

UNIVERSITÀ DEGLI STUDI DI SALERNO



DIPARTIMENTO DI INFORMATICA

Corso di Laurea Magistrale in Informatica

Curriculum Sicurezza Informatica

Rilevazione di stream nei servizi di Instant Messaging con tecniche di Intelligenza Artificiale nell'ambito della Network Forensics

Relatori:

prof. Raffaele Pizzolante

prof. Gianni D'Angelo

Candidato:

Egidio Giacoia

Matr.0522500488

Anno Accademico 2019/2020

Indice

1. Introduzione	7
2. La Digital Forensics	9
2.1 Introduzione.....	9
2.2 L' evoluzione delle Digital Forensics	11
2.2.1 La Digital Forensics in Italia	13
2.3 Il Processo Investigativo.....	16
2.3.1 L'evidenza digitale e le figure nell'ambito forense	17
2.4 La Network Forensics	20
2.4.1 Il Rapporto con la Network Security e Computer Forensics	21
2.4.2 Classificazione dei sistemi di Network Forensics	23
2.4.3 Applicazioni della Network Forensics.....	24
3. I servizi di Instant Messaging	27
3.1 L'evoluzione dell'Instant Messaging.....	27
3.2 Fondamenti dei sistemi di IM	30
3.2.1 Funzionamento	30
3.2.2 Le Principali Architetture.....	33
3.2.3 La tecnologia WebRTC.....	34
3.2.3.1 <i>Il Signaling Server</i>	34
3.2.3.2 <i>Il Server STUN</i>	35
3.2.3.3 <i>Il Server TURN</i>	36
3.3 I Protocolli.....	37
3.3.1 Gli Open Standard Protocols	38
3.3.1.1 <i>Il protocollo XMPP</i>	39
3.3.1.2 <i>Il protocollo SIMPLE</i>	40
3.3.1.3 <i>Il protocollo Signal</i>	41
3.3.2 I Protocolli Proprietari	42
3.4 La Sicurezza nei sistemi IM.....	43
3.4.1 L' Autenticazione	43
3.4.2 Security Threats.....	45
4. Cenni sulle Reti Neurali	49
4.1 Introduzione.....	49
4.1.1 Rete neurale biologica e artificiale	49
4.1.2 Risolvere un problema con le RN	51
4.1.2.1 <i>Data Classification, Regression Analysis e Clustering</i>	52
4.1.3 Training e Validazione.....	53

4.2	Modellizzazione e Strutturazione	54
4.2.1	Modello di un neurone.....	54
4.2.1.1	<i>Funzioni di attivazione</i>	55
4.2.2	Paradigmi di apprendimento	58
4.2.2.1	<i>Addestramento non supervisionato</i>	58
4.2.2.2	<i>Addestramento supervisionato</i>	59
4.2.3	Cenni sui tipi di architetture.....	59
4.2.3.1	<i>Reti Feed-forward</i>	59
4.2.3.2	<i>Reti Ricorrenti</i>	64
5.	Analisi Forense delle Attività delle App di Instant Messaging	66
5.1	La Network Forensics e l'Instant Messaging.....	66
5.2	Caso di studio: profiling delle attività dell'app IM	68
5.2.1	Le applicazioni WhatsApp e Instagram	68
5.2.1.1	<i>L'app WhatsApp</i>	69
5.2.1.2	<i>L'app Instagram</i>	70
5.2.2	Esempio di Analisi Forense Attività WhatsApp	72
5.2.3.1	<i>Distinzione tra chiamata con o senza flusso audio</i>	73
5.2.3.2	<i>Evidenze Individuate</i>	76
5.2.3	Esempio di Analisi Forense Attività Instagram	77
5.2.4.3	<i>Evidenze Individuate</i>	82
5.2.4	Conclusioni	83
6.	Il tool Instant Messaging Stream Detector Parser	85
6.1	Concept.....	85
6.1.1	Cenni sulla Side-Channel Analysis.....	86
6.1.2	Osservazione sull'estrapolazione delle features da un flusso	88
6.2	Implementazione del tool IMSD Parser.....	90
6.2.1	Struttura del progetto.....	91
6.2.2	Setup dell'ambiente	92
6.2.3	Un overview sullo script.....	93
6.2.3.1	<i>La classe FEATURES</i>	93
6.2.3.2	<i>La classe FEATURES_CSV_CREATOR</i>	96
6.2.3.3	<i>La classe Capture</i>	97
6.3	Modalità ed Esempi di Utilizzo	100
6.3.1	Il comando !SEARCH_SERVER.....	103
6.3.2	Il comando !SEARCH_ALL_CONN	104
6.3.3	Il comando !EXTRACT_FEATURE_FLUX.....	105
6.3.4	Il comando !EXTRACT_FEATURE_CONN.....	109
6.3.5	Il comando !EXTRACT_FEATURE_ALL_CONN.....	111

6.3.6	Il comando !CHANGE_IM	113
7.	Un esperimento: generazione di un dataset per il ML	115
7.1	Metodologia e sperimentazione	115
7.1.1	I file PCAP del Dataset	116
7.2	Un caso particolare	117
7.3	Analisi forense dei file PCAP acquisiti	120
7.3.1	Invio/Ricezione delle immagini	120
7.3.1.1	<i>Test Case: IMG_1MB</i>	120
7.3.1.2	<i>Test Case: IMG_4-6MB</i>	123
7.3.2	Invio/Ricezione dei video.....	125
7.3.2.1	<i>Test Case: VID_5-10MB</i>	125
7.3.2.2	<i>Test Case: VID_20MB</i>	127
7.3.3	Invio/Ricezione degli audio	130
7.3.3.1	<i>Test Case: AUD_1MB</i>	130
7.3.3.2	<i>Test Case: AUD_6MB</i>	132
7.3.4	Invio/Ricezione dei file.....	134
7.3.4.1	<i>Test Case: FIL_1-2MB</i>	135
7.3.4.2	<i>Test Case: FIL_10MB</i>	137
7.3.5	Conclusioni	139
7.4	Generazione del Dataset tramite IMSD Parser	141
7.4.1	Generazione Dataset: frequenza diversa per ogni oggetto	142
7.4.2	Generazione del Dataset: frequenza uguale per tutti gli oggetti	145
8.	Conclusione	147
	Bibliografia	149

1. Introduzione

Al giorno d'oggi, con la diffusione degli Smartphone, milioni di persone utilizzano app di Instant Messaging (IM) per comunicare tra loro in tempo reale e condividere oggetti multimediali. Inoltre, questi servizi sono affiancati da una sicurezza in termini di codifica dati che impedisce la violazione delle conversazioni, in maniera tale da garantire la privacy agli utenti. Le misure di sicurezza adottate per i consumatori però, sfortunatamente, proteggono anche i criminali che ne fanno un uso improprio di queste piattaforme, soprattutto in casi di cyberstalking e/o cyberbullismo.

In ambito di Digital Forensics per identificare delle possibili evidenze digitali sull'utilizzo illecito di un'applicazione di Instant Messaging si potrebbe analizzare i dispositivi fisici di un sospettato. Nel caso, però, non si dispone di quest'ultimi, un modo per individuare delle evidenze digitali è quello di osservare il traffico di rete prodotto dai servizi di Instant Messaging e i relativi client utilizzando gli strumenti della Network Forensics. Quello che accade comunemente in un'indagine è che: inizialmente viene acquisito il traffico raw, prodotto dalle interfacce di rete di un dispositivo informatico (pacchetti, eventuali log ecc..), per poi essere analizzato.

Per quanto riguarda il traffico di rete prodotto da tali servizi di Instant Messaging è sia stream di pacchetti UDP (es. chiamate VOIP), sia stream di pacchetti TCP/TLS per quanto riguarda la condivisione di vari oggetti multimediali.

Osservando il traffico TCP/TLS non si riesce a dire con chiarezza e non ambiguità che quel flusso identifica, ad esempio, un invio di una foto piuttosto che un file audio o video, in quanto il contenuto dei singoli pacchetti sono cifrati. Per cui un modo per rilevare la tipologia di un flusso di pacchetti è quello di utilizzare tecniche di side channel analysis, ovvero, di estrarre delle caratteristiche temporali/spaziali sul flusso di pacchetti (ad esempio header, protocollo, numero di pacchetto ecc.), in maniera tale da poter classificare il traffico.

Questo lavoro, ovviamente, se svolto in maniera manuale richiede tempo e un notevole sforzo, per cui non risulta fattibile se si vuole analizzare un traffico di rete voluminoso. Per tale motivo è necessario affidarsi a quelle che sono le tecniche di Intelligenza Artificiale per automatizzare tale processo.

Da questa idea è stato sviluppato un tool denominato Instant Messaging Stream Detector Parser, (sigla IMSD Parser) per l'estrapolazione delle features da stream TCP di oggetti (immagini, video, audio e file) nei servizi di Instant Messaging.

Il lavoro di tesi è organizzato come di seguito:

- Nel secondo capitolo si discuterà la disciplina della Digital Forensics, la sua evoluzione e il processo investigativo, analizzando anche un suo sottoramo, ovvero la Network Forensics;
- Nel terzo capitolo è dedicato alla presentazione dei servizi di Instant Messaging, analizzando i fondamenti e i protocolli utilizzati, soffermandosi anche sul lato della sicurezza;
- Nel quarto capitolo si darà un'introduzione alle reti neurali e dei cenni sulla modellizzazione e strutturazione;
- Nel quinto capitolo verrà mostrato un caso di studio dello svolgimento di un'analisi forense per il profiling di attività utente tramite app di Instant Messaging;
- Nel sesto capitolo verrà presentato il tool IMSD Parser, come nasce, come è stato sviluppato e la discussione su degli esempi di utilizzo;
- Nel settimo capitolo verrà effettuato un esperimento il cui obiettivo è quello di estrarre delle features da stream di oggetti per generare un dataset tramite l'utilizzo del tool IMSD Parser.
- L'ultimo capitulo sarà dedicato alle conclusioni, i problemi riscontrati e gli sviluppi futuri.

2. La Digital Forensics

Negli ultimi anni le evidenze digitali hanno avuto dei ruoli chiave in ambito legale, in quanto con l'aumento dei dispositivi informatici che ci circondano possono aiutare a poter ricostruire determinati eventi, ma possono essere usati, al tempo stesso, come strumenti per la criminalità informatica (cybercrime).

La Digital Forensics è una scienza per trovare prove dai media digitali come un computer, un telefono cellulare, un server o una rete. Tale materia fornisce alla squadra forense le migliori tecniche e strumenti per risolvere complicati casi relativi al digitale.

Nella sezione 2.1 è un'introduzione sulla definizione della Digital Forensics e i suoi obiettivi; in 2.2 si tratterà l'evoluzione della Digital Forensics nella storia e la visione in Italia di tale disciplina; nella sezione 2.3 viene presentato il processo investigativo forense e le figure forensi; in conclusione verrà trattato un ramo della Digital Forensics, ovvero, la Network Forensics nella sezione 2.4, definendone cos'è e i suoi obiettivi, il rapporto con la Network Security e Computer Forensics e i campi di applicazione.

2.1 Introduzione

Provando a dare una definizione tecnica, la Digital Forensics consiste nell'uso di metodi scientificamente provati, per le attività di conservazione, raccolta, convalida, identificazione, analisi, interpretazione, documentazione e presentazione di dati digitali, derivati da dispositivi informatici, con lo scopo di trovare delle possibili evidenze digitali (prove) per semplificare la ricostruzione di eventi criminali o azioni illegali e contribuire ad anticipare le azioni non autorizzate.

Alcuni obiettivi essenziali dell'utilizzo della Digital Forensics possono essere:

- aiutare a recuperare, analizzare e preservare i computer e i materiali correlati in modo tale da aiutare l'agenzia investigativa a presentarli come prove in un tribunale;
- aiuta a postulare il motivo dietro il crimine e l'identità del principale colpevole;
- progettare procedure su una presunta scena del crimine che possa aiutare a garantire che le prove digitali ottenute non siano corrotte;
- acquisizione e duplicazione dei dati: recupero di file cancellati e partizioni cancellate dai media digitali per estrarre le prove e convalidarle;
- aiutare a identificare rapidamente le prove e consente anche di stimare il potenziale impatto dell'attività dannosa sulla vittima;
- produzione di un documento forense informatico che offre un rapporto completo sul processo di indagine;
- preservare le prove seguendo la catena di custodia;

Col tempo, l'evoluzione tecnologica ha comportato ramificazioni della disciplina forense. Con il termine generico "Digital Forensics" si parla di varie discipline elencate di seguito. Possono essere viste come una sorta di specializzazioni, a seconda dell'oggetto dell'analisi e del dispositivo analizzato. [1]

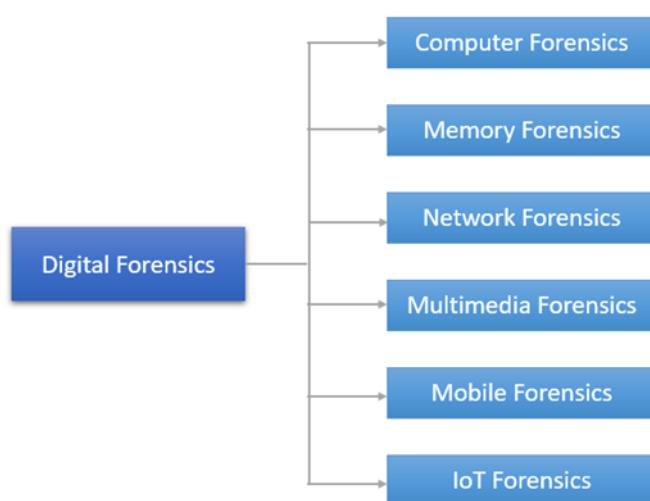


Figura 2.1 - Ramificazione della Digital Forensics

2.2 L'evoluzione delle Digital Forensics

L'epoca dagli anni 60' ai primi anni 80' è meno documentata perché gran parte di ciò che è accaduto non era incentrato sulla Digital Forensics. In questi anni i computer erano principalmente degli apparecchi industriali, di proprietà e gestiti da società, università, centri di ricerca e agenzie governative. Avevano bisogno di una grande infrastruttura fisica, comprese enormi quantità di energia e aria condizionata, e di personale altamente specializzato e dedicato. La loro funzione era in gran parte l'elaborazione dei dati. È in questo ruolo che i computer sono diventati per la prima volta di interesse per le comunità di sicurezza delle informazioni, legali e delle forze dell'ordine.

La crescita della criminalità informatica negli anni '80 e '90 ha indotto le forze dell'ordine a istituire gruppi specializzati, solitamente a livello nazionale, per gestire gli aspetti tecnici delle indagini. Per esempio, nel 1984 l'FBI lanciò un Computer Analysis and Response Team e l'anno seguente fu creato un dipartimento per la criminalità informatica all'interno della squadra antifrode della Metropolitan Police britannica. Oltre ad essere professionisti delle forze dell'ordine, molti dei primi membri di questi gruppi erano anche informatici per hobby e divennero responsabili della ricerca iniziale e della direzione del campo.[2] [3]

Il libro di Donn Parker del 1976, *Crime by Computer*, è forse la prima descrizione dell'uso delle informazioni digitali per indagare e perseguire i reati commessi con l'assistenza di un computer. Gli amministratori di sistema erano, per la maggior parte, responsabili della sicurezza dei propri sistemi, molti dei quali non erano collegati in rete verso il mondo esterno. Gli audit di sistema sono stati progettati per garantire l'efficienza e l'accuratezza del trattamento dei dati, che all'epoca era molto costoso. Questi audit costituivano il primo approccio sistematico alla sicurezza informatica. Inoltre, le informazioni raccolte durante gli audit potevano

essere utilizzate per indagare su illeciti [4]. Ciò viene comunque ancora adottato nella comunità delle forze dell'ordine.

Il libro di Cliff Stoll del 1990, *The Cuckoo's Egg* [5], cattura la pratica e l'etica della prima Digital Forensics. Sottolinea, inoltre, la riluttanza delle agenzie governative a impegnarsi in questa nuova area, infatti, è stato difficile per manager e investigatori tradizionali cogliere il potenziale dei computer di essere sia strumenti che vittime del crimine.

Durante gli anni '90 ci fu una forte richiesta di nuove e fondamentali risorse investigative. Vennero creati gruppi a livello regionale, e persino locale, per esempio, la National Hi-Tech Crime Unit nacque nel 2001 per fornire un'infrastruttura nazionale contro la criminalità informatica, con personale situato a livello centrale a Londra e con varie forze di polizia regionali (l'unità è stata trasformata nell'Agenzia Serious Organised Crime (SOCA) nel 2006).[6]

Dal 2000, in risposta all'esigenza di standardizzazione, vari organismi e agenzie pubblicarono linee guida per la Digital Forensics. Il Scientific Working Group on Digital Evidence(SWGDE) produsse un documento del 2002, "Best practice for Computer Forensics", che fu seguito, nel 2005, dalla pubblicazione di uno standard ISO (ISO 17025, General requirements for the competence of testing and calibration laboratories) .[7][8][9] La Convenzione sulla criminalità informatica, entrò in vigore nel 2004 con l'obiettivo di riconciliare le leggi nazionali sulla criminalità informatica, le tecniche investigative e la cooperazione internazionale. Il trattato fu firmato da 43 nazioni (tra cui Stati Uniti, Canada, Giappone, Sudafrica, Regno Unito e altre nazioni europee) e ratificato da 16. Le società commerciali (spesso sviluppatrici di software forense) iniziarono ad offrire programmi di certificazione e la Digital Forensics analysis venne inclusa come argomento per la formazione degli investigatori speciali del Regno Unito, Centrex.[7][10]

Dalla fine degli anni '90 i dispositivi mobile sono ampiamente aumentati, diventando ricche fonti di informazione, anche per reati non tradizionalmente associati alla Digital Forensics.[11] Nonostante questo, l'analisi digitale dei telefoni

è rimasta indietro rispetto ai tradizionali supporti informatici, in gran parte a causa di problemi relativi alla natura proprietaria dei dispositivi.[12]

2.2.1 La Digital Forensics in Italia

Il campo della Digital Forensics affronta ancora problemi irrisolti. Un documento del 2009, " Digital Forensic Research: The Good, the Bad and the Unaddressed ", di Peterson e Shenoi ha identificato un pregiudizio verso i sistemi operativi Windows nella ricerca scientifica forense.[13] Nel 2010 Simson Garfinkel ha identificato le problematiche relative alle indagini digitali in futuro, includendo l'incremento dei media digitali, l'ampia disponibilità di crittografia per i consumatori, una crescente varietà di sistemi operativi e formati di file, un crescente numero di individui che possiedono più dispositivi e con conseguenti limitazioni sulle investigazioni. Il documento ha inoltre identificato i problemi di formazione continua e il costo proibitivo dell'ingresso sul campo.[14]

L'esperienza insegna che gran parte degli esperti di computer forensics in Italia, per quanto riguarda l'aspetto delle metodologie informatiche, vanta un curriculum tecnico. La materia ha però trovato ottimi spunti di ricerca anche nell'ambiente giuridico e nel mondo delle professioni legali, grazie all'interesse accademico (soprattutto delle cattedre di informatica giuridica e di diritto processuale penale) e a numerosi avvocati che hanno iniziato ad analizzare il rapporto che occorre mantenere tra la professione forense e gli esperti informatici chiamati a svolgere perizie o interventi in tema di digital forensics, soprattutto in fase di indagini difensive.

Cesare Maioli definisce [15] la computer forensics come «la disciplina che studia l'insieme delle attività rivolte all'analisi e alla soluzione dei casi di criminalità informatica, comprendendo tra questi i crimini realizzati con l'uso di un computer, diretti a un computer, o in cui il computer può rappresentare comunque un elemento di prova». Secondo lo stesso autore, «gli scopi dell'informatica forense sono di conservare identificare acquisire documentare o interpretare i dati presenti in un computer».

Per Maioli sono essenzialmente cinque i punti cardine della disciplina della forensics. Il primo è quello relativo alla conservazione della fonte di prova, il secondo quello della identificazione della stessa, il terzo quello della acquisizione, cui seguono le fasi conclusive della documentazione (o reportistica) e interpretazione dei dati.

Gli aspetti eminentemente pratico-informatici, secondo Maioli, sono ben precisi e chiaramente identificabili. Lo scopo dell'analista forense in presenza di dati informatici sarebbe, prima di tutto, quello di adottare metodi di acquisizione della prova che non alterino il sistema informatico oggetto di analisi che contiene le fonti di prova. Occorrerebbe poi garantire un'egualanza originale-copia nel caso fosse fatta una copia di un supporto contenente dati per un'analisi in altro tempo o luogo. Infine, la procedura di analisi sulla copia deve comunque avvenire senza causare alterazioni del dato.

Al di là delle definizioni elencate, la disciplina non potrà essere circoscritta all'aspetto tecnologico, ma dovrà aprirsi alla considerazione di tutti gli aspetti legali interconnessi, come ad esempio:

- il problema dell'aggiornamento delle tecnologie di ricerca, che rischiano di diventare obsolete con conseguente perdita di una grande quantità di dati;
- il problema della “proceduralizzazione certa” nell’acquisizione e nell’utilizzo della prova informatica;
- il problema della privacy e della sua interconnessione con l’analisi digitale;
- il problema del rispetto delle norme di legge, la cui violazione ridurrebbe la digital forensics a pura interpretazione tecnologica.

Lo studio in campo penale di questa nuova disciplina non può prescindere: da un lato, dal tema della congruità dei nuovi mezzi di prova rispetto ai valori fondamentali dell’ordinamento; dall’altro, da quello dell’idoneità delle singole attrezzature e dei singoli protocolli applicativi a garantire i diritti della difesa.

A livello legislativo è interessante notare che, su una ricerca effettuata all'interno di 16 Stati europei [16], non è stata rilevata nessuna definizione di prova elettronica e/o digitale.

In Italia, l'art. 1, lett. p) del D.lgs. 82/05, anche denominato "codice dell'amministrazione digitale", definisce documento informatico qualsiasi «rappresentazione informatica di atti, fatti o dati giuridicamente rilevanti». Tramite la legge di ratifica della Convenzione di Budapest (legge 48/08), inoltre, è stata abrogata l'aporia normativa esistente nel nostro ordinamento, che vedeva la compresenza, accanto alla definizione appena citata, di quella contenuta nell'art. 491-bis c.p. con la quale si intendeva per documento informatico qualunque supporto informatico contenente dati o informazioni aventi efficacia probatoria, o programmi specificamente destinati ad elaborarli. Questa nozione faceva riferimento alla materialità del supporto informatico contenente dati o informazioni aventi efficacia probatoria.

Oggi, dunque, possiamo definire documento informatico qualsiasi file avente un quid rappresentativo espresso in linguaggio binario: un testo, un'immagine, un suono e, dunque, anche le pagine dei social network o le e-mail. Paolo Tonini ha evidenziato che la rappresentazione del fatto è la medesima, sia essa incorporata in uno scritto o in un file. Quello che cambia è soltanto il metodo di incorporamento su base materiale. Se, ad esempio, il file di testo viene stampato su carta, siamo di nuovo dinanzi ad un documento "tradizionale", che esplicita in modo visibile il contenuto del documento informatico.

La differenza tra i due concetti (documento tradizionale e documento informatico), dunque, sta tutta nel metodo di incorporamento, e non nel metodo di rappresentazione. I metodi di incorporamento, sempre secondo Tonini, si possono dividere in due categorie fondamentali: quella analogica e quella digitale. L'incorporamento analogico è "materiale", nel senso che la rappresentazione non esiste senza il supporto fisico sul quale è incorporata. Ad esempio, se all'interno di un documento scritto vengono operate delle cancellazioni, resta comunque traccia della manipolazione. Attraverso il metodo digitale, invece, una rappresentazione è incorporata su di una base «materiale mediante grandezze

fisiche variabili»: detto altrimenti, si tratta di una sequenza di bit. L'incorporamento digitale ha, dunque, la caratteristica dell'immaterialità, poiché la rappresentazione esiste indifferentemente dal tipo di supporto fisico sul quale il dato informatico è incorporato [17].

2.3 Il Processo Investigativo

Ci sono stati molti tentativi di sviluppare un modello di processo, ma finora nessuno è stato universalmente accettato. Parte del motivo potrebbe essere dovuto al fatto che molti dei modelli di processo sono stati progettati per un ambiente specifico, come le forze dell'ordine, e pertanto non possono essere prontamente applicati in altri ambienti.

Gli investigatori possono affidarsi a varie best practices, ovvero tecniche, metodologie, linee guida ecc., raccolte dalle esperienze più significative che si considera possano ottenere risultati migliori.

Il flusso di un'indagine forense inizia da una denuncia o segnalazione di un'attività illecita; dopodiché segue una fase di investigazione; in fine si ha un dibattimento in sede giudiziale.

Il processo investigativo [1] si suddivide in cinque fasi principali e consecutive:

- **Identificazione** - del crimine e di fonti contenenti potenzialmente prove digitali;
- **Raccolta** - (o acquisizione) di dati raw (grezzi), copiandoli, in maniera opportuna, dai dispositivi digitali
- **Ispezione** - dei dati raccolti con l'obiettivo di realizzarne una struttura migliore ai fini dell'analisi e della comprensione
- **Analisi** - si cerca di ottenere una migliore comprensione e si cerca di determinare i fatti di un evento o un'azione illegale;

- **Presentazione** - le prove digitali, individuate nelle fasi precedenti, vengono adeguatamente presentate nei tribunali e/o negli enti preposti.

Durante il processo di investigazione forense, possono esservi diverse iterazioni di una o più fasi.



Figura 2.2 - Fase del processo investigativo forense

2.3.1 L'evidenza digitale e le figure nell'ambito forense

Al centro di ogni investigazione digital forense, vi sono le digital evidence (evidenze digitali o prova digitale).

Una delle definizioni di prova digitale adottate a livello internazionale definisce la electronic evidence «è un'informazione generata, memorizzata e trasmessa attraverso un supporto informatico che può avere valore in tribunale» (International Organization on Computer Evidence (IOCE)); nonché quella adottata dallo Scientific Working Group on Digital Evidence (SWGDE) per cui costituisce digital evidence «qualsiasi informazione, con valore probatorio, che sia o meno memorizzata o trasmessa in un formato digitale» [4]. Stephen Mason osserva correttamente che i termini electronic evidence e digital evidence sono spesso usati impropriamente come sinonimi, anche se la digital evidence costituisce un sottosinsieme della electronic evidence, questa ha una portata definitoria più ampia, comprendente anche tutti i dati in formato analogico (analogue evidence).

Degli esempi di digital evidence sono le audio e video cassette, le pellicole fotografiche e le telefonate compiute attraverso la rete pubblica: tutte fonti di prova che possono essere “digitalizzate”, ma che non nascono in formato digitale. Sulla base di queste considerazioni, Mason definisce la prova elettronica come «l’insieme di tutti quei dati, inclusi quelli derivanti dalle risultanze registrate da apparati analogici e/o digitali, creati, processati, memorizzati o trasmessi da

qualsiasi apparecchio, elaboratore elettronico o sistema elettronico, o comunque disseminati a mezzo di una rete di comunicazione, rilevanti ai fini di un processo decisionale» [19].

Caratteristiche di una evidenza digitale:

- Il tempismo è una delle caratteristiche importanti dell'evidenza digitale, il primo soccorritore ha risposto immediatamente; in caso contrario, i dati potrebbero andare persi. Ad esempio, i dispositivi alimentati a batteria potrebbero spegnersi e la connessione di rete corrente potrebbe andare persa.
- Proprio come le impronte digitali o qualsiasi altra prova biometrica, anche l'evidenza digitale è nascosta o latente, il che richiede un processo da scoprire.
- Le prove digitali potrebbero essere distrutte o danneggiate. La risposta rapida e la catena di custodia sono la chiave della scienza forense del computer, è necessario agire in base alla situazione, altrimenti i dati importanti potrebbero essere danneggiati (intenzionalmente o non intenzionalmente).

Le forze dell'ordine e le agenzie di sicurezza sono responsabili delle indagini su un crimine informatico, tuttavia ogni organizzazione dovrebbe avere la capacità di risolvere autonomamente i problemi di base e le indagini.

Persino un'organizzazione può assumere esperti di società di investigazione informatica di piccole o medie dimensioni. Inoltre, è possibile creare la propria azienda che fornisce servizi di forensics. Per fare ciò, occorre un laboratorio di digital forensics, il permesso del governo di fondare un'azienda forense, gli strumenti adeguati e politiche per gestire l'azienda in modo efficace ed efficiente.

Tra le figure [1] chiave che un'azienda di investigazione informatica dovrebbe avere ritroviamo:

- Investigatori: si tratta di un gruppo di persone (il numero dipende dalle dimensioni dell'azienda) che gestiscono e risolvono il caso. È loro compito

usare gli strumenti e le tecniche forensi per trovare le prove contro l'indagato. Possono chiamare le forze dell'ordine, se necessario. Gli investigatori dovrebbero agire immediatamente dopo il verificarsi dell'evento sospettato di attività criminale.

- **Fotografo:** registrare la scena del crimine è importante quanto indagarla. Il lavoro del fotografo è scattare fotografie della scena del crimine (dispositivi IT e altre attrezzature)
- **Gestori degli incidenti (first responder):** ogni organizzazione, indipendentemente dal tipo, dovrebbe avere gestori degli incidenti nel proprio reparto IT. La responsabilità di queste persone è monitorare e agire in caso di danni della sicurezza del computer, come violazione dei criteri di rete, iniezione di codice, dirottamento del server, RAT o qualsiasi altra installazione di codice dannoso. In genere utilizzano la varietà di strumenti forense per svolgere il proprio lavoro.
- **Ingegneri e tecnici IT (altro personale di supporto):** questo gruppo di persone gestisce le attività quotidiane dell'azienda. Sono ingegneri e tecnici IT per la manutenzione del laboratorio forense. Il team dovrebbe essere composto da amministratore di rete, supporto IT, ingegneri della sicurezza IT e supporto desktop. Il ruolo chiave del team è quello di garantire il buon funzionamento delle funzioni organizzative, il monitoraggio, la risoluzione dei problemi, il recupero dei dati e di mantenere il backup richiesto.
- **Avvocato:** dal momento che la scientifica forense del computer si occupa direttamente delle indagini e di presentare il caso in tribunale, un avvocato dovrebbe far parte dell'organizzazione.

Le funzioni del primo soccorritore (first responder) sono fondamentali per le indagini. Egli è il primo a ricevere la notifica e ad agire in merito all'incidente di sicurezza. Questo è un ruolo che potrebbe essere assegnato a chiunque, inclusi

ingegneri della sicurezza IT, amministratore di rete e altri. La persona che ha la responsabilità di agire come tale dovrebbe avere le conoscenze, le abilità e il kit di strumenti adatti.

Il primo soccorritore dovrebbe essere pronto a gestire qualsiasi situazione e la sua azione dovrebbe essere pianificata e ben documentata. Alcune responsabilità fondamentali sono le seguenti:

- Comprendere la situazione, l'evento e il problema
- Mettere in sicurezza i dati e le periferiche
- Assicurare la scena da soggetti non autorizzati
- Preservare e impacchettare le tracce per il trasporto dalla scena del crimine
- Discutere le informazioni raccolte con gli altri membri del team
- Documentare la scena e la stanza integralmente

I gestori del primo soccorritore o di incidente dovrebbero avere un'esperienza di prima mano nella sicurezza delle informazioni, nei diversi sistemi operativi e nelle loro architetture.

2.4 La Network Forensics

La Network Forensics è una scienza forense che appartiene al ramo della Digital Forensics. Si tratta di una scienza relativamente recente, sorta in concomitanza con lo sviluppo e la progressiva diffusione di Internet.

La Network Forensics è definito in [18] come "uso di tecniche scientificamente provate per raccogliere, fondere, identificare, esaminare, correlare, analizzare e documentare prove digitali da più fonti, elaborando e trasmettendo attivamente fonti digitali allo scopo di scoprire fatti relativi all'intento pianificato o misurare il successo di non autorizzati attività intese a interrompere, corrompere o compromettere anche i componenti di sistema come fornire informazioni per assistere in risposta o recupero da queste attività."

