UNIVERSITA' DEGLI STUDI DI
NAPOLI FEDERICO II

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

Elaborato finale in **Reti Di Calcolatori**

Cattura ed analisi del traffico cifrato generato da app mobili tramite man-in-the-middle proxy

Anno Accademico 2021/2022

Relatore
Ch.mo prof.Pescapé Antonio

Candidato
Paolo Russo
matr. N46002315

"A continuos effort - not strengh or intelligence- is the key that unlesashes our potential."

SIR WINSTON CHURCHILL

Ringraziamenti

Prima di procedere con la trattazione,vorrei dedicare qualche riga a tutti coloro che mi sono stati vicini in questo percorso di crescita personale e professionale.Ringrazio il mio relatore Prof.Pescapè Antonio,il mio correlatore l'Ingegnere Montieri Antonio e l'Ingegnere Guarino Idio,sempre pronto a darmi le giuste indicazioni in ogni fase della realizzazione dell'elaborato.Grazie a voi ho accresciuto le mie conoscenze e le mie competenze.Un ringraziamento speciale va ai miei genitori,a mio fratello ed a Anna Maria che mi sono sempre stati accanto,con l'infinita pazienza che li contraddistingue.Grazie per avermi sempre sostenuto e per avermi permesso di raggiungere questo traguardo.Se oggi sono riuscito a raggiungere questo obiettivo devo anche ringraziare tutti i miei parenti per il loro sostegno e incoraggiamento a non mollare mai.Una dedica speciale va ai miei amici,in particolare a Raffaele e Francesco che mi hanno sempre mostrato il loro supporto e vicinanza sopratutto in questo ultimo anno.L'affetto e il sostegno che mi hanno dimostrato rendono questo traguardo ancora più gratificante.Un ringraziamento va a Pasquale,amicizia nata per caso su un ring,persona leale e sincera che, nonostante la distanza che ci separa,c'è sempre stato.Un ringraziamento ai miei colleghi Angelo, Alessandro,Fabio,Guido e Marco per il percorso condiviso.Un caloroso saluto va all'Atletica Marcianise e al suo presidente Angelo Garofalo,che sin dall'inizio mi ha accolto in questa grande famiglia ed in particolare un abraccio al gruppo "del-

le ore 6:00" con il quale condivido volentieri allenamenti e momenti di spensieratezza.Ringrazio tutto lo staff dello studio ingegneristico Ingegneria &Energia,per l'ospitalità offerta in questi mesi.

Grazie di cuore a tutti voi.

INTRODUZIONE

Al giorno d'oggi,soprattutto dopo l'avvento del COVID-19,ogni aspetto della nosta vita può essere associato all'utilizzo di Internet [1]. La quotidianità dell'uomo è stata stravolta dalla pandemia,che ha cambiato la nostra vita, il nostro modo di lavorare,il nostro modo di interagire.Tutto ciò, di conseguenza, ha determinato un utilizzo sempre più imponenete di applicazioni web o servzi online e di applicazioni mobile, portando ad un aumento esponenziale del traffico di dati generato in rete.L'utilizzo di queste applicazioni, dunque, ha assunto, nel corso degli anni,specie durante il periodo di lockdown, una rilevanza sempre più importante nel nostro modo di vivere.Tutti questi servizi online memorizzano o trasferiscono le informazioni sensibili dell'utente, che rappresentano un obiettivo chiave per gli hacker.Oltre ai singoli individui,gli hacker mirano principalmente alle imprese e alle organizzazioni,con l'obiettivo di effettuare gravi danni economici.In questo nuovo mondo di "persone e cose sempre connesse" tramite Internet, è molto comune leggere di attacchi riusciti a dispositivi informatici e servizi online.Il nostro obiettivo principale è quello di acquisire e ana-

lizzare il traffico HTTPS generato durante l'uso di un app quando si accede ad Internet.Un' app perde informazioni personali quando queste vengono trasmesse a terze parti senza il consenso dell'utente.Per intercettare il traffico di rete generato da un app per smartphone, andremo ad utilizzare un server proxy che fungerà da Man-in-the-Middle (MitM),che rappresenta uno degli attacchi di maggior successo compiuti dagli hacker e si traduce nell'ottenere il controllo sugli utenti finali ed i dati da essi trasferiti. Prima di procedere alla presentazione dell'elaborato, è necessario fare una doverosa precisazione: nel rispetto delle normative sulla privacy(**DECRETO LEGISLATIVO 30 giugno 2003,n.196**), l'utilizzo e la trattazione dei dati, che hanno esclusivamente un fine scientifico, avvengono con il pieno consenso dell'utente.

Indice

Capitolo 1

MAN-IN-THE-MIDDLE

Man-In-The-Middle (MITM) è un tipo di attacco dove una terza parte malintenzionata prende segretamente il controllo del canale di comunicazione tra due o più endpoint.Mediante un attacco MITM è possibile intercettare,modificare,cambiare o sostituire il traffico di comunicazione delle vittime che,convinte che il canale di comunicazione risulti sicuro, non sono a conoscenza dell'intruso.L'attacco MITM può essere eseguito su diversi canali di comunicazione come GSM, UMTS,LTE, Bluetooth,NFC e Wi-Fi. L'obiettivo di questi attacchi non è solo quello di catturare i dati effettivi che fluiscono tra gli endpoint, ma anche la riservatezza e l'integrità dei dati stessi. In particolare, l'attacco MITM mira a compromettere:

- riservatezza, intercettando la comunicazione;
- integrità, intercettando la comunicazione e modificando i messaggi;

- disponibilità, intercettando e distruggendo messaggi o modificare i messaggi per estromettere dal canale di trasmissione una delle due parti comunicanti;

Possiamo identificare almeno tre modi di caratterizzazione degli attacchi MITM:

1. MITM basato su tecniche di imitazione;
2. MITM basato sul canale di comunicazione in cui viene eseguito l'attacco;
3. MITM basato sulla posizione dell'attaccante e della vittima all'interno della rete;

Nello specifico, tra le varie tipologie di attacco, possiamo considerare:

- Il MITM basato sullo **spoofing**, in cui l'attaccante intercetta una comunicazione legittima tra due host. Attraverso questo meccanismo l'hacker può manipolare e controllare i dati trasferiti, senza che le vittime siano a conoscenza di un intermediario esistente. In alcuni casi, come spoofing DNS, l'aggressore può falsificare i dispositivi tra le vittime, in altri casi, come ARP spoofing, l'aggressore falsifica direttamente i dispositivi della vittima.
- **SSL/TLS MITM**, che è una forma di intercettazione di rete attiva, in cui l'attaccante si inserisce nel canale di comunicazione tra due vittime, di solito il browser e server web; l'attaccante,

successivamente, stabilisce due connessioni SSL separate con ciascuna vittima e trasmette messaggi tra di loro, in modo tale che entrambi non conoscano l'intermediario. Questa configurazione permette all' attaccante di salvare o modificare tutti i dati trasmessi.

- **BGP MITM** è un attacco che si basa sul dirottamento IP, in cui il traffico rubato passa attraverso l'attaccante ad un terzo dispositivo, dove può essere manipolato.

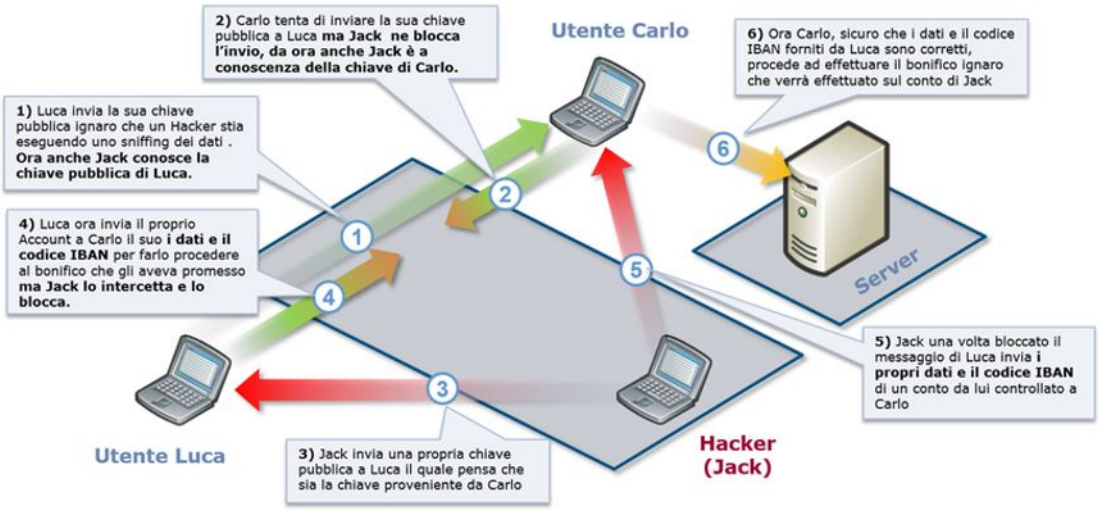
- MITM basato su false stazioni base detto **FBS-based**, è un attacco in cui una terza parte costringe la vittima a creare una connessione con una falsa stazione ricetrasmittente (BTS), che permette all'attaccante di manipolare il traffico delle vittime.

Un altro modo per classificare l'attacco MITM è considerare il canale di comunicazione in cui viene eseguito l'attacco. Nella seguente tabella elenchiamo gli attacchi MITM attraverso i livelli OSI e le reti cellulari. [2]

		MITM Attacks
OSI Layer	Application	BGP MITM, DHCP spoofing-based MITM, DNS spoofing-based MITM
	Presentation	SSL/TLS MITM
	Transport	IP spoofing-based MITM
	Network	
	Data Link	ARP spoofing-based MITM
Cellular networks	GSM	FBS-based MITM
	GSM/UTMS	

Ogni livello applica approcci diversi per garantire la sicurezza, tuttavia, nessuno di essi è esente da rischi di attacchi MITM, che trovano

terreno fertile grazie alla mancanza di autenticazione, dal momento che non è fattibile o pratico utilizzare l'autenticazione per ogni handshake, sessione e scambio di query/risposte, e allora ci sarà sempre un rischio di intercettazione del traffico. L'unica condizione attenuante è la distanza tra server e client per tali scambi. Uno scambio di query/risposte sulla stessa sottorete locale è uno scambio molto più sicuro rispetto ad uno con più hop. Indipendentemente dalla distanza che il traffico di dati percorre, il rischio di un attacco MitM è sempre presente.



Nel capitolo successivo andremo ad introdurre uno strumento fondamentale per poter simulare l'attacco Man-In-The-Middle.

