020	192.168.20.105	216.58.205.46	QUIC	1392	Client Hello, PKN: 3, CID: 18147491578658637633
27	44.958016	192.168.20.105	216.58.205.46	QUIC	1392	Payload (Encrypted), PKN: 4, CID: 18147491578658637633
28	44.962013	192.168.20.105	216.58.205.46	QUIC	1268	Payload (Encrypted), PKN: 5, CID: 18147491578658637633
29	44.974016	192.168.20.105	216.58.205.46	TCP	66	48109 → 443 [ACK] Seq=311 Ack=3939 Win=99072 Len=0 TSval=7232...
Frame 8: 1392 bytes on wire (11136 bits), 1392 bytes captured (11136 bits)						
Ethernet II, Src: XiaomiCo_18:cc:bf (38:a4:ed:18:cc:bf), Dst: RealtekS_53:44:58 (00:e0:4c:53:44:58)						
Internet Protocol Version 4, Src: 192.168.20.105, Dst: 216.58.205.46						
User Datagram Protocol, Src Port: 53887, Dst Port: 443						
QUIC (Quick UDP Internet Connections)						

È possibile filtrare, oltre per indirizzi ip sorgenti e destinazioni, anche per il tipo di porta o di protocollo.

Esempio: `tcp.port == 80`

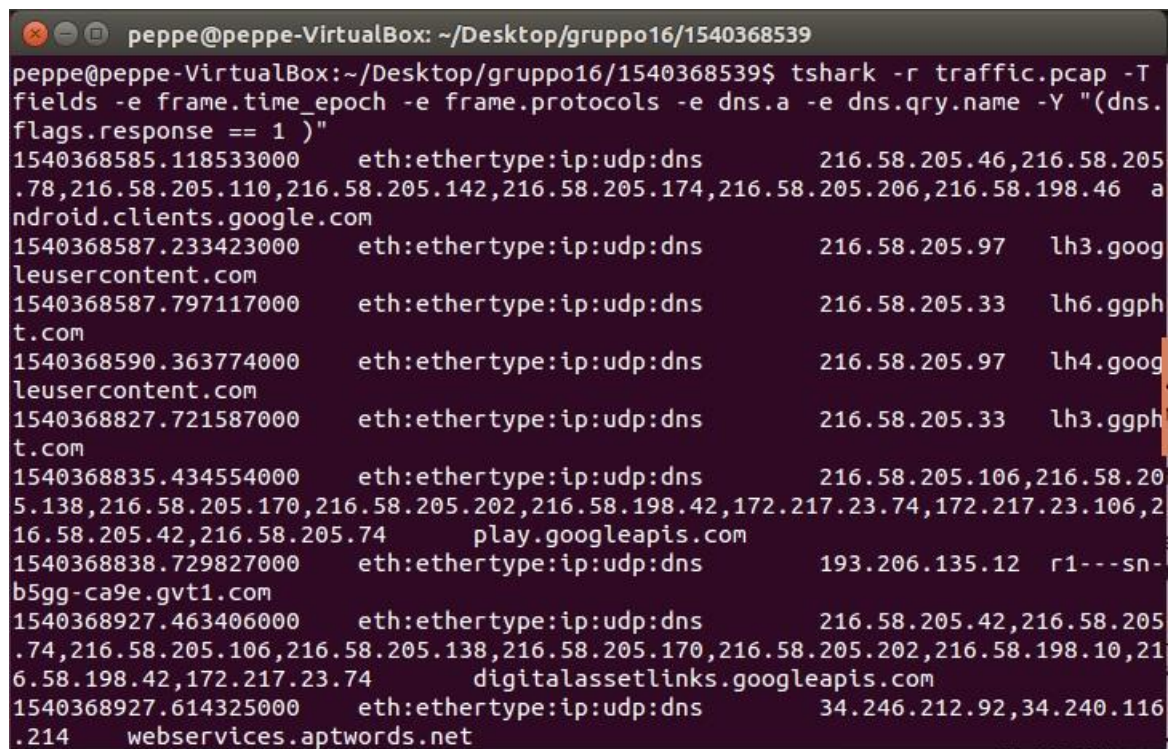
Esso filtra tutte le richieste che sono passate sulla porta 80. Se scriviamo `udp` o `tcp` nella barra dei filtri, vengono visualizzate solo le richieste UDP o TCP