Trattandosi di una scienza forense, anche la Network Forensics si occupa di individuare le prove di un reato (oltre ad una più generica analisi del traffico di rete) che spesso consiste in una “intrusione” all’interno del network da parte di un soggetto esterno non autorizzato. Poiché il traffico di rete consiste sostanzialmente in informazioni volatili e dinamiche, ossia dati che vengono trasmessi per poi andare perduti, spesso le indagini di Network Forensics hanno un carattere proattivo.

L’indagine di un cyber-criminale spesso comporta casi legati alla sicurezza nazionale, allo spionaggio aziendale, pedopornografia, criminalità tradizionale assistita da tecnologia informatica e di rete, monitoraggio dei dipendenti o cartelle cliniche, in cui la privacy svolge un ruolo importante.

2.4.1 Il Rapporto con la Network Security e Computer Forensics

Le analisi delle Network Forensics sembrano essere simili alla Network Security, tuttavia gli obiettivi delle due sono molto diversi.

La Network Forensics è un nascente scienza che si occupa di acquisizione, registrazione e analisi del traffico di rete. I dati sul traffico di rete vengono acquisiti mediante sniffer di pacchetti e alerts e logs vengono raccolti dagli strumenti di sicurezza di rete esistenti. Questi dati vengono analizzati per la caratterizzazione degli attacchi e studiati per rintracciare i sospetti.

L’approccio di Network Security utilizza meccanismi difensivi come firewall, per la prevenzione, e il sistema di rilevamento delle intrusioni (IDS), per il rilevamento. Questi approcci scoprono le vulnerabilità della rete e bloccano tutte le comunicazioni dannose dall’esterno. I firewall controllano come il traffico entra ed esce da una rete, in base agli indirizzi di origine e di destinazione e numeri di porta. Filtra il traffico di rete dannoso in base alle policy.

L’approccio Network Forensics raccoglie le prove richieste per la risposta agli incidenti e le indagini sul crimine, mentre la Network Security protegge il sistema da attacchi. Gli strumenti di Network Security sono generalizzati e monitorano

continuamente la rete per possibili comportamenti dannosi. La Network Forensics comprende le indagini post-mortem dell'attacco e viene avviata dopo la notifica del crimine.

Ogni scenario criminale è diverso in molti aspetti, ci possono essere alcuni reati che non violano le politiche di sicurezza di rete ma può essere perseguitabile legalmente. Questi crimini possono essere gestiti solo dalla Network Forensics [21]. Le principali differenze tra la sicurezza della rete e le analisi forensi di rete sono riportate nella Tabella 2.1.

Network Security	Network Forensics
Protezione del sistema contro gli attacchi	Nessuna protezione del sistema contro gli attacchi
Tipicamente in real-time	Post-mortem
Generalizzato - alla ricerca di eventuali comportamenti dannosi	Caso limitato - vuole ricostruire lo scenario criminale
Resta attento 24h ogni giorno	Dopo una notifica di un crimine
Processo continuo	Processo a tempo limitato
Campo del settore dell'informatica	Scienza immatura e giovane

Tabella 2.1 – Differenze tra Network Security e Network Forensics

Network Forensics è un'estensione naturale della Computer Forensics. Quest'ultima [22] è stata introdotta dalle forze dell'ordine e ha molti principi guida della metodologia investigativa del sistema giudiziario. La Computer Forensics comporta la conservazione, l'identificazione, l'estrazione, la documentazione e l'interpretazione dei dati informatici. La Network Forensics si è evoluta in risposta alla comunità degli hacker e prevede l'acquisizione, la registrazione e l'analisi di eventi di rete al fine di scoprire la fonte degli attacchi.

Nella Computer Forensics, l'investigatore e il cyber-criminale sono a due diversi livelli con l'investigatore avvantaggiato. Invece, nella Network Forensics hanno lo stesso livello di abilità. Il Cyber-criminale utilizza una serie di strumenti per lanciare l'attacco e lo specialista della rete forense utilizza strumenti simili per indagare sull'attacco.

Le principali differenze tra Computer Forensics e Network Forensics sono indicate nella tabella 2.2.

Computer Forensics	Network Forensics
Introdotta dalle forze dell'ordine per gestire i dati informatici	Sviluppata in risposta alla comunità di cyber-criminali
L'investigatore e l'attaccante sono su due livelli diversi	L'investigatore e l'attaccante sono sullo stesso livello di abilità
Generalizzato - alla ricerca di eventuali comportamenti dannosi	Caso limitato - vuole ricostruire lo scenario criminale
L'investigatore e l'attaccante usano tools differenti (Investigatore è avvantaggiato)	L'investigatore e l'attaccante usano gli stessi tools e pratiche
Contiene preservazione, identificazione, estrazione, documentazione e interpretazione dei dati	Involvi nella cattura, registrazione e analisi di eventi della rete
Si tratta di acquisire, fornire catena di custodia, autenticare e interpretazione	Si tratta di indagare su filtri di pacchetti, logs di firewall e IDS

Tabella 2.2 – Differenze tra Computer Forensics e Network Forensics

2.4.2 Classificazione dei sistemi di Network Forensics

I sistemi di Network Forensics sono classificati in base a varie caratteristiche:

Scopo – La General Network Forensics (GNF) si concentra sul miglioramento della sicurezza. Vengono analizzati i dati sul traffico di rete e vengono scoperti modelli di attacco. La Strict Network Forensics (SNF) comporta rigidi requisiti legali in quanto i risultati ottenuti saranno utilizzati come prove per perseguire i crimini della rete [24].

Packet Capture – I sistemi Catch-it-as-you-can acquisiscono tutti i pacchetti che passano un particolare punto di traffico e successivamente analizzarli, richiedendo grandi quantità di archiviazione. I sistemi stop-look-and-hear analizzano ogni pacchetto in memoria e solo alcune informazioni vengono salvate per analisi future, che richiedono un processore più veloce [23].

Platform – Il sistema Network Forensics può essere un dispositivo hardware con software preinstallato. Può acquisire dati, analizzarli e presentare i risultati su

un'interfaccia del computer. Può anche essere un software autonomo, che può essere installato su un host. Analizza i pacchetti acquisiti o i record di NetFlow, che vengono copiati e archiviati nell'host.

Time of Analysis – Le apparecchiature di Network Forensics commerciale implicano la sorveglianza della rete in real-time, il rilevamento di anomalie basato sulla firma, l'analisi dei dati e l'indagine forense. Molti strumenti software open source sono progettati per l'indagine post-mortem delle acquisizioni di pacchetti. I dati a pacchetto completo vengono acquisiti dagli strumenti sniffer, memorizzati in un host e analizzati offline in un secondo momento.

Data Source – I sistemi basati sul flusso (Flow-based system) raccolgono informazioni statistiche basate su alcuni criteri all'interno del traffico di rete mentre passano attraverso la rete. L'apparecchiatura di rete raccoglie questi dati e li invia a un raccoglitore di flusso che archivia e analizza i dati. I sistemi basati su pacchetti (Packet-based system) implicano l'acquisizione di pacchetti completi in vari punti della rete. I pacchetti vengono raccolti e archiviati per una deep packet inspection.

2.4.3 **Applicazioni della Network Forensics**

La Network Forensics veniva tradizionalmente applicata agli ambienti cablati e si concentrava sulla versione 4 del protocollo Internet e sui relativi protocolli a livello di rete della suite TCP / IP. Di seguito sono riportati alcuni dei recenti lavori nella Network Forensics:

Stenografia – Molti aggressori usano forme di crittografia in qualche modo “leggere” per rendere più difficile il riconoscimento di rootkit o schemi di attacco, che altrimenti sarebbero stati facilmente individuati da qualsiasi IDS [25].

Honeypot Forensics – Gli honeypot sono posti per essere compromessi e forniscono informazioni sulle tecniche e sugli strumenti black hat, prima e dopo l'intrusione nell'honeypot. È possibile scoprire nuove forme di rootkit, trojan e potenziali exploit zero-day. È possibile ottenere una migliore comprensione delle aree di interesse e collegamenti nascosti tra team di black hat [26, 27].

IP versione 6 Forensics – Internet IPv6 offre agli utenti malintenzionati un rifugio temporaneo sicuro, poiché gli eventi sono scarsamente registrati e monitorati. Molti broker di tunnel gratuiti offrono connettività semplice e relativamente anonima [28]. Il passaggio da IPv4 a IPv6 richiederà tempo ed entrambi i protocolli potrebbero coesistere per un certo periodo di tempo richiedendo un meccanismo di interoperabilità. Questa disposizione a doppio stack porterà nuove vulnerabilità e exploit di sicurezza che necessiteranno di analisi forense [29].

Botnet Forensics – Le macchine compromesse possono essere collegate per formare "botnet" sotto controllo esterno, che vengono utilizzate per inviare e-mail di spam o disabilitare siti Web con un flusso di richieste fasulle. È molto difficile rintracciare l'identità degli spammer semplicemente analizzando la traccia elettronica [30, 31].

Application Level – Gli attacchi forensi sono passati dal livello di rete e di trasporto al livello di applicazione della suite di protocolli TCP / IP. Gli attacchi alla sicurezza Web includono cross site scripting (XSS), iniezione SQL, buffer overflow, ecc. È possibile fornire prove digitali affidabili dal payload del traffico di dati di rete trasmesso da e verso il servizio Web [32]. Anche la forense del servizio dei nomi di dominio (DNS) è una sfida importante [33].

Grid Forensics – Il grid computing aggrega tutti i tipi di risorse eterogenee che sono geograficamente distribuite e richiedono servizi di sicurezza approfonditi per proteggere risorse e dati. Implica inoltre adeguate tecniche forense che possono essere impiegate per valutare la responsabilità dei trasgressori. I team di sicurezza mancano dell'esperienza della scientifica forense poiché il grid computing è esso stesso una tecnologia in crescita [34].

Cloud Forensics – Il cloud computing richiederà un cambiamento nelle politiche aziendali e di sicurezza relative all'accesso remoto, all'uso dei dati su un browser, meccanismi di privacy e audit, sistemi di reportistica e sistemi di gestione che incorporano il modo in cui i dati sono protetti su un sistema informatico noleggiato che può trovarsi ovunque nel mondo. La complessa serie di interconnessioni tra il fornitore del cloud e il consumatore del cloud offre un terreno fertile per hacker e

criminali. La Network Forensics nel cloud computing richiede una nuova mentalità investigativa, in cui alcuni dati non saranno disponibili, alcuni saranno sospetti e solo alcuni saranno pronti per il tribunale [35].

Intelligent Network Forensics – Un sistema di Network Forensics intelligente ricostruisce scenari di intrusione e rende le attribuzioni di attacco che richiedono conoscenza di firme, prove, impatti e obiettivi delle intrusioni. La conoscenza del problem solving che descrive come il sistema può usare la conoscenza del dominio per analizzare attività dannose è essenziale. Saad e Traore [36] adattano le recenti ricerche nel Web semantico, nell'architettura dell'informazione e nell'ingegneria dell'ontologia per progettare metodi di ontologia per l'analisi della Network Forensics.

3. I servizi di Instant Messaging

L'Instant Messaging (Messaggistica Istantanea) ha portato una comunicazione efficace ed efficiente in tempo reale basata su testo alla comunità di Internet. Inoltre, la maggior parte delle applicazioni di messaggistica istantanea offrono funzioni extra come il trasferimento di file, elenchi di contatti e possibilità di conversazioni simultanee, il che rafforza la dipendenza di settori più ampi di utenti su queste applicazioni.

In questo capitolo inizialmente verrà presentate l'evoluzione dell'Instant Messaging, sezione 3.1; a seguire, nella sezione 3.2, si discuterà del funzionamento, architettura e del WebRTC relativo all'IM; nella sezione 3.3 sarà fatto luce sui diversi tipi di protocolli utilizzati per implementare le applicazioni di messaggistica istantanea; in conclusione, nella sezione 3.4, si discuterà del lato relativo alla sicurezza.

3.1 L'evoluzione dell'Instant Messaging

Internet ha rivoluzionato il modo in cui comunichiamo. L'e-mail è stata la forma di comunicazione più rapidamente adottata mai conosciuta. Ma a volte anche l'e-mail non è abbastanza veloce. Tra l'altro non si ha un riscontro se una persona a cui si desidera inviare e-mail è online in quel momento. Per colmare questi svantaggi ci pensa la messaggistica istantanea.

Oggi, la messaggistica istantanea (IM) è una delle applicazioni Internet più importanti e molti utenti la utilizza per motivi personali, sociali, educativi e aziendali. È un metodo di comunicazione che consente agli utenti di condividere istantaneamente informazioni in maniera digitale come testo, audio, video e monitorare la disponibilità di un elenco di utenti in tempo reale su una rete di computer, come Internet.

Nasce, inizialmente, come un'applicazione casual, utilizzata principalmente da adolescenti e studenti universitari, mentre ora anche per collegare varie aziende e basi militari [1].

La messaggistica istantanea precede Internet, infatti le applicazioni di IM sono state utilizzate per la comunicazione di testo live tra diversi utenti di un computer multiutente avente come sistema operativo Unix. L'utilizzo della messaggistica istantanea è aumentato con le prime implementazioni del sistema di notifica Athena Zephyr del Progetto Institute of Technology (MIT) del Massachusetts e Internet Relay Chat (IRC) che sono state avviate all'Università di Oulu in Finlandia [2]. Prima che Internet diventasse popolare, le persone comuni potevano connettersi e comunicare tra loro utilizzando servizi online come America Online (AOL) e CompuServe.

Negli anni '90, le chat room basate sul web hanno iniziato a fornire agli utenti generali funzionalità di comunicazione istantanea. In una chat room, un gruppo di persone può digitare i messaggi che vengono visualizzati da tutti nella room. La messaggistica istantanea combina le funzionalità di e-mail e chat room. Consente la comunicazione in tempo reale come le chat room, mantenendo l'atmosfera personale e la privacy della posta elettronica. La messaggistica istantanea è stata introdotta per gli utenti di Internet nel 1996 [3] con l'introduzione di ICQ, un'utilità di messaggistica istantanea gratuita che poteva essere utilizzata da una vasta gamma di utenti. Di recente, l'altissima prevalenza di Internet ha portato l'IM ad un enorme aumento di popolarità.

Nel 1997, AOL, considerato il pioniere della comunità online, ha dato ai suoi utenti la possibilità di parlare in tempo reale tra loro attraverso chat room e messaggi istantanei. Il modello ICQ è la base per la maggior parte delle utility di messaggistica istantanea sul mercato oggi.

Non molto tempo dopo che AOL acquistò ICQ, AOL Instant Messenger (AIM) divenne il leader della messaggistica istantanea. Negli ultimi anni, tuttavia, numerosi servizi hanno attirato il pubblico di AIM. Windows Live Messenger (precedentemente MSN Messenger) e Yahoo! Messenger, in particolare, sono

diventati ampiamente utilizzato in tutto il mondo. Nel 2005 Google ha introdotto il suo sistema di messaggistica istantanea, Google Talk.

I vantaggi della tecnologia di messaggistica istantanea e della comunicazione in tempo reale l'hanno aiutata ad evolversi da uno strumento utilizzato dagli utenti normali a uno strumento di comunicazione tra i dipendenti delle loro organizzazioni. Un problema fondamentale affrontato dalle organizzazioni è la progettazione di protocolli e architetture di messaggistica istantanea standardizzati per adattarsi a un gran numero di utenti simultanei. Nelle organizzazioni aziendali devono essere garantiti adeguati meccanismi di sicurezza in grado di proteggere le comunicazioni di messaggistica istantanea dalle potenziali minacce.

Dal 2010, grazie alla diffusione degli smartphone, più persone hanno iniziato a utilizzare app di messaggistica come WhatsApp, WeChat, Facebook Messenger, Signal e Line. Le app di messaggistica differiscono dalla precedente generazione di piattaforme di messaggistica istantanea come AIM, Yahoo! Messenger e Windows Live Messenger, in quanto vengono utilizzate principalmente tramite app mobile su smartphone rispetto ai personal computer, anche se alcune app di messaggistica offrono versioni basate su Web o software per sistemi operativi per PC.

Con il tempo le persone passarono dalle chiamate tradizionali e dagli SMS (servizi a pagamento) alle app di messaggistica che sono gratuite o comportano solo piccoli addebiti sui dati. [4] Un altro vantaggio delle app di messaggistica per smartphone è la messaggistica di gruppo che ha contribuito alla loro adozione. [5]

Prima dell'introduzione delle app di messaggistica, gli utenti di smartphone potevano partecipare alle interazioni individuali tramite chiamate vocali o SMS, il cui coordinamento di gruppo è organizzato tramite singole chiamate / messaggi di testo per ogni singolo membro. Tuttavia, una persona può inviare lo stesso messaggio di testo a più membri del gruppo, però i membri del gruppo non possono vedere le risposte di tutti gli altri e nemmeno sapere chi fa parte del gruppo. Essi hanno accesso solo ai messaggi che inviano o ricevono, il che ostacola

il coordinamento del gruppo. Con l'introduzione delle app di messaggistica, gli utenti possono formare chat di gruppo, in cui tutti i membri possono vedere un intero thread di risposte da parte di tutti. I membri possono anche rispondere direttamente gli uni agli altri, anziché dover consultare il membro che ha invitato il messaggio di gruppo per inoltrare le informazioni. [6]

3.2 Fondamenti dei sistemi di IM

In questa sezione verranno esplorate alcune delle funzionalità e dei meccanismi sottostanti su come funzionano i sistemi di messaggistica istantanea. Infine, si presenteranno le varie architetture che differenziano i sistemi di messaggistica istantanea e la tecnologia WebRTC.

Attraverso la messaggistica istantanea un utente possiede un elenco di persone con cui può interagire. I messaggi vengono digitati in una piccola finestra di dialogo e sono visualizzati su entrambi gli schermi degli utenti in tempo reale.

La maggior parte dei programmi di messaggistica istantanea offre queste funzionalità:

- Inviare messaggi istantanei con utenti online e creare chat room con più utenti;
- Condividere video, immagini, audio, file e collegamenti Web;
- Conversazione tramite Internet anziché telefonica e streaming di contenuti.

3.2.1 Funzionamento

Inoltre, i client di messaggistica istantanea possono fornire informazioni sulla presenza degli utenti (indipendentemente dal fatto che un utente abbia effettuato l'accesso a un server di messaggistica istantanea) e sulla disponibilità degli utenti (indipendentemente dal fatto che un utente sia o meno disposto a inviare o ricevere messaggi). Di conseguenza, i client di messaggistica istantanea

consentono la creazione di elenchi di contatti bloccati (elenchi di ID utente esplicitamente vietati di ottenere la presenza e le informazioni sulla disponibilità dell'utente corrente) ed elenchi di contatti consentiti (elenco di ID utente autorizzati a inviare messaggi all'utente corrente e che possono tracciare le informazioni dell'utente riguardo presenza e disponibilità).

Di solito, gli utenti installano un'applicazione software sui propri computer o smartphone per lavorare come client sui server di IM. Il client di messaggistica istantanea aiuta gli utenti a registrare nomi utente e password univoci e utilizzare queste credenziali per connettersi ai server al fine di ricevere vari servizi. Una volta che il client ha effettuato l'accesso utilizzando le credenziali giuste, invia le informazioni di connessione come l'indirizzo IP e il numero di porta al server di IM. Quest'ultimo crea un file temporaneo che contiene le informazioni di connessione del client e l'elenco dei suoi contatti e verifica se qualcuno degli utenti nell'elenco è attualmente connesso.

Se il server rileva uno dei contatti client connessi, invia al client le informazioni sulla connessione di quell'utente e, inoltre, invia le informazioni sulla connessione client agli utenti connessi nell'elenco dei contatti. Tali informazioni di connessione consentono ai client di recapitare i messaggi alle macchine previste direttamente o tramite server. La Figura mostra un modello standard di comunicazioni di messaggistica istantanea che coinvolge un server e client.

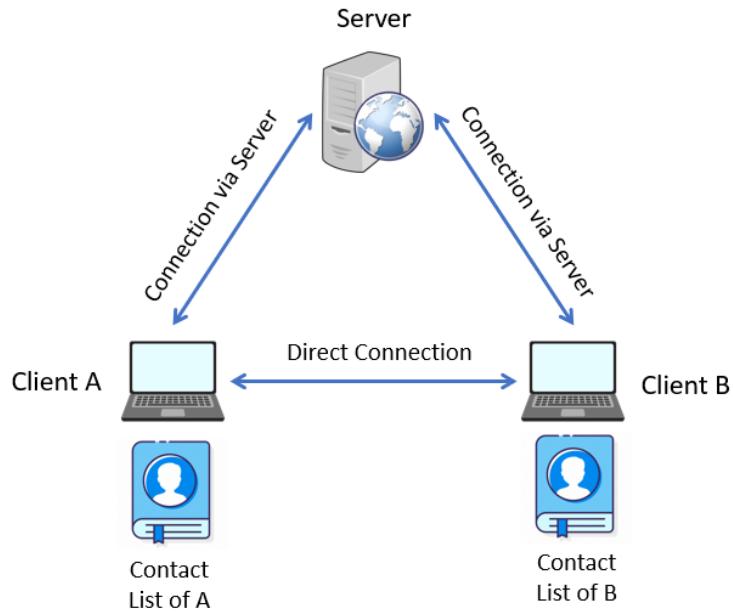


Figura 3.1 - Illustrazione del modello standard di comunicazione tra server IM e clients

Quando il client ottiene le informazioni di connessione per una persona nell'elenco dei contatti, lo stato di quella persona cambia in "online". Cliccando sul nome di una persona nell'elenco contatti che è online, si apre una finestra in cui si può iniziare la chat. [7]

Poiché il client ha l'indirizzo IP e il numero di porta per il computer della persona a cui è stato inviato il messaggio, la comunicazione avviene direttamente tra i due client, per cui il server non è coinvolto a questo punto. L'altro utente riceve il messaggio istantaneo e risponde. Man mano che la conversazione va avanti, la finestra che ognuno degli utenti vede sui rispettivi computer si espanderà come una finestra di dialogo a scorrimento. I messaggi istantanei di ogni persona vengono visualizzati in questa finestra su entrambi i computer.

Al termine della conversazione, si chiude la finestra di dialogo, si passa allo stato "offline" e si esce dall'applicazione. In questo caso, il client invia un messaggio al server per terminare la sessione. Il server invia un messaggio al client di ogni persona nell'elenco contatti che è attualmente online per indicare che l'utente si è disconnesso. Infine, il server elimina il file temporaneo che conteneva le

informazioni di connessione. Agli altri client dei contatti che sono online, il nome dell'utente appena disconnesso, passa allo stato “offline”. [8]

3.2.2 Le Principali Architetture

La maggior parte dei sistemi di messaggistica istantanea utilizza un'architettura client-server per inviare e ricevere messaggi e file. In questo schema, i messaggi tra gli utenti vengono in genere inoltrati attraverso un server. Tuttavia, un server di messaggistica istantanea sembra essere una singola entità per un client, in realtà, può essere un gruppo di server controllati da un singolo fornitore di servizi di messaggistica istantanea o una raccolta di server da fornitori di servizi di messaggistica istantanea indipendenti.

Se l'utente A desidera comunicare con l'utente B, entrambi devono accedere allo stesso servizio di messaggistica istantanea. I messaggi da A a B verranno recapitati dal server in base alle impostazioni sulla privacy di B [2]. La Figura mostra una tipica architettura client-server.

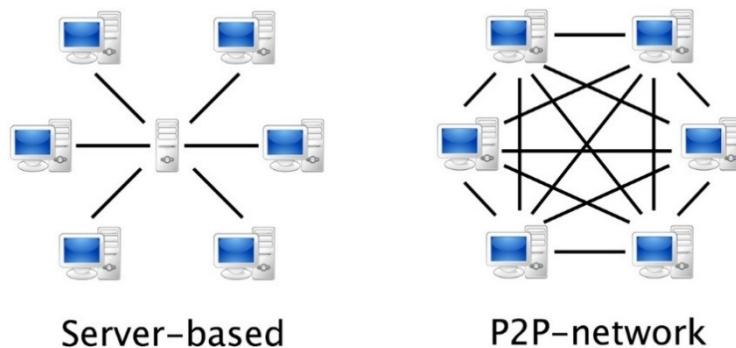


Figura 3.2 - Illustrazione delle architetture utilizzate dai servizi di IM

Nell'architettura client-server sono disponibili due approcci: simmetrico e asimmetrico. In un'architettura simmetrica, ciascun server svolge funzioni identiche, in modo tale che un client non debba distinguere quale server contattare per impegnarsi in un'attività [9]. In un approccio asimmetrico, ogni server è dedicato a una particolare attività come l'accesso, la scoperta di altri

utenti sulla rete, la manutenzione di una chat room o l'inoltro di un messaggio istantaneo. Il fattore principale che determina quale di questi due approcci scegliere è la scalabilità del sistema di messaggistica istantanea al crescere del numero degli utenti.

D'altra parte, alcuni sistemi di messaggistica istantanea utilizzano l'architettura peer-to-peer principalmente per sessioni non testuali come sessioni vocali e video. A differenza del client-server, gli utenti finali peer-to-peer possono trasferire file o messaggi video tra loro direttamente senza l'ausilio di server centralizzati, come mostrato nella Figura. Poiché non vi sono requisiti di archiviazione dei file ed è necessaria solo la larghezza di banda per i trasferimenti di file, è un'opzione economica e scalabile.

3.2.3 La tecnologia WebRTC

Il protocollo Voice over Internet Protocol (VoIP) è uno degli standard più popolari per le chiamate vocali e video sul Web. Questo protocollo VoIP viene utilizzato da varie piattaforme come WhatsApp, Skype, Messenger, Facebook ecc. Sia la chiamata vocale che la videochiamata dipendono dal modo in cui si trasmettono i media tra i due client collegati tra loro, compito destinato al WebRTC.

La tecnologia WebRTC è open source che fornisce ai browser e alle applicazioni mobili funzionalità di comunicazione in tempo reale (RTC). Solo WebRTC non basta per completare l'implementazione, servono anche altri componenti, quali: Signaling Server, TURN Server e STUN Server.

3.2.3.1 *Il Signaling Server*

Il Signaling Server permette di stabilire una connessione tra i due client che vogliono comunicare, condividendogli le informazioni di servizio.

I Peers interagiscono con il Signaling server per condividere gli handshakes e iniziare una trasmissione peer-to-peer diretta (occorre conoscere gli IP pubblici dei peer).

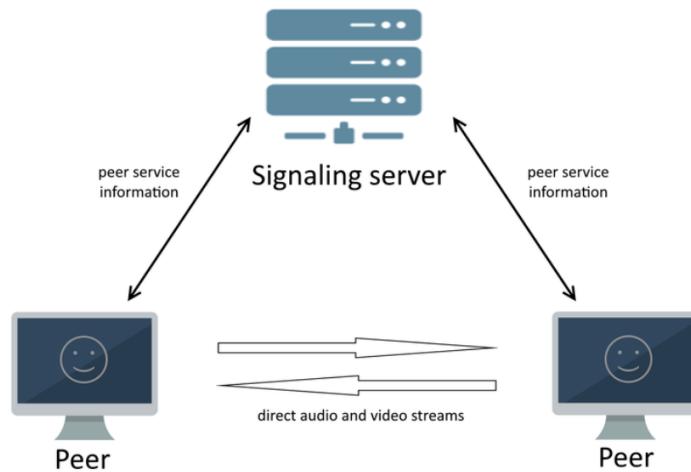


Figura 3.3 - Illustrazione del funzionamento del Signaling server

Il carico di traffico e di calcolo del Signaling server è relativamente basso, ma è un elemento centrale del sistema di connessione WebRTC.

3.2.3.2 *Il Server STUN*

Siccome i peer possono essere nascosti dietro NAT, per cui non hanno alcuna informazione sugli indirizzi esterni l'uno dell'altro, viene utilizzato il server STUN. Esso consente di rilevare gli indirizzi di rete pubblica dei peer e stabilire una connessione peer-to-peer dietro un NAT. [10].

I messaggi STUN sono costituiti da un'intestazione fissa contenente un metodo, una classe e un ID transazione. L'intestazione è seguita da attributi che indicano informazioni aggiuntive per un determinato messaggio. Quando un endpoint (client STUN) desidera apprendere il suo indirizzo pubblico, invia un messaggio STUN chiamato Binding Request a un determinato server STUN. Se nel percorso è presente un NAT, questo modifica l'indirizzo di trasporto di origine della richiesta.

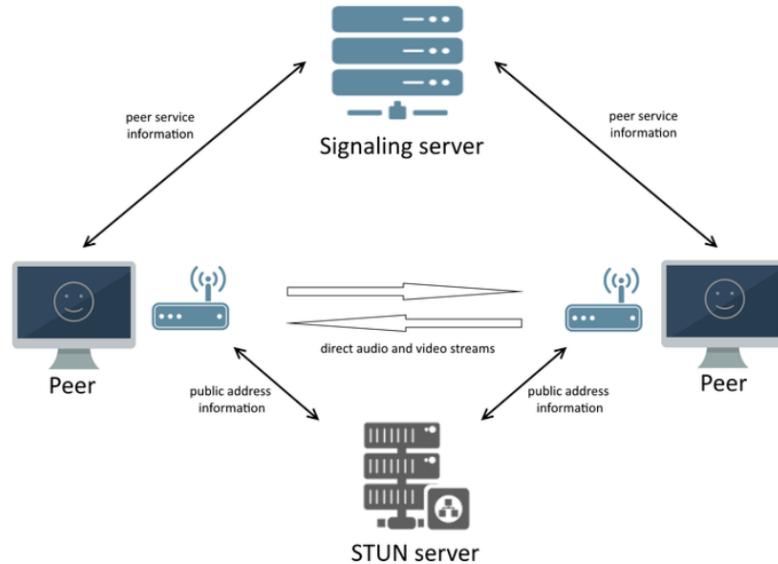


Figura 3.4 - Illustrazione del funzionamento del server STUN

Il carico di traffico e di calcolo del server STUN è relativamente basso.

3.2.3.3 *Il Server TURN*

C'è un altro problema che può impedire la trasmissione diretta peer-to-peer: ovvero l'utilizzo di un firewall. Esso può essere posizionato in qualsiasi punto della rete e tagliare il traffico WebRTC diretto.

Un modo per risolvere questo problema è utilizzare un server TURN. Esso ha un indirizzo pubblico, quindi entrambi i peer possono interagire con il server TURN anche dietro i firewall. Pertanto, quando non è disponibile alcuna connessione diretta peer-to-peer, il server TURN trasmette flussi audio / video di entrambi i peer proprio come un comune server multimediale.

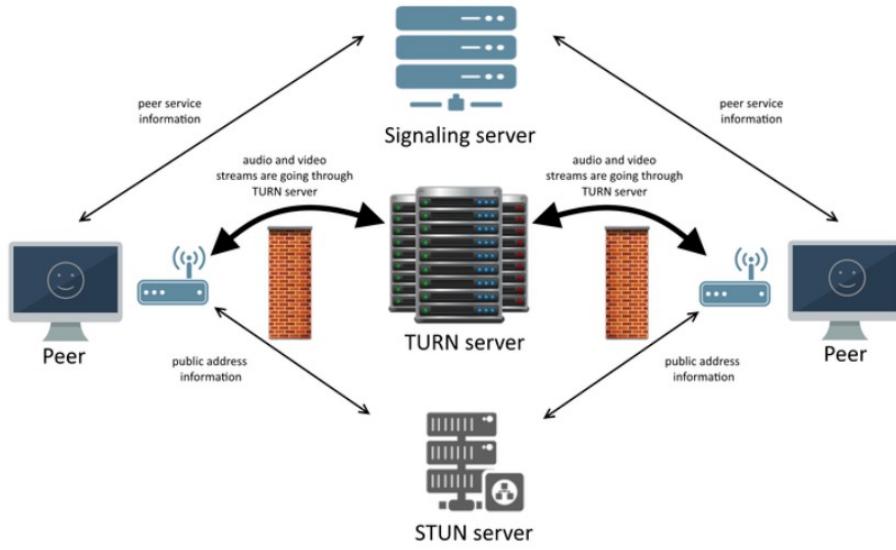


Figura 3.5 - Illustrazione del funzionamento del server TURN

Pertanto, il server STUN riceve la richiesta di associazione con un indirizzo di trasporto di origine che corrisponde all'indirizzo pubblico degli endpoint, ovvero il suo indirizzo riflessivo. Il server STUN inserisce questo indirizzo riflessivo nell'attributo XOR-MAPPED-ADDRESS nel corpo del messaggio STUN chiamato Binding Response e lo restituisce all'endpoint. Pertanto, l'endpoint può imparare il suo indirizzo.