Capitolo 2

MITMPROXY

Mitmproxy[3] è uno strumento open source utile per intercettare, ispezionare, modificare e riprodurre il traffico web come HTTP/1, HTTP/2, WebSocket e qualsiasi altro protocollo protetto da SSL/TLS. Mitmproxy può anche fungere da proxy TCP generico: in questa modalità, rileverà la presenza TLS all'inizio di una connessione ed eseguirà un attacco Man-in-the-Middle, se necessario, ma inoltrerà i messaggi senza modifiche.È possibile intercettare e decodificare una varietà di tipi di messaggi che vanno da HTML a Protobuf, dirottandoli su client o server specifici.Le principali caratteristiche sono:

- intercettare e decodificare le richieste e le risposte HTTP e HTTPS;
- salvare le conversazioni HTTP complete per riproduzioni e analizzarle successivamente;
- riprodurre il lato client e lato server di una conversazione HTTP;

- inoltro del traffico a uno specifico server;
- è possibile modificare tramite l'aggiunta di uno script Python il traffico HTTP;
- i certificati SSL/TLS per l'intercettazione vengono generati al volo;

L'ambiente mitmproxy mette a disposizione un insieme di front-end che espongono funzionalità comuni, dunque, quando parliamo di "mitmproxy" di solito ci riferiamo a uno qualsiasi dei tre strumenti elencati di seguito, che sono associati allo stesso proxy principale, pertanto:

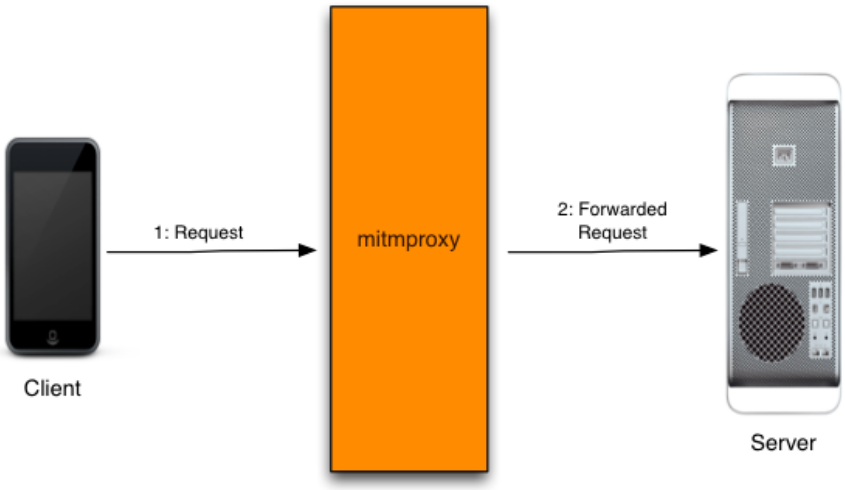
- **mitmproxy** è un proxy di intercettazione interattivo e di modifica del traffico HTTP. Tutti i flussi sono tenuti in memoria, il che significa che è possibile prelevare e manipolare flussi di piccole dimensioni da parte dell'utente;
- **mitmweb** è l'interfaccia utente basata sul web di mitmproxy.Tutti i flussi sono tenuti in memoria, quindi anche in questo caso, l'utente può prelevare e manipolare campioni di flussi;
- **mitmdump** è la versione da riga di comando di mitmproxy che fornisce funzionalità simili a tcpdump per consentire di visualizzare, registrare e trasformare a livello di codice il traffico HTTP.

2.1 Come lavora Mitmproxy

Mitmproxy è uno strumento estremamente flessibile, il cui meccanismo di funzionamento presenta vari modalità di implementazione.

2.1.1 Http Esplicito

Configurare il client per utilizzare mitmproxy come proxy esplicito è il modo più semplice e affidabile per intercettare il traffico. Il protocollo proxy è codificato nell'HTTP RFC , quindi il comportamento sia del client che del server è ben definito e in genere risulta affidabile.



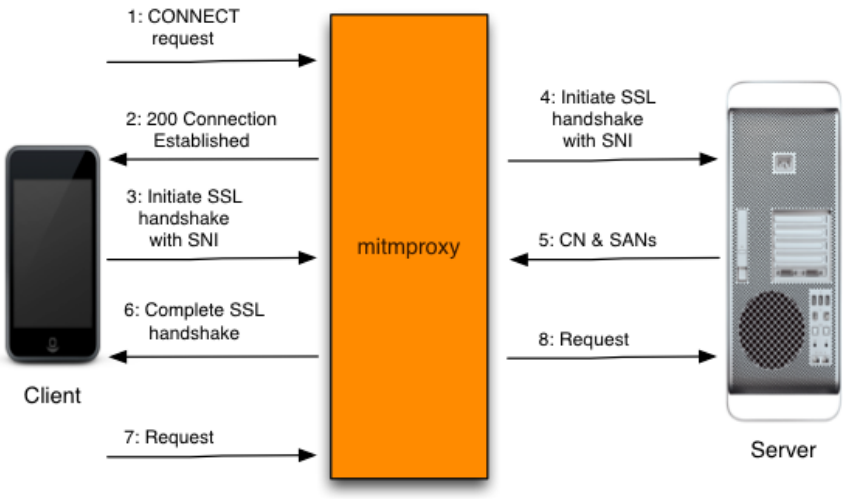
Il client si connette al proxy e fa una richiesta. Dal canto suo, Mitmproxy si connette al server upstream a cui inoltra semplicemente la richiesta.

2.1.2 Https Esplicito

Il processo per una connessione HTTPS con proxy esplicito è abbastanza diverso. Un proxy convenzionale non può né visualizzare né manipolare un flusso di dati crittografato con TLS; pertanto, una richiesta di CONNECT chiede semplicemente al proxy di aprire una pipe tra il client e il server; dopo di che il proxy inoltra alla cieca i dati in entrambe le direzioni senza sapere nulla dei contenuti.La negoziazione della connessione TLS avviene su questa pipe e il successivo flusso di richieste e risposte è completamente opaco al proxy.Nelle connessioni https,l'idea di base è simulare di essere il server del client e allo stesso tempo essere il client del server, mentre ci poniamo nel mezzo della decodifica del traffico da entrambi i lati.Sostanzialmente,si attua un processo d'implementazione **SSL/TLS MITM**.La parte difficile è capire come gestire l' autorità di certificazione, infatti, poiché il sistema è progettato per prevenire questi attacchi, consente a una terza parte fidata di firmare crittograficamente i certificati di un server per verificarne la legittimità.Se questa firma non corrisponde o proviene da una parte non attendibile, un client sicuro interromperà semplicemente la connessione e si rifiuterà di procedere, il che renderà impossibile al MITM accedere su una connessione TLS per la cattura. Tale problema viene risolto grazie alla capacità del Mitmproxy d'includere un'implementazione CA completa che genera certificati di intercettazione prontamente. Per fare in modo che il client conside-

ri attendibili questi certificati, registriamo manualmente mitmproxy come CA attendibile sul dispositivo che stiamo utilizzando, pertanto, abbiamo bisogno di conoscere il nome di dominio da utilizzare nel certificato di intercettazione: il client verificherà che il certificato sia generato per il dominio a cui si sta connettendo, altrimenti, in caso contrario, abortirà la connessione.Consideriamo una connessione del tipo: **CONNECT 10.1.1.1:443 HTTP/1.1**. L'utilizzo dell'indirizzo IP è perfettamente legittimo perché ci fornisce informazioni sufficienti per avviare la pipe, anche se non rivela il nome dell'host remoto. Mitmproxy ha un meccanismo astuto che attenua questo sniffing di certificati over-upstream, infatti, non appena arriva una richiesta di CONNECT, viene messa in pausa la parte client della conversazione e avviata una connessione simultanea al server, grazie alla quale è possibile terminare l'handshake TLS con il server ed ispezionare i certificati utilizzati.Ora,mediante il **Common Name (CN)** nei certificati upstream si può generare il certificato fittizio per il client, in questo modo si ottiene l'hostname corretto da presentare al client.Esistono dei casi in cui il nome comune del certificato non è, di fatto,il nome host a cui il client si sta connettendo.Ciò è possibile attraverso il campo opzionale **Subject Alternative Name (SAN)** all'interno del certificato che consente di specificare un numero arbitrario di domini alternativi.Se il dominio previsto corrisponde a uno di questi, il client procederà, anche se il dominio non corrisponde al certificato CN.Quando si estrae il CN

dal certificato upstream, si va a prelevare anche i SAN e aggiunti al certificato fittizio generato. Uno dei grandi limiti del protocollo TLS consiste nel fatto che ogni certificato richieda il proprio indirizzo IP, tale per cui non sarà possibile fare hosting virtuale laddove più domini con certificati indipendenti condividano lo stesso indirizzo IP.Come ben sappiamo, il pool di indirizzi IPv4 è in rapida diminuzione; per ovviare a questo problema, mitmproxy offre una soluzione attraverso il **Server Name Indication (SNI)** per i protocolli TLS. Ciò consente al client di specificare il nome del server remoto all'inizio dell'handshake TLS,lasciando che il server selezioni il certificato corretto per completare il processo.SNI interrompe il processo di sniffing dei certificati upstream quando ci si connette senza interpellare il server, servendo un certificato predefinito che potrebbe non avere nulla a che fare con il certificato previsto dal client. La soluzione prevista da Mitmproxy è quella di agire sul processo di connessione del client. In altre parole, dopo che il client si è connesso, viene consentito all'handshake TLS di continuare fino a quando il valore SNI non viene inviato, in questo modo possiamo mettere in pausa la conversazione e avviare una connessione upstream utilizzando il valore SNI corretto, che poi servirà il certificato upstream relativo, dal quale è possibile estrarre i CN e SAN previsti. Quindi nel caso di HTTPS esplicito, risulta:



1. il client effettua una connessione a mitmproxy inviando una richiesta HTTP CONNECT;

2. mitmproxy risponde con un **200 Connection Established**, come se avesse impostato la pipe CONNECT;

3. il client crede di parlare con il server remoto e avvia la connessione TLS. Utilizza SNI per indicare il nome host a cui si sta connettendo;

4. mitmproxy si connette al server e stabilisce una connessione TLS utilizzando il nome host SNI indicato dal client;

5. il server risponde con il certificato corrispondente, che contiene i valori CN e SAN necessari per generare il certificato di intercettazione;

6. mitmproxy genera il certificato di intercettazione e continua l'handshake TLS del client messo in pausa nel passaggio 3;

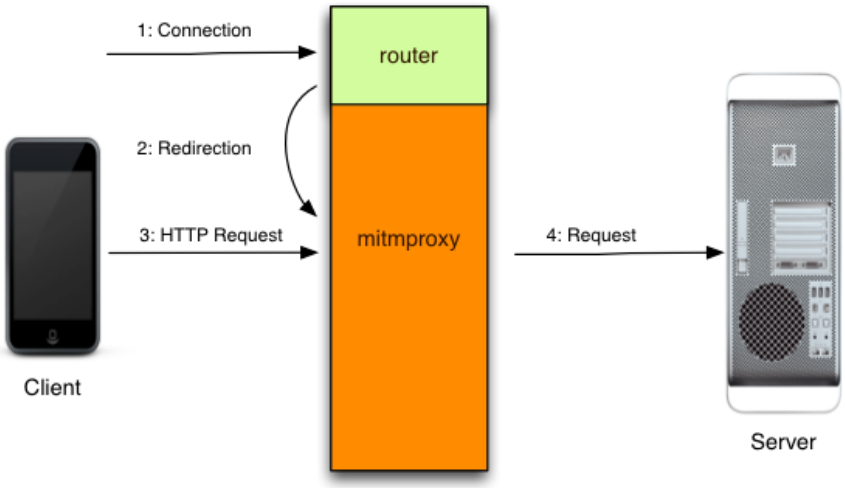
7. il client invia la richiesta tramite la connessione TLS stabilita;

8. mitmproxy passa la richiesta al server tramite la connessione TLS avviata nel passaggio 4.