In una navigazione web, il `destination port` raggruppa le richieste che il nostro computer invia al server. In `source port` verranno raggruppate le risposte del server. Nei filtri delle porte possiamo specificare se il pacchetto sta entrando o uscendo da una determinata porta.

2.1.3 tShark

Esiste una versione di Wireshark a linea di comando, chiamata `tshark`, utilizzata principalmente per estrarre le risoluzioni DNS presenti nelle tracce di traffico. Il comando utilizzato per tali richieste è: `tshark -r traffic.pcap -T fields -e frame.time_epoch -e frame.protocols -e dns.resp.addr -e dns.qry.name -Y "(dns.flags.response == 1)"`

Risultato di esempio:



```
peppe@peppe-VirtualBox: ~/Desktop/gruppo16/1540368539
peppe@peppe-VirtualBox:~/Desktop/gruppo16/1540368539$ tshark -r traffic.pcap -T
fields -e frame.time_epoch -e frame.protocols -e dns.a -e dns.qry.name -Y "(dns.
flags.response == 1)"
1540368585.118533000    eth:ethertype:ip:udp:dns      216.58.205.46,216.58.205
.78,216.58.205.110,216.58.205.142,216.58.205.174,216.58.205.206,216.58.198.46 a
ndroid.clients.google.com
1540368587.233423000    eth:ethertype:ip:udp:dns      216.58.205.97    lh3.goog
leusercontent.com
1540368587.797117000    eth:ethertype:ip:udp:dns      216.58.205.33    lh6.ggph
t.com
1540368590.363774000    eth:ethertype:ip:udp:dns      216.58.205.97    lh4.goog
leusercontent.com
1540368827.721587000    eth:ethertype:ip:udp:dns      216.58.205.33    lh3.ggph
t.com
1540368835.434554000    eth:ethertype:ip:udp:dns      216.58.205.106,216.58.20
5.138,216.58.205.170,216.58.205.202,216.58.198.42,172.217.23.74,172.217.23.106,2
16.58.205.42,216.58.205.74    play.googleapis.com
1540368838.729827000    eth:ethertype:ip:udp:dns      193.206.135.12   r1---sn-
b5gg-ca9e.gvt1.com
1540368927.463406000    eth:ethertype:ip:udp:dns      216.58.205.42,216.58.205
.74,216.58.205.106,216.58.205.138,216.58.205.170,216.58.205.202,216.58.198.10,21
6.58.198.42,172.217.23.74    digitalassetlinks.googleapis.com
1540368927.614325000    eth:ethertype:ip:udp:dns      34.246.212.92,34.240.116
.214    webservices.aptwords.net
```