Se tutto va bene, si ottengono gli indirizzi IP pubblici di entrambi i client e quindi WebRTC connette entrambi i client e gestisce tutto lo streaming multimediale. Il server TURN viene utilizzato per connettere entrambi i client in caso di errore peer to peer fungendo da mediatore. Fondamentalmente, prende i dati da un client e li invia a un altro client, quindi, ha il compito di inoltrare i media [11].

Poiché il server TURN trasmette i flussi multimediali tra peer, consuma molto traffico e richiede molta potenza di calcolo, soprattutto in caso di elaborazione di più peer, per cui è consigliabile disporre di un server dedicato per questa attività e che la larghezza di banda del traffico sia sufficientemente grande per gestire questa attività.

3.3 I Protocolli

I protocolli di messaggistica istantanea vengono visualizzati nel livello applicazione nel modello TCP/IP, come mostrato nella Figura, e vengono utilizzati dai client e dai server di messaggistica istantanea per comunicare tra loro in rete.

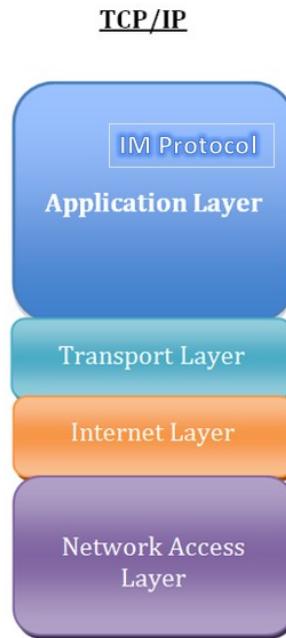


Figura 3.6 - Illustrazione del modello TCP/IP

I client di messaggistica istantanea utilizzano i protocolli per sfruttare appieno i servizi di messaggistica istantanea forniti dai server. A seconda della natura dell'applicazione di messaggistica istantanea e dei dati trasferiti, alcuni protocolli di messaggistica istantanea si collegano direttamente a un server tramite una connessione TCP (Transmission Control Protocol), mentre altri utilizzano User Datagram Protocol (UDP). Per sviluppare il servizio di messaggistica istantanea, lo sviluppatore deve selezionare quale protocollo utilizzare. In questa sezione esploriamo alcuni protocolli proprietari e standard aperti.

3.3.1 Gli Open Standard Protocols

Un protocollo è definito come standard aperto se i suoi diritti di utilizzo e dettagli di progettazione sono disponibili pubblicamente. La disponibilità di protocolli standard aperti consente agli sviluppatori che dispongono di un nome di dominio e di una connessione Internet adeguata di avviare il proprio server di messaggistica

istantanea e persino di regalare account gratuiti che gli utenti possono utilizzare con apposite applicazioni client di messaggistica immediata. Sfortunatamente, la maggior parte delle applicazioni di presenza e messaggistica istantanea attualmente utilizza protocolli indipendenti, non standard e non interoperabili sviluppati da vari fornitori. [12].

L'Internet Engineering Task Force (IETF) esegue la standardizzazione dei protocolli Internet da parte dei suoi vari gruppi di lavoro. Uno di questi gruppi di lavoro è il gruppo di lavoro IMPP (Instant Messaging and Presence Protocol) che definisce un protocollo standard in modo che le applicazioni sviluppate in modo indipendente di messaggistica istantanea possano interagire su Internet [12]. Definisce l'insieme minimo di requisiti di spazio dei nomi e amministrazione, scalabilità, controllo degli accessi, topologia di rete, formato dei messaggi, affidabilità, prestazioni e considerazioni sulla sicurezza che il protocollo di messaggistica istantanea deve soddisfare.

Inoltre, con la crescente necessità di interoperabilità tra diversi protocolli di messaggistica istantanea, la IETF ha pubblicato la specifica Common Profile for Instant Messaging (CPIM) [9] che definisce semantica e formati di dati comuni per la messaggistica istantanea che soddisfano i requisiti specificati dall'IMPP per facilitare la creazione dei gateway tra i servizi di messaggistica istantanea per consentire l'interoperabilità tra ampie gamme di sistemi di messaggistica istantanea. I gateway CPIM possono inoltrare payload comuni trasportati da diversi protocolli di messaggistica istantanea.

Di seguito abbiamo fatto luce su alcuni protocolli IM standard aperti, vale a dire XMPP, SIMPLE e Signal.

3.3.1.1 *Il protocollo XMPP*

All'XMPP fu inizialmente dato il nome di Jabber che fu ispirato dalla comunità open source Jabber trovata da Jeremie Miller. Nel gennaio 1999 Jeremie Miller ha rilasciato il codice sorgente per la versione iniziale del server Jabber. Nel 2002 il protocollo Jabber è stato ribattezzato con il nome neutro Extensible Messaging

and Presence Protocol (XMPP). XMPP è molto più di un semplice messaggio istantaneo: è un protocollo XML (Extensible Markup Language) aperto per servizi di messaggistica, presenza e richiesta-risposta in tempo quasi reale. La sintassi e la semantica di base sono state sviluppate originariamente all'interno della comunità open source Jabber, principalmente nel 1999 [14]. A partire dal 2002, il gruppo di lavoro IETF (L'Internet Engineering Task Force) XMPP ha lavorato sulle funzionalità e sulle estensioni di XMPP per fornire funzionalità di messaggistica istantanea e presenza.

I punti di forza del protocollo XMPP sono:

- **Sistema decentralizzato** - L'architettura di XMPP è simile alle e-mail; chiunque può realizzare il proprio server XMPP e non si identificano server centrali.
- **Standard aperto** - IETF ha formalizzato XMPP come tecnologia approvata per la messaggistica istantanea (definito negli RFC 6120 e RFC 6121). Non sono previste royalty per l'implementazione di queste specifiche.
- **Diffusione** - Le tecnologie XMPP sono utilizzate dal 1999. Esistono molte implementazioni dello standard XMPP per client, server e sono stati realizzati molti componenti e librerie.
- **Sicurezza** - I server XMPP possono essere isolati dalla rete pubblica, e la sicurezza viene affidata a protocolli come SASL e TLS.
- **Flessibilità** - Si possono realizzare funzioni proprietarie usando XMPP come base; per mantenere interoperabilità, la XMPP Standards Foundation gestisce estensioni al protocollo. Le estensioni permettono di realizzare funzionalità come chat room, gestione di rete, groupware, file sharing, videogiochi, controllo remoto di sistemi e monitoraggio, geolocalizzazione, middleware, cloud computing e VoIP. [15 , 16]

3.3.1.2

Il protocollo SIMPLE

SIMPLE, che sta per Session Initiation Protocol for Instant Messaging and Presence Leveraging Extensions, è un protocollo di messaggistica istantanea e presenza standard aperto. Come si evince dal suo nome, SIMPLE si basa sul più famoso SIP (Session Initiation Protocol).

Session Initiation Protocol (SIP) è un protocollo di controllo a livello di applicazione che può stabilire, modificare e terminare sessioni multimediali come le chiamate telefoniche via Internet [17].

SIMPLE applica SIP ai problemi di:

- registrarsi per informazioni sulla presenza e ricevere notifiche quando si verificano tali eventi, ad esempio quando un utente accede o ritorna da pranzo;
- invio di brevi messaggi, analogo a SMS o paging bidirezionale;
- gestire una sessione di messaggi in tempo reale tra due o più partecipanti.

I meccanismi forniti da SIP sono più utili per le applicazioni di presenza e per le applicazioni di comunicazione orientate alla sessione che per la messaggistica istantanea [12]. Per la messaggistica istantanea, SIP definisce due modalità, vale a dire la modalità di pagina (che invia messaggi senza stabilire una sessione) e la modalità di sessione (che inizia con la creazione di una sessione prima di inviare messaggi istantanei).

3.3.1.3 *Il protocollo Signal*

L'implementazione della crittografia end-to-end (E2EE) in un servizio di messaggistica significa che i contenuti di un determinato messaggio sono disponibili solo per il mittente e il destinatario previsto. Senza E2EE, un messaggio potrebbe essere crittografato mentre viene trasmesso al server, ma il server potrebbe essere in grado di leggerlo. Ad esempio, alcuni fornitori di servizi potrebbero farlo per generare annunci più specifici per un utente.

Con E2EE, un messaggio viene crittografato in ogni momento mentre si fa strada attraverso eventuali intermediari. Nessuno, tranne il destinatario previsto, ha la chiave per decrittografarlo. Con un buon protocollo E2EE, né gli intermediari (server di app di messaggistica, database), né chiunque abbia intenzioni dannose sarebbe in grado di leggere i messaggi inviati.

Con la quantità di informazioni sensibili che potremmo condividere tramite messaggi di testo / istantanei, questo sta diventando un problema importante. Signal, un'applicazione di messaggistica di Open Whisper Systems (OWS), ha guadagnando popolarità tra le persone di tutto il mondo, in quanto fornisce una crittografia end-to-end ai suoi utenti. Utilizza il Signal Protocol, un protocollo di crittografia end-to-end open source, anch'esso sviluppato da OWS nel 2013.

Il protocollo fornisce riservatezza, integrità, autenticazione, coerenza dei partecipanti, convalida della destinazione, perfect forward secrecy, segretezza futura, conservazione della causalità, non collegabilità dei messaggi, ripudio dei messaggi, ripudio della partecipazione e asincronismo. [18 a] Non fornisce l'anonimato e richiede server per l'inoltro di messaggi e la memorizzazione della public key material. [19]

3.3.2 I Protocolli Proprietari

A differenza dei protocolli standard aperti, un protocollo proprietario è di proprietà di una singola organizzazione o individuo. Gli sviluppatori di protocolli proprietari mantengono le specifiche e i dettagli del protocollo lontano dall'accesso del pubblico e applicano restrizioni in modo che solo gli inventori o i licenziatari del protocollo siano in grado di sviluppare client software che dipende dal protocollo. Ci sono anche problemi finanziari e legali associati all'uso di protocolli proprietari che aggiungono ancora più restrizioni. Molti protocolli di messaggistica istantanea sono proprietari.

Due esempi sono Microsoft Notification Protocol (MSNP) e RVP, sviluppati rispettivamente da Microsoft nel 1999 e nel 1997 e utilizzati da Windows Live Messenger. Altri esempi di protocolli proprietari sono il protocollo Open System

for Communication in Realtime (OSCAR), il protocollo Skype, il protocollo Talk to OSCAR (TOC), il protocollo TOC2 e Yahoo! Messenger Protocol (YMSG).

3.4 La Sicurezza nei sistemi IM

Gli sviluppatori di software di solito si concentrano sulle funzionalità che attraggono gli utenti finali piuttosto che sui problemi di sicurezza durante la progettazione e l'implementazione dei loro prodotti software. Milioni di utenti beneficiano dei servizi di messaggistica istantanea, per cui è molto probabile che cyber-criminali provano a sfruttare le vulnerabilità dei sistemi di messaggistica istantanea per infettare un ampio settore della comunità di Internet. In questa sezione, discutiamo alcune delle funzionalità di sicurezza nei sistemi di messaggistica istantanea e le minacce che li circondano.

3.4.1 L' Autenticazione

L'autenticazione è definita come il processo di verifica di un'identità rivendicata da o per un'entità di sistema [20]. Nel contesto di messaggistica istantanea, un'entità di sistema potrebbe essere rappresentata da un utente. Le applicazioni di messaggistica istantanea dovrebbero fornire un meccanismo di sicurezza per autenticare gli utenti quando utilizzano client di messaggistica istantanea per accedere ai server. Inoltre, l'applicazione deve assicurarsi che agli utenti venga concessa l'autorizzazione per utilizzare le risorse di sistema appropriate. Ad esempio, Bob non dovrebbe accedere all'elenco dei contatti di Alice.

Il processo di autenticazione si basa generalmente su uno username e una password che vengono scelti da un utente al momento della registrazione. L'username è distribuito dall'utente a tutti i suoi contatti, mentre la password rimane riservata. La Figura mostra un tipico processo di autenticazione utilizzato da un'applicazione di messaggistica istantanea.



Figura 3.7 - Illustrazione del processo di autenticazione utilizzato da un'app di IM

Un meccanismo utilizzato per l'autenticazione è Single Sign-On (SSO), mediante il quale l'utente si autentica una sola volta e viene automaticamente registrato nei Service Provider (server di messaggistica istantanea), se necessario, senza richiedere ulteriori interazioni manuali [21]. Un metodo di autenticazione utilizzato da IM è Simple Authentication and Security Layer (SASL) che fornisce un metodo generalizzato per l'aggiunta del supporto di autenticazione ai protocolli basati sulla connessione [14] ed è pubblicato da IETF. SASL è supportato da XMPP per l'autenticazione.

D'altro canto, è importante garantire la riservatezza per i messaggi istantanei, poiché gran parte della messaggistica istantanea sta avvenendo su reti pubbliche e non attendibili come Internet. Secure Socket Layer (SSL) è un protocollo per la gestione della sicurezza di una trasmissione di messaggi su Internet ed è stato sviluppato da Netscape Communications Corporation. Successivamente, IETF ha utilizzato la versione 3.0 di SSL per sviluppare un protocollo standard, che è diventato il protocollo TLS (Transport Layer Security).

Il protocollo TLS consente alle applicazioni client / server di comunicare in modo tale da prevenire intercettazioni, manomissioni o falsificazione di messaggi [22]. Funziona a livello di trasporto per autenticare gli endpoint e crittografare i messaggi tra le entità autenticate. XMPP utilizza TLS per crittografare e proteggere lo stream XML attraverso la rete e anche il protocollo SIMPLE si basa su TLS.

La progettazione del protocollo TLS fino a TLS 1.2, non era incentrata sulla privacy, il suo obiettivo era semplicemente consentire a due parti, un client e un server, di stabilire in modo sicuro le chiavi di sessione. Il design di TLS 1.3 rivoluziona lo scambio di chiavi autenticato nel mondo reale, impiegando moderni meccanismi crittografici, riducendo così l'handshake.

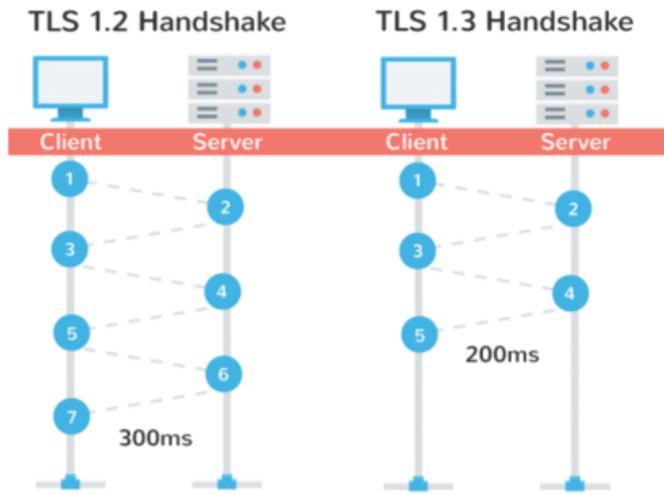


Figura 3.8 - Illustrazione dell'handshake TLS 1.2 e 1.3

Una minaccia comune negli ambienti di messaggistica istantanea è quando gli aggressori sfruttano programmi per iscrivere account illimitati in maniera automatizzata per consumare risorse dei server di messaggistica immediata. I server superano questa minaccia utilizzando la tecnica nota come Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA). CAPTCHA è un test che verifica la risposta di una sfida proposta al creatore dell'account di messaggistica istantanea. Tipicamente l'utente deve inserire una stringa di caratteri richiesta in base a un'immagine visualizzata sullo schermo. Di solito, l'immagine viene visualizzata in un modo che solo gli umani possono riconoscere.

3.4.2 Security Threats

È importante notare che la messaggistica istantanea non è considerata un modo sicuro per comunicare. I messaggi e le informazioni sulla connessione vengono mantenuti sui server controllati dal provider dell'utilità di messaggistica

istantanea. La maggior parte delle utility fornisce un certo livello di crittografia, ma non sono così sicure che è necessario inviare informazioni riservate attraverso il sistema. Sono stati segnalati casi in cui i log degli utenti di messaggistica istantanea sono stati acquisiti e utilizzati da tipi nefasti; I client di messaggistica istantanea offrono la possibilità di trasferire file tra utenti oltre allo scambio di messaggi istantanei, tale funzionalità mette gli utenti IM a rischio di essere infettati da programmi dannosi nascosti nei file trasferiti. Si ritiene che Voice Over Internet Protocol (VoIP) sia più vulnerabile alle infiltrazioni rispetto alla messaggistica istantanea basata su testo.

Di seguito esploriamo alcune delle minacce che possono arrecare danni e violazione della privacy agli utenti e indisponibilità del servizio di IM.

Worm

Un Worm è un fastidioso programma in grado di autoreplicarsi utilizzando una rete per copiarsi su altre macchine. Una caratteristica importante dei worm è che non hanno bisogno di un programma host per essere attivi e possono eseguire la replica senza intervento umano. Nel contesto della messaggistica istantanea, i worm possono infettare i computer degli utenti in diversi modi. Ad esempio, un worm può essere inviato a un utente di messaggistica istantanea in un file utilizzando la richiesta di trasferimento automatico di file da un client infetto all'avvio di una conversazione. Un altro modo è inviare Uniform Resource Locators (URL) malevoli di pagine Web come messaggi di testo che incoraggiano il destinatario ad aprirle.

Quando il rilevamento è disponibile per un determinato worm, i file infetti possono essere arrestati sul gateway di bordo. Nel caso di messaggistica istantanea, tuttavia, il software di rilevamento non monitora attualmente il traffico a livello di gateway. Se un worm inizia a diffondersi utilizzando la messaggistica istantanea, non può essere arrestato prima che raggiunga il computer dell'utente [23].

Trojan-Horse

Un Trojan-Horse è un programma malizioso che si cela all'interno di un programma benigno. Induce l'utente a eseguirlo e di solito permette un'apertura di una backdoor attraverso la quale un utente malintenzionato può accedere al sistema infetto [24]. Essi sfruttano la capacità di messaggistica istantanea di trasferire file configurando l'IM per condividere tutti i file nel sistema con pieno accesso a tutti utilizzando la porta IM. Di conseguenza, viene aperta una backdoor per i cyber-criminali per accedere alle risorse della macchina sfruttata. I firewall possono proteggere dai Trojan-Horse inibendo il numero di porta che bloccherà l'intero traffico.

Denial of Services (DoS) Attack

Un attacco DoS mira a sovraccaricare il sistema o il servizio a cui è diretto per negare agli utenti legittimi l'accesso al servizio [24]. Gli aggressori causano attacchi DoS nel sistema di messaggistica istantanea inondando la vittima di messaggi indesiderati che consumano una grande quantità di energia della CPU causando il blocco, crash o instabilità del client di messaggistica istantanea. I client di messaggistica immediata possono proteggersi da DoS ignorando determinati utenti utilizzando l'elenco dei contatti bloccati.

Man In The Middle (MITM) Attack

Un attacco MITM che mira ad ascoltare una conversazione tra due parti comunicanti, a catturare i messaggi scambiati in una comunicazione, a modificare i messaggi prima della ritrasmissione o persino a inviare messaggi per ingannare una parte comunicante nel credere che provengano da un vero mittente. Le soluzioni standard aperte offrono una migliore protezione contro gli attacchi man-in-the-middle, mentre le soluzioni di messaggistica istantanea proprietarie non offrono praticamente alcuna protezione contro di loro perché i dati vengono generalmente inviati non crittografati e non autenticati, rendendo la lettura e la produzione di messaggi facile per un avversario [24].

Buffer Overflow Attack

Un attacco Buffer Overflow tenta di inserire dati, messaggi estremamente lunghi o file di grandi dimensioni, che superano la capacità di un'applicazione di

messaggistica istantanea. Tale attacco può corrompere o sovrascrivere i dati esistenti nel buffer. Le vulnerabilità del buffer overflow sono state rilevate in molte delle principali soluzioni proprietarie di messaggistica istantanea [24]. Le versioni recenti di tutti i principali client di messaggistica istantanea includono un'opzione per l'utilizzo di software antivirus che può essere avviato automaticamente ad ogni download di file IM [2] in grado di proteggere i client di messaggistica istantanea da una varietà di minacce.

4. Cenni sulle Reti Neurali

I computer possono eseguire numerose operazioni molto più velocemente di un essere umano. Tuttavia, più veloce non è sempre meglio per la risoluzione dei problemi. Esistono molte attività per le quali il computer è notevolmente inferiore alla sua controparte umana. Ad esempio, date due immagini, un bambino in età infantile può facilmente dire la differenza tra un gatto e un cane. Tuttavia, questa stessa semplice operazione è estremamente difficile per i computer di oggi.

Il capitolo è organizzato come segue: nella sezione 4.1 si darà un'introduzione alle reti neurali; mentre nella sezione 4.1 dei cenni sulla modellizzazione e strutturazione di una rete neurale.

4.1 Introduzione

In questa sezione si descriverà la similitudine della rete neurale biologica e artificiale, quando devono e non devono essere utilizzate le reti neurali e, infine, il training e validazione di una rete neurale.

4.1.1 Rete neurale biologica e artificiale

Il termine Rete Neurale (RN), come viene normalmente utilizzato, è in realtà un termine improprio. I computer tentano di simulare reti neurali biologiche implementando reti neurali artificiali. Tuttavia, la maggior parte delle pubblicazioni usano il termine "rete neurale", piuttosto che "rete neurale artificiale" (ANN).

Per costruire un computer in grado di "pensare come un essere umano", i ricercatori hanno usato l'unico modello di lavoro disponibile: il cervello umano. Tuttavia, il cervello umano nel suo insieme è troppo complesso per essere modellato. Piuttosto, vengono studiate le singole cellule che compongono il cervello umano e le reti neurali artificiali tentano di simulare il comportamento di queste cellule.

Una cellula neuronale, come mostrato nella figura sottostante, accetta segnali dai dendriti. Quando un neurone accetta un segnale, quel neurone può “sparare”. Quando un neurone spara, un segnale viene trasmesso sull'assone del neurone. Alla fine, il segnale lascerà il neurone mentre viaggia verso i terminali degli assoni. Il segnale viene quindi trasmesso ad altri neuroni o nervi.

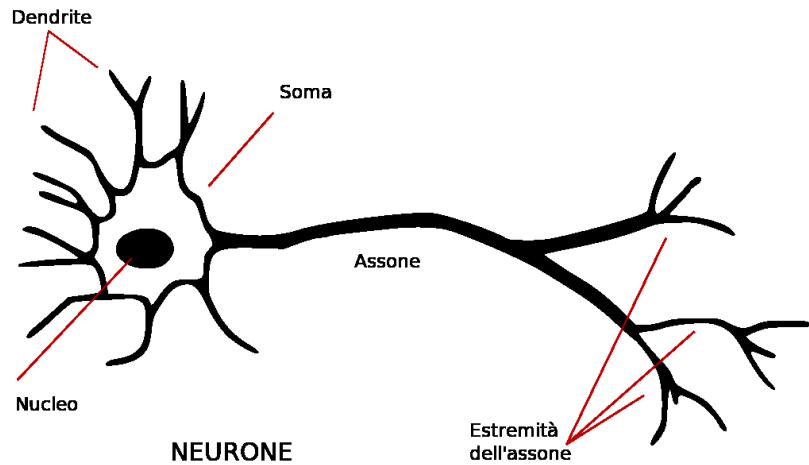


Figura 4.1 - Illustrazione schematica di un singolo neuro

Questo segnale, trasmesso dal neurone, è un segnale analogico. I computer sono macchine digitali e quindi richiedono un segnale digitale: la presenza di un segnale elettrico è indicata con un valore di uno, mentre l'assenza di un segnale elettrico è indicata con un valore di zero.

Un neurone prende una decisione sparando o non sparando. Le decisioni prese sono decisioni di livello estremamente basso. Le decisioni di livello superiore sono il risultato dell'input e dell'output collettivo di molti neuroni. [1]

Le reti neurali artificiali (ANN "Artificial Neural Network" o NN "Neural Network in inglese) si basano su questo modello: in termini pratici sono strutture non-lineari di dati statistici organizzate come strumenti di modellazione. Esse possono essere utilizzate per simulare relazioni complesse tra ingressi e uscite che altre funzioni analitiche non riescono a rappresentare.

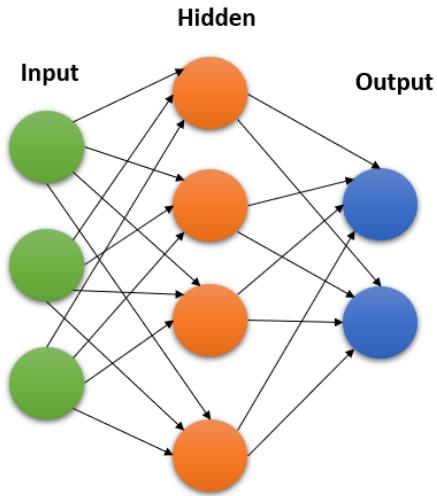


Figura 4.2 - Illustrazione rete neurale artificiale

Una rete neurale artificiale riceve segnali esterni su uno strato di nodi (unità di elaborazione) d'ingresso, ciascuno dei quali è collegato con numerosi nodi interni, organizzati in più livelli. Ogni nodo elabora i segnali ricevuti e trasmette il risultato a nodi successivi.

Le reti neurali sono in genere stratificate con un livello di input e output, ma possono esserci anche livelli nascosti. Tuttavia, il livello di input e il livello di output sono sempre presenti e possono essere incorporati nello stesso layer.

4.1.2 Risolvere un problema con le RN

Un programmatore di reti neurali, devi capire quali problemi sono adatti alle soluzioni di reti neurali e quali no. Inoltre, bisogna conoscere anche quale struttura di rete neurale è più applicabile a un determinato problema.

I programmi che possono essere facilmente scritti come diagrammi di flusso sono esempi di problemi per i quali le reti neurali non sono appropriate. Se un programma prevede passaggi ben definiti, saranno sufficienti le normali tecniche di programmazione.

Una delle caratteristiche principali delle reti neurali è la loro capacità di apprendimento. Per cui se l'algoritmo utilizzato per risolvere il problema utilizza

una regola immutabile, non vi è motivo di utilizzare una rete neurale. In effetti, potrebbe essere dannoso per l'applicazione se la rete neurale tenti di trovare una soluzione migliore e inizi a divergere dal processo desiderato e produca risultati inaspettati.

Infine, le reti neurali spesso non sono adatte a problemi in cui è necessario sapere esattamente come è stata derivata la soluzione. Una rete neurale può essere molto utile per risolvere il problema per il quale è stata addestrata, ma non può spiegare il suo ragionamento, la serie di serie di passaggi per ottenere la risposta.

Sebbene ci siano molti problemi per i quali le reti neurali non sono adatte, ci sono anche molti problemi per i quali una soluzione fornita da esse è piuttosto utile. Inoltre, le reti neurali possono spesso risolvere problemi con meno righe di codice rispetto a un algoritmo di programmazione tradizionale.

Le reti neurali sono particolarmente utili per risolvere problemi che non possono essere espressi come una serie di passaggi, come il riconoscimento di schemi, classificazione, previsione delle serie e data mining.

4.1.2.1 *Data Classification, Regression Analysis e Clustering*

Di seguito sono riportati i tre principali modelli problematici per le reti neurali (si parla di apprendimento supervisionato e non supervisionato, la cui trattazione è specificata nelle sezioni successive). [2]

Data Classification

La classificazione tenta di determinare in quale classe rientrano i dati di input. La classificazione è di solito un'operazione di addestramento supervisionato, il che significa che l'utente fornisce dati e risultati previsti alla rete neurale. Per la classificazione dei dati, il risultato atteso è l'identificazione della classe di dati.

Le reti neurali supervisionate sono sempre addestrate con dati noti. Durante l'addestramento, le reti vengono valutate su come classificano i dati noti. La

speranza è che la rete neurale, una volta addestrata, sia in grado di classificare anche dati sconosciuti.

Una classe è generalmente un attributo di dati non numerico e come tale, l'appartenenza alla classe deve essere ben definita, cioè: se una rete neurale viene addestrata, ad esempio, su tre tipi, non ci si può aspettare che identifichi un altro tipo. Tutti i membri della classe devono essere conosciuti al momento della formazione.

Regression Analysis

La regressione è concettualmente simile alla classificazione con la differenza che l'output non è semplicemente una classe, ma un numero che può assumere un numero illimitato (o comunque una grande quantità) di valori continui. Un esempio è la predizione del valore del tasso di cambio di una valuta nel futuro, dati i suoi valori in tempi recenti.

Clustering

Nel clustering un insieme di dati viene diviso in gruppi che però non sono noti a priori (nella classificazione lo sono). La natura stessa dei problemi appartenenti a questa categoria li rende tipicamente dei task di apprendimento non supervisionato.

4.1.3 Training e Validazione

I singoli neuroni che compongono una rete neurale sono interconnessi attraverso le loro sinapsi. Queste connessioni consentono ai neuroni di segnalarsi a vicenda e le informazioni vengono elaborate. A ogni connessione viene assegnato un peso di connessione, che determina l'output della rete neurale; pertanto si può dire che i pesi di connessione formano la memoria della rete neurale.

Il Training è il processo mediante il quale vengono assegnati questi pesi di connessione. La maggior parte degli algoritmi di training inizia assegnando numeri casuali a una matrice di pesi. Quindi, viene esaminata la validità della rete neurale. Successivamente, i pesi vengono regolati in base alla prestazione della rete

neurale e alla validità dei risultati. Questo processo viene ripetuto fino a quando l'errore di convalida non rientra in un limite accettabile.

Esistono molti modi per addestrare le reti neurali. I metodi di addestramento delle reti neurali generalmente rientrano nelle categorie di approcci supervisionati, non supervisionati e ibridi.

Una volta addestrata la rete, il passaggio finale, è la convalida di una rete neurale, che consente di determinare se è necessario un addestramento aggiuntivo.

Per validare correttamente una rete neurale, bisogna utilizzare dei dati di validazione (validation set) che devono essere diversi dai dati di addestramento (training set). [3] Se la rete neurale si comportasse male sul set di dati di validazione potrebbe significare che i pesi casuali iniziali non erano appropriati oppure che i dati di addestramento non siano stati scelti correttamente.

In alcune situazioni potrebbe essere impossibile raccogliere dati aggiuntivi da utilizzare come dati di addestramento o di convalida. In questo caso, non rimane altra scelta che combinare tutto o parte del set di validazione con il set di addestramento. Sebbene questo approccio rinuncerà alla sicurezza di una buona convalida, se non è possibile acquisire dati aggiuntivi, questa potrebbe essere l'unica alternativa.

4.2 Modellizzazione e Strutturazione

In questa sezione si accennerà alla modellazione di un neurone, i paradigmi di apprendimento e cenni sulle architetture delle reti neurali.

4.2.1 Modello di un neurone

Un neurone è l'unità di calcolo fondamentale della rete neurale ed è formato da tre elementi di base nel modello neurale:

- Un' insieme di sinapsi o connessioni ciascuna delle quali è caratterizzata da un peso; a differenza del modello umano, il modello artificiale può avere pesi sia negativi che positivi;
- un sommatore che somma i segnali in input pesati dalle rispettive sinapsi, producendo in output una combinazione lineare degli input;
- una funzione di attivazione per limitare l'ampiezza dell'output di un neurone.

Il modello neuronale include anche un valore soglia, denominato bias, che ha l'effetto, a seconda della sua positività o negatività, di aumentare o diminuire l'input netto alla funzione di attivazione. [1]

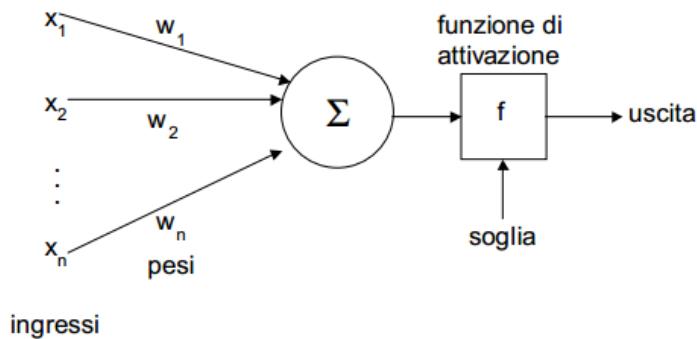


Figura 4.3 - Modello di un singolo neurone

4.2.1.1 *Funzioni di attivazione*

La maggior parte delle reti neurali passa l'output dei loro layer attraverso le funzioni di attivazione. Queste funzioni di attivazione scalano l'output della rete neurale in intervalli adeguati.