2.1.3 Http Trasparente

Quando viene utilizzato un proxy trasparente, la connessione viene reindirizzata in un proxy a livello di rete, senza che sia richiesta alcuna configurazione del client. Ciò rende il proxy trasparente ideale per quelle situazioni in cui non è possibile modificare il comportamento del client: le applicazioni Android ignare del proxy sono un esempio comune. Introduciamo due componenti aggiuntivi: il primo è un meccanismo che reindirizza in modo trasparente una connessione TCP destinata a un server su Internet a un server proxy in ascolto, che di solito assume la forma di un firewall sullo stesso host del server proxy. Una volta che il client ha avviato la connessione, effettua una richiesta HTTP del tipo: **GET/index.html HTTP/1.1**. Questa richiesta differisce dalla variazione del proxy esplicito, in quanto omette lo schema e il nome host.È possibile capire a quale host inoltrare la richiesta grazie al meccanismo di routing,che permette di eseguire il reindirizzamento della richiesta e,quindi, tener traccia della destinazione originale. Ogni meccanismo di routing ha un modo diverso di esporre questi dati, dunque, introduciamo il secondo componente necessario per il funzionamento del proxy trasparente,ossia un modulo host che

sa come recuperare l'indirizzo di destinazione originale dal router.In mitmproxy, questo assume la forma di un insieme integrato di moduli che sanno come comunicare con il meccanismo di reindirizzamento di ciascuna piattaforma. Una volta che abbiamo queste informazioni, il processo risulta abbastanza semplice.Andando a schematizzare il tutto:



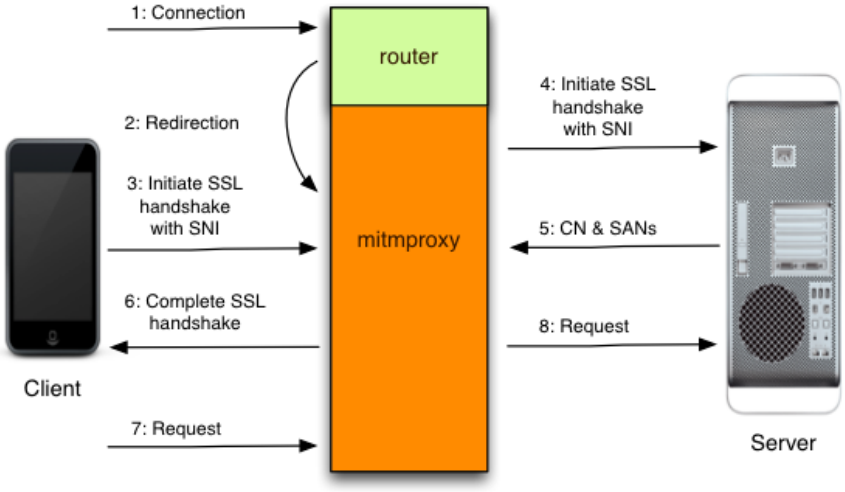
1. Il client effettua una connessione al server;

2. il router reindirizza la connessione a mitmproxy, che in genere è in ascolto su una porta locale dello stesso host.Mitmproxy consulta quindi il meccanismo di routing per stabilire quale fosse la destinazione originale;

3. infine,leggiamo semplicemente la richiesta del cliente per poi inoltrarlo a monte.

2.1.4 Https Trasparente

Per implementare questa modalità, usiamo il meccanismo di routing per scoprire qual è la porta di destinazione originale. Tutte le connessioni in entrata passano attraverso diversi livelli che possono determinare il protocollo effettivo da utilizzare. Il rilevamento automatico di TLS funziona per SSLv3, TLS 1.0, TLS 1.1 e TLS 1.2 con l'obiettivo di cercare un messaggio ClientHello all'inizio di ogni connessione.Funziona indipendentemente dalla porta TCP utilizzata. Da qui,il processo è una fusione dei metodi che abbiamo descritto per il proxy HTTP trasparente e il proxy HTTPS esplicito.



1. Il client effettua una connessione al server;

2. il router reindirizza la connessione a mitmproxy, che in genere è in ascolto su una porta locale dello stesso host.Mitmproxy consulta quindi il meccanismo di routing per stabilire quale fosse la destinazione originale;

3. il client crede di parlare con il server remoto e avvia la connessione TLS.Utilizza SNI per indicare il nome host a cui si sta connettendo;

4. mitmproxy si connette al server e stabilisce una connessione TLS utilizzando il nome host SNI indicato dal client;

5. il server risponde con il certificato corrispondente, che contiene i valori CN e SAN necessari per generare il certificato di intercettazione;

6. mitmproxy genera il certificato di intercettazione e continua l'handshake TLS del client messo in pausa nel passaggio 3;

7. il client invia la richiesta tramite la connessione TLS stabilita;

8. mitmproxy passa la richiesta al server tramite la connessione TLS avviata nel passaggio 4.

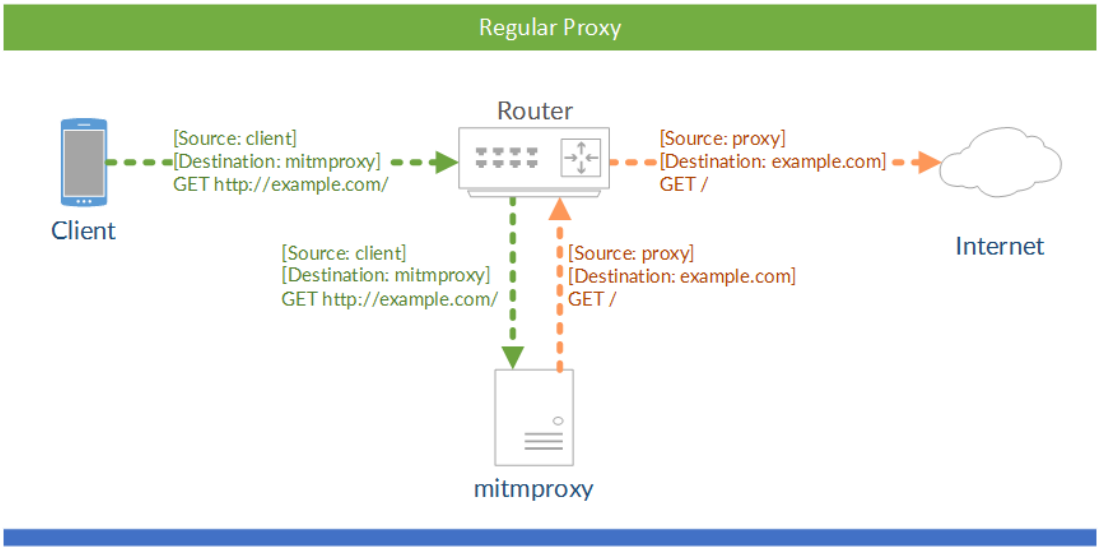
La modalità trasparente può essere esclusivamente implementata su distribuzioni Linux, attraverso il meccanismo di reindirizzamento iptables e distribuzioni OSX con pf. Nel caso di sottosistemi WLS, ciò non è implementabile in quanto WLS attualmente non supporta le interfacce del kernel Linux iptables.

2.2 Meccanismi di implementazione

Ora andiamo ad illustrare come avviene lo scambio dei dati tra i vari endpoint che contribuiscono al funzionamento dell'intero sistema Mitmproxy, in termine di pacchetti.

2.2.1 Proxy Regolare

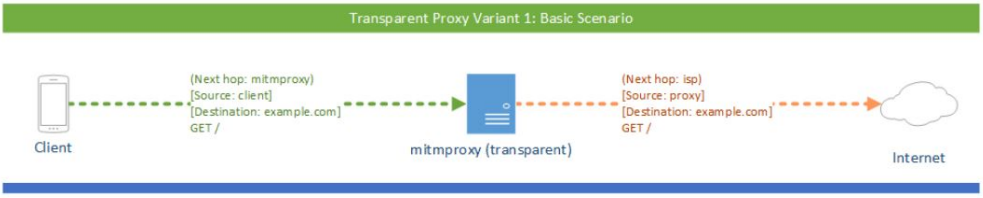
In questa modalità, dopo aver impostato ed avviato esplicitamente un Proxy HTTP, il client si connetterà al mitmproxy che a sua volta si connette esplicitamente al server di destinazione. I pacchetti, quindi, verranno reindirizzati dal ruoter al dispositivo su cui è avviato il mitmproxy, per poterli catturare ed eventualmente modificarli.



2.2.2 Proxy Trasparente

Come abbiamo visto,in questa modalità il traffico viene diretto in un proxy a livello di rete, senza che sia richiesta alcuna configurazione

del client. La macchina che esegue mitmproxy viene interposta tra il router e Internet.Pertanto:



Quando il pacchetto arriva alla macchina mitmproxy,deve ancora essere indirizzato al server di destinazione.Ciò significa che la traduzione degli indirizzi di rete non dovrebbe essere applicata prima che il traffico raggiunga il mitmproxy,poiché ciò rimuoverebbe le informazioni di destinazione,lasciando il mitmproxy incapace di determinare la reale destinazione.In questa modalità è possibile reindirizzare in modo trasparente una connessione TCP destinata inizialmente a un server in Internet, a un server proxy in ascolto. Quando il proxy riceve una connessione reindirizzata, vede una richiesta HTTP senza una specifica proveniente dal host, e allora mediante un nuovo modulo host è possibile interrogare il redirector per la destinazione originale della connessione TCP.