Capitolo 3: Cattura e analisi del traffico mobile

3.1 Cattura del traffico

La cattura del traffico è stata effettuata in un laboratorio predisposto alla cattura di traffico, nel quale è stato messo a disposizione un sistema che permette di generare una ground truth in maniera semiautomatica, quindi senza necessità dell'utente di osservare il traffico, per poi assegnarci un'etichetta. In particolare, la generazione del traffico è stata effettuata in maniera manuale, ma automaticamente il sistema ha catturato i pacchetti trasmessi dalle applicazioni. In questo caso la ground truth, viene utilizzata per addestrare i classificatori. Il sistema di cattura è costituito da uno smartphone e da un tablet Android connessi ad una workstation che funge da access point verso Internet. Il collegamento tra i dispositivi e la workstation viene effettuato tramite un cavetto che a sua volta è collegato ad un Mobile Capture Terminal. Durante le due sedute, sono state utilizzate le seguenti applicazioni: E-Bay, Slither.io, OneDrive, Google Maps e Weather. Come prima azione, è stato necessario terminare le attività di Google Play con un arresto forzato e disinstallare, se presenti, le app da dover utilizzare. Appena collegati i dispositivi all'hub usb ha avuto inizio la fase di cattura vera e propria. Sono state scaricate le app interessate e, accedendo con un proprio account, si è cominciato a generare più traffico possibile. Allo scadere dei 5 minuti prefissati, è terminata la fase di cattura, quindi sono stati scollegati i cavi ed è stato fatto un arresto forzato delle applicazioni in questione. Questo ciclo di operazioni è stato ripetuto, esclusa la fase di login, per cinque volte per ogni applicazione.

3.1.1 Dati e Metadati

Il PCAP (Packet CAPture) è un insieme di API (Application Programming Interface) che si occupano di catturare il traffico di rete. Queste API sono scritte nel linguaggio di programmazione C ed offrono diverse possibilità, tra cui: Catturare i pacchetti che circolano sulla nostra rete; filtrare i pacchetti catturati a seconda di vari parametri che vedremo successivamente; Salvare i pacchetti su un file.

Al termine della cattura, sono generati diversi file tra cui traffic.pcap e strace.log, contenuti nelle cartelle rinominate con i TimeStamp, che identificano la data della cattura.

L'estensione .pcap (packet capture) è associabile principalmente a Wireshark, programma utilizzato per analizzare le reti e visualizzare informazioni sui singoli pacchetti. I dati e i risultati dell'analisi di rete sono salvati utilizzando appunto l'estensione .pcap, ed è per questo che si parla di .pcap file.

Il file traffic.pcap contiene tracce di traffico, ovvero i pacchetti catturati da specifici programmi. Questo file è molto utile per analizzare le caratteristiche della rete e lo stato, consentendo di

individuare i problemi che si sono verificati sulla rete e ciò può essere fatto tramite il packet analyzer Wireshark.

Il file `strace.log` contiene metadati e, in particolare, le system call delle socket aperte dai dispositivi ed il nome del package (esempio: `com.wish.it`) che ha generato quel particolare biffuso. Nel file sono presenti le varie quintuple (ip sorgente, ip destinazione, porto sorgente, porto destinazione, protocollo di comunicazione). Esso rappresenta la ground truth, cioè la nostra verità, per poi essere confrontata con il file `.pcap` e l'output di classificazione di TIE.

3.2 Preparazione analisi

Come prima cosa, è stata decompressa la cartella relativa al gruppo 16, contenente tante sottocartelle quanti i cicli di cattura effettuati in laboratorio. Successivamente, è stato utilizzato il seguente comando da shell: `“./TIE -a ndping_1.0 -r <path_to_pcap_file>.”`

Attraverso tale comando, è stato possibile ottenere un file `.tie` del file `.pcap` indicato nel percorso digitato. Una volta rinominato, tale file è stato inserito in una cartella creata appositamente. La procedura è stata ripetuta per ogni sottocartella.

3.3 Script per l'automazione delle analisi

La procedura di analisi del traffico mobile è stata eseguita considerando contemporaneamente il file `.pcap`, l'output di classificazione di TIE e il log `strace`. Nel corso della validazione è stato creato un altro file di testo costituito da una copia del file di `.tie`, con l'aggiunta delle colonne Package, DNS, DIG e WHOIS. Per tale file si è utilizzata, per condizioni prestabilite di progetto, l'estensione `.gt.tie`.

Data la grande quantità di righe da analizzare, si è deciso di automatizzare i vari passaggi utili alla classificazione, scrivendo uno script in linguaggio bash. In generale, l'obiettivo principale è stato quello di verificare, per ogni biffuso considerato da Wireshark, se TIE lo ha classificato, con quale identificativo e se corrisponde al package associato al biffuso nel log `strace`.