È possibile notare che sarebbe possibile utilizzare una diversa funzione di attivazione per ogni livello della rete neurale.

Sigmoide

Una funzione di attivazione sigmoide utilizza la funzione sigmoide per determinarne l'attivazione.

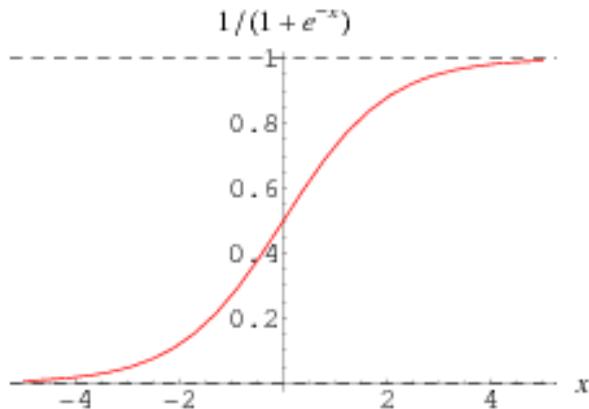


Figura 4.4 - Grafico della funzione lineare

Una cosa importante da notare su questa funzione di attivazione è che restituisce solo valori positivi. Se è necessaria la rete neurale per restituire numeri negativi, la funzione sigmoide non sarà adatta.

Alcuni metodi di allenamento richiedono la derivata della funzione di attivazione, ad esempio la backpropagation. Questa tecnica di allenamento non può essere utilizzata su una rete neurale che utilizza una funzione di attivazione che non ha una derivata. Tuttavia, è ancora possibile utilizzare un algoritmo genetico o un simulated annealing.

Tangente iperbolica

Come accennato in precedenza, la funzione di attivazione sigmoide non restituisce valori inferiori a zero. Tuttavia, è possibile "spostare" la funzione sigmoide in una regione del grafico in modo che fornisca numeri negativi. Questo viene fatto usando la funzione tangente iperbolica.

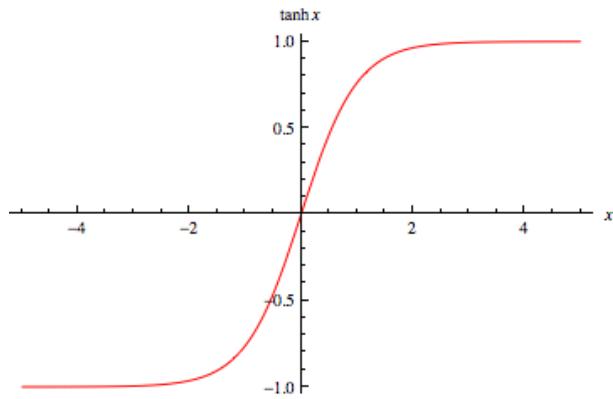


Figura 4.5 - Grafico della funzione tangente iperbolica

Sebbene appaia notevolmente più complesso della funzione sigmoide, può essere considerato come una versione compatibile positiva e negativa della funzione sigmoide. Per cui la funzione di attivazione tangente iperbolica restituisce valori positivi e negativi. Se è necessario per la rete neurale restituire numeri negativi allora questa è una delle funzioni di attivazione che si possono utilizzare.

Lineare

La funzione di attivazione lineare è essenzialmente nessuna funzione di attivazione. È probabilmente la meno utilizzata delle funzioni di attivazione. Essa non modifica un modello prima di emetterlo.

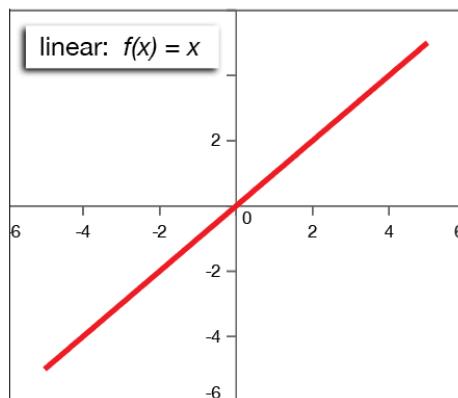


Figura 4.6 - Grafico della funzione lineare

La funzione di attivazione lineare potrebbe essere utile in situazioni in cui è necessario emettere l'intero intervallo di numeri. Tipicamente, si può pensare ai

neuroni come attivi o non attivi. Poiché le funzioni di attivazione tangente iperbolica e sigmoide hanno entrambe stabilito limiti superiori e inferiori, tendono ad essere utilizzate maggiormente per le operazioni di tipo booleano. La funzione di attivazione lineare non fa altro che restituire ciò che viene dato.

Non è possibile utilizzare la tecnica di backpropagation per addestrare una rete neurale che utilizza la funzione lineare (derivata della funzione lineare è pari ad 1).

4.2.2 Paradigmi di apprendimento

L' apprendimento è un processo molto importante per una rete neurale. Esistono due forme di formazione che possono essere impiegate, supervisionate e non supervisionate. L' addestramento supervisionato (Supervised Training) prevede di fornire alla rete neurale set di addestramento e risultati previsti. Invece nell' addestramento senza supervisione (Unsupervised Training), la rete neurale è anche dotata di set di addestramento, ma non di risultati previsti. Questa sezione fornirà una breve descrizione di ciascun approccio. [2]

4.2.2.1 *Addestramento non supervisionato*

Come accennato in precedenza, vengono definiti dei set di addestramento, che vengono forniti in input alla rete neurale. La rete neurale non supervisionata non conosce quali sono gli output anticipati corrispondenti ai set di dati di addestramento dati in input.

Questa tipologia di training viene in genere utilizzato per addestrare le reti neurali che rispondono a problemi di classificazione. Una rete neurale di classificazione riceve schemi di input, che vengono presentati ai neuroni di input. Questi schemi di input vengono quindi elaborati, provocando l'attivazione di un singolo neurone sul livello di output. Questo neurone attivo fornisce la classificazione per lo schema e identifica a quale gruppo appartiene lo schema.

Un'altra applicazione comune per l'addestramento senza supervisione è il data mining. In questo caso, si ha una grande quantità di dati da elaborare e non si

conosce bene l'output. Si vuole che la rete neurale classifichi questi dati in diversi gruppi. Non si fornisce in anticipo alla rete neurale quale modello di input debba essere classificato in quale gruppo. Mentre la rete neurale si allena, i modelli di input si dividono in gruppi con altri input con caratteristiche simili. Ciò consente di vedere quali schemi di input condividono somiglianze.

4.2.2.2 *Addestramento supervisionato*

Il metodo di addestramento supervisionato è simile al metodo di addestramento non supervisionato, in quanto vengono forniti set di addestramento. Proprio come con l'addestramento senza supervisione, questi insiemi di addestramento specificano i segnali di input alla rete neurale. La differenza principale è che nell'addestramento supervisionato vengono forniti i risultati previsti, ciò consente alla rete neurale di regolare i valori nella matrice del peso in base alle differenze tra l'uscita prevista e l'uscita effettiva.

Esistono diversi algoritmi di addestramento supervisionato popolari. Uno dei più comuni è l'algoritmo di backpropagation. È anche possibile utilizzare simulated annealing o algoritmi genetici per implementare l'addestramento supervisionato.

4.2.3 *Cenni sui tipi di architetture*

Esistono due categorie principali di architetture di rete che dipendono dal tipo di connessioni tra i neuroni: reti feed-forward e reti ricorrenti. Se non vi è alcun feedback dalle uscite dei neuroni verso gli ingressi in tutta la rete, la rete viene definita rete neurale feed-forward. Altrimenti, se esiste un tale feedback, cioè una connessione sinaptica dalle uscite verso gli ingressi (o i loro ingressi o gli ingressi di altri neuroni), la rete viene chiamata rete neurale ricorrente.

4.2.3.1 *Reti Feed-forward*

La rete neurale feed-forward è il tipo di rete neurale artificiale più semplice. In questa rete, le informazioni si spostano in una sola direzione, in avanti, dai nodi di

input, attraverso i nodi nascosti (se presenti) e ai nodi di output. Non ci sono cicli o loop nella rete.

Tipicamente le reti feed-forward ricadono in due categorie in dipendenza dal numero di livelli: ad un livello (o single layer) o a più livelli (multi-layer).

- **Single Layer** - in questa struttura abbiamo i nodi di input (input layer) e uno strato di neuroni (output layer). Il segnale nella rete si propaga in avanti in modo aciclico, partendo dal layer di input e terminando in quello di output tramite una serie di pesi. La somma dei prodotti dei pesi e degli input viene calcolata in ciascun nodo e se il valore è al di sopra di una soglia (tipicamente 0) il neurone spara e assume il valore attivato (tipicamente 1); altrimenti prende il valore disattivato (in genere -1). In letteratura il termine percettrone si riferisce spesso a reti costituite solo da una di queste unità.

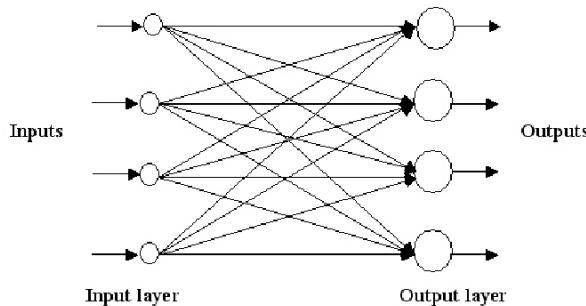


Figura 4.7 - Illustrazione di una rete feed-forward single layer

- **Multi-Layer** - Una rete neurale feedforward multi-layer si distingue dalla precedente dal fatto che tra il livello di input e quello di output si possono avere uno o più livelli nascosti (hidden layer). Ogni livello ha connessioni entranti dal livello precedente e uscenti in quello successivo, per cui la propagazione del segnale avviene in avanti senza cicli e connessioni trasversali.

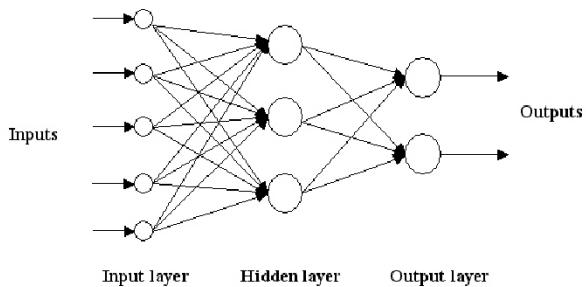


Figura 4.8 - Illustrazione di una rete feed-forward multi-layer

Per definire una rete feed-forward bisogna stabilire quanti neuroni saranno presenti nei livelli di input e output; quanti livelli nascosti utilizzare e quanti neuroni ci saranno in ciascuno di essi.

Input Layer

Il livello di input è il condotto attraverso il quale l'ambiente esterno presenta un modello alla rete neurale. Una volta che un modello viene presentato al livello di input, il livello di output produrrà un altro pattern. In sostanza, questo è tutto ciò che fa la rete neurale. Il livello di input dovrebbe rappresentare la condizione per la quale stiamo addestrando la rete neurale. Ogni neurone di input dovrebbe rappresentare una variabile indipendente che ha un'influenza sull'output della rete neurale.

È importante ricordare che gli ingressi alla rete neurale sono numeri in virgola mobile. Questo non vuol dire che è possibile elaborare solo dati numerici con la rete neurale; se si desidera elaborare una forma di dati non numerica, è necessario sviluppare un processo che normalizzi questi dati in una rappresentazione numerica.

Output Layer

Il livello di output della rete neurale è ciò che presenta effettivamente un modello per l'ambiente esterno. Il modello presentato dal livello di output può essere ricondotto direttamente al livello di input.

Per determinare il numero di neuroni da utilizzare nel livello di output, è necessario innanzitutto considerare l'uso previsto della rete neurale. Se la rete neurale deve essere utilizzata per classificare gli elementi in gruppi, è spesso preferibile avere un neurone di uscita per ciascun gruppo in cui devono essere assegnati gli elementi di input. Se la rete neurale deve eseguire la riduzione del rumore su un segnale, è probabile che il numero di neuroni in ingresso corrisponda al numero di neuroni in uscita. In questo tipo di rete neurale, quello che si vuole è che i modelli lascino la rete neurale nello stesso formato in cui sono entrati.

Hidden Layer

Per quanto riguarda i livelli nascosti occorre determinare:

- quanti strati nascosti devono effettivamente avere nella rete neurale
- quanti neuroni saranno presenti in ciascuno di questi strati.

Raramente si verificano problemi che richiedono due livelli nascosti. Tuttavia, le reti neurali con due livelli nascosti possono rappresentare funzioni con qualsiasi tipo di forma.

Decidere il numero di neuroni negli strati nascosti è una parte molto importante nel decidere la struttura globale della rete neurale. [4] Sebbene questi layer non interagiscano direttamente con l'ambiente esterno, hanno un'enorme influenza sull'output finale. Sia il numero di strati nascosti che il numero di neuroni in ciascuno di questi strati nascosti devono essere attentamente considerati.

L'uso di un numero troppo limitato di neuroni negli strati nascosti comporterà qualcosa chiamato underfitting. Esso si verifica quando ci sono pochi neuroni negli strati nascosti per rilevare adeguatamente i segnali in un set di dati complicato.

L'uso di troppi neuroni negli strati nascosti può causare diversi problemi. In primo luogo, troppi neuroni negli strati nascosti possono causare un eccesso di adattamento. L'overfitting si verifica quando la rete neurale ha così tanta capacità di elaborazione delle informazioni che la quantità limitata di informazioni contenute nel set di addestramento non è sufficiente per addestrare tutti i neuroni

negli strati nascosti. Un secondo problema può verificarsi anche quando i dati di allenamento sono sufficienti.

Un numero eccessivamente elevato di neuroni negli strati nascosti può aumentare il tempo necessario per addestrare la rete. La quantità di tempo di addestramento può aumentare al punto che è impossibile addestrare adeguatamente la rete neurale. Ovviamente, è necessario raggiungere un compromesso tra troppi e troppo pochi neuroni negli strati nascosti.

Esistono molti metodi empirici per determinare il numero corretto di neuroni da utilizzare negli strati nascosti, come i seguenti:

- Il numero di neuroni nascosti dovrebbe essere compreso tra la dimensione del livello di input e la dimensione del livello di output.
- Il numero di neuroni nascosti dovrebbe essere pari a $2/3$ della dimensione del livello di input, più la dimensione del livello di output.
- Il numero di neuroni nascosti dovrebbe essere inferiore al doppio della dimensione del livello di input.

Backpropagation

Per determinare i valori dei pesi e della soglia si utilizza l'algoritmo di backpropagation. Questo è un algoritmo molto utile per allenare le reti neurali. L'algoritmo funziona eseguendo la rete neurale presentandole i dati di training. Poiché ogni elemento di dati di allenamento viene presentato alla rete neurale, viene calcolato l'errore tra l'output effettivo della rete neurale e l'output previsto (e specificato nel set di training). I pesi e la soglia vengono quindi modificati in accordo a tale errore, in maniera tale si ha una maggiore possibilità che la rete restituisca il risultato corretto alla successiva presentazione della stessa rete con lo stesso input. Per addestrare la rete neurale, dobbiamo cercare di ridurre al minimo questo errore (una tecnica è il metodo di discesa gradiente)

Per addestrare la rete neurale, è necessario determinare un metodo per calcolare l'errore. Man mano che la rete neurale viene addestrata, la rete viene presentata con campioni dal set di addestramento. Il risultato ottenuto dalla rete neurale

viene quindi confrontato con il risultato atteso che fa parte del set di addestramento. Il grado in cui l'uscita dalla rete neurale differisce da questa uscita prevista è l'errore.

La backpropagation può essere utilizzata con qualsiasi rete feed-forward che utilizza una funzione di attivazione differenziabile.

4.2.3.2 Reti Ricorrenti

Una rete neurale ricorrente (RNN) è una classe di reti neurali artificiali in cui le connessioni tra i nodi formano un grafo diretto lungo una sequenza temporale. Ciò gli consente di esibire un comportamento dinamico temporale. Le RNN possono utilizzare il loro stato interno (memoria) per elaborare sequenze di input di lunghezza variabile. Ciò le rende applicabili a compiti come ad esempio il riconoscimento vocale.

Il termine "rete neurale ricorrente" è usato indiscriminatamente per riferirsi a due ampie classi di reti con una struttura generale simile [5]:

- Una rete ricorrente a impulso finito è un grafo aciclico diretto che può essere srotolato e sostituito con una rete neurale strettamente feedforward;
- Una rete ricorrente a impulso infinito è un grafo ciclico diretto che non può essere srotolato.

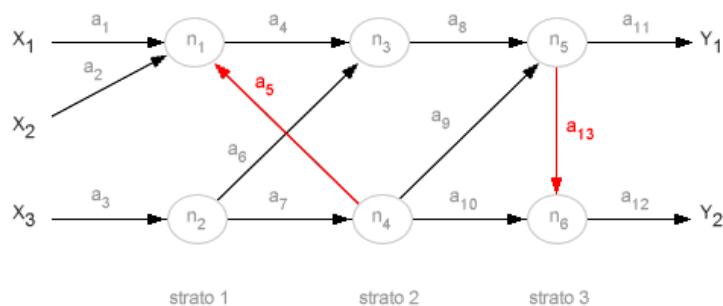


Figura 4.9 - Illustrazione di una rete ricorrente

Nelle reti ricorrenti sono previste connessioni di feedback (in genere verso neuroni dello stesso livello, ma anche all'indietro). Questo complica notevolmente il flusso delle informazioni e l'addestramento, richiedendo di considerare il comportamento in più istanti temporali (*unfolding in time*).

5. Analisi Forense delle Attività delle App di Instant Messaging

L'analisi forense delle attività delle applicazioni di Instant Messaging è onerosa e non banale da effettuare. Le informazioni e/o i pattern trovati, osservando il traffico di rete acquisito, possono portare a determinare delle evidenze digitali, andando ad arricchire il quadro investigativo.

In questo capitolo inizialmente verrà presentata il rapporto tra la Network Forensics e l'Instant Messaging, sezione 5.1; a seguire, nella sezione 5.2, si discuterà di un caso di studio riguardante il profiling di alcune attività utente su WhatsApp e Instagram.

5.1 La Network Forensics e l'Instant Messaging

Gli investigatori e gli analisti stanno sperimentando sempre più grandi set di dati quando conducono indagini sulla criminalità informatica, utilizzando le tecniche di Network Forensics e di sniffing.

L'analisi dei registri delle chiamate è una delle strategie di indagine criminale critica per le forze dell'ordine (LEA). I registri delle chiamate forniscono informazioni importanti, quali date, orari e durata delle chiamate in uscita e in entrata, che è estremamente rilevante per l'esame della scena del crimine [1]. Tuttavia, l'ubiquità delle app di messaggistica istantanea sugli smartphone ha fornito ai criminali la possibilità di poter comunicare con canali che sono difficili da tracciare utilizzando le tradizionali tecnologie di indagine. Al giorno d'oggi, la maggior parte dei criminali utilizza app di messaggistica istantanea invece di telefonate vocali per evitare di essere presi di mira dai LEA. Trovare l'identità del cyber-criminale senza l'aiuto dell'autorità straniera è difficile su Internet, il che fornisce anonimato e privacy completa [2]. Lo sviluppo di nuove tecniche per analizzare i record delle chiamate moderne è un compito urgente.

La principale difficoltà nel recuperare informazioni preziose da app di messaggistica istantanea specifiche è come filtrare i record delle connessioni di rete su Internet. I dati grezzi acquisiti da Internet sono pieni di pacchetti prodotti da diverse app da vari dispositivi. I protocolli, le porte, le frequenze di connessione sono tutte diverse tra le diverse app. Inoltre, gli smartphone possono persino stabilire connessioni attraverso le sue diverse interfacce di rete. Sebbene il recupero dei registri delle chiamate o dei registri delle connessioni di rete dagli smartphone sia una grande sfida, i suoi dati sono in grado di fornire informazioni più avanzate e dettagliate rispetto ai registri telefonici tradizionali. Ad esempio, l'indirizzo IP rivela le posizioni delle chiamate e i pacchetti di rete acquisiti forniscono il contenuto multimediale delle comunicazioni.

Gli analizzatori di pacchetti vengono applicati spesso nel campo della Network Security per analizzare il traffico non elaborato, rilevare attacchi, sniffing e risoluzione dei problemi di rete [3].

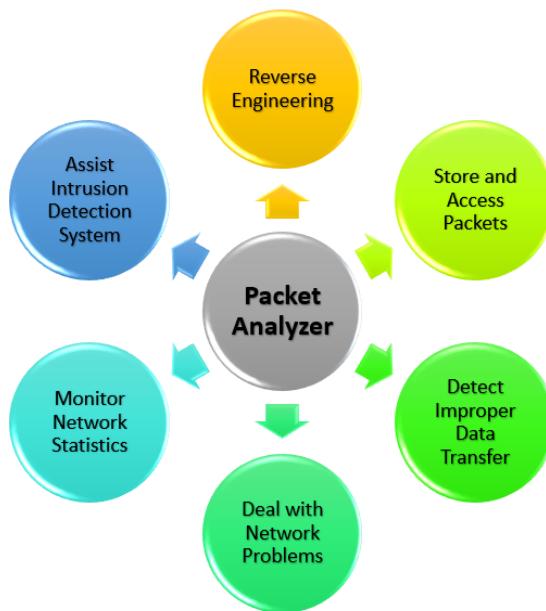


Figura 5.1 - Illustrazioni delle funzioni dell'analizzatore di pacchetti

L'analizzatore di pacchetti aiuta a eseguire un controllo di sicurezza attraverso i pacchetti; per gli amministratori di rete, diventa uno strumento per diagnosticare i problemi in una rete; per i white hat, aiutano a trovare le vulnerabilità delle

applicazioni software; per gli sviluppatori di protocolli, possono essere utilizzati per diagnosticare problemi relativi al protocollo.

L'analizzatore di pacchetti può anche essere utilizzato in modo immorale, ad esempio ispezionando il payload dei pacchetti per decrittografare le password o sniffare il traffico per inscenare un attacco man-in-the-middle.

L'analisi dei pacchetti viene in genere eseguita da uno sniffer di pacchetti, che è uno strumento utilizzato per acquisire dati di rete grezzi che attraversano le interfacce cablate o wireless. Essa può aiutare a comprendere le caratteristiche della rete, determinare chi o cosa sta utilizzando la larghezza di banda disponibile, trovare applicazioni non sicure, identificare i tempi di utilizzo della rete o capire attività dannose. Esistono vari tipi di programmi di sniffing dei pacchetti, inclusi quelli gratuiti e commerciali. Ogni programma è progettato con diversi obiettivi in mente. Alcuni programmi di analisi di pacchetti popolari sono Tcpdump e Wireshark. Tcpdump è un programma da riga di comando, mentre Wireshark ha interfacce utente grafica [4].

5.2 Caso di studio: profiling delle attività dell'app IM

In questa sezione si discuterà un lavoro di analisi forense denominato “Network Forensics di WhatsApp e Instagram: Profiling delle principali attività dei Cybercriminali” [5].

L'obiettivo del progetto è quello di analizzare il traffico di rete generato dalle app WhatsApp e Instagram in seguito a delle attività di un utente, delineando metodologie e pattern che considerino le sole informazioni a disposizione (in quanto il traffico è tipicamente cifrato), in maniera tale da individuare evidenze che possano arricchire il quadro investigativo di una indagine per un cyber-reato (o per un reato in cui sono coinvolti dispositivi informatici).

5.2.1 Le applicazioni WhatsApp e Instagram

In questa sezione si darà un'infarinatura sulle applicazioni WhatsApp e Instagram.

5.2.1.1 *L'app WhatsApp*

Uno studio della compagnia di analisi di mercato SimilarWeb (Aprile 2016) evidenzia che WhatsApp è l'app di messaggistica più usata su dispositivi Android. Infatti, WhatsApp risulta l'app prima in classifica in 109 paesi diversi al mondo, Italia inclusa.

Attualmente, WhatsApp vanta di oltre 1,5 miliardi di utenti attivi al mese, i quali si scambiano circa 60 miliardi di messaggi giornalmente [6]. In virtù dell'enorme numero di utenti e delle interazioni tra gli stessi, è possibile affermare che WhatsApp sia uno dei più grandi servizi sociali esistenti al giorno d'oggi.

Tramite questa applicazione gli utenti possono scambiarsi: messaggi, foto, video, registrazioni messaggi vocali e file.

Tutto avviene utilizzando la rete ed utilizzando come identificativo il proprio numero di telefono. È inoltre possibile creare chat di gruppo per messaggiare con più persone contemporaneamente. WhatsApp utilizza la crittografia end-to-end per messaggi, foto, video, messaggi vocali, documenti, aggiornamenti allo stato e chiamate, in modo tale che terze parti non possano leggere i contenuti scambiati tra due persone. La crittografia end-to-end assicura che solo il legittimo mittente e destinatario possano leggere ciò che viene inviato; neppure i server WhatsApp hanno accesso alle chiavi private degli utenti. Il protocollo Voice over Internet Protocol (VoIP) è uno degli standard più popolari per le chiamate vocali e video sul Web. Questo protocollo VoIP viene utilizzato da varie piattaforme come WhatsApp, Skype, Messenger, Facebook ecc. Sia la chiamata vocale che la videochiamata dipendono dal modo in cui si trasmettono i media tra i due client collegati tra loro, compito destinato al WebRTC.

Solo WebRTC non basta per completare l'implementazione, servono anche altri componenti, quali: Signaling Server, TURN Server e STUN Server.

5.2.1.2

L'app Instagram

Instagram ha modificato il modo di comunicare sui social media. Esso è focalizzato sulla condivisione di foto fatte da smartphone (e non solo). L'app ha riscosso un notevole successo e si è diffusa su una vasta base di utenti al punto di suscitare interesse nell'azienda Facebook, che la ha poi acquisita nel 2012. Il 32% degli utenti Internet utilizza Instagram. Sono 77,6 milioni di utenti di Instagram che provengono dagli Stati Uniti, ma l'80% degli utenti di Instagram si trova fuori degli USA. Instagram viene perlopiù utilizzato da giovani nella fascia di età tra i 18 e 29 anni, come dimostrato da uno studio del Pew Research Center [7].

L'app permette di scattare foto e realizzare video, applicando eventualmente filtri in tempo reale prima di condividerli. Tra gli altri strumenti di Instagram che si possono utilizzare vi sono le Storie e le Dirette. Un'altra opzione è quella dei Direct, ovvero i messaggi privati relativi alla chat. Inoltre, è possibile condividere le foto, video con chiunque mandandoli in privato. Come protocollo di cifratura, Instagram si basa su TLS.

5.2.2 Metodologia della sperimentazione

Sono stati definiti cinque Test Case, i quali simulano cinque diverse situazioni realistiche (come ad esempio l'utilizzo di VPN ecc.) in cui degli utenti malintenzionati e/o cybercriminali potrebbero agire per compiere azioni malevoli.

Di seguito descriviamo lo scenario di acquisizione relativo ad un Test Case:

- un utente A, che svolge il ruolo di inviante, che utilizza lo smartphone connesso alla rete hotspot del pc PC1;
- Il PC1 è connesso al modem wireless (fastweb) e viene utilizzato come punto di intercettazione del traffico;
- un utente B, che svolge il ruolo di ricevente, che utilizza lo smartphone connesso alla rete hotspot del pc PC2;

- Il PC2 è connesso al modem wireless (alice) e viene utilizzato come punto di intercettazione del traffico;

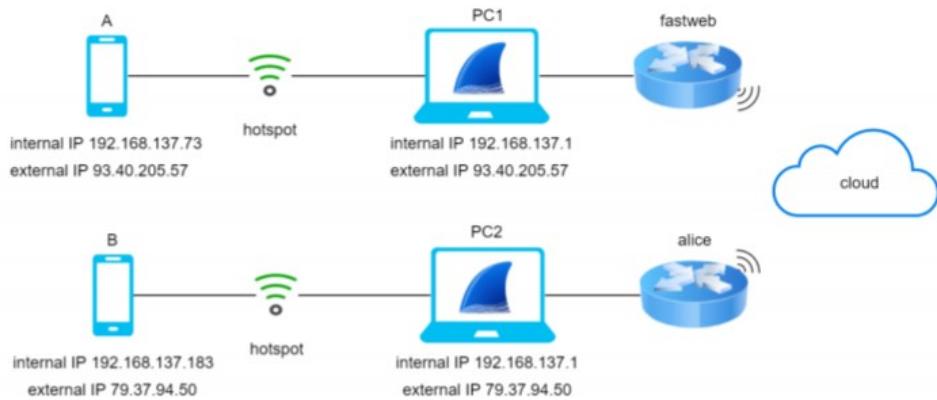


Figura 5.2 - Scenario di rete di riferimento per l'acquisizione del traffico

In questi scenari di rete l'utente/cyber-criminale compiono varie attività utilizzando l'app di messaggistica WhatsApp e Instagram.

L'acquisizione del traffico di rete avviene utilizzando il tool Wireshark e sono stati memorizzati nel formato PCAP. Per analizzare le informazioni a disposizione e effettuare un profiling sul tipo di attività svolta sono stati utilizzati filtri di pacchetti e grafici I/O. Inoltre, si è utilizzato anche il tool Xplico per una detection sulla tipologia dei dati catturati.

Di seguito sono riportate le varie attività utente svolte relativamente con le app di IM, le quali sono state soggette di un'analisi forense.

WhatsApp:

- Chiamate effettuate/ricevute
- Rifiuto/terminazione di chiamata
- Chiamate con blocco
- Chiamate di gruppo

Instagram:

- Interazioni utente (avvio/chiusura dell'app, like e unlike su una foto/video, commento di una foto/video, invio/ricezione di un messaggio testuale in Direct)
- Scambio di elementi multimediali in Direct (foto, video, audio)
- Videochiamata

A proseguire sono riportate come esempio di analisi, l'attività relativa al rifiuto/terminazione di una chiamata vocale utilizzando l'app di WhatsApp e l'invio/ricezione di foto e video in Direct utilizzando l'app di Instagram.

5.2.2 Esempio di Analisi Forense Attività WhatsApp

Il caso preso in esame riguarda l'individuazione di evidenze, relative al rifiuto/terminazione chiamata (ad esempio, modello per distinguere il rifiuto da una terminazione, ecc.), mediante l'analisi forense del traffico di rete. In particolare, si analizzeranno i pacchetti di rete, acquisiti (e memorizzati in file PCAP) da una fase di sperimentazione che è descritta precedentemente.

Con questo tipo di esperimento si vuole simulare situazione un'altra realistica dove il sospetto effettua una chiamata WhatsApp alla vittima che rifiuta la chiamata. Inoltre, si verifica cosa succede quando, durante la fase di ringing, è stesso il sospetto a terminare la chiamata prima che la vittima risponda. In ultimo si verifica anche cosa succede se si termina la chiamata dopo aver risposto.

Sono stati effettuate un paio di chiamate dove il chiamante è A e il chiamato è B e viceversa. Tutti gli esperimenti hanno dato lo stesso risultato su entrambi i terminali come mostrato di seguito.