2.2.3 Proxy Socks

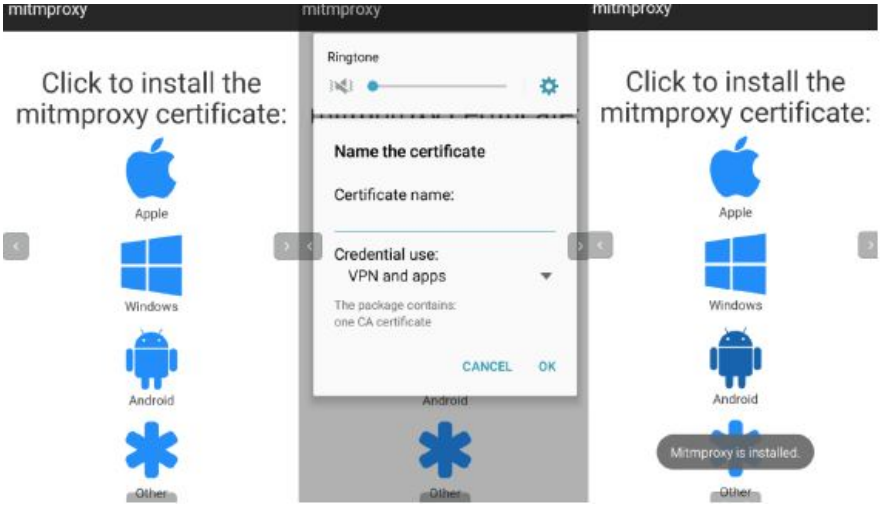
In questa modalità, mitmproxy funge da proxy SOCKS5.È simile alla modalità di proxy regolare, ma utilizza il protocollo SOCKS il cui acronimo è Socket Secure, comunemente usato per attività ad alta intensità di traffico, come lo streaming di contenuti o la condivisione

P2P.SOCKS utilizza una connessione TCP progettata per inviare e ricevere pacchetti di dati su Internet, nonché per garantire la corretta consegna delle risorse sulle reti.I proxy SOCKS, dunque, consentono a un dispositivo di inviare dati a un altro tramite un terzo dispositivo, chiamato server SOCKS o server proxy SOCKS, attraverso cui è possibile instaurare una connessione con un qualsiasi altro server dietro il firewall e scambiare pacchetti di rete tra il client e il server effettivo.Il traffico scorrerà attraverso un server proxy che assegna un indirizzo IP arbitrario prima di raggiungere la sua destinazione permettendo al client di agire nell'anonimato mascherando il proprio indirizzo IP, ma ciò non significa che il traffico sia sicuro.

Mentre i proxy HTTPS sono pensati per il protocollo HTTP/HTTPS, i proxy SOCKS sono di basso livello e possono gestire la maggior parte dei protocolli Internet supportando anche connessioni TCP e UDP rispetto a HTTPS che, invece, supporta solo TCP.

2.3 Certificato

Mitmproxy può decriptare il traffico purché il client si fidi dell'**autorità di certificazione (CA)** integrata di mitmproxy che deve essere installata nei dispositivi mobili, altrimenti il client può rifiutare l'handshake SSL/TLS e annullare la comunicazione. Per ogni server di destinazione crittografato SSL/TLS, il mitmproxy CA genera prontamente un certificato fittizio per impersonare il sito web visitato.La CA viene creata la prima volta che mitmproxy sarà stato avviato.I messaggi di avviso verranno attivati se il client riceve un certificato non attendibile che non è configurato per essere accettato.L'installazione del certificato CA nei dispositivi mobili può avvenire attraverso una configurazione rapida o manuale. Nel primo caso, è previsto un built-in dell'applicazione di installazione del certificato a cui è possibile accedere, utilizzando quello che viene chiamato dominio magico mitm.it,tramite un browser web [4]. Ecco un esempio delle tre fasi dell'installazione rapida di mitmproxy:



Il client seleziona l'icona che corrisponde al suo smartphone, permettendo l'installazione della CA.Alla fine dell'operazione lo smartphone conferma la corretta installazione.D'altra parte, la configurazione manuale viene eseguita attraverso un elenco di puntatori al certificato manuale, forniti dal mitmproxy. Scorrendo la documentazione sarà possibile capire come effettuare questo tipo d'installazione a seconda della piattaforma. In molte applicazioni per prevenire attacchi Man-in-the-Middle, è presente un meccanismo di sicurezza chiamato **certificate pinning**, utilizzato per autenticare l'identità di un server noto sul lato client con il quale un' applicazione instaura una connessione sicura ,al fine di impedire l'uso di certificati illegittimi. Ci sono due modi di eseguire il pinning del certificato: confronto tra certificati e confronto delle chiavi pubbliche. Nella primo caso, l'intero certificato del server viene confrontato con il certificato attendibile incluso nell' applicazione;nel secondo caso viene confrontata solo la chiave pubblica contenuta all'interno del certificato.Con il certificate pinning attivato, l'applicazione mobile non si accontenta che il certificato sia valido, bensì verifica puntualmente che la parte pubblica, ossia quella fornita dal server in fase di instaurazione della connessione,sia esattamente quella che si aspetta,di fatto non fidandosi semplicemente della lista di certificati installata sul dispositivo mobile.Ciò significa che i certificati di mitmproxy non possono essere accettati da queste applicazioni senza che esse siano modificate.Per intercettare le connessioni

bloccate,bisogna patchare l'applicazione manualmente.Per i dispositivi Android e iOS, esistono vari strumenti per farlo, quello che andremo ad introdurre e usare, sarà apk-mitm [5].

2.3.1 Apk-Mitm

È una Comand Line Applications [6] ossia un programma attraverso il quale è possibile interagire interamente attraverso terminale o shell. Con il seguente comando **apk-mitm <app.apk> - - certificate <certificato.pem>**,andremo a generare un file app-patched.apk, per rendere possibile l'ispezione e l'analisi HTTPS. Apk-mitm automatizzerà l'intero processo, pertanto ogni file sarà sottoposto al seguente processo:

- Decodifica del file APK mediante Apktool:in questa fase,andiamo a sottoporre il file apk all'operazione di reverse engineering, in cui si va a decodificare il file apk da cui è possibile rigenerare il relativo codice sorgente.

- Aggiunta del certificato:in questa fase si va ad agire sul codice,le varie implementazioni di pinning del certificato vengono disabilitate e si va a modificare la configurazione della sicurezza di rete dell'app per consentire l'aggiunta del certificato richiesto dall'utente.La funzionalità di configurazione della sicurezza di rete permette alle app di personalizzare le proprie impostazioni di