3.2.1 Script

Le righe iniziali del documento `“script.sh”` servono per specificare quale file si vuole andare ad analizzare. In particolare, l'utente, prima di ogni esecuzione dello script, dovrà andare ad inserire:

- Il percorso del file `.tie` nella variabile denominata `“classtie”`.
- Il percorso del file `strace.log` nella variabile denominata `“strace”`.
- Il percorso del file `.gt.tie` nella variabile denominata `“classgttie”`.
- Il percorso del file `.pcap` nella variabile denominata `“traffic”`.
- Il percorso del file in cui si desidera salvare le richieste dns nella variabile denominata `“dnsfile”`.

All'esecuzione da terminale dello script, la prima operazione che avviene è l'esecuzione del comando fornito da `tshark` per effettuare una richiesta dns per ogni riga del file `traffic.pcap`.

Quindi, il risultato viene salvato in un vettore denominato *dnscmd*. Il contenuto di tale vettore viene, infine, scritto nel file *dns.txt*, creato appositamente nel percorso precedentemente specificato in *dnsfile*.

A questo punto, viene scritta la prima riga nel file *.gt.tie*, la quale è composta dai campi di colonna del file originale *.tie*, con l'aggiunta dei campi *Package*, *DNS*, *DIG* e *WHOIS* prima citati. Può così iniziare l'analisi del traffico. In modo ciclico, vengono salvati i vari campi di riga. Ciò è necessario soprattutto per la composizione della parola chiave da cercare nel *strace.log*. Tale parola, nel formato *src_ip:sport->dst_ip:d_port*, viene utilizzata per estrarre il package associato nel *.log*. Il risultato della ricerca viene salvato nella variabile *rpackage*. Si procede, quindi, con la verifica della presenza del dns relativo al *dst_ip* della riga in questione, effettuando una ricerca nel file *dns.txt*. Il risultato dell'operazione è salvato nella variabile *rdns*. Allo stesso modo, si ottengono ulteriori informazioni sulla compagnia intestataria relativa al *dst_ip*, tramite l'utilizzo del comando *whois*, memorizzando il risultato nel vettore *rwhois*. Per concludere il ciclo, viene valutato il contenuto delle variabili appena citate, assegnando "UNKNOWN" se corrisponde ad una stringa vuota. Nel caso in cui *rdns* risulti "UNKNOWN", viene eseguito un reverse dns attraverso il comando *dig*, salvando il risultato nel vettore *rdig*. Se anche in questo caso non si ottengono risultati, *rdig* viene posto anch'esso "UNKNOWN". Se, invece, in *rdns* vi sia un risultato valido, allora *rdig* viene posto "NOT_REQUIRED". Infine, viene salvata la nuova riga nel file *.gt.tie* e viene ripetuto il ciclo.

Capitolo 4: Risultati sperimentali

4.1 Riepilogo dei risultati

Al termine dell'esecuzione dello script per ogni file .tie, è stato necessario andare a modificare manualmente eventuali errori o classificazioni incomplete. In particolare, per ogni riga, si è andati a confrontare il campo `app_details` generato da TIE con il campo `Package` aggiunto dallo script.

4.1.1 File 1

I flussi contenuti nel file .gt.tie sono 101:

36 richieste DNS, 1 richiesta ARP, 6 classificazioni come Other UDP, 4 classificazioni come Other TCP, 15 crittografati e 39 classificati.

Tra quelli classificati, ve ne sono 6 errati e 1 incompleto.

I flussi classificati da TIE come appartenenti a Yahoo, infatti, sono flussi di Ebay con OrgName: Akamai Technologies (un importante CDN).