No.	Time	Source	Destination	Protocol	Length	Info
L	365 88.202648	192.168.137.73	31.13.86.48	STUN	168	Unknown Request
	366 88.202649	192.168.137.73	185.60.216.51	STUN	168	Unknown Request
	367 88.202649	192.168.137.73	179.60.192.48	STUN	168	Unknown Request
	368 88.202650	192.168.137.73	31.13.92.50	STUN	168	Unknown Request
	369 88.203524	192.168.137.73	157.240.20.51	STUN	168	Unknown Request

Figura 5.3 -Elenco dei pacchetti acquisiti filtrati durante un esperimento del dispositivo A

Source 192.168.137.73 = A internal IP / Destination: diversi IP di ISP WhatsApp

No.	Time	Source	Destination	Protocol	Length	Info
89	5.110643	192.168.137.183	31.13.86.48	STUN	168	Unknown Request
90	5.110645	192.168.137.183	185.60.216.51	STUN	168	Unknown Request
91	5.111384	192.168.137.183	179.60.192.48	STUN	168	Unknown Request
92	5.111645	192.168.137.183	31.13.92.50	STUN	168	Unknown Request
93	5.111856	192.168.137.183	157.240.20.51	STUN	168	Unknown Request

Figura 5.4 - Elenco dei pacchetti acquisiti filtrati durante un esperimento del dispositivo B

Source: 192.168.137.183 = B internal IP / Destination: diversi indirizzi IP di ISP WhatsApp

Da entrambe le parti si può osservare che a prescindere da chi termina o rifiuta la chiamata, questa azione provoca l'invio di cinque pacchetti sequenziali STUN di tipo Unknown Request da entrambi i dispositivi, inoltre nel campo source compare sempre A o sempre B, nel campo destination compaiono sempre gli ISP.

Da vari test case si evince che colui che terminata la chiamata invia per primo i pacchetti STUN Unknown Request, l'altro interlocutore invia gli stessi pacchetti circa 1 secondo dopo.

Questi pacchetti non danno nessuna informazione su chi ha rifiutato o terminato la chiamata, inoltre, non si riesce a distinguere se la chiamata è stata rifiutata senza risposta (assenza di flusso audio) o terminata dopo aver risposto (presenza di flusso audio).

5.2.3.1 *Distinzione tra chiamata con o senza flusso audio*

Una possibile fase che distingue se la chiamata è stata rifiutata senza risposta (assenza di flusso audio) o terminata dopo aver risposto (presenza di flusso audio), è quella immediatamente prima del rifiuto/terminazione, ovvero, fase di ringing (prima del rifiuto), oppure, fase di flusso audio (prima della terminazione).

Esaminando meglio il file PCAP, sia di una chiamata con flusso audio che non, si nota che prima dell'invio dei pacchetti STUN di tipo Unknown Request vengono inviati dei pacchetti UDP con caratteristiche diverse. In una chiamata con flusso

audio i pacchetti UDP sono di lunghezza 2 o 6 (Figura 5.5) e identificano, molto probabilmente, la fase di ringing. Quando, invece, viene accettata la sessione, i pacchetti UDP sono di maggiore lunghezza (Figura 5.6) e indentificano, molto probabilmente, il flusso audio. Grazie a questa discrepanza si può capire con più facilità se la terminazione di chiamata è avvenuta con o senza flusso audio.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	31.13.86.48	192.168.137.73	UDP	44	3478 + 63582 Len=2
2	0.778773	192.168.137.73	31.13.86.49	TCP	91	49338 + 5222 [PSH, ACK] Seq=1 Ack=1 Win=2048 Len=25 TSval=41729695 TSecr=3847341120 [TCP
3	0.889158	31.13.86.49	192.168.137.73	TCP	66	5222 + 49338 [ACK] Seq=1 Ack=26 Win=159 Len=8 TSval=3847344983 TSecr=41729695
4	0.983668	31.13.86.49	192.168.137.73	TCP	182	5222 + 49338 [PSH, ACK] Seq=1 Ack=26 Win=159 Len=26 TSval=3847345091 TSecr=41729695 [TCP
5	0.987083	192.168.137.73	31.13.86.49	TCP	66	49338 + 5222 [ACK] Seq=26 Ack=37 Win=8 TSval=41729987 TSecr=3847345091
6	2.842264	192.168.137.73	239.255.255.250	SSDP	175	H-SEARCH * HTTP/1.1
7	2.999538	192.168.137.73	31.13.86.48	UDP	48	63582 + 3478 Len=6
8	3.181996	31.13.86.48	192.168.137.73	UDP	44	3478 + 63582 Len=2
9	5.846898	192.168.137.73	239.255.255.250	SSDP	143	H-SEARCH * HTTP/1.1
10	6.426151	192.168.137.73	31.13.86.48	UDP	48	63582 + 3478 Len=6
11	6.946388	31.13.86.48	192.168.137.73	UDP	44	3478 + 63582 Len=2
12	8.753531	192.168.137.73	224.0.0.251	DNS	128	Standard query 0x0000 PTR _raop._tcp.local, "QH" question PTR _airplay._tcp.local, "QH"
13	8.755534	Fe:00:00:00:00:cd6	f-[REDACTED]	DNS	148	Standard query 0x0000 PTR _raop._tcp.local, "QH" question PTR _airplay._tcp.local, "QH"
14	9.848545	192.168.137.73	31.13.86.48	UDP	48	63582 + 3478 Len=6
15	9.872489	31.13.86.48	192.168.137.73	UDP	44	3478 + 63582 Len=2
16	10.995616	192.168.137.73	31.13.86.49	TCP	54	[TCP Keep-Alive] 49338 + 5222 [ACK] Seq=25 Ack=37 Win=2048 Len=0
17	11.823658	31.13.86.49	192.168.137.73	TCP	66	[TCP Keep-Alive ACK] 5222 + 49338 [ACK] Seq=37 Ack=26 Win=159 Len=0 TSval=3847355199 TSe
22	12.153180	192.168.137.73	31.13.86.48	UDP	48	63582 + 3478 Len=6
23	12.173864	31.13.86.48	192.168.137.73	UDP	44	3478 + 63582 Len=2
24	15.169914	192.168.137.73	31.13.86.48	UDP	48	63582 + 3478 Len=6
25	15.191155	31.13.86.48	192.168.137.73	UDP	44	3478 + 63582 Len=2

Figura 5.5 - Elenco (1) dei pacchetti acquisiti durante un esperimento del dispositivo A

No.	Time	Source	Destination	Protocol	Length	Info
35	17.342429	31.13.86.48	192.168.137.73	UDP	158	3478 + 57461 Len=108
36	17.407683	31.13.86.49	192.168.137.73	TCP	115	5222 + 49339 [PSH, ACK] Seq=1 Ack=129 Win=147 Len=49 TS
37	17.411156	192.168.137.73	31.13.86.49	TCP	66	49339 + 5222 [ACK] Seq=129 Ack=50 Win=2047 Len=0 TSval=
38	17.411537	192.168.137.73	31.13.86.48	UDP	263	57461 + 3478 Len=221
39	17.488766	31.13.86.48	192.168.137.73	UDP	159	3478 + 57461 Len=117
40	17.532572	31.13.86.48	192.168.137.73	UDP	164	3478 + 57461 Len=122
41	17.573585	192.168.137.73	31.13.86.48	UDP	217	57461 + 3478 Len=175
42	17.622423	31.13.86.48	192.168.137.73	UDP	157	3478 + 57461 Len=115
43	17.695763	31.13.86.49	192.168.137.73	TCP	327	5222 + 49339 [PSH, ACK] Seq=50 Ack=129 Win=147 Len=261
44	17.698637	192.168.137.73	31.13.86.49	TCP	66	49339 + 5222 [ACK] Seq=129 Ack=311 Win=2043 Len=0 TSval=
47	17.706793	Apple_00:[REDACTED]	Broadcast	ARP	42	Who has 192.168.137.183? Tell 192.168.137.73
48	17.707253	192.168.137.73	31.13.86.49	TCP	199	49339 + 5222 [PSH, ACK] Seq=129 Ack=311 Win=2048 Len=13
51	17.729237	31.13.86.49	192.168.137.73	TCP	66	5222 + 49339 [ACK] Seq=311 Ack=262 Win=151 Len=0 TSval=
52	17.731942	192.168.137.73	31.13.86.48	UDP	158	57461 + 3478 Len=116
53	17.783105	31.13.86.48	192.168.137.73	UDP	168	3478 + 57461 Len=118
55	17.894197	192.168.137.73	31.13.86.48	UDP	164	57461 + 3478 Len=122
56	17.945388	31.13.86.48	192.168.137.73	UDP	168	3478 + 57461 Len=118
57	18.053863	192.168.137.73	31.13.86.48	UDP	157	57461 + 3478 Len=115
58	18.123989	192.168.137.73	31.13.86.48	UDP	48	57461 + 3478 Len=6
59	18.125013	31.13.86.48	192.168.137.73	UDP	152	3478 + 57461 Len=110
60	18.143871	31.13.86.48	192.168.137.73	UDP	44	3478 + 57461 Len=2
61	18.200352	31.13.86.48	192.168.137.73	UDP	68	3478 + 57461 Len=26
62	18.214644	192.168.137.73	31.13.86.48	UDP	157	57461 + 3478 Len=115

Figura 5.6 - Elenco (2) dei pacchetti acquisiti durante un esperimento del dispositivo A

Nelle figure seguenti viene mostrato il grafico I/O delle due chiamate in questione, in cui vengono rappresentati il numero di pacchetti sull'asse delle ordinate e la linea temporale sull'asse ascisse. Per una facile lettura del grafico si utilizza un filtro per i pacchetti TCP di colore viola e uno per i pacchetti UDP di colore azzurro che sono gli elementi sostanziali per ripercorrere tutte le fasi della chiamata.

Chiamata con flusso audio

Grafico rappresentante il numero di pacchetti TCP e UDP, di una chiamata con flusso audio, catturati durante un esperimento, tramite PC1, del dispositivo A.

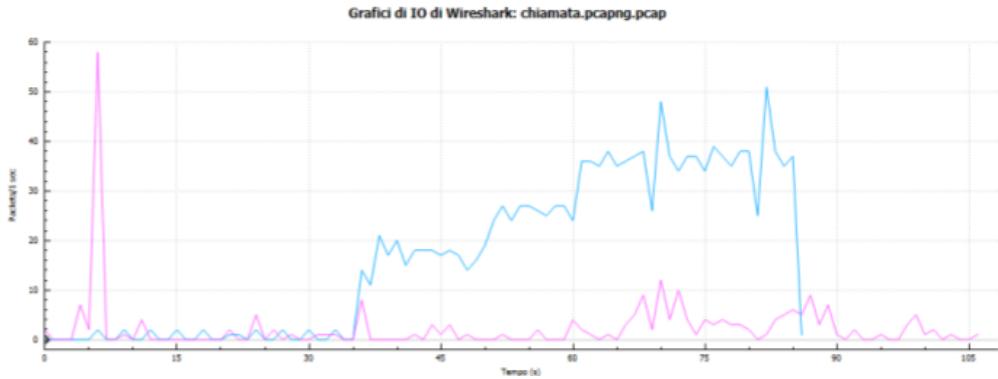


Figura 5.7 - Grafico I/O del numero di pacchetti della chiamata con flusso audio del dispositivo A

Intervallo	Fase	Descrizione
5 - 7 sec.	Setup	Picco TCP in cui viene instaurata una sessione tra A e B
8 - 35 sec.	Ringing	Invio pacchetti UDP di lunghezza 2 o 6
35 - 36 sec.	Accettazione	Breve picco TCP in cui viene accettata la sessione
36 - 85 sec.	Flusso Audio	Alti picchi UDP che identificano il flusso audio
86 - 87 sec.	Terminazione	A termina la chiamata, invio di pacchetti STUN <i>Unknown Request</i>

Figura 5.8 - Fasi della chiamata con flusso audio

Chiamata senza flusso audio

Grafico rappresentante il numero di pacchetti TCP e UDP, di una chiamata senza flusso audio, catturati durante un esperimento, tramite PC1, del dispositivo A.

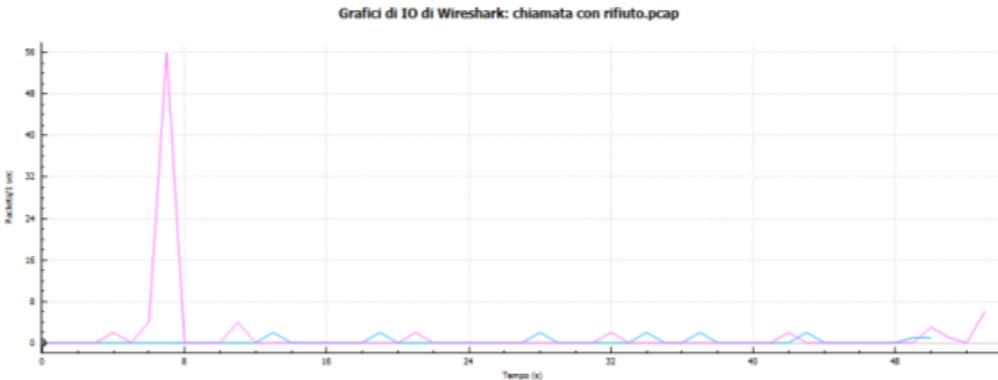


Figura 5.9 - Grafico I/O del numero di pacchetti chiamata senza flusso audio del dispositivo A

Intervallo	Fase	Descrizione
6 - 8 sec.	Setup	Picco TCP in cui viene instaurata una sessione tra A e B
8 - 50 sec.	Ringing	Invio pacchetti UDP di lunghezza 2 o 6
50 - 51 sec.	Terminazione	B rifiuta la chiamata, invio di pacchetti STUN <i>Unknown Request</i>

Figura 5.10 - Fasi della chiamata senza flusso audio

Dai grafici si nota chiaramente che il numero dei pacchetti UDP aumenta in modo notevole durante la fase di flusso audio, invece resta basso nella fase di ringing. Si può quindi provare che se prima della fase di terminazione ci sono pochi pacchetti UDP di breve lunghezza allora la vittima ha rifiutato la chiamata, al contrario, se prima della fase di terminazione ci sono molti pacchetti UDP di lunghezza maggiore allora la vittima aveva accettato la chiamata in precedenza e poi l'ha terminata.

5.2.3.2 Evidenze Individuate

In questo paragrafo si raccolgono le evidenze individuate a seguito di una analisi forense delle tracce relative al rifiuto/terminazione di chiamata. Dal traffico di una telefonata, si può riconoscere se c'è stato un rifiuto/terminazione di chiamata. Si presume che questa azione venga identificata dal traffico di cinque pacchetti STUN Unknown Request che, purtroppo, non permettono di riconoscere chi, tra gli interlocutori, ha compiuto la suddetta azione, tantomeno riconoscere se la chiamata è stata rifiutata senza flusso audio, che identificherebbe un possibile caso di cyberstalking (ove la vittima rifiuta le chiamate provenienti dal presunto cyber-stalker). Queste limitazioni possono essere superate grazie a due possibili pattern. Il primo pattern identifica l'interlocutore che ha rifiutato/terminato la chiamata, semplicemente si va a osservare quale, tra i due dispositivi, per primo ha inviato i pacchetti STUN Unknown Request grazie al campo arrival time. Il secondo pattern riconosce se la chiamata è stata rifiutata o terminata. Per fare ciò, non ci si può basare soltanto sui pacchetti STUN Unknown Request, che non danno nessuna informazione in merito, ma bisogna esaminare anche quelli antecedenti ad essi, questo perché, prima di una terminazione di chiamata, sono solo due gli eventi che possono concretizzarsi: fase di ringing o fase di flusso audio.

Questi due eventi sono distinguibili grazie ad una particolare caratteristica dei pacchetti UDP, precedenti a quelli STUN Unknown Request, infatti, durante la fase di ringing, vengono inviati pochi pacchetti UDP di breve lunghezza, durante la fase di flusso audio, invece, vengono inviati molti pacchetti UDP di lunghezza maggiore. Si può quindi provare che se prima della terminazione di chiamata ci sono pochi pacchetti UDP di breve lunghezza allora la vittima ha rifiutato la chiamata, al contrario, se prima della terminazione di chiamata ci sono molti pacchetti UDP di lunghezza maggiore allora la vittima aveva accettato la chiamata in precedenza e poi l'ha terminata.

5.2.3 Esempio di Analisi Forense Attività Instagram

Gli esperimenti simulati sono i seguenti:

- Invio/Ricezione di una foto in Direct
- Invio/Ricezione di un video in Direct

Osserviamo il traffico di ogni esperimento. Per via dell'elevato numero di pacchetti TCP e TLS inviati in questi esperimenti, il traffico viene mostrato attraverso il grafico I/O, in cui vengono rappresentati sull'asse delle ordinate la quantità di Byte dei pacchetti, e sull'asse delle ascisse la linea temporale. Per una facile lettura del grafico si utilizza un filtro per i pacchetti TCP di colore viola e uno per i pacchetti TLS di colore blu che sono gli elementi sostanziali per capire quali sono le azioni che sono state eseguite. Inoltre, viene utilizzato il tool Xplico come strumento di supporto per rilevare eventuali tracce.

5.2.4.1 *Invio/Ricezione di una foto in Direct*

In questo esperimento, A invia 3 foto in Direct a B, con il peso di 1.817 MB, 1.008 MB, 2.257 MB rispettivamente.

Grafico rappresentante la quantità in Bytes dei pacchetti TCP e TLS, catturati durante un esperimento, tramite PC1, del dispositivo A.

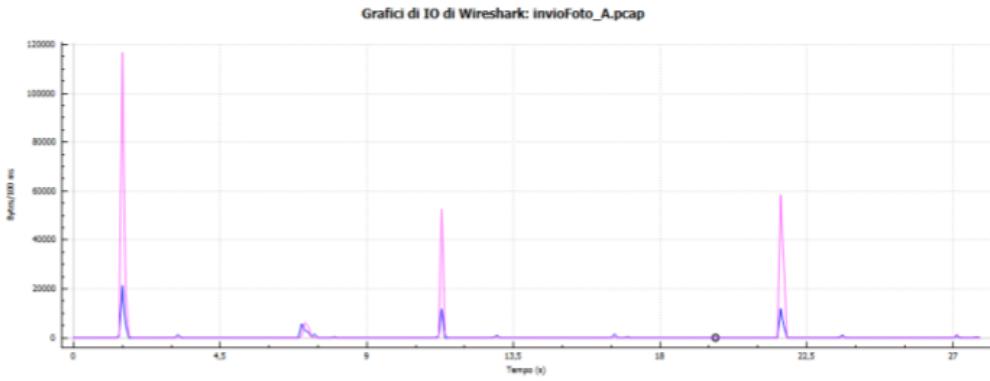


Figura 5.11 - Grafico I/O dei bytes dell'invio della foto del dispositivo A

Intervallo	Fase	Descrizione
1 - 2 sec.	Invio	Invio foto, picchi TCP alti e TLS bassi
11 - 12 sec.	Invio	Invio foto, picchi TCP alti e TLS bassi
21 - 22 sec.	Invio	Invio foto, picchi TCP alti e TLS bassi

Figura 5.12 - Azioni eseguite da A durante l'esperimento

Destination	Port	Protocol	Duration [s]	Size [byte]	Info
31.13.86.2	443	Unknown	21	2940	info.xml
instagram.c10.facebook.com	443	SSL.Iagram	0	19560	info.xml
instagram.c10.facebook.com	443	Unknown	26	315609	info.xml
239.255.255.250	1900	Unknown	9	548	info.xml

Figura 5.13 - Screen della sezione Undecoded di Xplico relativo al traffico catturato del dispositivo A

L'invio di foto è rappresentato da picchi TCP alti e TLS bassi, la dimensione dei pacchetti, però, non rispecchia quella delle foto, Xplico, infatti, nella sezione Undecoded, rileva un traffico complessivo pari a 338 kB circa.

Di seguito abbiamo il grafico rappresentante la quantità in Bytes dei pacchetti TCP e TLS, catturati durante un esperimento, tramite PC2, del dispositivo B.

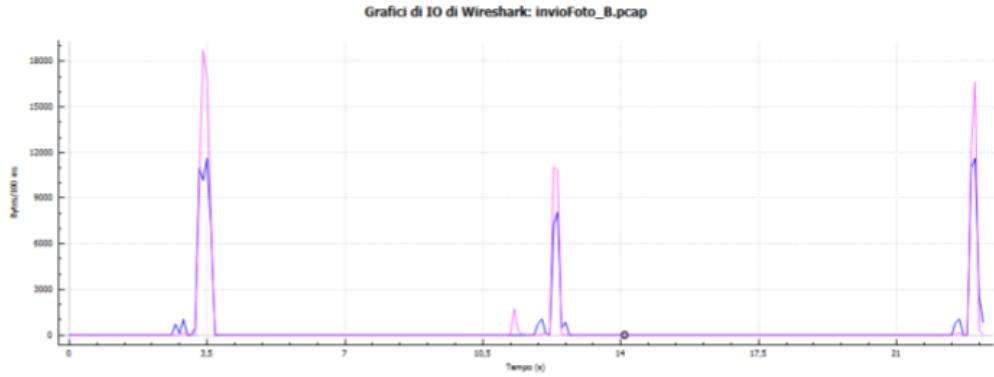


Figura 5.14 - Grafico I/O dei bytes della ricezione della foto del dispositivo B

Intervallo	Fase	Descrizione
3 - 4 sec.	Ricezione	Ricezione foto, picchi TCP e TLS alti
12 - 13 sec.	Ricezione	Ricezione foto, picchi TCP e TLS alti
23 - 24 sec.	Ricezione	Ricezione foto, picchi TCP e TLS alti

Figura 5.15 - Azioni eseguite da B durante l'esperimento

Destination	Port	Protocol	Duration [s]	Size [byte]	Info
192.168.137.183	42343	Unknown	20	2150	info.xml
192.168.137.183	57919	Unknown	20	3010	info.xml
scontent-mxp1-1.cdninstagram.com	443	Unknown	20	3471	info.xml
scontent-mxp1-1.cdninstagram.com	443	SSL_No_Cert.WhatsApp	20	169117	info.xml

Figura 5.16 - Screen della sezione Undecoded di Xplico relativo al traffico catturato del dispositivo B

La ricezione di foto è rappresentata da picchi TCP e TLS alti, la dimensione dei pacchetti, però, non rispecchia quella delle foto, Xplico, infatti, nella sezione Undecoded, rileva un traffico complessivo pari a 177 kB circa.

5.2.4.2 Inviò/Ricezione di un video in Direct

In questo esperimento, A invia 2 video a B. Il primo video è di 11 secondi e pesa 30.266 MB, il secondo video è di 31 secondi e pesa 84.603 MB.

Dal grafico rappresentante la quantità in Bytes dei pacchetti TCP e TLS, catturati durante un esperimento, tramite PC1, del dispositivo A.

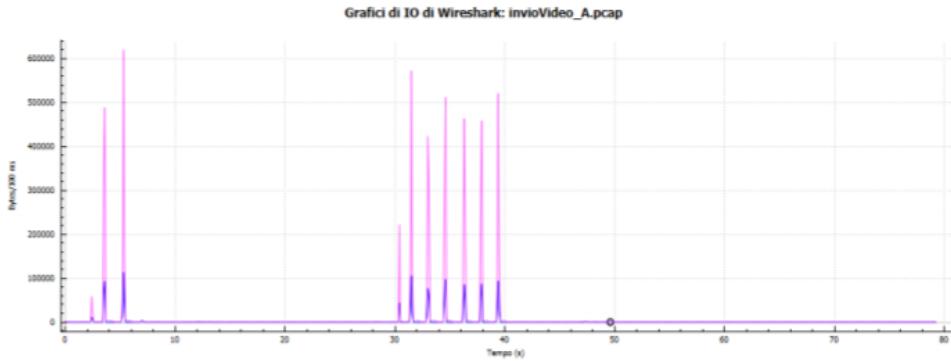


Figura 5.17 - Grafico I/O dei bytes dell'invio del video del dispositivo A

Intervallo	Fase	Descrizione
2 - 6 sec.	Invio	Invio video, picchi TCP alti e TLS bassi
30 - 40 sec.	Invio	Invio video, picchi TCP alti e TLS bassi

Figura 5.18 - Azioni eseguite da A durante l'esperimento

Destination	Port	Protocol	Duration [s]	Size [byte]	Info
239.255.255.250	1900	Unknown	3	696	info.xml
31.13.86.2	443	Unknown	60	2318	info.xml
Instagram.c10cffacebook.com	443	SSL.Instagram	65	6560	info.xml
Instagram.c10cffacebook.com	443	Unknown	48	7444319	info.xml

Figura 5.19 - Screen della sezione Undecoded di Xplico relativo al traffico catturato del dispositivo A

L'invio di un video è rappresentato da picchi TCP alti e TLS bassi, la dimensione dei pacchetti, però, non rispecchia quella dei video, Xplico, infatti, nella sezione Undecoded, rileva un traffico complessivo pari a 7.4 MB circa.

Il grafico rappresenta la quantità in Bytes dei pacchetti TCP e TLS, catturati durante un esperimento, tramite PC2, del dispositivo B.

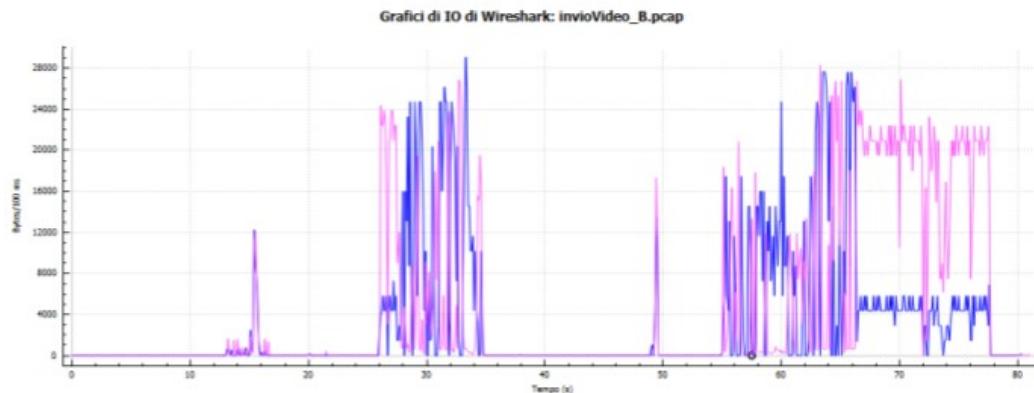


Figura 5.20 - Grafico I/O dei bytes della ricezione del video del dispositivo B

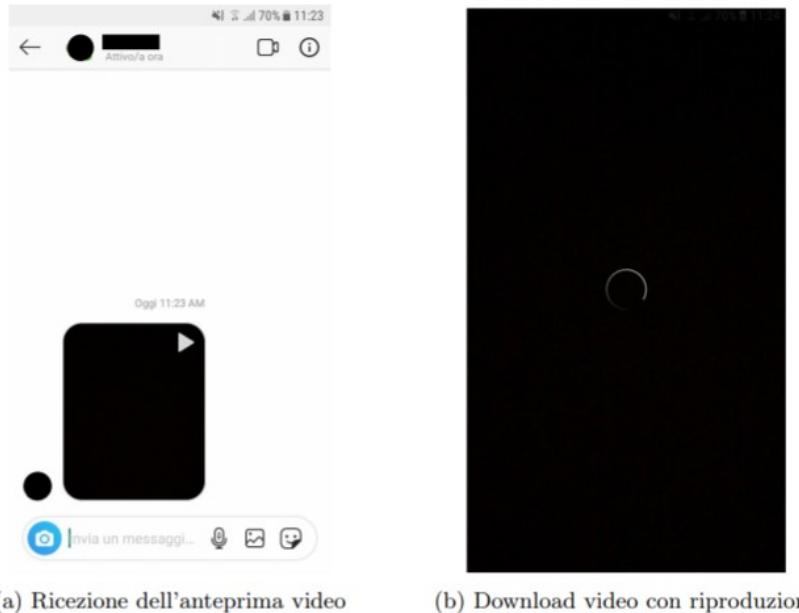
Intervallo	Fase	Descrizione
13 - 17 sec.	Ricezione	Ricezione antemprima video, picchi TCP e TLS bassi
26 - 35 sec.	Download	Download del video, picchi TCP e TLS alti
49 - 50 sec.	Ricezione	Ricezione antemprima video, picchi TCP e TLS bassi
55 - 78 sec.	Download	Download del video, picchi TCP e TLS alti

Figura 5.21 - Azioni eseguite da B durante l'esperimento

Destination	Port	Protocol	Duration [s]	Size [byte]	Info
239.255.255.250	1900	Unknown	8	8751	info.xml
192.168.137.183	1900	Unknown	8	8895	info.xml
239.255.255.250	1900	Unknown	15	822	info.xml
i.instagram.com	443	SSL_No_Cert.WhatsApp	52	5694173	info.xml
i.instagram.com	443	SSL.Instagram	35	4571	info.xml
i.instagram.com	443	SSL.Instagram	36	120594	info.xml
192.168.137.183	58025	Unknown	36	1997	info.xml
i.instagram.com	443	SSL.Instagram	67	9181	info.xml

Figura 5.22 - Screen della sezione Undecoded di Xplico relativo al traffico catturato del dispositivo B

La ricezione di video è rappresentata da picchi TCP e TLS alti, la dimensione dei pacchetti, però, non rispecchia quella dei video, Xplico, infatti, nella sezione Undecoded, rileva un traffico complessivo pari a 5.8 MB circa. Quando viene ricevuto un video, inizialmente il destinatario riceve una sua anteprima, dopoi, cliccando su di essa viene scaricato il video con conseguente riproduzione. Il rilevamento di questa azione può essere molto importante nei casi di pedopornografia, per confermare che il sospettato ha intenzionalmente scaricato il video per visionarlo.



(a) Ricezione dell'anteprima video (b) Download video con riproduzione

Figura 5.23 - Screen del dispositivo B alla ricezione del video

5.2.4.3 Evidenze Individuate

In questo paragrafo si raccolgono le evidenze individuate a seguito di una analisi forense dello scambio di elementi multimediali tramite Direct. Il traffico dell'invio/ricezione di foto in Direct, rappresentato da un picco TCP, le cui caratteristiche sono quelle di essere breve e alto, sembra essere uguale al traffico delle normali interazioni utente come (like, unlike, commento, ecc), tuttavia c'è un particolare. L'invio/ricezione della foto risulta avere maggiore peso rispetto alle normali interazioni, l'investigatore, avendo a disposizione l'intero traffico, può escludere i picchi di peso minore, dare importanza a quelli di peso maggiore e riconoscere che si tratta di una foto

Il traffico relativo all'invio di video è rappresentato da picchi TCP stretti, lunghi e consequenziali, quasi dello stesso peso. Il traffico relativo alla ricezione dei video è caratterizzato da un'anteprima video (picchi TCP simili a quelli delle foto) e, se si clicca sul pulsante "riproduci" viene eseguito il download del video. Quest'ultima azione è molto importante per una investigazione, per esempio sulla

pedopornografia, per confermare che un utente che riceve un video ha intenzionalmente visto il video.

5.2.4 Conclusioni

Grazie all’analisi statistica ispirata alla Channel-Side Analysis, è stato possibile individuare evidenze che possono migliorare il quadro investigativo di una indagine per un possibile cybercrimine, utilizzando le informazioni estraibili del flusso di pacchetti come ad esempio protocollo utilizzato, flag ecc. Questo approccio è particolarmente rilevante, nel caso in cui gli investigatori non dispongano del dispositivo informatico con cui è stato condotto il cyber-reato.

Dall’analisi manuale dei pacchetti relativi al traffico di una normale chiamata su WhatsApp, è stato possibile individuare le seguenti evidenze sugli indirizzi IP degli interlocutori nei pacchetti STUN Binding Request, non sempre presenti, con i quali è possibile effettuare una ricerca sul database whois per risalire a informazioni molto utili sul provider che ha fornito tali indirizzi.

L’analisi statistica ispirata alla channel-side analysis è stata di notevole supporto, grazie alla quale, osservando i grafici relativi al traffico generato da una chiamata, è stato possibile risalire a tutte le azioni compiute dal dispositivo dell’interlocutore intercettato (Setup, Ringing, Accettazione, Flusso Audio, Terminazione), caratterizzate da picchi TCP o UDP. Lo stesso approccio vale anche per le chiamate di gruppo dove, una volta individuate le suddette azioni, si può risalire al numero di partecipanti e chi ha compiuto una determinata azione osservando semplicemente i picchi TCP/UDP.