sicurezza di rete in un file di configurazione sicuro e dichiarativo senza alterare il principio di funzionamento dell'app.Queste impostazioni possono essere configurate per domini specifici. La piattaforma Android fornisce un nuovo e semplice strumento per gestire la configurazione di rete: **Network Security Configuration (NSC)**, che è disponibile da Android 7.0. Con NSC, è possibile dichiarare metodi di comunicazione sicuri,utilizzando un file XML.Si verifica la modifica del file Manifest.xml dell'app a cui viene associato un ulteriore file di configurazione xml, grazie all'attributo networkSecurityConfig presente nel tag dell'applicazione. Pertanto,il Manifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application android:networkSecurityConfig="@xml/nsc_mitm"
                 ... >
        ...
    </application>
</manifest>
```

Il file nsc_mitm.xml presenterà un implementazione:

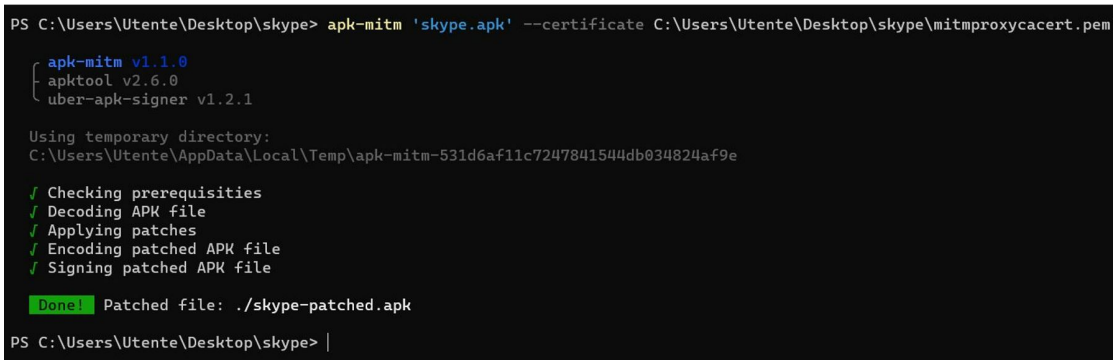
```
<?xml version="1.0" encoding="utf-8"?>
<!-- Intentionally lax Network Security Configuration (generated by apk-mitm) -->
<network-security-config>
  <!-- Allow cleartext traffic -->
  <base-config cleartextTrafficPermitted="true">
    <trust-anchors>
      <!-- Allow user-added (proxy) certificates -->
      <certificates src="user" />
      <certificates src="@raw/mitmproxycacert" />
      <certificates src="system" />
    </trust-anchors>
  </base-config>
</network-security-config>
```

Attraverso questo meccanismo andiamo a rendere visibile il traffico dei dati settando true l'attributo **cleartextTrafficPermitted**.Con il parametro **user**, invece, andiamo a definire l'aggiunta del **certificato**

utente, seguito dal riferimento. In questo modo sarà possibile visionare in chiaro il traffico sul mitmproxy, senza essere rilevati come una possibile minaccia.

- Codifica del file APK con patch mediante Apktool:in questa fase si va a riassemblare il codice sorgente in un apk,ottenendo un file del tipo app-patched.apk;

- Firma app-patched con il tool uber-apk-signer: in quest'ultima fase, quindi, si va ad utilizzare uno strumento che aiuta a verificare, ottimizzare e firmare APK con il certificato di rilascio fornito dall'utente.



Capitolo 3

PRINCIPIO DI FUNZIONAMENTO

Al termine del processo di modifica,dopo aver disattivato il certificate pinning e aggiunto il certificato associato al Mitmproxy , saremo pronti per avviare il sistema andando ad eseguire un vero attacco MitM alla nostra app, mediante un server proxy, con l'obiettivo di effettuare le catture necessarie per analizzare il traffico generato dalla nostra app.Questo processo sarà costituito da diverse fasi che vanno dalla cattura dei traffici fino all'analisi, passando per assemblaggio ed elaborazione.

3.1 Processo di Cattura

Andiamo a definire un file bash che chiameremo StartMitm.sh, costituito:

SSLKEYLOGFILE="$PWD/.script_bash/sslkeylogfile.txt" mitmproxy –mode socks5 –listen-port 8080

In base alle considerazioni fatte in precedenza, consideriamo il mitmproxy in modalità Proxy server SOCKS5,in ascolto sulla porta 8080.Bisogna effettuare una particolare osservazione al campo SSLKEYLOGFILE: le chiavi principali SSL/TLS possono essere registrate da mitmproxy in modo che i programmi esterni possano decrittografare le connessioni SSL/TLS sia da che verso il proxy.Le versioni recenti di Wireshark [7] possono utilizzare questi file di registro per decrittografare i pacchetti.Abilitiamo Il key logging impostando la variabile di ambiente in SSLKEYLOGFILE in maniera tale che punti a un file di testo scrivibile,che abbiamo chiamato sslkeylogfile.txt.Questo file prende il nome di master-secret il quale abilita la decriptazione TLS in Wireshark e può essere fornito tramite il file di registro delle chiavi. Il pre-master secret è il risultato dello scambio di chiavi e può essere convertito in un master secret sempre da Wireshark.Il pre-master secret può essere ottenuto quando viene fornita una chiave privata RSA ed è in atto uno scambio di chiavi RSA.Per capire come si ottiene e come funziona facciamo un passo indietro: per avviare una nuova connessione TLS [8] ,il client e il server si scambiano Messaggi "Hello" che

contengono una sequenza di byte casuali che chiameremo client random (CR) per quanto riguarda il client, si tratta di una sequenza di 32 byte univoca per ogni connessione e dovrebbe contenere un timestamp di quattro byte seguito da 28 byte casuali, mentre, da parte del server, verrà generato un server random (SR), che presenterà le stesse caratteristiche del client random.Dopo i messaggi "Hello", ulteriori dettagli possono essere scambiati, compresi i certificati digitali a seconda della suite di crittografia selezionata. Inoltre, entrambe le parti generano un pre-master secret (PS) utilizzando l'algoritmo di scambio di chiavi come,per esempio, il Diffie Hellmann specificato nella suite di cifratura.PS, CR e SR vengono utilizzati per calcolare il master secret (MS), il quale non viene mai trasferito da un end-point all'altro, bensì, per ogni sessione, TLS crea un master secret lato client e server e ne ricava, per ognuno, quattro chiavi per l'hashing e crittografia. Pertanto:

- il client utilizza la prima chiave per calcolare il MAC per ogni messaggio in uscita. Il server utilizza la stessa chiave per convalidare il MAC di tutti i messaggi in arrivo dal client;

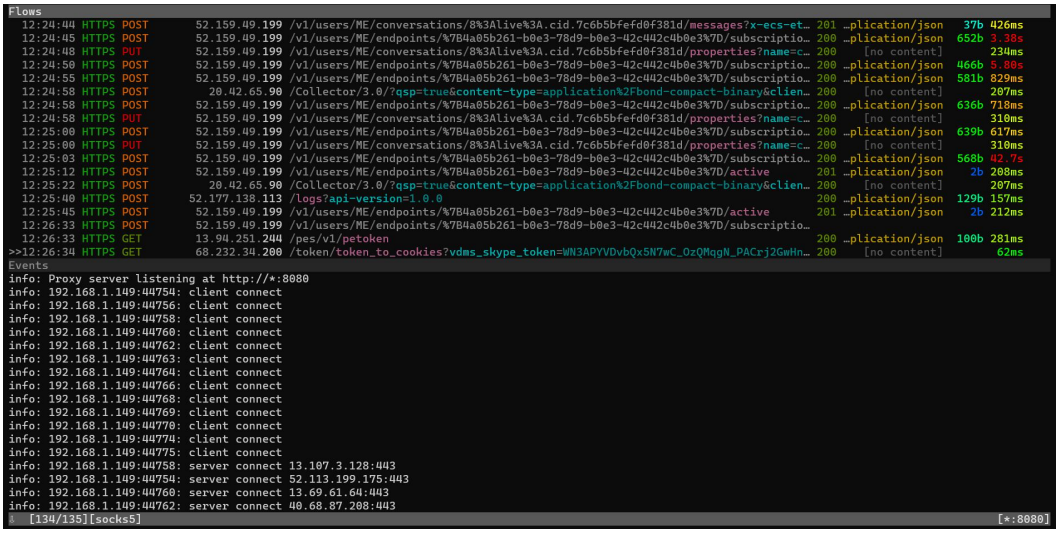
- il server usa la seconda chiave per calcolare il MAC per ogni messaggio in uscita. Il client utilizza la stessa chiave per convalidare il MAC di tutti messaggi in arrivo dal server;

- il client utilizza la terza chiave per crittografare in uscita messaggi e il server utilizza la stessa chiave per decrittografare tutti

i messaggi in arrivo;

- il server utilizza la quarta chiave per crittografare i messaggi in uscita e il client utilizza la stessa chiave per decrittografare tutti i messaggi in arrivo;

Al termine del Handshake TLS, MS, CR e SR sono utilizzati con una funzione pseudo-casuale (PRF) da parte del client e del server per derivare la stessa chiave di sessione, che rappresenta una chiave per una cifratura simmetrica e consente al client e al server di crittografare i messaggi tra loro.Pertanto, da questo momento in poi, qualsiasi comunicazione successiva tra le due parti verrà crittografata con la chiave di sessione.Inoltre, è necessario ricordare che le chiavi sono "effimere", quindi sono adatte per una sola sessione TLS, ovvero il registro di una sessione non è idoneo per decifrare il traffico di un'altra sessione.Dopo queste doverose osservazioni,per avviare la cattura del traffico generato dall' applicazione che andremo ad analizzare ovvero Skype,bisogna innanzitutto configurare l'app PcapDroid, con il nostro MitM, andando, quindi,sulle impostazioni di PcapDroid, abilitiamo la modalità Proxy ed andiamo a determinare l'indirizzo IP e la porta del Proxy, che corrisponde al calcolatore sui cui lanceremo lo script bash.Lanciando da terminale, il file bash ./StartMitm.sh:



Un flusso è definito come una sequenza di pacchetti IP che condividono gli stessi indirizzi IP di sorgente e di destinazione. Ogni flusso scambiato tra la sorgente, nonché il nostro dispositivo mobile, su cui stiamo lanciando la nostra app, e destinatario o viceversa, verrà riportato sul nostro MitM.

{"eventMessages":[{"id":1349,"type":"EventMessage","resourceType":"NewMessage","time":"2021-09-27T16:35:22Z","resourceLink":"https://azwus1-client-s.gateway.messenger.live.com/v1/users/ME/conversations/8:live:.cid.7c6b5bfefd0f381d/messages/1632760522279","resource":{"ackrequired":"https://azwus1-client-s.gateway.messenger.live.com/v1/users/ME/conversations/ALL/messages/1632760522279/ack","type":"Message","from":"https://azwus1-client-s.gateway.messenger.live.com/v1/users/ME/contacts/8:live:.cid.1fc6afbd01519b00","clientmessageid":"8780957228405555601","receiverdisplayname":"Francesco Russo","version":"1632760522279","messagetype":"RichText","counterpartymessageid":"1632760522279","imdisplayname":"Paolo Russo","content":"Ciaooooooo","composetime":"2021-09-27T16:35:22.270Z","origincontextid":"0","originalarrivaltime":"2021-09-27T16:35:22.270Z","threadtopic":"live:.cid.1fc6afbd01519b00,
live:.cid.7c6b5bfefd0f381d","contenttype":"text","conversationLink":"https://azwus1-client-s.gateway.messenger.live.com/v1/users/ME/conversations/8:live:.cid.7c6b5bfefd0f381d","isactive":true,"id":"1632760522279"}}]}

Come possiamo notare, grazie ai procedimenti eseguiti, saremo in grado di visualizzare la struttura dei traffici HTTPS ed i contenuti dei messaggi scambiati tra sorgente e destinatario. Durante la fase di cattura, mentre sul nostro dispositivo mobile verranno generati dei file pcap, che in seguito andremo ad analizzare, sul nostro calcolatore, per

ogni flusso, il nostro master secret inizierà a riempirsi, presentando una struttura del tipo:

```
CLIENT_RANDOM 636b9cbf3cb8cc8923690b6ea8f7ce6e5490986c7a3ce07d624
023e1605922a93c256e48e797a5fc4b4fdfc1fdf847fa366d7332de5f5e313169
CLIENT_HANDSHAKE_TRAFFIC_SECRET 25bfecf534f21ea196ed0547f5149f3b7
ea7b34ceb97c0311573a51a2649184d5f5d72e441291be9d628595793fc11d38
SERVER_HANDSHAKE_TRAFFIC_SECRET 25bfecf534f21ea196ed0547f5149f3b7
3d4f3ce944c1b81b960bf4728bcaef893730f5bbdc8a0d5901f6b42a5504dc54
CLIENT_TRAFFIC_SECRET_0 25bfecf534f21ea196ed0547f5149f3b7c42cab04
cdc8fe421ba9b6db476635b65f4ba27d36c178b4d50a835fb7f2c16e4cdb45ff
SERVER TRAFFIC SECRET 0 25bfecf534f21ea196ed0547f5149f3b7c42cab04
```

Al termine della cattura, abbiamo generato il traffico di rete, che andiamo a decriptare mediante l'uso del master secret.Il nostro obiettivo è quello di effettuare un matching tra più catture appartenenti alla stessa app, ma che si differenzino a seconda dell'attività ,della granularità e dello stream. [9].Pertanto,andremo ad effettuare trenta catture, di cui dieci riguardano come attività i mesaggi, dieci riguardano le call e le altre dieci le videocall.

3.2 Processo di Assemblaggio

Dopo la fase di cattura, andiamo ad assemblare e decriptare i singoli file pcap mediante tshark, che è una versione orientata al terminale di Wireshark, un packet sniffer o analizzatore di protocollo, ovvero un software per catturare ed interpretare i dati che transitano su una rete; esso consente l'analisi in dettaglio dei pacchetti in ingresso e in uscita per ogni livello della pila dei protocolli. Nel nostro caso ci focalizzeremo sull'osservazione dei pacchetti TLS/SSL; quindi, andiamo a definire un

ulteriore file bash che chiameremo PcapAss.sh:

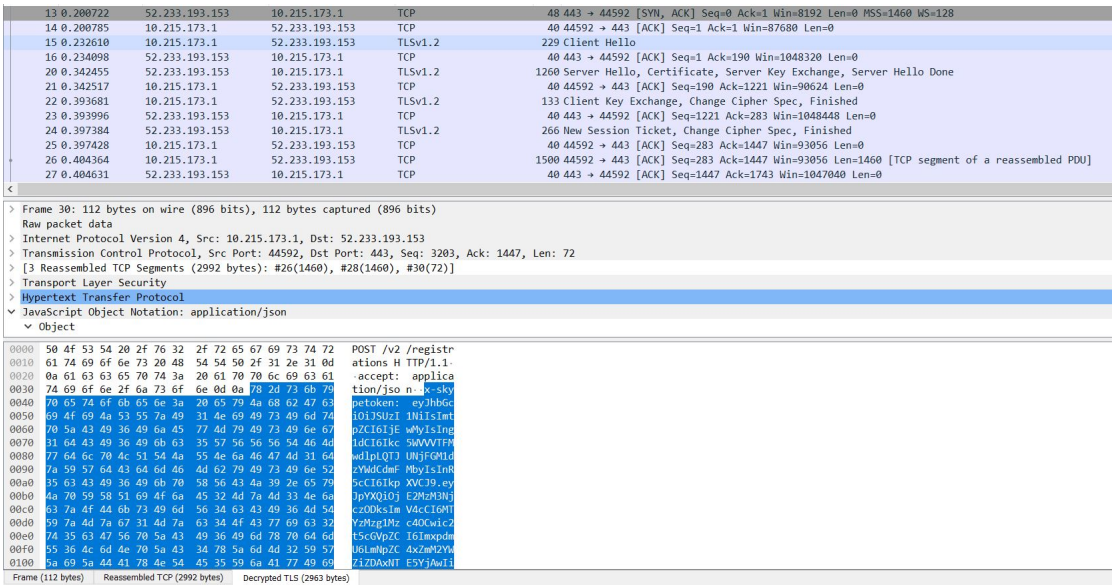
```
tshark -r $catt/*.pcap* -V -o "tls.debug_file:$catt/Analisi/ssldebug.log" -o
"tls.desegment_ssl_records: TRUE" -o "tls.desegment_ssl_application_data: TRUE" -o
"tls.keys_list:C:/Users/Utente/.script_bash/sslkeylogfile.txt" -F pcap -w
$catt/Analisi/$NameAss.pcap
```

Per ogni file pcap andiamo ad assemblare e decifrare tutti i pacchetti. Analizzando lo specifico comando di tshark ,per ogni file pcap processato, andiamo a riassemblare i dati dell' applicazione TLS/SSL,che si estendono su più record TLS/SSL. Affinché tale operazione vada a buon fine è fondamentale fornire il riferimento al master secret, generato, durante la fase di cattura, dal server proxy. Il tutto verrà riscritto su un nuovo file pcap. Al termine della fase di assemblaggio ,aprendo un file pcap con Wireshark , possiamo osservare una lista principale di pacchetti da cui possiamo approfondire il traffico SSL/TLS. Infatti,possiamo selezionare qualsiasi pacchetto con il protocollo SSL/TLS e selezionare la funzione "Segui flusso SSL/TLS" nel menù contestuale attraverso il quale sarà possibile visualizzare il traffico decrittografato rappresentato di seguito:

```
POST /v2/registrations HTTP/1.1
accept: application/json
x-skypetoken:
eyJhbGciOiJSUzI1NiIsImtpZCI6IjEwMyIsIng1dCI6Ikc5WVVVTFMwdlpLQTJUNjFGM1dzYWdCdmFMbyIsInR5cCI6IkpXVCJ9.eyJpYXQiOjE2MzM3NjczODksImV4cCI6MT
YzMzg1Mzc4OCwic2t5cGVpZCI6ImxpdmU6LmNpZC4xZmM2YWZiZDAxNTE5YjAwIiwic2NwIjo5NTYsImNzaSI6IjE2MzM3NjczODgiLCJjaWQiOiIxZmM2YWZiZDAxNTE5YjAwI
iwiYWF0IjoxNjMyNDcyNTk3fQ.T3ND4gy77rdw9ZQ9-b6_nheb6oyl-OOiMeMLVpFpxuOLps69SNY0yiUQqf-
lLnp4tlQSHBvODzAOXoPGdwJ35Ng4q9gaq1oXr6BBwE1Kusq2Mms3nmGLrvnrxqp0qGSrsR0dT3k-1Z95lWAqa6YnrnPrWJhrCkBoBvDM_9Q-
aEzET47zs5sAdwVi_UXGCW1Vhe98llLWXJ3edP5RdGyT_oje-
bui1KRClkliEfkxwnZL97oP7urVNVKiClsDx2Hg8weZ5IXkLjTT_yn1ZdRcI6q4rVnWVDLzH7bmMykkbD1dzBT5SpxjRJFqRsYjC3mBD0H_0zpx22sUwvemh0ja59w
Content-Type: application/json
Content-Length: 2088
Host: prod.registrar.skype.com
Connection: Keep-Alive
Accept-Encoding: gzip
User-Agent: okhttp/3.12.1