Uno dei flussi classificato da TIE come Google, invece, risulta essere un package di Ebay con DNS `googleads.g.doubleclick.net` (DoubleClick è un'agenzia che offre servizi web per la pubblicità online, entrata a far parte del circuito pubblicitario del network di Google da qualche anno).

4.1.2 File 2

I flussi contenuti nel file .gt.tie sono 120:

39 richieste DNS, 1 richiesta ARP, 17 classificazioni come Other TCP, 10 crittografati e 67 classificati.

Tra quelli classificati, ve ne sono 10 errati e 3 incompleti.

9 flussi classificati da TIE come appartenenti a Yahoo, infatti, sono flussi di Ebay, mentre 1 flusso classificato da TIE come appartenente sempre a Yahoo è un flusso di Ebay con OrgName: Akamai Technologies.

3 flussi classificati da TIE come Ebay, invece, risultano essere un package di Ebay con OrgName: Akamai Technologies.

4.1.3 File 3

I flussi contenuti nel file .gt.tie sono 50:

14 richieste DNS, 1 richiesta ARP, 1 classificato come HTTP, 3 classificazioni come Other TCP, 6 crittografati e 26 classificati.

Tra quelli classificati, ve ne sono 1 errato e 4 incompleti.

Il flusso classificato da TIE come appartenente a Yahoo, infatti, è un flusso di Ebay con OrgName: Akamai Technologies.

I flussi classificati da TIE come Ebay, invece, risultano essere package di Ebay con OrgName: Akamai Technologies.

4.1.4 File 4

I flussi contenuti nel file .gt.tie sono 70:

18 richieste DNS, 1 richiesta ARP, 4 classificati come HTTP, 3 classificazioni come Other UDP, 24 classificazioni come Other TCP, 3 crittografati e 17 classificati.

Tra quelli classificati, ve ne sono 5 incompleti.

I flussi classificati da TIE come Google, infatti, risultano essere un package di Slither.io con DNS static.doubleclick.net.

4.1.5 File 5

I flussi contenuti nel file .gt.tie sono 140:

46 richieste DNS, 1 richiesta ARP, 4 classificazioni come Other UDP, 4 classificazioni come Other TCP, 29 crittografati e 56 classificati.

Tra quelli classificati, ve ne sono 12 errati e 2 incompleti.

I flussi classificati da TIE come appartenente a Yahoo, infatti, sono flussi di Ebay con OrgName: Akamai Technologies.

I flussi classificati da TIE come Google, invece, risultano essere package di Ebay con DNS googleads.g.doubleclick.net.

4.1.6 File 6

I flussi contenuti nel file .gt.tie sono 141:

46 richieste DNS, 1 richiesta ARP, 3 classificati come HTTP, 23 classificazioni come Other TCP, 46 crittografati e 22 classificati.

Tra quelli classificati, ve ne sono 6 incompleti.

2 flussi classificati da TIE come Google, infatti, risultano essere un package di Slither.io con DNS static.doubleclick.net, mentre 4 flussi classificati da TIE come Google, infatti, risultano essere un package di org.cyanogenmod.gello.browser con DNS static.doubleclick.net.

4.1.7 File 7

I flussi contenuti nel file .gt.tie sono 174:

59 richieste DNS, 1 richiesta ARP, 16 classificazioni come Other TCP, 35 crittografati e 63 classificati.

Tra quelli classificati, ve ne sono 16 errati e 18 incompleti.

I flussi classificati da TIE come appartenente a Yahoo, infatti, sono flussi di Ebay con OrgName: Akamai Technologies.

I flussi classificati da TIE come Ebay, invece, risultano essere package di Ebay con OrgName: Akamai Technologies.

4.1.8 File 8

I flussi contenuti nel file .gt.tie sono 50:

15 richieste DNS, 1 richiesta ARP, 2 classificati come HTTP, 20 classificazioni come Other TCP e 12 classificati.

Tra quelli classificati, ve ne è 1 incompleto.