Per quanto riguarda, invece, l’analisi manuale effettuata su Instagram, la cifratura dei pacchetti risulta molto più robusta di quella di WhatsApp, rendendo indistinguibile il traffico generato da normali azioni (ad esempio, like, commento, ecc.). Per le azioni più articolate (scambio di elementi multimediali, videochiamate) la channel-side analysis è di aiuto. Si è visto come l’invio/ricezione di un video provochi l’invio di pacchetti TCP/TLS caratterizzati da picchi contigui e costanti. Alla ricezione del video, viene inviata inizialmente una anteprima,

quando l'interlocutore clicca su di essa parte il download che genera picchi TCP/TLS. Le videochiamate di Instagram generano traffico con picchi simili a quelli di WhatsApp, quindi si riesce a risalire a tutte le suddette azioni.

Tutto questo lavoro ha portato a dei risultati che, al di fuori degli aspetti strettamente tecnici e metodologici, possono supportare le forze dell'ordine nella scoperta di payload di comunicazioni criminali e per perseguire penalmente i criminali informatici al fine di consegnarli alla giustizia.

Vantaggi

L'analisi proposta potrebbe essere più robusta, rispetto ad una potenziale analisi forense dello smartphone. Inoltre, è da considerare che non è detto che vi sia effettiva disponibilità dello smartphone stesso (lo smartphone potrebbe essere danneggiato o non rinvenuto). Inoltre, è verosimilmente più complesso applicare tecniche anti-forensi efficaci che vadano a mitigare l'analisi forense dei pacchetti.

Svantaggi e Limitazioni

Ci sono tuttavia delle limitazioni nella metodologia:

- Cifratura dei pacchetti che, di fatto, rende "illeggibile" il contenuto di questi ultimi;
- Possibile difficoltà e onerosità nell'effettuare delle acquisizioni di rete;
- Può essere utilizzata solo per arricchire/integrare il quadro investigativo di una indagine;
- Analisi è manuale, osservando appunto il flusso dei pacchetti e le caratteristiche.

6. Il tool Instant Messaging Stream Detector Parser

Il punto centrale dell'intero lavoro di Tesi è la realizzazione del tool Instant Messaging Stream Detector Parser (IMSD Parser), il cui scopo è l'estrapolazione delle features da stream TCP di oggetti (immagini, video, audio e file) nei servizi di Instant Messaging con la possibilità di riportandole all'interno di un file CSV.

Nella sezione 6.1 si discuterà dell'idea progettuale che ha portato a sviluppare il tool; in 6.2 si analizzerà la struttura, il setup e un'overview sull'implementazione del tool; infine nella sezione 6.3 viene presentato le modalità ed esempi di utilizzo del tool.

6.1 Concept

Gli strumenti della Network Forensics aiutano ad analizzare il traffico di rete catturato per carpire delle evidenze digitali che possono arricchire il quadro investigativo. Per analizzare i casi in cui sono coinvolti dispositivi mobile che utilizzano servizi di Instant Messaging occorre osservare il traffico di rete tra i client e i relativi server.

Il traffico di rete coinvolge stream di pacchetti TCP/TLS per cui non si è in grado di poter estrarre il contenuto essendo crittografati. Per cui un modo per rilevare la tipologia di un flusso di pacchetti è quello di utilizzare tecniche di side channel analysis, ovvero, di estrarre delle caratteristiche temporali/spaziali sul flusso di pacchetti (ad esempio header, protocollo, numero di pacchetto ecc.), in maniera tale da poter classificare il traffico.

Nel capitolo precedente abbiamo analizzato che a partire dalle caratteristiche del traffico è stato possibile delineare metodologie e pattern che permettono di individuare evidenze per effettuare profiling delle attività utente attraverso l'utilizzo di un'applicazione di Instant Messaging.

La difficoltà maggiore di un lavoro del genere è manualmente determinare le caratteristiche temporali/spaziali dei pacchetti (richiede tempo e un notevole sforzo), per cui non risulta fattibile se si vuole analizzare un traffico di rete voluminoso. Per tale motivo è opportuno automatizzare tale processo, affidandosi a quelle che sono le tecniche di Intelligenza Artificiale, in maniera tale da estrapolare le features.

Da questa idea è stato sviluppato un tool denominato Instant Messaging Stream Detector Parser, (sigla IMSD Parser) per l'estrapolazione delle features dei flussi TCP di traffico nei servizi di Instant Messaging.

Un'applicazione del tool che è stata effettuata riguarda la generazione/preparazione di un file CSV di features a partire da numerose catture di traffico che simulano lo scambio di oggetti (immagini, video, audio e file) tra utenti attraverso applicazioni di Instant Messaging.

Questo file di features potrà essere, ad esempio, utilizzato come dataset di addestramento per una rete neurale in maniera tale da poter permettere, poi, la rilevazione della tipologia di stream di una cattura di traffico di rete che coinvolge un servizio di IM.

6.1.1 Cenni sulla Side-Channel Analysis

Una side channel analysis [1, 2, 3] si basa sulle informazioni ottenute dall'implementazione di un sistema informatico, piuttosto che da punti deboli nello stesso algoritmo implementato (ad esempio crittografia e bug del software). Le informazioni di temporizzazione, il consumo di energia, le perdite elettromagnetiche o persino il suono possono fornire una fonte aggiuntiva di informazioni che possono essere sfruttate.

Prendiamo in esame, ad esempio, il Secure Sockets Layer (SSL) e il suo successore Transport Layer Security (TLS). Essi sono protocolli crittografici che si occupano della crittografia di una connessione o sessione. Il protocollo inizia con la verifica dell'identità di una o più parti comunicanti e se l'identità può essere verificata con successo viene scambiata una chiave di crittografia simmetrica che rimane privata

sia per il client che per il server. Questa chiave simmetrica viene utilizzata per crittografare tutto il traffico nella sessione tra client e server, nonché per generare codici di autenticazione dei messaggi (MAC) per garantire l'integrità dei messaggi mentre si spostano attraverso una rete non sicura.

Nonostante l'uso della crittografia, l'osservazione di una sessione crittografata SSL / TLS rivela ancora le seguenti informazioni:

- quantità approssimativa di dati trasferiti in ciascuna sessione;
- orari di inizio e fine di ciascuna sessione;
- indirizzo IP del client e dei server, nonché i loro nomi di dominio;
- ordine delle sessioni.

Queste informazioni possono essere in molte situazioni, sufficienti per rivelare le seguenti informazioni private sull'interazione di un utente con un sito Web protetto SSL / TLS:

- le pagine Web che l'utente ha caricato;
- input privati che hanno causato il caricamento di una pagina Web;
- i tempi relativi e assoluti in cui si sono verificati gli eventi in tempo reale;
- contenuti della pagina Web in base al modo in cui la pagina risponde al ridimensionamento della finestra.

Analogamente, queste informazioni possono essere utili anche dal punto di vista forense per identificare tracce di attività che possono rendersi utili durante un'indagine investigativa.

Principalmente, la Network Forensics viene eseguita sui live system, dove viene acquisito il traffico raw, prodotto dalle interfacce di rete di un dispositivo informatico (pacchetti, eventuali log, ecc.), che viene poi analizzato.

Un compito tedioso della Network Forensics è la correlazione dei dati, che può essere causale o temporale (i timestamp devono essere registrati). Un utente malintenzionato può crittografare il traffico, in genere utilizzando una connessione VPN SSL. Per un investigatore di rete, l'indirizzo e la porta sono ancora visibili; tuttavia, il flusso di dati non è disponibile. Un'altra sfida aggiuntiva è determinare l'origine di un attacco, poiché un utente malintenzionato può utilizzare un host intermedio per eseguire un attacco o semplicemente utilizzare un server proxy remoto. Ciò rende difficile per un investigatore di rete seguire l'indirizzo originale degli aggressori.

Tenendo conto di questi inconvenienti, il compito principale di un investigatore forense di rete è analizzare i pacchetti di rete, al fine di risalire a elementi importanti come: protocolli utilizzati, indirizzi IP, numeri di porta, timestamp, file trasferiti, user agent, versioni dei server e versioni del sistema operativo. Queste informazioni possono essere estratte da diversi tipi di traffico e possono essere utili per migliorare il quadro investigativo.

6.1.2 Osservazione sull'estrapolazione delle features da un flusso

Avendo un'acquisizione di un traffico dire è possibile determinare dei valori guardando le caratteristiche temporali/spaziali sul flusso di pacchetti. Queste caratteristiche (features) possono essere collezionate per poi utilizzare, ad esempio, come dei dataset di training set o training evalutation per i modelli di machine learning.

Le features possono essere calcolate guardando l'intero flusso di pacchetti, ma ciò potrebbe portare a delle valutazioni poco accurate sul traffico di rete osservato. Al fine di avere dei dataset adeguati, il flusso viene suddiviso, di volta in volta, in una finestra temporale scorrevole. Tale finestra contiene soltanto un certo numero di pacchetti e su questi ultimi vengono estrarre le features. Man mano si sposta la finestra in avanti, andando a considerare un nuovo sottoinsieme di pacchetti dell'intero flusso.

In dettaglio, la finestra temporale è un intervallo [START, END] i cui valori si basano sul tempo di arrivo dei pacchetti. Il valore START riguarda il tempo di arrivo del pacchetto nella posizione i-esima (inizialmente del primo pacchetto del flusso). Il valore END, invece è pari ad START+1 secondi. All'interno di questa finestra, rientrano un determinato numero di pacchetti del flusso il cui tempo di arrivo appartiene all'intervallo [START, END].

Per determinare una nuova finestra temporale quello che si fa è rideterminare l'intervallo [START,END] sulla base di un parametro denominato stride.

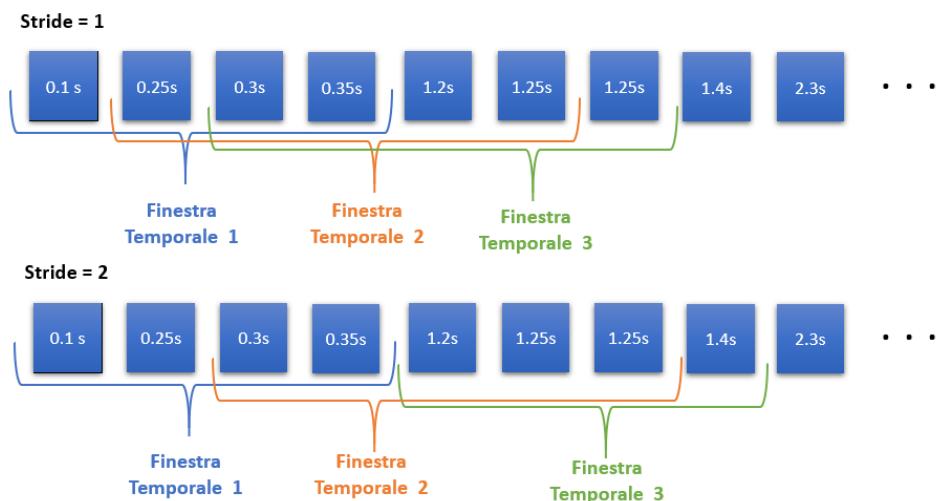


Figura 6.1 - Esempio delle finestre temporali utilizzando stride pari a 1 e 2

Indichiamo un pacchetto i-esimo con P_i e il suo tempo di attivo $T(P_i)$, fissiamo lo stride pari ad X e consideriamo la finestra temporale $[START = T(P_i), END = T(P_i)+1]$. Una volta calcolate le feature su tale finestra, occorre definire una nuova finestra temporale su un altro sotto-insieme di pacchetti del flusso. La nuova finestra temporale si sposterà in avanti, cioè il valore iniziale NEW_START sarà determinato dal tempo di arrivo del pacchetto in posizione i-esima + X , che indichiamo con P_j . Per cui il nuovo intervallo della finestra temporale sarà dato da $[NEW_START = T(P_j), END = T(P_j)+1]$ e così via, fin quando non si arriva a considerare il tempo di arrivo dell'ultimo pacchetto dell'intero flusso.

Si può osservare che lo stride va a determinare sostanzialmente il numero di finestre temporali di un flusso. Se lo stride ha un valore alto, meno sarà il numero

di finestre temporali e, di conseguenza, il tempo di elaborazione nell'estrapolare le features diminuisce, ma minore accuratezza sulla valutazione del flusso; Viceversa se lo stride ha un valore basso, allora si avranno molte finestre temporali e il tempo di elaborazione nell'estrapolare le features aumenta, ma si avrà una maggiore accuratezza sulla valutazione del flusso.

Un'importante osservazione è che durante il processo di estrapolazione delle features è che quando si considera la finestra temporale non si esaminano tutti i pacchetti che vi appartengono, ma soltanto alcuni di questi, ovvero si effettua un campionamento (frequenza). Ad esempio, se la frequenza è 6hz allora si stanno prendendo 6 pacchetti al secondo dalla finestra temporale.

6.2 Implementazione del tool IMSD Parser

Il tool IMSD Parser è composto da un uno script in Python in grado di effettuare il parsing di un file PCAP utilizzando la libreria Scapy, in maniera tale da permettere una detection di possibili connessioni di rete tra un client e un servizio di Instant Messaging con il fine ultimo di poterne estrarre delle features dal traffico di rete che verranno riportate in un file CSV.



Figura – 6.2 Logo del tool IMSD Parser

Di seguito sono riportate le funzionalità del tool:

- Individuare la presenza di server di Instant Messaging selezionato;
- Individuare le connessioni tra client e server di IM;
- Individuare i flussi TCP tra client e server di IM;
- Estrapolazione delle feature da un singolo flusso di rete di una connessione di rete individuata;
- Estrapolazione delle feature di tutti i flussi di una connessione di rete individuata;
- Estrapolazione delle feature di tutti i flussi di tutte le connessioni di rete individuate;
- Salvare le features all'interno di un file CSV nuovo o aggiornare uno esistente;

6.2.1 Struttura del progetto

La cartella principale del progetto è **IMSD_Parser**, in cui all'interno troviamo:

- **IMSD_parser.py**: script principale in Python in cui sono definite classi e funzioni necessarie per effettuare il parsing di un file PCAP utilizzando la libreria Scapy;
- **IMSD_parser_CLI_.py**: script Python che importa **IMSD_parser.py** in modo tale da istanziare il parser ed eseguirlo tramite una Command Line Interface;
- **TMP_FEATURE**: cartella che contiene tutti i file CSV che vengono creati quando si estrapolano le features da un determinato flusso protocollare appartenente ad una connessione tra un client e un server di Instant Messaging;
- **test_file**: cartella che contiene alcuni file PCAP di test;

- **requirements.txt**: file di testo che contiene una lista dei moduli Python che devono essere installati.

È importante notare che il contenuto della cartella **TMP_FEATURE** viene sovrascritto ad ogni esecuzione del tool e ad ogni estrapolazioni delle features di una connessione tra un client e un server di Instant Messaging.

6.2.2 Setup dell'ambiente

Il tool **IMSD Parser** è un insieme di script in Python che possono essere lanciato utilizzando l'interprete Python versione 2. In particolare, Il sistema operativo su cui il tool IMSD Parser è stato sviluppato, eseguito e testato è Ubuntu 18.04.2 LTS attraverso l'utilizzo della Bash di Linux su Windows 10 e la versione di Python 2.7.15+.

Lo script **IMSD_parser.py** richiede di importare diversi moduli Python, che devono essere opportunamente installati. Di seguito vengono riportati i moduli necessari:

- **ipwhois** – Un pacchetto Python focalizzato sul recupero e sull'analisi dei dati whois per indirizzi IPv4 e IPv6.
- **scapy_ssl_tls** – Una libreria Python che consente all'utente di inviare, sniffare, dissezionare e creare pacchetti di rete (Include anche i Livelli SSL/TLS).

All'interno del progetto è presente un file requirements.txt in cui vengono riportati i moduli e le rispettive versioni utilizzate. Per installare i moduli lanciare il seguente comando:

```
>> python -m pip install -r IMSD_Parser/requirements.txt
```

Per un'installazione pulita è possibile ricorrere alla creazione di un ambiente virtuale utilizzando virtualenv di Python3.

```
>> sudo apt-get install python3-pip  
>> sudo pip3 install virtualenv  
>> virtualenv -p /usr/bin/python2.7 venv  
>> source venv/bin/activate
```

Sposta il contenuto della cartella `IMSD_Parser` nella cartella `venv`, entrare all'interno della cartella `venv/ IMSD_Parser` e lanciare il comando Python precedente per installare i moduli necessari.

6.2.3 Un overview sullo script

Lo script `IMSD_parser.py` è il fulcro centrale in cui vengono definite le classi e funzioni necessarie per effettuare il parsing di un file PCAP utilizzando la libreria Scapy.

Nella seguente sezione analizzeremo, brevemente, le classi definite all'intero dello script `parser.py`

Le classi `Host`, `Connection`, `Flux`, `Features_Dict` non saranno descritte, ma sono semplicemente dei contenitori di dati ed oggetti.

6.2.3.1 La classe FEATURES

Descrizione

La classe `FEATURES` contiene vari metodi statici che permettono di calcolare varie features a partire da una lista di pacchetti.

Le features possono essere estrapolate considerando il flusso tra un Client e Server IM, andando a suddividerlo in finestre temporali scegliendone la frequenza di campionamento e stride.

Sono state definite tre categorie di features secondo la tabella 6.1, dove le label di **flow features** fanno riferimento all'intero flusso protocollare di pacchetti tra un client e Server di IM; mentre **Client to Server Features** ai soli pacchetti che hanno

come sorgente il Client; viceversa **Server to Client Features** ai soli i pacchetti che hanno come sorgente il Server IM.

Flow Features	
Label Feature	Descrizione
TAG	Etichetta che identifica se il flusso è un audio, video, foto o file
DURATION_TIME	Tempo trascorso in secondi tra il tempo di arrivo del primo pacchetto e il tempo di arrivo dell'ultimo pacchetto
NUMB_OF_PACKETS	Numero di pacchetti scambiati
TOTAL_BYTE	Dimensione Totale in Byte dei pacchetti scambiati
MIN_BYTE	Dimensione minima in Byte tra i pacchetti scambiati
MAX_BYTE	Dimensione massima in Byte tra i pacchetti scambiati
AVG_BYTE	Dimensione media in Byte dei pacchetti scambiati
Client to Server Features	
Label Feature	Descrizione
CLI_NUMB_OF_PACKETS	Numero di pacchetti inviati dal Client
CLI_TOTAL_BYTE	Dimensione Totale in Byte dei pacchetti inviati dal Client
CLI_MIN_BYTE	Dimensione minima in Byte dei i pacchetti inviati dal Client
CLI_MAX_BYTE	Dimensione massima in Byte dei i pacchetti inviati dal Client
CLI_AVG_BYTE	Dimensione media in Byte dei i pacchetti inviati dal Client
Server to Client Features	
Label Feature	Descrizione
SVR_NUMB_OF_PACKETS	Numero di pacchetti inviati dal Server
SVR_TOTAL_BYTE	Dimensione Totale in Byte dei pacchetti inviati dal Server
SVR_MIN_BYTE	Dimensione minima in Byte dei i pacchetti inviati dal Server
SVR_MAX_BYTE	Dimensione massima in Byte dei i pacchetti inviati dal Server

SVR_AVG_BYTE	Dimensione media in Byte dei i pacchetti inviati dal Server
--------------	---

Tabella 6.1 – CATEGORIA E DESCRIZIONE DELLE FEATURES

Variabili

La classe contiene, in accordo alla tabella precedente, tre liste:

- LIST_FLOW_FEATURES_NAME – contiene le label relative a flow features;
- LIST_CLI_FEATURES_NAME – contiene le label relative Client to Server Features;
- LIST_SRV_FEATURES_NAME – contiene le label relative Server to Client Features.

Inoltre, vi è una lista LIST_OBJECT che contiene i tag (audio, video, immagini, file e sconosciuto) che identificano la tipologia di un flusso.

Metodi

All'interno della classe sono definiti dei metodi statici che data una lista di pacchetti, calcola il valore della rispettiva feature e lo restituisce in output.

Esempio di metodo che calcola i byte totali di una lista di pacchetti:

```

1. @staticmethod
2. def TOTAL_BYTE(list_packets):
3.     total_length = 0
4.     for pkt in list_packets:
5.         total_length += len(pkt)
6.     return total_length

```

I metodi statici definiti nella classe hanno come nome del metodo esattamente i nomi delle label definiti nelle liste LIST_FLOW_FEATURES_NAME.

In particolare, sono presenti due metodi statici:

- **def extract_flow_features(flux, list_packets_window = None, tag = "?"):-**
Data una lista di pacchetti, permette di calcolare tutte le features definite in LIST_FLOW_FEATURES_NAME, LIST_CLI_FEATURES_NAME e

LIST_SRV_FEATURES_NAME richiamando i rispettivi metodi statici ed inserirle in un oggetto Features_Dict che verrà restituito; Il parametro tag fa riferimento alla tipologia del flusso.

- **def temporal_extract_features(flux, frequency, stride, tag):** -

Permette di estrapolare le features applicando una finestratura temporale (frequency fa riferimento alla frequenza di campionamento e stride indica l'offset dei pacchetti a cui applicare la successiva finestratura); restituisce un dizionario di Features_Dict, ovvero viene inserito all'interno di esso le features relative ad ogni finestra temporale individuata, richiamando il metodo *extract_flow_features(...)*. Il parametro tag fa riferimento alla tipologia del flusso.

Altro

Per aggiungere una nuova feature da calcolare basta inserire una label all'interno delle liste LIST_FLOW_FEATURES_NAME, LIST_CLI_FEATURES_NAME e LIST_SRV_FEATURES_NAME (anteporre CLI_ o SRV_ se si vuole calcolarla per client o server) e definire un metodo statico all'interno della classe FEATURES che ha lo stesso nome della label appena inserita.

6.2.3.2 La classe FEATURES_CSV_CREATOR

Descrizione

La classe **FEATURES_CSV_CREATOR** permette di creare file CSV nella cartella TMP FEATURE (file con stesso nome vengono riscritti) oppure di creare o aggiornare (se già esiste) un file CSV con un dato nome file. Questa classe viene utilizzate, dunque, per salvare le features in file CSV relative al flusso analizzato o per generare e aggiornare dataset.

Il formato del file CSV è organizzato nel modo seguente:

- le colonne CSV sono le features definite nelle liste in FEATURES.LIST_FLOW_FEATURES_NAME, FEATURES.LIST_CLI_FEATURES_NAME e FEATURES.LIST_SRV_FEATURES_NAME;

- le righe fanno riferimento al numero di finestra temporale osservate per quel determinato flusso.

Variabili

- PATH_FOLDER_TMP_FEATURE – contiene il path in cui inserire i file CSV; impostata come cartella “TMP_FEAUTURE” nel percorso relativo del file IMSD_parser.py

Metodi

La classe **FEATURES_CSV_CREATOR** contiene in tutto quattro metodi statici:

- **def make_folder():** permette di creare la cartella “TMP_FEAUTURE” specificata dal path PATH_FOLDER_TMP_FEATURE;
- **def delete_folder():** permette di cancellare tutto il contenuto della cartella “TMP_FEAUTURE” specificata dal path PATH_FOLDER_TMP_FEATURE;
- **def write_csv_file(dict_window):** permette di creare un file CSV nella cartella “TMP_FEAUTURE” specificata dal path PATH_FOLDER_TMP_FEATURE e scrivere i dati del dizionario, ovvero le feature, definite in dict_window nel file CSV;
- **def write_csv_dataset(dict_window, name_file):** permette di creare o aggiornare (se già esiste) un file CSV nella percorso relativo specificato da name_file e scrivere i dati del dizionario, ovvero le feature, definite in dict_window nel file CSV;

6.2.3.3 La classe Capture

Descrizione

La classe **Capture** permette di poter di poter trovare varie informazioni su una struttura Scapy, utilizzando i metodi statici delle classi FEATURES. Le azioni possibili sono quelle di recuperare la lista degli host, dei client, dei server, delle connessioni e flussi TCP, estrarre da una flusso TCP le rispettive features applicando una finestratura temporale.

Variabili

- LIST_SUPPORTED_IM – contiene i servizi di Instant Messaging che possono essere analizzati con il tool; ritroviamo WhatsApp e Telegram,

Costruttore

```
1. def __init__(self,capture):  
2.     self._capture = capture  
3.     self._list_hosts = []  
4.     self._list_servers = []  
5.     self._list_client_hosts = [];  
6.     self._list_connections = [];  
7.     self._asn = [];
```

Il costruttore prende in input un riferimento ad un oggetto Scapy (per i nostri scopi ottenibile tramite il metodo della libreria Scapy *rdpcap* (“file.PCAP”), che permette di leggere un file PCAP) che viene mantenuto tramite l’attributo `self._capture`; inizializza poi una lista degli host, dei client, dei server e delle connessioni e la lista degli ASN come liste vuote;

Classi Ereditarie

All’interno del file `IMSD_parser.py` sono state definite anche le classi `WHATSAPP_Capture(Capture)` e `TELEGRAM_Capture(Capture)` che vengono istanziate in base a quale servizio di Instant Messaging si desidera analizzare. Ciascuna delle classi figlie imposta il parametro `self._asn` come una lista contenente i numeri degli ASN associati al servizio di IM.

Ad esempio, per il servizio di IM “WhatsApp” abbiamo che l’Autonomous System Number corrisponde a AS32934 di proprietà di Facebook.

```
1. class WHATSAPP_Capture(Capture):  
2.  
3.     def __init__(self,capture):  
4.         self._capture = capture  
5.         self._list_hosts = []  
6.         self._list_servers = []  
7.         self._list_client_hosts = [];  
8.         self._list_connections = [];  
9.         self._asn = ["32934"]
```

Metodi

I metodi definiti all'intero troviamo:

- **def is_server(self,ip_indr):**– Metodo booleano che effettua il controllo se un dato indirizzo IP è un server di IM, utilizzato il servizio di lookup WHOIS in maniera tale da verificare che l'Autonomous System Number (ASN) associato a quell'indirizzo IP corrisponde agli ASN dei servizi di IM supportati;
- **def search_hosts(self):**– Trova all'intero dell'oggetto capture gli indirizzi IP e MAC, creando oggetti Host ed inserendoli nella lista degli host self._list_hosts.
- **def search_servers(self):**– Verifica che gli host individuati sono dei server di IM invocando il metodo booleano *is_server(self,ip_indr)* e, se lo sono, li inserisce nella lista dei server self._list_servers;
- **def search_connections(self):**– setaccia i server dalla lista degli host ottenendo una lista dei client self._list_client_hosts e per ogni client e server IP crea un oggetto Connection(Host_Client, Host_Server) inserendolo nella lista delle connessioni self._list_connections.
- **def search_connection_flux_TCP(self, connection):**– Trova i flussi TCP di pacchetti all'intero del oggetto capture in base all'oggetto Connection, definendo un dizionario di flussi (ip sorgente, porta sorgente, ip destinazione, porta destinazione, protocollo TCP) e mantenendone il riferimento in connection._list_flux.
- **def extract_conn_features(self, connection, protocol=None, number_window = 1):**– Viene restituito un dizionario contenente le features riguardanti la connessione, il protocollo e il numero di finestre in cui suddividere il flusso; se non specificato il valore protocol vengono trovati tutti i protocolli supportati; se non specificato il valore number_window il flusso non viene suddiviso;
- **def extract_flux_features(self, flux, frequency = 1, stride = 1, tag = FEATURES.UNKNOWN):** - Restituisce un dizionario di features relativo alle

finestre temporali definite in base ai parametri frequency e stride a partire da un oggetto flux che contiene una lista di pacchetti di un determinato flusso. Richiama il metodo FEATURES.temporal_extract_features(flux, frequency, stride, tag).

- **def extract_conn_features(self, flux, frequency = 1, stride = 1, tag = FEATURES.UNKNOWN):** – Viene restituita una lista di dizionari. Per una data connessione in maniera ciclica si va a scorrere i flussi appartenenti a quella connessione e per ognuno di essi viene invocato il metodo *extract_flux_features(flux, frequency, stride, tag)* il quale restituisce un dizionario di features di quel flusso.
- **def extract_all_conn_features(self, frequency = 1, stride = 1, tag = FEATURES.UNKNOWN):** - Viene restituita una lista di dizionari. Per tutte le connessione in maniera ciclica si va a scorrere i flussi appartenenti ad una singola connessione e per ognuna di esse, in maniera iterativa, va a scorrere i flussi appartenenti a quella connessione e per ognuno di essi viene invocato il metodo *extract_flux_features(flux, frequency, stride, tag)* il quale restituisce un dizionario di features di quel flusso.
- **def get_connection(self, ip_addr_1, ip_addr_2):**– Viene restituito un oggetto Connection dalla lista delle connessioni, se il client host e il server hanno gli indirizzi IP passati in input.
- **def get_flux(self, conn, port_1, port_2, protocol):** - Trova un flusso di una connessione dalla lista dei flussi, che abbia porte e protocollo specificati in input. Restituisce in output l'oggetto Flux.

Altro

Per poter aggiungere un nuovo IM da analizzare occorre inserire nello script IMSD_parser.py una nuova classe figlia “**EXAMPLE_Capture(Capture):**” ed indicare i vari ASN all'interno della lista self._asn.

6.3 Modalità ed Esempi di Utilizzo

Il file **IMSD_Parser_CLI.py** è uno script Python che importa lo script **IMSD_parser.py** in modo tale da istanziare il parser ed eseguirlo tramite una Command Line Interface.

Lo scopo del tool è quello di analizzare il file PCAP dato in input in maniera da poter estrarre dai i flussi di una connessione tra un client e un server IM le features e salvarle all'interno di un file CSV.

Le features vengono estratte da un flusso applicando una suddivisione dello stesso in finestre temporali in base alla frequenza di campionamento e allo stride dati in input.



Figura 6.3 - Screen del banner iniziale

Per un corretto funzionamento del tool installare tutti i moduli specificati nel paragrafo 4.2.2 e il file **IMSD_parser.py** deve essere nella stessa cartella del file **IMSD_Parser_CLI.py**.

Viene eseguito lanciando da una Command Line Interface tramite l'interprete Python versione 2.7.15+, posizionandosi all'interno della cartella **IMSD_Parser** ed inserendo il seguente comando:

```
>> python IMD_Parser_CLI.py [path_file_PCAP]
```

Per comprendere le azioni possibili che il tool ci permette di fare su un file PCAP, prendiamo in considerazione un esempio di analisi del file PCAP “whatsapp_video.pcap” contenuto nella cartella **test_file**. Tale cattura contiene il traffico di rete generato da un invio di un video da parte di un client utilizzando l'applicazione WhatsApp.

Al momento del lancio del tool ci viene mostrato il banner del tool e ci viene fornita la lista di servizi di Instant Messaging supportati, ovvero WhatsApp e Telegram. Essendo che nel nostro file di esempio è una cattura riguardante il traffico WhatsApp, inseriamo “WhatsApp”.



The screenshot shows the IMD Parser tool's interface. At the top, it displays the title "IMSD PARSER" in large, bold, white, pixelated letters. Below the title, there is some small, illegible text. The main area is a terminal-like window with a black background and white text. It starts with a banner: "Dip.Informatica UNISA - @Egidio Giacoia" followed by a list of supported messaging services: ["WHATSAPP", 'TELEGRAM']. A prompt "[+] Choose Instant Messaging: WHATSAPP" is shown, followed by a command menu: "Command: !SEARCH_SERVER - !SEARCH_ALL_CONN - !EXTRACT_FEATURE_FLUX - !EXTRACT_FEATURE_CONN - !EXTRACT_FEATURE_ALL_CONN - !CHANGE_IM - !EXIT - >>".

Figura 6.4 -Screen relativo ai comandi disponibili dopo aver seleziona il servizio di IM

Una volta scelto il servizio di Instant Messaging ci viene mostrato a video la lista dei comandi che è possibile eseguire.

Di seguito sono riportati i comandi disponibili:

- !SEARCH_SERVER
- !SEARCH_ALL_CONN
- !EXTRACT_FEATURE_FLUX
- !EXTRACT_FEATURE_CONN
- !EXTRACT_FEATURE_ALL_CONN
- !CHANGE_IM
- !EXIT

Nelle seguenti sezioni vedremo le azioni svolte del tool quando vengono inseriti i comandi definiti, prendendo sempre in considerazione il file PCAP “whatsapp_video.pcap” come esempio.

6.3.1 Il comando !SEARCH_SERVER

Il comando !SEARCH_SERVER cerca all'interno del file dato in input tutti gli indirizzi IP presenti e, tra questi, verifica se sono degli indirizzi IP di server dell'Instant Messaging selezionato. Tale procedimento viene effettuato tramite un servizio di WHOIS lookup, prendendo per ciascuno degli indirizzi IP, l'Autonomous System Number (ASN) e lo confronta con l'ASN dell'Instant Messaging selezionato.