{"registrationId":"4a05b261-b0e3-78d9-b0e3-42c442c4b0e3","nodeId":"","clientDescription":
{"appId":"com.skype.raider","platform":"android","languageId":"it-IT","templateKey":"com.skype.raider:
8.14","platformUIVersion":"1419/8.75.0.140/"},"transports":{"FCM":[{"ttl":
15552000,"context":"NGCS2S","creationTime":"","path":"fVDdpvT2TQeEvwvlx_pTK5:APA91bG2S-
Vc7h9e2IwWF2KlFDRxrpotMbXMdYDmpEJYIEFAhU2RltktL8TTdpU_XxNIRX27sVpHXegCZRUzqWfV6SbzmNyokmJyx1l9g7KZQkN9pmG1vglny5X3yi6GN_Ewx5y3A78N"},
{"ttl":15552000,"context":"NGCGVC","creationTime":"","path":"fVDdpvT2TQeEvwvlx_pTK5:APA91bG2S-
Vc7h9e2IwWF2KlFDRxrpotMbXMdYDmpEJYIEFAhU2RltktL8TTdpU_XxNIRX27sVpHXegCZRUzqWfV6SbzmNyokmJyx1l9g7KZQkN9pmG1vglny5X3yi6GN_Ewx5y3A78N"},
{"ttl":15552000,"context":"NGCL2S","creationTime":"","path":"fVDdpvT2TQeEvwvlx_pTK5:APA91bG2S-
Vc7h9e2IwWF2KlFDRxrpotMbXMdYDmpEJYIEFAhU2RltktL8TTdpU_XxNIRX27sVpHXegCZRUzqWfV6SbzmNyokmJyx1l9g7KZQkN9pmG1vglny5X3yi6GN_Ewx5y3A78N"},
{"ttl":15552000,"context":"","creationTime":"","path":"fVDdpvT2TQeEvwvlx_pTK5:APA91bG2S-
Vc7h9e2IwWF2KlFDRxrpotMbXMdYDmpEJYIEFAhU2RltktL8TTdpU_XxNIRX27sVpHXegCZRUzqWfV6SbzmNyokmJyx1l9g7KZQkN9pmG1vglny5X3yi6GN_Ewx5y3A78N"},
{"ttl":15552000,"context":"NGC_CAST","creationTime":"","path":"fVDdpvT2TQeEvwvlx_pTK5:APA91bG2S-
Vc7h9e2IwWF2KlFDRxrpotMbXMdYDmpEJYIEFAhU2RltktL8TTdpU_XxNIRX27sVpHXegCZRUzqWfV6SbzmNyokmJyx1l9g7KZQkN9pmG1vglny5X3yi6GN_Ewx5y3A78N"},
{"ttl":15552000,"context":"NGC_STOP_RINGER","creationTime":"","path":"fVDdpvT2TQeEvwvlx_pTK5:APA91bG2S-
Vc7h9e2IwWF2KlFDRxrpotMbXMdYDmpEJYIEFAhU2RltktL8TTdpU_XxNIRX27sVpHXegCZRUzqWfV6SbzmNyokmJyx1l9g7KZQkN9pmG1vglny5X3yi6GN_Ewx5y3A78N"},
{"ttl":15552000,"context":"NGCP2S","creationTime":"","path":"fVDdpvT2TQeEvwvlx_pTK5:APA91bG2S-
Vc7h9e2IwWF2KlFDRxrpotMbXMdYDmpEJYIEFAhU2RltktL8TTdpU_XxNIRX27sVpHXegCZRUzqWfV6SbzmNyokmJyx1l9g7KZQkN9pmG1vglny5X3yi6GN_Ewx5y3A78N"},
{"ttl":15552000,"context":"NGC_WARM_PUSH","creationTime":"","path":"fVDdpvT2TQeEvwvlx_pTK5:APA91bG2S-
Vc7h9e2IwWF2KlFDRxrpotMbXMdYDmpEJYIEFAhU2RltktL8TTdpU_XxNIRX27sVpHXegCZRUzqWfV6SbzmNyokmJyx1l9g7KZQkN9pmG1vglny5X3yi6GN_Ewx5y3A78N"}]}}
HTTP/1.1 202 Accepted
Content-Length: 0
Server: Microsoft-HTTPAPI/2.0
Strict-Transport-Security: max-age=31536000; includeSubDomains
X-Content-Type-Options: nosniff
MS-CV: 1oXGrZBcJ0+gjucZ+C7XzQ.0
Date: Sat, 09 Oct 2021 09:59:25 GMT
```


In pratica, andiamo a ricreare l'intero scambio. E' presente una scheda "Dati SSL/TLS decrittografati" aggiuntiva per i dati dell'applicazione in cui i pacchetti SSL/TLS mostrano anche i contenuti decifrati. Possiamo riscontrare che i dati decifrati non sembrano immediatamente traffico HTTPS. Ciò è dovuto al fatto che Wireshark sta decifrando rigorosamente il traffico TLS/SSL e non esegue l'ispezione dei dati associati ad un protocollo aggiuntivo [10].



3.3 Processo di Elaborazione

Nella fase successiva,per ogni file pcap generato, andiamo ad estrarre delle features, mediante Scapy [11], un tool scritto in Python, grazie a cui è possibile assemblare o decodificare i pacchetti di un ampio numero di protocolli, inviarli, catturarli, filtrare richieste e risposte, e molto altro ancora.Il nostro obiettivo è quello di racchiudere in unico file CSV, tutte le features, associate ai singoli biflussi, estratte all' interno di ogni file pcap assemblato e decifrato.Un generico pacchetto di rete in Scapy viene rappresentato in layers separati, secondo una struttura piramidale, nel rispetto della pila ISO/OSI e ciascuno di essi viene rappresentato da un insieme di attributi che definiscono i parametri associati a quel livello. Infatti , considerando la figura sottostante:

Per ogni pacchetto relativo allo specifico file pcap selezionato andiamo a definire una tupla, in cui andiamo ad estrarre:

- Indirizzo Ip sorgente e destinatario;
- Type Strem, in cui andiamo a definire se la comunicazione è di

tipo UpStrem o DownStream;

- Il tipo di protocollo, utilizzato nel livello di trasporto, quindi TCP o UDP;
- porto sorgente;
- porto destinatario;
- la dimensione effettiva del pacchetto;
- inter-arrival-time(iat) packet, calcolandolo come la differenza tra il timestamp del pacchetto successivo e il pacchetto precedente. Chiaramente, poichè siamo interessanti alla direzione dello stream, questo parametro andrà calcolato in maniera indipendete in funzione del UpSteam che del DownStream;
- iat message, calcolandolo come la differenza tra il timestamp del messaggio successivo e il messaggio precedente.Anche in questo caso , valgono le considerazioni fatte nel punto precedente.
- la dimensione del contenuto decifrato, relativo ai dati effettivi generati dall'applicazione durante la cattura.

Chiaramente, a questa aggregazione di dati, andremo ad aggiungere ,il nome dell'app, l'attività e l'indice correlato alla cattura.Questa operazione verrà estesa per ogni pcap.Ora, partendo da questi dati "grezzi", bisogna raffinarli, ecco che subentra la libreria pandas [12], che consiste in un package Python che fornisce strutture dati veloci, flessibili

ed espressive, progettate per rendere il lavoro con dati "relazionali" o "etichettati" facile ed intuitivo.Attraverso la libreria pandas, è possibile effettuare molteplici attività, nel nostro caso,ci concentreremo sull'uso della classe DataFrame, che fornisce una struttura dati tabulare bidimensionale, potenzialmente eterogenea, con assi etichettati in righe e colonne.Partendo dalla lista di tuple appena generata, andiamo a creare il nostro DataFrame, caricando il set di dati.Da questo momento in poi, il DataFrame appena generato, lo possiamo vedere come un grande contenitore di dati,che andremo a salvare all'interno di un file csv.In questo modo otteniamo una visione in termini di pacchetto per i vari flussi relativi ad ogni singolo pcap. La classe DataFrame permette di selezionare le righe in base a valori di una o più colonne. Possiamo anche ottenere righe da DataFrame che soddisfano o non soddisfano una o più condizioni, tramite il metodo di indicizzazione booleana, che prevede, inizialmente,la creazione di una maschera che consiste in un set di valori booleani che rappresentano se la colonna contienel'elemento specifico o meno.In questo modo, applicando questa maschera al DataFrame di partenza mi permette di geneare un nuovo DataFrame filtrato il quale contiene solo le righe che hanno lo specifico valore per la colonna che stiamo considerando.Ad ogni ciclo, generiamo una cattura, cx(c1,c2,c3. . .),che andiamo a filtrare all'interno del nostro DataFrame; nel caso in cui non c'è corrispondenza,vuol dire che sono state prelevate tutte le catture precedenti e/o non sono presenti

catture correlate.In caso contrario, filtriamo il nostro DataFrame in funzione della specifica cattura cx.Poiché in ogni cattura sono presenti più biflussi che si ripetono,li comprimiamo in unico set di dati.Quindi, grazie ai metodi offerti dalla classe DataFrame, andiamo ad estrarre solo i biflussi univoci.Di seguito, per ogni biflusso univoco,andiamo a prelevare le informazioni sul nome dell' app e sul tipo di attività, e lo andiamo a filtrare con il DataFrame contenente esclusivamente il set di valori associati alla singola cattura cx. Ogni feature prelevata verrà aggiunta in un'apposita lista. Al termine del filtraggio associato allo specifico biflusso univoco, tutte le features associate insieme al nome dell' app ,l'attività correlata, l'indice di cattura cx ed il biflusso stesso, convergono in un nuovo set di dati compattati.Questo ragionamento verrà iterato per ogni singolo biflusso. Dopodichè andiamo a sovrascrivere il set di dati ricavato all'interno di un nuvo DataFrame,che salviamo come file CSV,cosicché ricaviamo una visione più dettagliata dei flussi,che ci consente di ricostuire per ogni singola coppia univoca sorgente destinatario l'intero scambio.

3.4 Processo di Analisi

In quest'ultima fase siamo pronti per analizzare i dati delle varie catture, contenute all'interno del file CSV in cui abbiamo una visone dei dati raccolti in termine di pacchetto, elaborate e generate nelle fasi precedenti.Consideriamo l' analisi del traffico in funzione delle

singole attività e dell'app.Pertanto,possiamo considerare tre modalità operative:

1. Ragruppiamo tutti i biflussi in maniera generica senza filtrare la direzione relativa allo stato dello stream che caratterizza il verso di percorrenza della comunicazione tra gli endpoint;

2. Ragruppiamo tutti i biflussi in maniera tale da considerare la direzione in funzione dell' UpStream che caratterizza il verso di percorrenza della comunicazione dal client al server, definendo in questo modo l'operazione di upload;

3. Definiamo l'operazione di download, in cui il verso di percorrenza della comunicazione è dal server al client, ragruppando tutti i biflussi in maniera tale da considerare la direzione in funzione del DownStream;