Il flusso classificato da TIE come Google, infatti, risulta essere un package di Slither.io con DNS static.doubleclick.net.

4.1.9 File 9

I flussi contenuti nel file .gt.tie sono 153:

57 richieste DNS, 1 richiesta ARP, 9 classificazioni come Other TCP, 30 crittografati e 56 classificati.

Tra quelli classificati, ve ne sono 13 errati e 3 incompleti.

I flussi classificati da TIE come appartenente a Yahoo, infatti, sono flussi di Ebay con OrgName: Akamai Technologies.

I flussi classificati da TIE come Ebay, invece, risultano essere package di Ebay con OrgName: Akamai Technologies.

4.1.10 File 10

I flussi contenuti nel file .gt.tie sono 44:

9 richieste DNS, 1 richiesta ARP, 2 classificati come HTTP, 1 classificazioni come Other UDP ,20 classificazioni come Other TCP e 11 classificati.

Tra quelli classificati, ve ne è 1 incompleto.

Il flusso classificato da TIE come Google, infatti, risulta essere un package di Slither.io con DNS static.doubleclick.net.

4.1.11 File 11

I flussi contenuti nel file .gt.tie sono 51:

7 richieste DNS, 1 richiesta ARP, 2 classificati come HTTP, 35 classificazioni come Other TCP e 6 classificati.

Tra quelli classificati, ve ne è 1 incompleto.

Il flusso classificato da TIE come Google, infatti, risulta essere un package di Slither.io con DNS static.doubleclick.net.

4.1.12 File 12

I flussi contenuti nel file .gt.tie sono 57:

18 richieste DNS, 1 richiesta ARP, 2 classificati come HTTP, 23 classificazioni come Other TCP e 13 classificati.

Tra quelli classificati, ve ne sono 4 incompleti.

Il flusso classificato da TIE come Google, infatti, risulta essere un package di Slither.io con DNS static.doubleclick.net, mentre, altri 3 flussi classificati da TIE come Google, risultano essere un package di Slither.io con DNS googleads.g.doubleclick.net.

Tutto il seguente traffico, proveniente da OneDrive, è risultato crittografato.

4.1.13 File 13

I flussi contenuti nel file .gt.tie sono 54:

19 richieste DNS, 1 richiesta ARP, 1 classificato come MDNS, 3 classificazioni come Other TCP, 28 crittografati e 2 classificati.

4.1.14 File 14

I flussi contenuti nel file .gt.tie sono 51:

19 richieste DNS, 1 richiesta ARP, 2 classificati come MDNS, 28 crittografati e 2 classificati.

4.1.15 File 15

I flussi contenuti nel file .gt.tie sono 78:

24 richieste DNS, 1 richiesta ARP, 1 classificato come MDNS, 5 classificazioni come Other TCP, 44 crittografati e 4 classificati.

4.1.16 File 16

I flussi contenuti nel file .gt.tie sono 116:

65 richieste DNS, 1 richiesta ARP, 1 classificato come MDNS, 9 classificazioni come Other UDP, 7 classificazioni come Other TCP, 9 crittografati e 24 classificati.

4.1.17 File 17

I flussi contenuti nel file .gt.tie sono 80:

15 richieste DNS, 1 richiesta ARP, 1 classificato come MDNS, 8 classificazioni come Other UDP, 36 classificazioni come Other TCP, 6 crittografati e 13 classificati.

4.1.18 File 18

I flussi contenuti nel file .gt.tie sono 80:

7 richieste DNS, 1 richiesta ARP, 1 classificato come MDNS, 5 classificazioni come Other UDP, 22 classificazioni come Other TCP, 1 crittografati e 43 classificati.

Conclusioni

Considerazioni sui risultati

I flussi totali analizzati sono stati 2102. Di questi, 290 risultano crittografati e 476 sono stati classificati da TIE, dove 166 sono quelli classificati in modo errato.