Potrebbe accadere che non viene trovata nessun indirizzo IP che corrisponde ad un server di IM, a questo punto si potrebbe provare a cambiare il servizio di Instant Messaging da analizzare, selezionando il comando !CHANGE_IM.

```
Command: !SEARCH_SERVER - !SEARCH_ALL_CONN - !EXTRACT_FEATURE_FLUX - !EXTRACT_FEAT
>>!SEARCH_SERVER
[+] Searching for IP Address

    --> IP Found: 31.13.86.51
    --> IP Found: 31.13.86.49
    --> IP Found: 192.168.137.121

[+] Searching IP WHATSAPP Server

Wait for WHOIS lookup service (make sure you're connected to the internet)

    --> WHATSAPP Server IP: [ 31.13.86.51 31.13.86.49 ]
```

Figura 6.5 - Screen relativo ai server WhatsApp rilevati

Nel nostro esempio sono stati rilevati:

- indirizzi IP: "31.13.86.51" - "31.13.86.49" - "192.168.137.121";
- indirizzi IP di un server di WhatsApp: "31.13.86.51"- "31.13.86.49".

Infatti, utilizzando il servizio WHOIS Lookup del sito web <http://whois.domaintools.com/> si può verificare che effettivamente gli indirizzi IP corrispondono ad un server WhatsApp.

The screenshot shows the DomainTools website with the navigation bar: PROFILE ▾, CONNECT ▾, MONITOR ▾, SUPPORT, Whois Lookup, and a search icon. Below the navigation is a breadcrumb trail: Home > Whois Lookup > 31.13.86.51. The main content is titled "IP Information" for 31.13.86.51. Under "Quick Stats", it lists:

IP Location	Italy Milan Facebook Ireland Ltd
ASN	AS32934 FACEBOOK, US (registered Aug 24, 2004)
Resolve Host	whatsapp-cdn-shv-01-mxp1.fbcn.net
Whois Server	whois.ripe.net
IP Address	31.13.86.51

Figura 6.6 - Applicazione del servizio WHOIS Lookup sull'indirizzo IP 31.13.86.51

6.3.2 Il comando !SEARCH_ALL_CONN

Il comando !SEARCH_ALL_CONN cerca all'interno del file dato in input tutti le connessioni tra un client e un server di Instant Messaging e per ognuna delle connessioni i rispettivi flussi (IP Client, Porta Client, IP Server, Porta Server, Protocollo TCP).

Potrebbe accadere che non viene restituita nessuna connessione se, implicitamente, non viene trovato nessun indirizzo IP che corrisponde ad un server di IM.

```
SCOMMUNICATOR>>!SEARCH_ALL_CONN
[+] Searching for all CLIENT - WHATSAPP_SERVER connections
--> Connection: CLIENT_IP:192.168.137.121 - WHATSAPP_IP:31.13.86.51
    --> Flux: CLIENT_IP:PORT=192.168.137.121:49572 - WHATSAPP_IP:PORT=31.13.86.51:443 [TCP]
    --> Flux: CLIENT_IP:PORT=192.168.137.121:49571 - WHATSAPP_IP:PORT=31.13.86.51:443 [TCP]
--> Connection: CLIENT_IP:192.168.137.121 - WHATSAPP_IP:31.13.86.49
    --> Flux: CLIENT_IP:PORT=192.168.137.121:44446 - WHATSAPP_IP:PORT=31.13.86.49:5222 [TCP]
```

Figura 6.7 - Screen che mostra i flussi TCP delle connessioni rilevate

Nel nostro esempio sono presenti due connessioni tra un client “192.168.137.121” e due server WhatsApp “31.13.86.51” e “31.13.86.49”.

I flussi relativi alla connessione client 192.168.137.121 – server 31.13.86.51 sono:

- 192.168.137.121:49572 - 31.13.86.51:433 [tcp]
- 192.168.137.121:49571 - 31.13.86.51:433 [tcp]

I flussi relativi alla connessione client 192.168.137.121 – server 31.13.86.49 sono:

- 192.168.137.121:44446 – 31.13.86.49:5222 [tcp]

Quindi il client inviante è coinvolto con due server WhatsApp, con cui si sono instaurati due flussi verso uno di questi e un unico flusso verso l’altro.

6.3.3 Il comando !EXTRACT_FEATURE_FLUX

Il comando !EXTRACT_FEATURE_FLUX permette di estrarre le features da flusso di una connessione tra un client e un server di IM. Il tool richiede di inserire in input:

- L’indirizzo IP Client e Porta Client
- L’indirizzo IP Server e Porta Server
- Tag Flusso: definire che tipologia di stream dei pacchetti, ovvero se è relativa ad audio (A), video (V), Immagine (I), file (F) o di natura sconosciuta (?);
- Un numero intero – indica la frequenza di campionamento (in hz)
- Un numero intero – indica lo stride

```
Command: !SEARCH_SERVER - !SEARCH_ALL_CONN - !EXTRACT_FEATURE_FLUX - !EXTRACT_FEATURE_CONN - !EXTRACT_FEATURE_ALL
>>>!EXTRACT_FEATURE_FLUX
[+] Searching Stream

[+][+] Insert Client IP:Port: 192.168.137.121:44446
[+][+] Insert Server IP:Port 31.13.86.49:5222
[+][+] List of Supported Tag Object: {'UNKNOWN': '?', 'IMAGE': 'I', 'AUDIO': 'A', 'VIDEO': 'V', 'FILE': 'F'}
Type '?' (UNKNOWN) if you don't know the type of stream.

[+][+] Choose Tag Object Stream: V
[+][+] Choose the frequency (hz) in which to split the flow temporally: 7
[+][+] Choose the stride: 1
[+][+] FLUX: CLIENT_IP:PORT=192.168.137.121:44446 - WHATSAPP_IP:PORT=31.13.86.49:5222 [TCP]

Frequency=7, Stride=1
[-----] WINDOW 1 [-----]
```

*Figura 6.8 - Screen relativo all’input dato relativo all’ estrazione delle features del flusso
192.168.137.121:44446 – 31.13.86.49:5222*

Nel nostro esempio si vuole estrarre le features del flusso della connessione client 192.168.137.121 – server 31.13.86.49, ovvero 192.168.137.121:44446 – 31.13.86.49:5222 [tcp]. Per cui in input abbiamo:

- Indirizzo IP Client e Porta Client: 192.168.137.121:44446
- Indirizzo IP Server e Porta Server: 31.13.86.49:5222
- Tag Object Stream: V (essendo che si tratti di un video)
- Frequenza di campionamento: 7
- Stride: 1

A questo punto una volta inseriti i parametri, il tool restituisce le varie finestre temporali definite dallo stride e per ognuna di esse le rispettive features dei pacchetti considerati in base frequenza di campionamento. Il processo di estrapolazione delle features avviene come definito in 5.1.2.

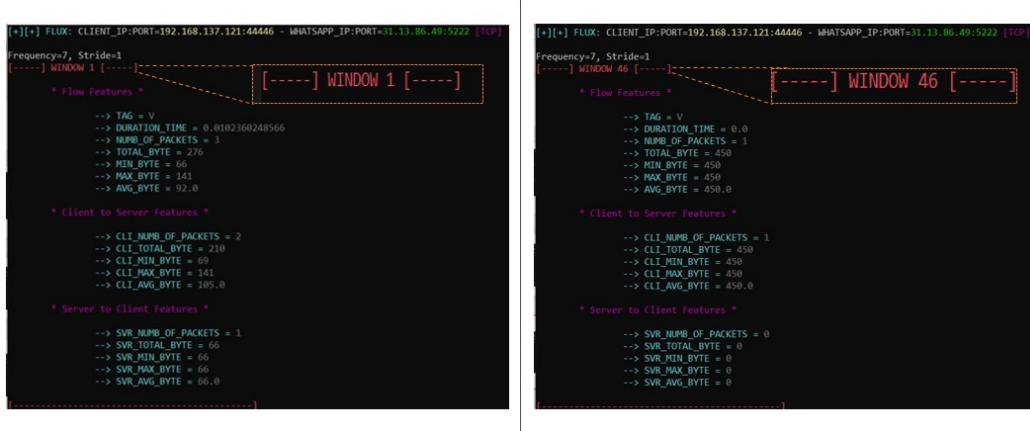


Figura 6.9 - Screen delle finestre 1 e 46 e le rispettive features del flusso

192.168.137.121:44446 – 31.13.86.49:5222

Nel nostro caso dall'intero flusso di partenza si hanno 46 finestre. Per quanto riguarda la frequenza di campionamento è pari a 7 Hz, per cui si prendono al più 7 pacchetti al secondo nella finestra temporale considerata.

L'immagine seguente mostra, come esempio, le features estrapolate dalla finestra 5 del flusso TCP della connessione 192.168.137.121:44446 - 31.13.86.49:5222.

```

[+] [+] FLUX: CLIENT_IP:PORT=192.168.137.121:44446 - WHATSAPP_IP:PORT=31.13.86.49:5222 [TCP]
Frequency=7, Stride=1
[-----] WINDOW 5 [-----]

    * Flow Features *

--> TAG = V
--> DURATION_TIME = 0.00579595565796
--> NUMB_OF_PACKETS = 7
--> TOTAL_BYTE = 1213
--> MIN_BYTE = 66
--> MAX_BYTE = 345
--> AVG_BYTE = 173.285714286

    * Client to Server Features *

--> CLI_NUMB_OF_PACKETS = 2
--> CLI_TOTAL_BYTE = 132
--> CLI_MIN_BYTE = 66
--> CLI_MAX_BYTE = 66
--> CLI_AVG_BYTE = 66.0

    * Server to Client Features *

--> SVR_NUMB_OF_PACKETS = 5
--> SVR_TOTAL_BYTE = 1081
--> SVR_MIN_BYTE = 66
--> SVR_MAX_BYTE = 345
--> SVR_AVG_BYTE = 216.2

[-----]

```

Figura 6.10 - Screen che mostra le features della finestra 5 del flusso

192.168.137.121:44446 – 31.13.86.49:5222

Come presentato nella sezione 5.2.3.1 vengono mostrate a video le tre categorie di features, dove:

- le label di **flow features** fanno riferimento all'intero flusso protocollare di pacchetti tra un client e Server di IM;
- le label **Client to Server Features** ai soli pacchetti che hanno come sorgente il Client;
- le label **Server to Client Features** ai soli i pacchetti che hanno come sorgente il Server IM.

Dopo aver mostrato a video tutte le features per ognuna delle finestre, il tool chiede se si vogliono salvare all'intero di una file CSV rispondendo con "Y" per confermare o con "N" per negare. Se si vuole effettuare il salvataggio occorre fornire il nome del file CSV e se già esiste allora viene aggiornato inserendo alla fine del file le features.

```
[-----]  
See TMP_CAPTURE directory for csv file  
All files in TMP_CAPTURE will be deleted every time the features are extracted  
[+][+] Do you want update DATASET with this features? (y/n): y  
[+][+] Type name file csv DATASET (without extension) : DATASET_1  
    --> DATASET update  
  
Command: !SEARCH_SERVER - !SEARCH_ALL_CONN - !EXTRACT_FEATURE_FLUX - !EXTRACT_FEATURE  
=>
```

Figura 6.11 - Screen relativo al salvataggio delle features in un file CSV

Nel nostro esempio le memorizziamo all'interno del file CSV “DATASET_1” che viene creato nella main folder del progetto IMSD Parser.

Figura 6.12 - Screen relativo al contenuto del file CSV “DATASET_1”

Come mostrato nella prima riga del file sono riportate le label delle features, mentre le restanti righe sono le varie finestre temporali. Ognuna delle righe contiene i valori di ciascuna feature.

Inoltre, indipendentemente se si vuole memorizzare o meno l'estrapolazione delle features, all'interno della cartella “TMP_FEATURE” vengono mantenuti in maniera temporanea i file CSV relativi ad ogni flusso considerato. Ad ogni nuova estrapolazione delle features, i file in questa cartella vengono rimossi.

The screenshot shows a file explorer window with the path 'LocalState > rootfs > home > egix'. The current view is 'TMP_FEATURE'. A single file is listed: '192.168.137.121_44446-31.13.86.49_5222-TCP' (Type: File con valori separati da virgola (CSV) di Microsoft Excel, Last modified: 06/03/2020 12:09, Size: 6 KB).

Nome	Ultima modifica	Tipo	Dimensio...
192.168.137.121_44446-31.13.86.49_5222-TCP	06/03/2020 12:09	File con valori separati da virgola (CSV) di Microsoft Excel	6 KB

Figura 6.13 - Screen del contenuto temporaneo della cartella TMP FEATURE

6.3.4 Il comando !EXTRACT FEATURE Conn

Il comando !EXTRACT FEATURE Conn permette di estrarre le features di tutti i flussi di una connessione tra un client e un server di IM. Il tool richiede di inserire in input:

- L'indirizzo IP Client;
- L'indirizzo IP Server;
- Tag Flusso: definire che tipologia di stream dei pacchetti, ovvero se è relativa ad audio (A), video (V), Immagine (I), file (F) o di natura sconosciuta (?);
- Un numero intero – indica la frequenza di campionamento (in hz) della finestra;
- Un numero intero – indica lo stride;

Nel nostro esempio si considera la connessione client 192.168.137.121 – server 31.13.86.51. I flussi di questa connessione sono:

- 192.168.137.121:49572 - 31.13.86.51:433 [tcp]
- 192.168.137.121:49571 - 31.13.86.51:433 [tcp]

Per tanto, diamo in input al tool i seguenti valori:

- Indirizzo IP Client: 192.168.137.121
- Indirizzo IP Server: 31.13.86.51
- Tag Object Stream: V (essendo che si tratti di un video)

- Frequenza di campionamento: 8
- Stride: 1

A questo punto una volta inseriti i parametri, il tool restituisce per ciascun flusso di una connessione le varie finestre temporali definite dallo stride e per ognuna di esse le rispettive features dei pacchetti considerati in base frequenza di campionamento. Il processo di estrapolazione delle features avviene come definito in 5.1.2.

```

[+] (+) FLUX: CLIENT_IP:PORT=192.168.137.121:49572 - WHATSAPP_IP:PORT=31.13.86.51:433 [TCP]
Frequency=8, Stride=1
[----] WINDOW 19 [----]
1
* Flow Features *
--> TAG = V
--> DURATION_TIME = 0.200211048126
--> NUMB_OF_PACKETS = 8
--> SVR_AVG_BYT = 936
--> MIN_BYT = 66
--> MAX_BYT = 456
--> AVG_BYT = 117.0
* Client to Server Features *
--> CLT_NUMB_OF_PACKETS = 6
--> CLT_TOTAL_BYT = 414
--> CLT_MIN_BYT = 72
--> CLT_MAX_BYT = 66
--> CLT_AVG_BYT = 69.0
* Server to Client Features *
--> SVR_NUMB_OF_PACKETS = 2
--> SVR_TOTAL_BYT = 522
--> SVR_MIN_BYT = 0
--> SVR_MAX_BYT = 456
--> SVR_AVG_BYT = 261.0
[----]
[+] (+) FLUX: CLIENT_IP:PORT=192.168.137.121:49572 - WHATSAPP_IP:PORT=31.13.86.51:433 [TCP]
Frequency=8, Stride=1
[----] WINDOW 19 [----]
1
* Flow Features *
--> TAG = V
--> DURATION_TIME = 0.200211048126
--> NUMB_OF_PACKETS = 1
--> SVR_AVG_BYT = 390
--> MIN_BYT = 390
--> MAX_BYT = 390
--> AVG_BYT = 390.0
* Client to Server Features *
--> CLT_NUMB_OF_PACKETS = 1
--> CLT_TOTAL_BYT = 390
--> CLT_MIN_BYT = 390
--> CLT_MAX_BYT = 390
--> CLT_AVG_BYT = 390.0
* Server to Client Features *
--> SVR_NUMB_OF_PACKETS = 0
--> SVR_TOTAL_BYT = 0
--> SVR_MIN_BYT = 0
--> SVR_MAX_BYT = 0
--> SVR_AVG_BYT = 0
[----]
[+] (+) FLUX: CLIENT_IP:PORT=192.168.137.121:49571 - WHATSAPP_IP:PORT=31.13.86.51:433 [TCP]
Frequency=8, Stride=1
[----] WINDOW 11 [----]
2
* Flow Features *
--> TAG = V
--> DURATION_TIME = 0.00050609500004

```

Figura 6.14 - Screen di alcune finestre dei due flussi della connessione 192.168.137.121 – 31.13.86.51

L'immagine mostra come, in maniera sequenziale, vengono estrapolate le features tra le varie finestre temporali trovate nel flusso 192.168.137.121:49572 - 31.13.86.51:433 [tcp] (1); e poi di seguito quelle relative al flusso 192.168.137.121:49571 - 31.13.86.51:433 [tcp] (2).

Nel nostro caso il flusso 192.168.137.121:49572 - 31.13.86.51:433 [tcp] produce 19 finestre temporali, mentre il flusso 192.168.137.121:49571 - 31.13.86.51:433 [tcp] produce 7 finestre temporali. Per quanto riguarda la frequenza di campionamento pari a 8 vuol dire che si prendono al più 8 pacchetti al secondo in ogni finestra temporale considerata.

Analogamente al comando precedente, il tool chiede se si vogliono salvare all'intero di una file CSV le features di entrambi i flussi della connessione. Possiamo aggiornare un file CSV già creato uno nuovo.

TMP_FEATURE				
Condividi		Visualizza		
<< LocalState > roots > home > egix >		> TMP_FEATURE		Cerca in TMP_FEATURE
#	Nome	Ultima modifica	Tipo	Dimensione
1	192.168.137.121_49571-31.13.86.51.443-TCP	06/03/2020 12:13	File con valori separati da virgola (CSV) di Microsoft Excel	2 KB
2	192.168.137.121_49572-31.13.86.51.443-TCP	06/03/2020 12:13	File con valori separati da virgola (CSV) di Microsoft Excel	3 KB

Figura 6.15 - Screen del contenuto temporaneo della cartella TMP_FEATURE

Nel nostro caso si è voluto aggiornare il file CSV “DATASET_1” creato in precedenza, per cui le features di ogni flusso vengono inseriti alla fine del file CSV.

Figura 6.15 - Screen relativo al contenuto del file CSV "DATASET 1" aggiornato

Dall'immagine possiamo notare che il riquadro (1) fa riferimento alle features relative alla precedente estrapolazione; in coda al file si vanno ad aggiungere al riquadro (2) le features relative alle finestre temporali del flusso 192.168.137.121:49572 - 31.13.86.51:433 [tcp] (19 finestre temporali), mentre nel riquadro (3) le features relative alle finestre temporali del flusso 192.168.137.121:49571 - 31.13.86.51:433 [tcp] (7 finestre temporali).

6.3.5 Il comando !EXTRACT FEATURE ALL CONN

Il comando `!EXTRACT_FEATURE_ALL_CONN` permette di estrarre le features di tutti i flussi di tutte le connessione tra un client e un server di IM trovate. Il tool richiede di inserire in input:

- Tag Flusso: definire che tipologia di stream dei pacchetti, ovvero se è relativa ad audio (A), video (V), Immagine (I), file (F) o di natura sconosciuta (?);
- Un numero intero – indica la frequenza di campionamento (in Hz) della finestra;
- Un numero intero – indica lo stride.

Nel nostro caso ci sono due connessioni, ovvero:

- Connessione 192.168.137.121 - 31.13.86.51 ha i seguenti flussi:
 - 192.168.137.121:49572 - 31.13.86.51:433 [tcp]
 - 192.168.137.121:49571 - 31.13.86.51:433 [tcp]
- Connessione 192.168.137.121 - 31.13.86.49 ha i seguenti flussi:
 - 192.168.137.121:44446 – 31.13.86.49:5222 [tcp]

Al tool viene dato in input i seguenti valori:

- Tag Object Stream: V (essendo che si tratti di un video)
- Frequenza di campionamento: 8
- Stride: 1

A questo punto una volta inseriti i parametri, il tool restituisce per ciascun flusso di ogni connessione le varie finestre temporali definite dallo stride e per ognuna di esse le rispettive features dei pacchetti considerati in base frequenza di campionamento. Il processo di estrappolazione delle features avviene come definito in 6.1.2.

```

Command: !SEARCH_SERVER !SEARCH_ALL_CONN - !EXTRACT_FEATURE_ALL_CONN - !EXTRACT_FEATURE_FLUX - !EXTRACT_FEATURE_CONN - !EXTRACT_FEATURE_ALL_CONN
[+] All flows
[+] List of Supported Tag Object: {UNKNOWN: '?', 'IMAGE': 'I', 'AUDIO': 'A', 'VIDEO': 'V', 'FILE': 'F'}
type '?' (UNKNOWN) if you don't know the type of stream.

[+] Choose Tag Object Stream V
[+] Choose the frequency (Hz) in which to split the flow temporally: 7
[+] Choose the stride: 1
[+] FLUX: CLIENT_IP:PORT=192.168.137.121:49572 - WHATSAPP_IP:PORT=31.13.86.51:433 [TCP]
frequency=7, Stride=1
[+] DYNAMIC [-----]

* Flow Features *

--> TAG = V
--> DURATION_TIME = 0.820380923721
--> NUMB_OF_PACKETS = 4
--> SVR_NUMB_OF_PACKETS_PER_SEC = 4.87621173479
--> TOTAL_BYTE = 450
--> MIN_BYTE = 66
--> MAX_BYTE = 450
--> AVG_BYTE = 112.5
--> NUMB_OF_BYTE_PER_SEC = 607.88836981

* Client to Server Features *

--> CLT_NUMB_OF_PACKETS = 3
--> CLT_NUMB_OF_PACKETS_PER_SEC = 3.6571588169
--> CLT_TOTAL_BYTE = 424
--> CLT_NUMB_OF_BYTE_PER_SEC = 516.878443887

* Server to Client Features *

--> SVR_NUMB_OF_PACKETS = 1
--> SVR_NUMB_OF_PACKETS_PER_SEC = 0
--> SVR_TOTAL_BYTE = 74
--> SVR_NUMB_OF_BYTE_PER_SEC = 74

[-----]

```

See CAPTURE directory for csv file
All files in CAPTURE will be deleted every time the features are extracted
[+][+] Do you want update DATASET with this features? (y/n):

Figura 6.16 - Screen di alcune finestre e le rispettive features di alcuni flussi

L’immagine mostra come, in maniera sequenziale, vengono estrapolate le features dei vari flussi di connessione tutte le connessioni; infatti nell’immagine a sinistra viene mostrata la window 1 del flusso 192.168.137.121:49572 - 31.13.86.51:433 [tcp]; mentre a destra viene mostrata la window 46 del flusso di un’altra connessione, ovvero il flusso 192.168.137.121:44446 - 31.13.86.49:5222 [tcp].

Analogamente ai comandi precedente, il tool chiede se si vogliono salvare all’intero di una file CSV tutte le features di tutti i flussi di tutte le connessioni estrapolate. Possiamo aggiornare un file CSV già esistente o creare uno nuovo.

6.3.6 Il comando !CHANGE_IM

Il comando !CHANGE_IM permette di selezionare il servizio di Instant Messaging che si vuole analizzare. Tale comando può essere utilizzando se ad esempio non viene trovata nessun indirizzo IP che corrisponde ad un server di IM selezionato precedentemente.

Nel nostro esempio avendo precedentemente selezionato “WHATSAPP”, lanciando il comando !CHANGE_IM il tool permette di effettuare un cambio di servizio di IM da analizzare, e selezioniamo “TELEGRAM”.

A questo punto, essendo che il file PCAP di prova contiene solo tracce di traffico relative all’utilizzo del servizio di IM di WhatsApp, se si prova a lanciare il comando

!SEARCH_SERVER notiamo che il tool non restituisce nessun server IP di Telegram in quanto effettivamente all'interno del file PCAP non vi sono tracce di interazioni con i server Telegram.

```
Command: !SEARCH_SERVER - !SEARCH_ALL_CONN - !EXTRACT_FEATURE_FLUX - !EXTRACT_FEATURE_IM  
>>!CHANGE_IM  
[+] List of supported Instant Messaging: ['WHATSAPP', 'TELEGRAM']  
[+] Choose Instant Messaging: TELEGRAM  
Command: !SEARCH_SERVER - !SEARCH_ALL_CONN - !EXTRACT_FEATURE_FLUX - !EXTRACT_FEATURE_IM  
>>!SEARCH_SERVER  
[+] Searching for IP Address  
    --> IP Found: 31.13.86.51  
    --> IP Found: 31.13.86.49  
    --> IP Found: 192.168.137.121  
[+] Searching IP TELEGRAM Server  
Wait for WHOIS lookup service (make sure you're connected to the internet)  
    --> No TELEGRAM Server IP Detected  
Command: !SEARCH_SERVER - !SEARCH_ALL_CONN - !EXTRACT_FEATURE_FLUX - !EXTRACT_FEATURE_IM  
>>
```

Figura 6.17 - Screen del cambio del servizio di IM e segnalazione della non presenza di Server Telegram

7. Un esperimento: generazione di un dataset per il ML

Come esperimento sono stati effettuati delle simulazioni dell'invio/ricezione di oggetti (immagini, video, audio e file) tra due utenti che utilizzano il servizio di Instant Messaging WhatsApp. L'obbiettivo è quello di estrapolare le features da questi stream di oggetti per generare un dataset tramite l'utilizzo del tool IMSD Parser.

Nella sezione 7.1 verrà presentata la metodologia e sperimentazione adottata; a seguire, sezione 7.2, è stato valutato un caso particolare riguardante alla metodologia di acquisizione del traffico; Nella sezione 7.3 viene effettuata un'analisi forense sui test case di acquisizioni di rete; infine, sezione 7.4, verranno generati dei dataset di features utilizzando il tool IMSD Parser.

7.1 Metodologia e sperimentazione

I file di test contenuti nella cartella **test_file** sono dei file PCAP generati tramite il tool Wireshark in cui vengono scambiati oggetti (immagini, video, audio e file) tramite le app di Instant Messaging supportate da IMSD Parser.

Ciascun file di test PCAP viene generato seguendo il medesimo scenario di riferimento:

- un utente A, che svolge il ruolo di inviante, che utilizza lo smartphone SP1 connesso alla rete hotspot del pc PC1;
- un utente B, che svolge il ruolo di ricevente, che utilizza lo smartphone SP2 connesso alla rete hotspot del pc PC2;

Entrambi i computer PC1 e PC2 sono connessi al modem Wireless (Globalcom) e vengono utilizzati come punti di intercettazione del traffico, utilizzando il tool Wireshark.

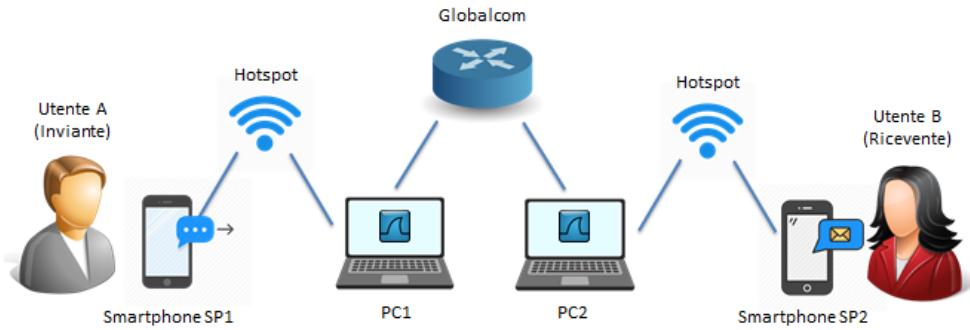


Figura 7.1 - Scenario di rete di riferimento per l'acquisizione del traffico

Per ogni file di test PCAP la cattura dei pacchetti viene avviata un'istante prima dell'invio/ricezione degli oggetti e termina un'istante dopo.

I dispositivi utilizzati vengono elencati di seguito:

Modello	Ruolo	Indirizzo IP interno
Huawei P10 Lite (SP1)	Inviante (A)	192.168.137.141 192.168.137.193*
Huawei Y6 (SP2)	Ricevente (B)	192.168.137.5 192.168.137.30*

Tabella 7.1 – Dispositivi utilizzati nella sperimentazione

Per i test case degli oggetti file è stato riassegnato l'indirizzo IP interno dei dispositivi, ovvero l'indirizzi contrassegnati con un * nella tabella precedente.

7.1.1 I file PCAP del Dataset

I file PCAP da cui sono state estratte le features per comporre il dataset utilizzano gli oggetti presi da [1, 2, 3, 4, 5 ,6]. Nella seguente tabella sono riportati le caratteristiche relative agli oggetti scambiati e i rispettivi file PCAP generati.

Ciascun test case consiste inviando più volte vari oggetti della stessa categoria di dimensione simile, per una durata di circa 10-11 minuti. Inoltre, dal lato del ricevente viene effettuato il download degli oggetti ricevuti.

Nella tabella sono riportati le caratteristiche relative ai test case effettuati. Le catture sono state effettuate secondo il modello di rete definito in 7.1.

Oggetto	Test Case	Nome PCAP	Dimensione	Campioni Inviati	Formato
Immagine	IMG_1MB	IMG_1MB_SENDER.pcap IMG_1MB_RECEIVER.pcap	1 MB	36	JPG
Immagine	IMG_4-6MB	IMG_4-6MB_SENDER.pcap IMG_4-6MB_RECEIVER.pcap	6.5 – 4.3 MB	33	JPG
Video	VID_5-10MB	VID_5-10MB_SENDER.pcap VID_5-10MB_RECEIVER.pcap	5 – 10 MB	7	MP4
Video	VID_20MB	VID_20MB_SENDER.pcap VID_20MB_RECEIVER.pcap	20 MB	3	MP4
Audio	AUD_1MB	AUD_1MB_SENDER.pcap AUD_1MB_RECEIVER.pcap	1.3 MB	6	WAV
Audio	AUD_6MB	AUD_6MB_SENDER.pcap AUD_6MB_RECEIVER.pcap	6 MB	5	MP3
File	FIL_1-2MB	FIL_1-2MB_SENDER.pcap FIL_1-2MB_RECEIVER.pcap	1-2MB	19	PDF
File	FIL_10MB	FIL_10MB_SENDER.pcap FIL_10MB_RECEIVER.pcap	10 MB	5	ZIP

Tabella 7.2 – Caratteristiche dei Test case

7.2 Un caso particolare

Una prima acquisizione dei pacchetti di rete prevedeva di inviare sempre lo stesso oggetto in maniera ripetitiva al ricevente. Da ciò è stato notato che: durante l'invio del singolo oggetto in maniera ripetitiva quello che accade è che viene effettivamente inviato la prima volta, dopodiché quello che fa WhatsApp non è rinviare più volte lo stesso oggetto, ma mostra soltanto una thumbnails dell'invio/ricezione dell'oggetto (essendo già stato inviato precedentemente). Infatti, nella memoria dei dispositivi dell'inviaente e ricevente è presente una sola copia.

Nella seguente tabella sono riportati le caratteristiche relative all'oggetto scambiato e i rispettivi file PCAP generati. Le catture sono state effettuate secondo il modello di rete definito in 7.1.

Ciascuno degli oggetti viene inviato in maniera ripetitiva per circa 10-11 minuti.

Oggetto	Label	Nome	Dimensione	Campioni Inviati	Formato
Immagine	IMG_1MB	SampleJPGImage_1mbmb	0.99 MB	97	JPG
Immagine	IMG_5MB	SampleJPGImage_5mbmb	5.02 MB	110	JPG
Immagine	IMG_10MB	SampleJPGImage_10mbmb	10.02 MB	97	JPG
Video	VID_5MB	SampleVideo_1280x720_5mb	5.01 MB	37	MP4
Video	VID_20MB	SampleVideo_1280x720_20mb	20.09 MB	18	MP4

Tabella 7.3 – Campioni utilizzati nelle catture di rete

Osserviamo i grafici I/O della cattura PCAP dal lato dell'inviaente relativa all'invio del video VID_5MB.