Le tre modalità appena elencate, possono essere implementate grazie all'ausilio di Pandas DataFrame e di seaborn [13],libreria usata per creare grafici ai fini statistici in Python, con l'obiettivo di migliorare e velocizzare l'analisi esplorativa dei dati. Fornisce un'interfaccia di alto livello costruita su Matplotlib e si integra perfettamente con le strutture di dati pandas.Una volta fornito il set di dati, caricato su un dataFrame pandas, specificata la traccia da realizzare, seaborn mappa automaticamente i valori dei dati e calcola internamente la mappatura semantica e l'aggregazione statistica necessarie per produrre traccia-

ti informativi.Nel nostro caso, andremo a plottare le features sotto forma di funzione di probabilità cumulata nota come **CDF**.Seaborn , pertanto ,permette l'uso della funzione ecdf [14],dove in tutti e tre i casi,il plottaggio avviene fissando una alla volta le singole features in funzione del campo attività, a cui si aggiunge il plotaggio relativo ad una panoramica generale delle feauters in funzione dell' app e non dell'attività.Come risultato finale,avremo i segunti grafici associati alle relative feauters:

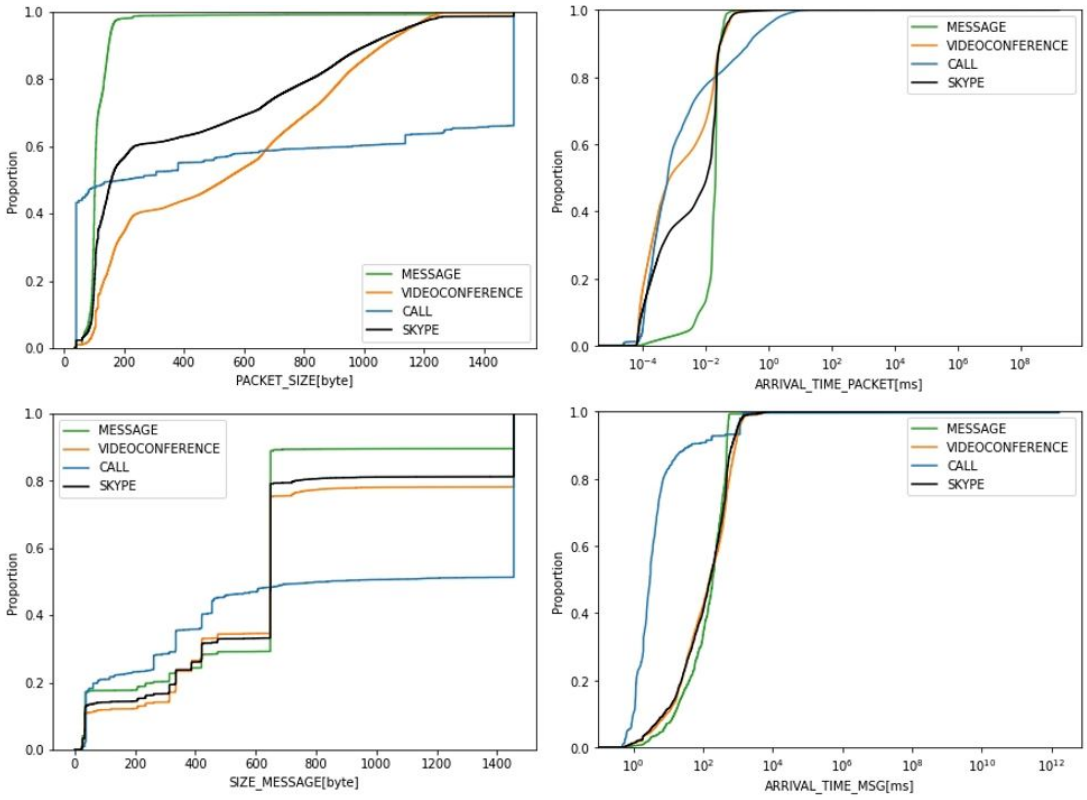


Figura 3.1: Grafici senza considereare la direzione dello stream

Questo primo gruppo di grafici rappresenta la nostra analisi qualitativa senza considerare la direzione dello stream. L'attività che presenta pacchetti con dimensioni maggiori rispetto alle altre attività,è

quella dei messaggi.Per quanto riguarda l'iat packet in tutti e quattro le attività vengono generati valori molti piccoli, espressi mediante una scala semilogaritmica,in cui il lasso di tempo più breve tra un pacchetto e quello immediatamente successivo viene misurato durante l'attività di messagistica;al contrario,per l'attività di call l'iat assume valori più rilevanti. Il maggior volume di dati generati tra i due endpoint si deve all'attività relativa ai messaggi.Per quanto concerne l'iat message, esso assume in tutti e quattro le attività valori molto piccoli che presentano il medesimo andamento,tranne per l'attività call,per cui l'iat message esprime un lasso di tempo maggiore.

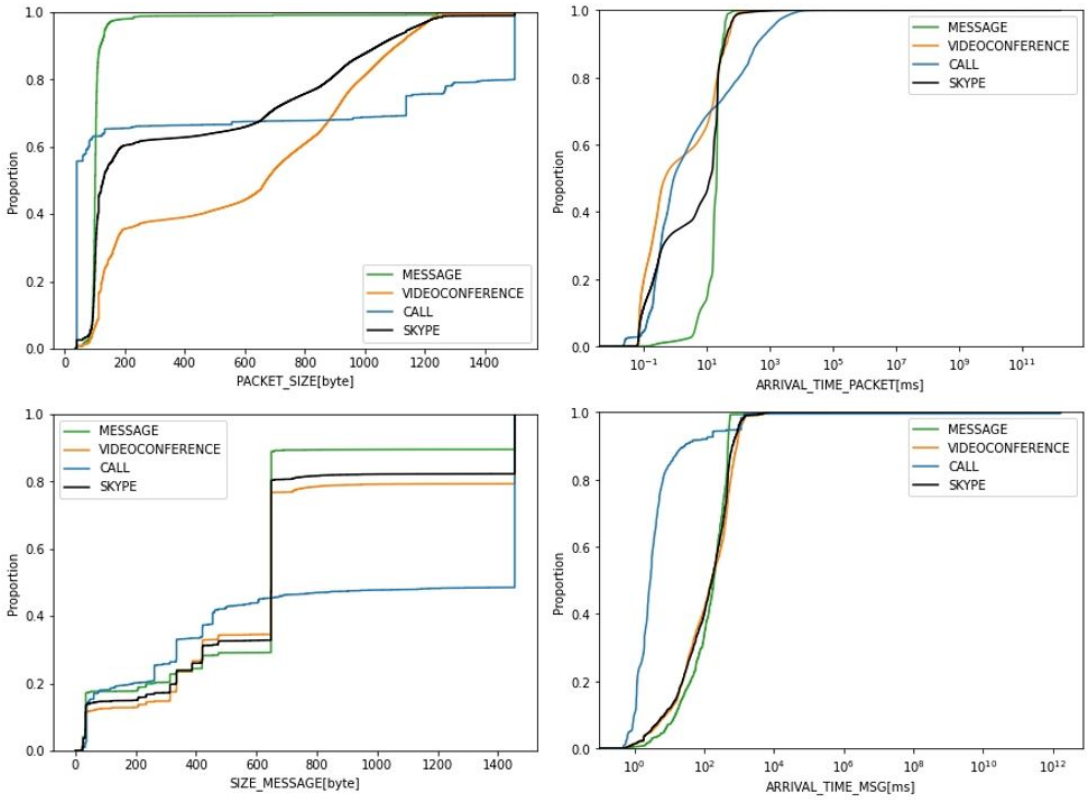


Figura 3.2: Grafici UpStream

In questo secondo gruppo di grafici, andiamo ad analizzare l'Up-

stream. L'attività che presenta pacchetti che hanno una maggior dimensione è quella di messaggistica,che, per quanto riguarda l'iat packet, presenta il tempo d'invio e recezione più breve.Possiamo notare che il valore iat packet più grande è quello fornito dall'attività di call. Anche in termini di dati effettivi, valgono le stesse considerazioni fatte nel caso del packet size,tuttavia nell'attività call il volume di dati effettivi è inferiore.In termini di iat message,infine,abbiamo valori simiiari tranne per l'attività call,che prensenta l'iat con il lasso di tempo maggiore mentre quello più breve è asocciato all'attvità dei messaggi.

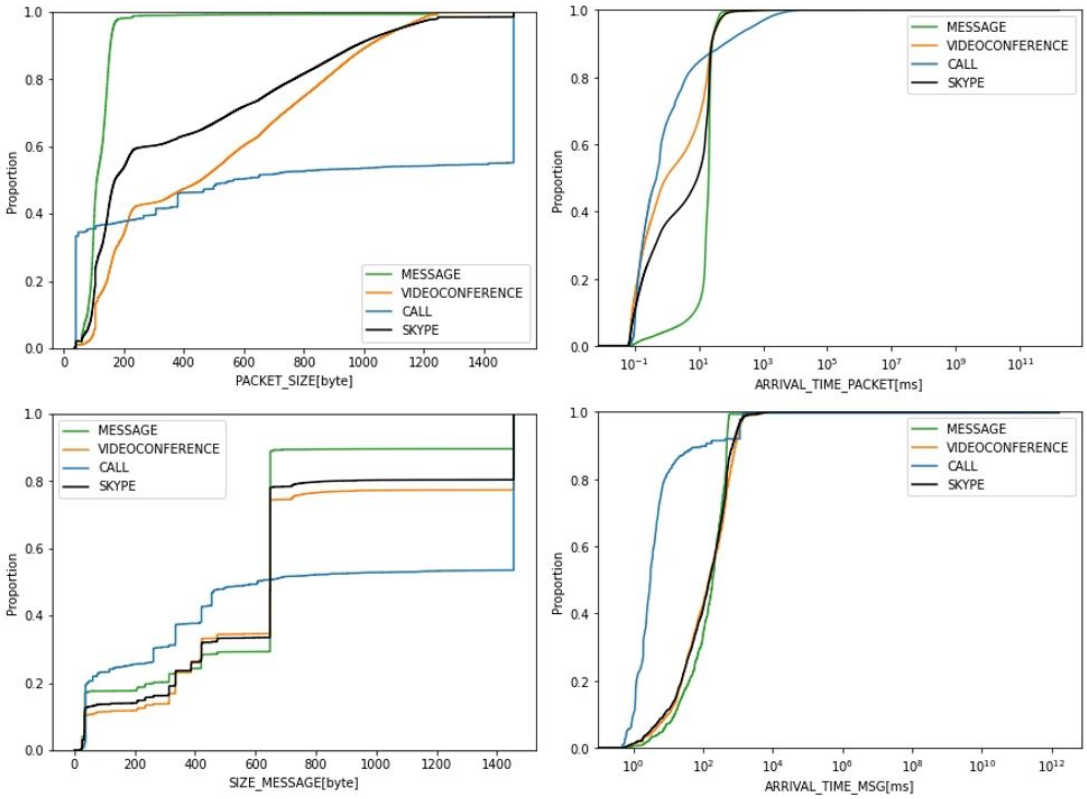


Figura 3.3: Grafici DownStream

Per quanto riguarda il downStream, possiamo osservare che in ter-

mini di pacchetto, anche in questo caso l'attività associata ai messaggi rispetto alle altre è quella più intensa.Nell'attività di call la dimensione dei pacchetti catturati è minore rispetto l'upstream. Per l'iat packet, valgono le considerazioni fatte nei casi precedenti, tuttavia il lasso di tempo tra un pacchetto e quello successivo calcolato durante l'attività di videoconference si riduce leggermente.Per il volume di dati generati valgono le stesse considerazioni fatte per l'UpStream, tranne che per l'attività di call, in cui la quantità di dati è maggiore. Per l'iat message, infine, le attività hanno un andamento similare tranne per l'attività call che presenta ancora una volta l'iat maggiore, tuttavia rispetto all'upstream, il tempo di trasmissione di un messaggio e quello succssivo, per videoconference e call, si riduce.Effettuando un confronto esteso esclusivamente all'app,possiamo constatare che l'andamento in termini di dimensione dei pacchetti risulta essere similare sia per l'upstream che downstream.Per quanto riguarda il volume di dati generati l'andamento risulta essere leggermente più intenso nel upstream rispetto al downstream.In termini dell'iat packet possiamo notare un andamento tale per cui nel caso dell'upstream si registra un lasso di tempo tra un pacchetto ed un altro leggermente maggiore rispetto al downstream.Infine, per quanto riguarda l'iat message l'andamento è quasi uguale.

Capitolo 4

Conclusioni

La simulazione di un attacco mitm, attaverso l'applicativo mitmproxy è semplice da implementare e risulta compatibile con qualsiasi piattaforma e dispositivo; fondamentale è la generazione e l'aggiunta del certificato sugli endpoint per l' esecuzione.Dal momento che molte applicazioni presentano dei meccanismi di sicurezza basati su pinning, per accedere al traffico https è necessario disabilitare ciò ed incorpare il certificato generato all'interno dell'app.E' stato constatato che questo tipo di soluzione non è univoca per tutte le applicazioni, in quanto queste potrebbero presentare dei meccanismi di sicurezza più robusti, e difficili da bypassare.Dopo queste dovute osservazioni, sarà possibile effettuare le nostre catture a seconda dell'attività. Il master secret generato durante le catture, rappresenta l'elemento chiave per la buona riuscita della criptazione dei messaggi.Scapy è stato uno strumento molto efficace nell'analisi dei file pcap,in quanto ci permette di ispezio-

nare in maniera dettagliata i pacchetti secondo lo standard ISO/OSI,ed estrarre le features necessarie (pkt size, iat pkt, msg size, iat msg) ai fini dell'analisi.Questi dati raccolti,grazie anche al modulo pandas che è uno strumento per la manipolazione dei dati, sono stati salvati in un formato Csv, il quale, anche se risulta meno versatile rispetto al formato JSON, presenta la caratteristica di essere più compatto e permette l'analisi del set di dati in maniera molto semplice.Combinando il csv ottenuto con pandas e seaborn, modulo Python per la creazione di grafici, molto diffuso nel settore data scientist e data analyst,è possibile plottare il set di dati generato, andando a descrivere i parametri raccolti in funzione dell'app e delle singole attività eseguite.Sia dal punta di vista dell'UpStream che DownStream è stato constatato che l'attività che è risultata più intensa,in termini di dimensioni di pacchetti e volume di dati generati,è l'attività di messaggistica,che, tra l'altro,per quanto riguarda il tempo di ricezione dei pacchetti e dei dati,presenta il lasso di tempo più breve. L'identificazione del traffico di rete generato da un app, è fonte di analisi per valutare la presenza di eventuali falle di sicurezza che potrebbero rilevare una distribuzione anomala del traffico, quindi per poter agire in maniera tempestiva ad eventuali attacchi; essa, inoltre, permette di valutare ed identificare i requisiti di Qualità del Servizio (QoS) specifici per quell'applicazione (per esempio, in termini di throughput e ritardo).

Bibliografia

[1] Antonia Affinito, Giorgio Ventre, and Alessio Botta. *The impact of covid on network utilization: an analysis on domain popularity.* IEEE, 2021.

[2] Mauro Conti, Nicola Dragoni, and Viktor Lesyk. *A Survey of Man In The Middle Attacks.* IEEE, 2016.

[3] Aldo Cortesi, Maximilian Hils, and Raum Fresser. *https://docs.mitmproxy.org/stable/.* MIT License, 2014.

[4] Timothy A.Chadza, Francisco J.Aparicio-Navarro, Konstantinos G.Kyriakopoulos, and Jonathon A.Chambers. *A look into the information your smartphone leaks.* IEEE, 2017.

[5] Felipe Sierra and Anthony Ramirez. *Defending Your Android App.* Association for Computing Machinery, 2015.

[6] Niklas Higi. *https://github.com/shroudedcode/apk-mitm.* MIT license, 2019.

[7] Charit Mishra. *Wireshark 2 Quick Start Guide : Secure Your Network Through Protocol Analysis;Decrypting encrypted traffic (SSL/TLS).* Packt Publishing, 2018.

[8] Prabath Siriwardena. *Advanced API Security:How Transport Layer Security Works?* Apress, 2020.

[9] Jessey Bullock. *Wireshark for Security Professionals: Using Wireshark and the Metasploit.Chapter 7 Decrypting TLS Capturing and USB and Keyloggers and Network Graphing.* John Wiley and Sons Incorporated, 2017.

[10] Tamara Radivilova, Lyudmyla Kirichenko, Dmytro Ageyev, Maxim Tawalbeh, and Vitalii Bulakh. *Decrypting SSL/TLS Traffic for Hidden Threats Detection.* IEEE, 2015.

[11] Shipra Bansal and Nitin Bansal. *Scapy–A Python Tool For Security Testing.* Bansal, 2015.

[12] Thomas Kluyver and Wes McKinney. *https://pandas.pydata.org.* pandas development team, 2011.

[13] Michael L. Waskom. *seaborn: statistical data visualization.* Lorena Pantano, 2021.

[14] Michael Waskom. *https://seaborn.pydata.org.* Michael Waskom, 2012.