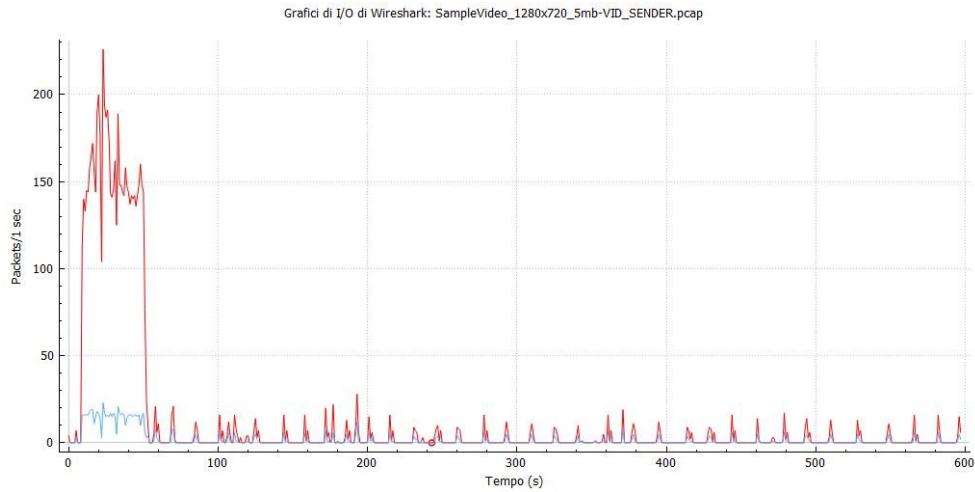


Figura 7.2 - Grafico I/O Inviaente - In rosso TCP, in blu TLS (Y=Packetti)

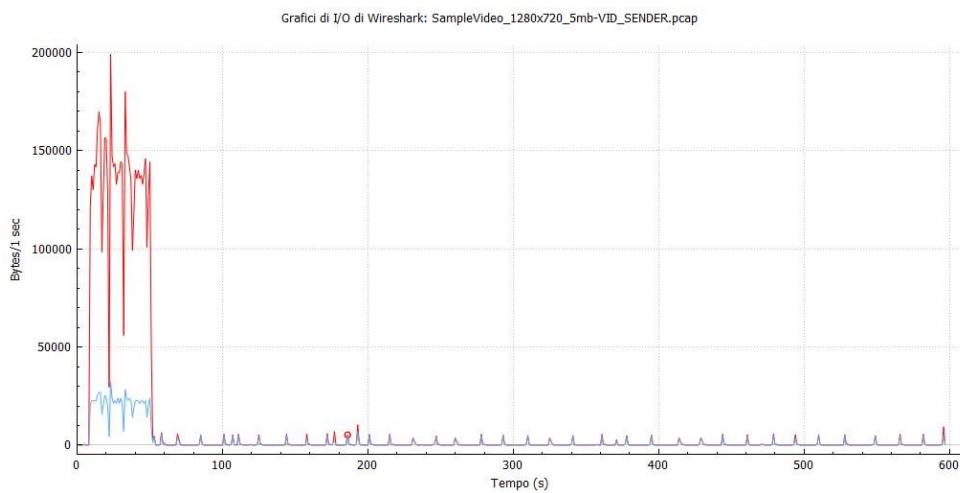


Figura 7.3 - Grafico I/O Inviaente - In rosso TCP, in blu TLS (Y=Bytes)

L'invio di un video è rappresentato da picchi TCP alti e TLS bassi, ma essendo che il video viene inviato più volte notiamo un grande picco di bytes/pacchetti soltanto all'inizio della cattura. Da questo si può dedurre che le successive volte non viene effettivamente rinviaito il video, ma probabilmente notificato l'invio di un oggetto già precedentemente inviato.

Osserviamo, adesso, i grafici I/O della cattura PCAP dal lato del ricevente relativa alla ricezione del video VID_5MB.

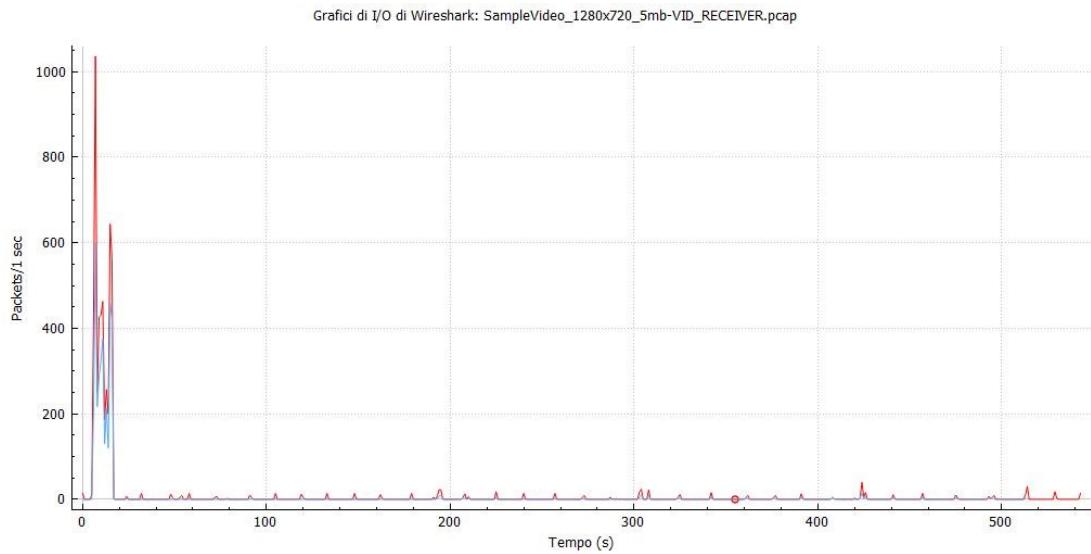


Figura 7.4 - Grafico I/O Ricevente - In rosso TCP, in blu TLS (Y=Pacchetti)

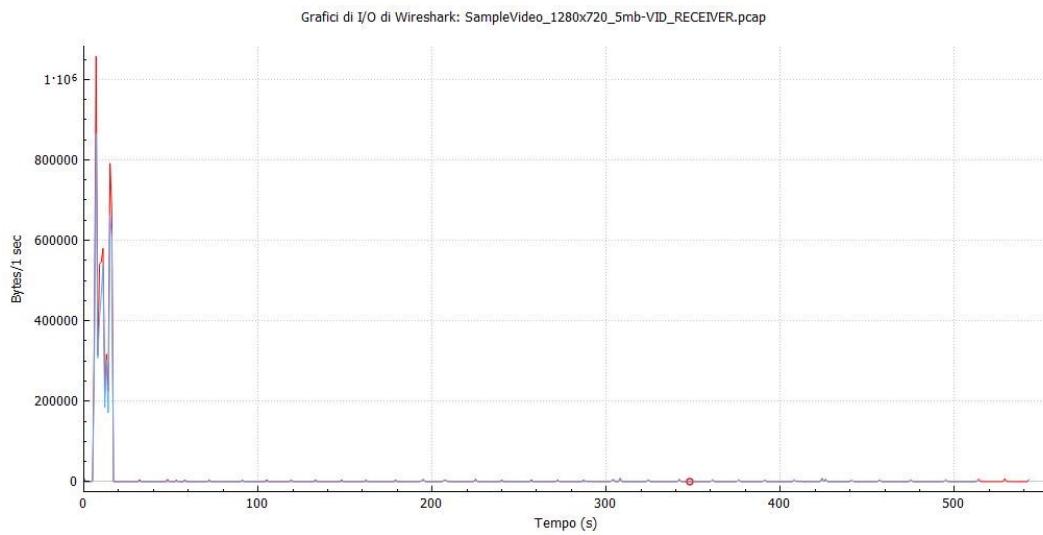


Figura 7.5 - Grafico I/O Inviaente - In rosso TCP, in blu TLS (Y=Bytes)

La ricezione del video è rappresentata da picchi TCP e TLS alti. Una nota sull'esperimento è che al momento della ricezione del video il destinatario riceve una sua anteprima, ovvero non lo scarica sul dispositivo.

L'invio di un video è rappresentato da picchi TCP alti e TLS bassi, ma essendo che il video viene inviato più volte notiamo un grande picco di bytes/pacchetti soltanto all'inizio della cattura. Da questo si può dedurre che le successive volte non viene

effettivamente rinvia il video, ma probabilmente notificato l'invio di un oggetto già precedentemente inviato.

Analogamente al caso dell'invio, il video viene effettivamente ricevuto solo una prima volta in quanto dai grafici notiamo un grande picco di bytes/pacchetti soltanto all'inizio della cattura. Da questo si può dedurre che le successive volte non viene effettivamente ricevuto il video, ma probabilmente viene soltanto notificato la ricezione dell'oggetto già precedentemente inviato.

Durante l'esperimento, infatti, su entrambi i dispositivi veniva mostrano una miniatura dell'invio/ricezione del video ogni volta, ma in realtà è stato inviato/ricevuto una sola volta.

Questo comportamento per la generazione di dataset non è efficiente, per cui è stato accantonato questo metodo di acquisizione dei file PCAP, ovvero inviare lo stesso singolo oggetto per 10-11 minuti.

7.3 Analisi forense dei file PCAP acquisiti

In questa sezione verrà condotta un'analisi forense sui file PCAP acquisiti per ogni test case, mostrando i grafici I/O generati con Wireshark (sia dal lato inviante che ricevente) e i server WhatsApp coinvolti.

7.3.1 Invio/Ricezione delle immagini

7.3.1.1 *Test Case: IMG_1MB*

Prendiamo in esame l'invio/ricezione delle immagini di peso 1MB (IMG_1MB), andando ad analizzare i rispettivi file PCAP generati dal lato inviante e dal lato ricevente.

Invio Immagini 1MB

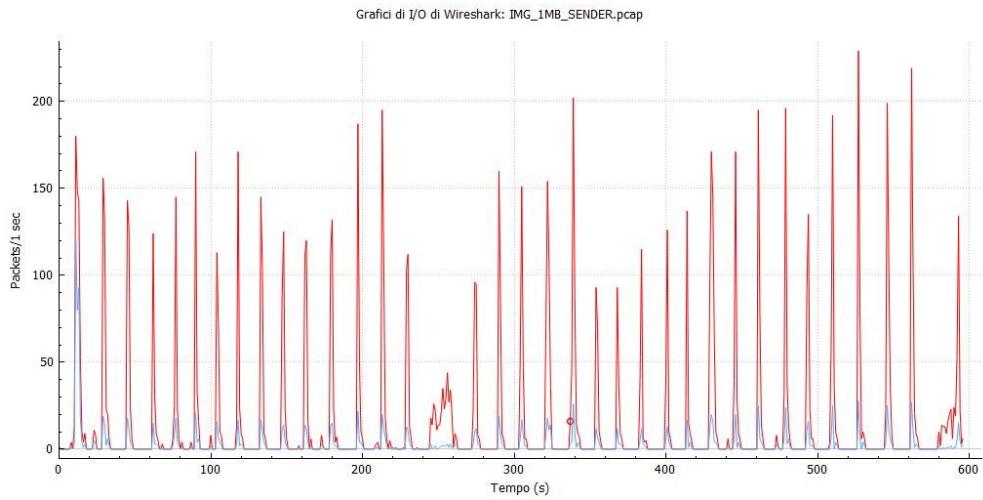


Figura 7.6 - Grafico I/O Invio Immagini 1MB - In rosso TCP, in blu TLS (Y=Packetti)

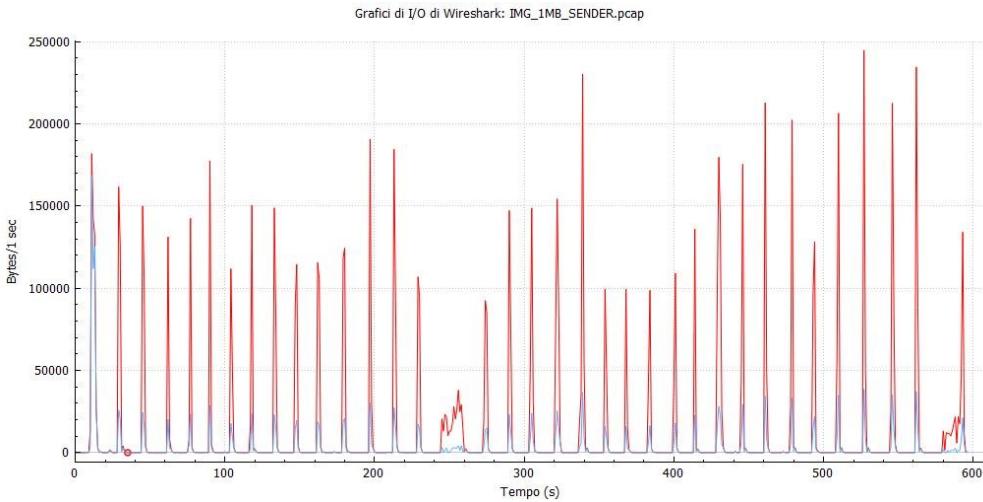


Figura 7.7 - Grafico I/O Invio Immagini 1MB - In rosso TCP, in blu TLS (Y=Bytes)

Dai grafici si evincono i vari picchi relativi all'invio delle immagini. Come si può notare, l'invio di foto è rappresentato da picchi TCP alti e TLS bassi.

Dal lato dell'invitante sono state individuati i seguenti server WhatsApp:

- 182.60.216.54 – (static-mum-182.60.216.54.mtnl.net.in) India
- 157.240.20.15 – (edge-star-shv-02-frt3.facebook.com) Germany
- 157.240.193.50 – (whatsapp-cdn-shv-01-fco1.fbcdn.net) USA

Ricezione Immagini 1MB

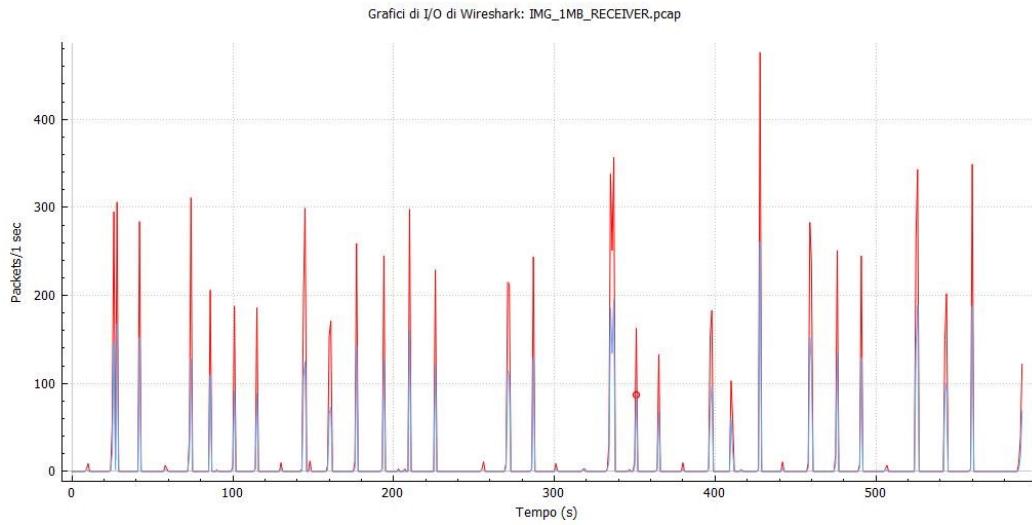


Figura 7.8 - Grafico I/O Ricezione Immagini 1MB - In rosso TCP, in blu TLS (Y=Packetti)

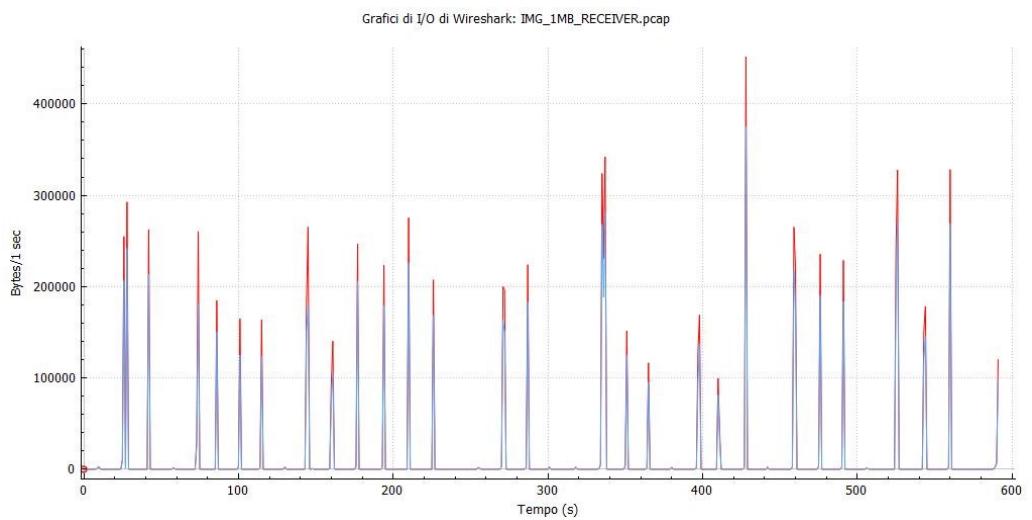


Figura 7.9 - Grafico I/O Ricezione Immagini 1MB - In rosso TCP, in blu TLS (Y=Bytes)

Di grafici si evincono i picchi relativi alla ricezione di un'immagine. Come si può notare, La ricezione di foto è rappresentata da picchi TCP e TLS alti.

Dal lato del ricevente sono state individuati i seguenti server WhatsApp:

- 157.240.193.55 – (whatsapp-chatd-edge-shv-01-fco1.facebook.com) USA
- 157.240.193.50 – (whatsapp-cdn-shv-01-fco1.fbcdn.net) USA

7.3.1.2

Test Case: IMG_4-6MB

Prendiamo in esame l'invio/ricezione delle immagini di peso tra 4-6MB (IMG_4-6MB), andando ad analizzare i rispettivi file PCAP generati dal lato inviante e dal lato ricevente.

Invio Immagini 4-6MB

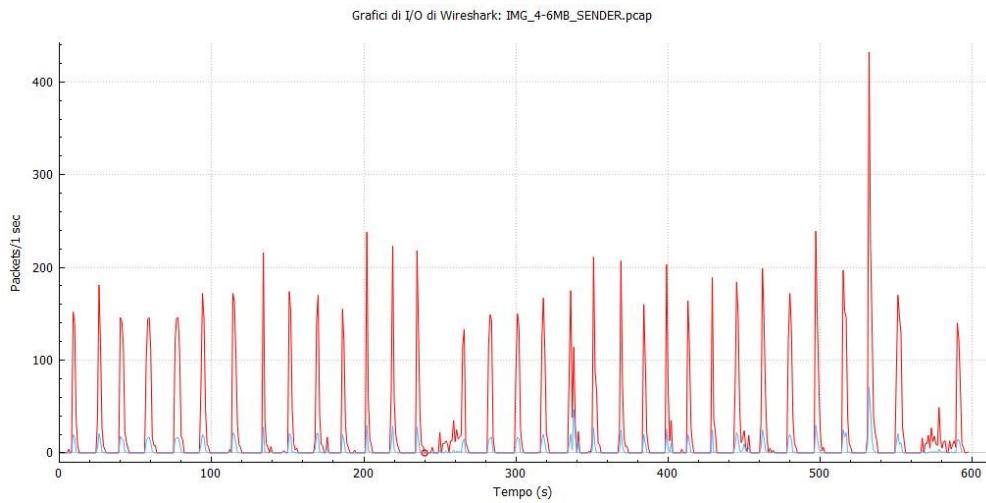


Figura 7.10 - Grafico I/O Invio Immagini 4-6MB - In rosso TCP, in blu TLS (Y=Packetti)

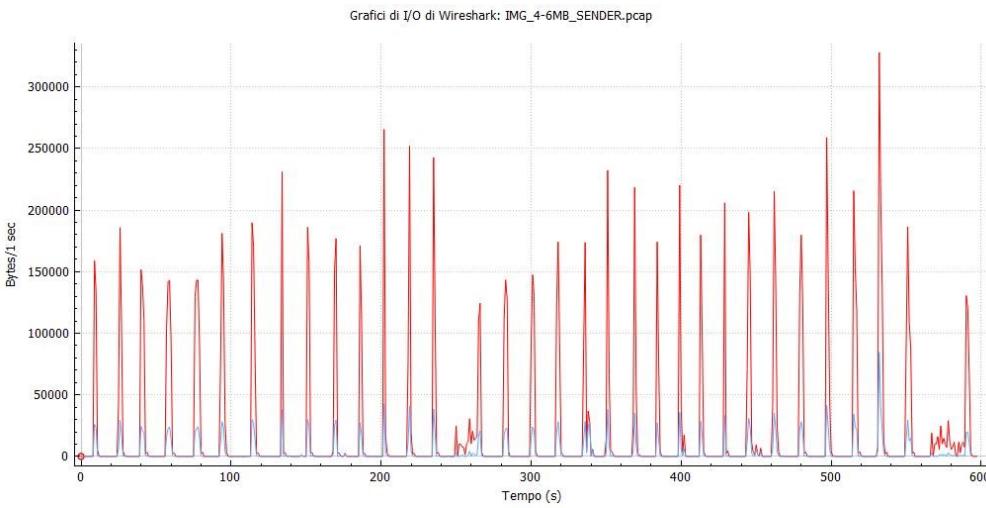


Figura 7.11 - Grafico I/O Invio Immagini 4-6MB - In rosso TCP, in blu TLS (Y=Bytes)

Dai grafici si evincono i picchi relativi all'invio di un'immagine. Come si può notare, l'invio di foto è rappresentato da picchi TCP alti e TLS bassi.

Dal lato dell'inviante sono state individuati i seguenti server WhatsApp:

- 157.240.193.50 – (whatsapp-cdn-shv-01-fco1.fbcdn.net) USA
- 157.240.193.55 – (whatsapp-chatd-edge-shv-01-fco1.facebook.com) USA

Ricezione Immagini 4-6MB

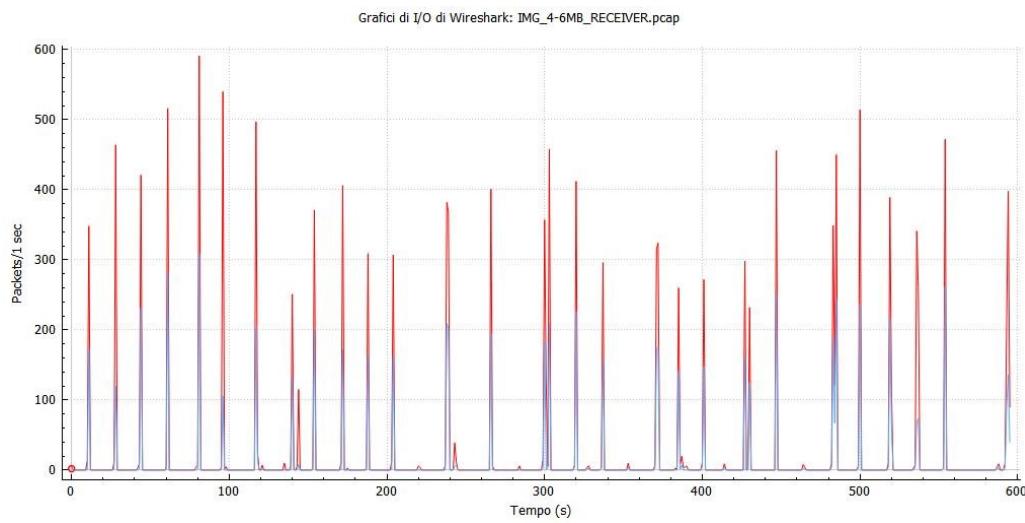


Figura 7.12 - Grafico I/O Ricezione Immagini 4-6MB - In rosso TCP, in blu TLS
(Y=Pacchetti)

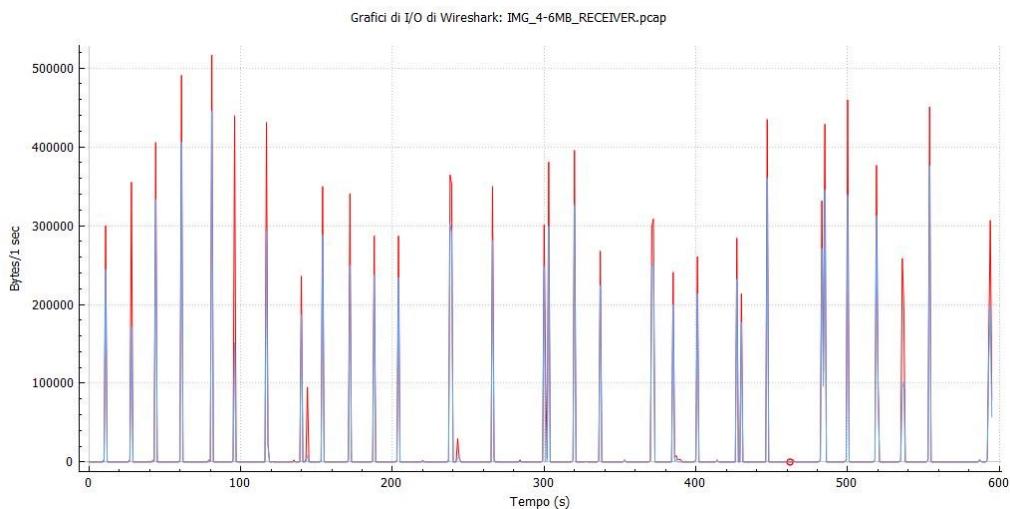


Figura 7.13 - Grafico I/O Ricezione Immagini 1MB - In rosso TCP, in blu TLS (Y=Bytes)

Di grafici si evincono i picchi relativi alla ricezione di un'immagine. Come si può notare, la ricezione di foto è rappresentata da picchi TCP e TLS alti.

Dal lato del ricevente sono state individuati i seguenti server WhatsApp:

- 157.240.193.50 – (whatsapp-cdn-shv-01-fco1.fbcdn.net) USA
- 157.240.193.55 – (whatsapp-chatd-edge-shv-01-fco1.facebook.com) USA

7.3.2 Invio/Ricezione dei video

7.3.2.1 *Test Case: VID_5-10MB*

Prendiamo in esame l'invio/ricezione di video di peso 5-10MB (VID_5-10MB), andando ad analizzare i rispettivi file PCAP generati dal lato inviante e dal lato ricevente.

Invio Video 5-10MB

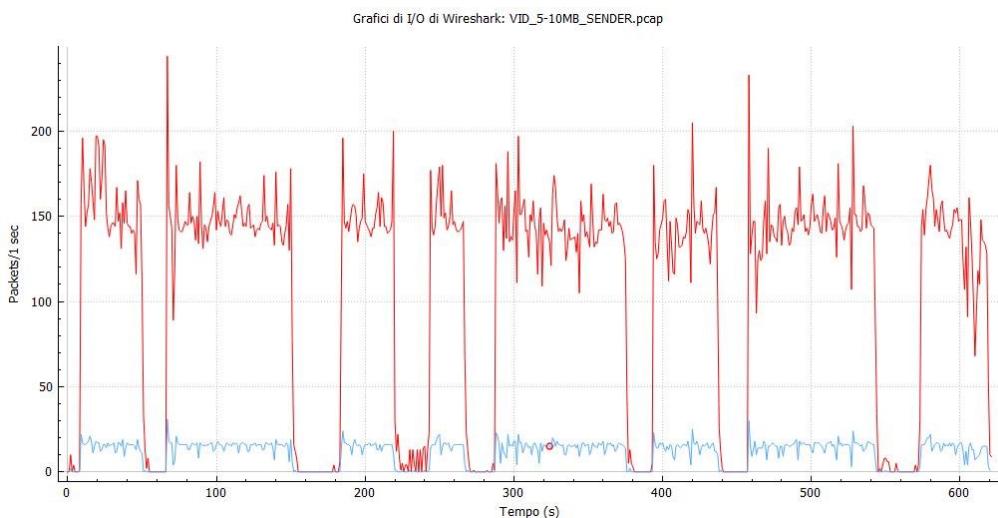


Figura 7.14 - Grafico I/O Invio Video 5-10MB - In rosso TCP, in blu TLS (Y=Pacchetti)

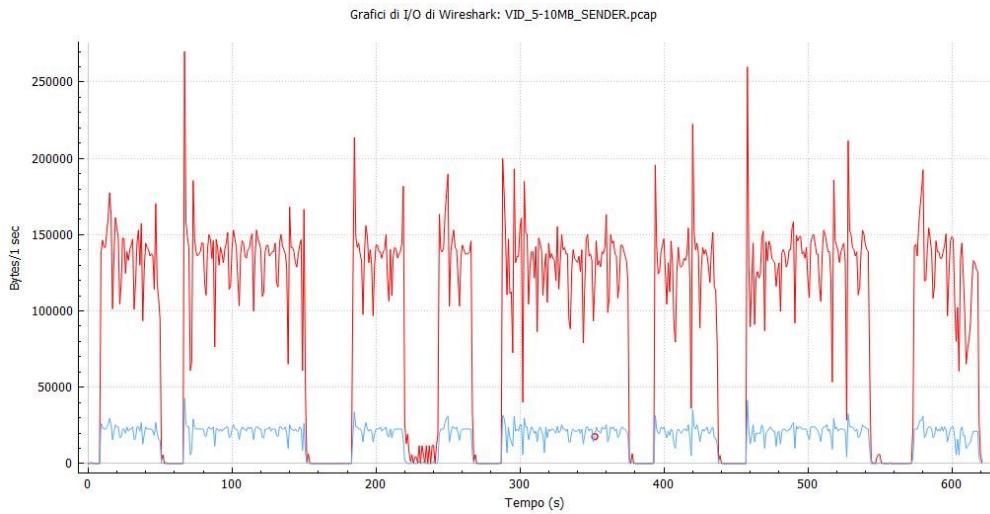


Figura 7.15 - Grafico I/O Ricezione Video5-101MB - In rosso TCP, in blu TLS (Y=Bytes)

Dai grafici si evincono i picchi relativi all'invio di un video. Come si può notare, l'invio di un video è rappresentato da picchi TCP alti e TLS bassi.

Dal lato dell'inviaente sono state individuati i seguenti server WhatsApp:

- 157.240.193.55 – (whatsapp-chatd-edge-shv-01-fco1.facebook.com) USA
- 157.240.193.50 – (whatsapp-cdn-shv-01-fco1.fbcdn.net) USA

Ricezione Video 5-10MB

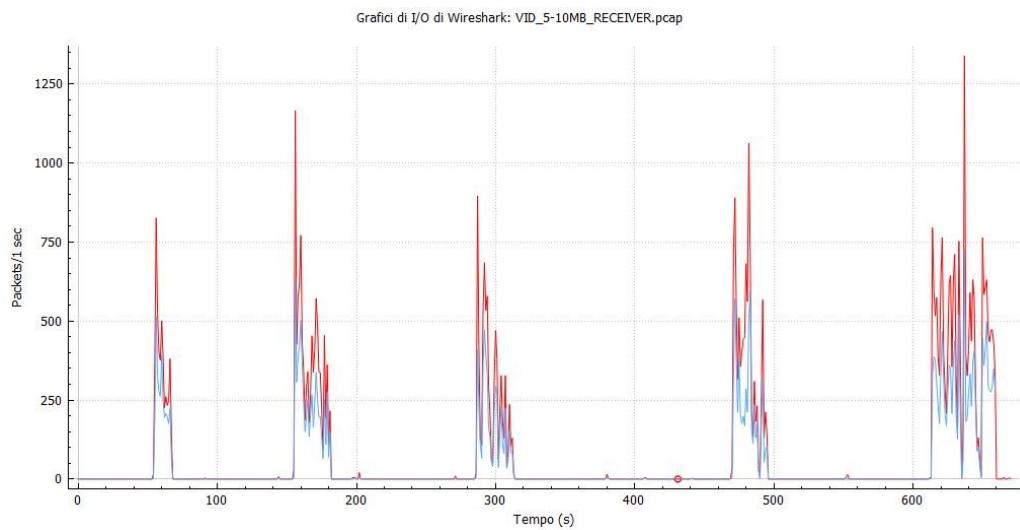


Figura 7.16 - Grafico I/O Ricezione Video 5-10MB - In rosso TCP, in blu TLS (Y=Pacchetti)

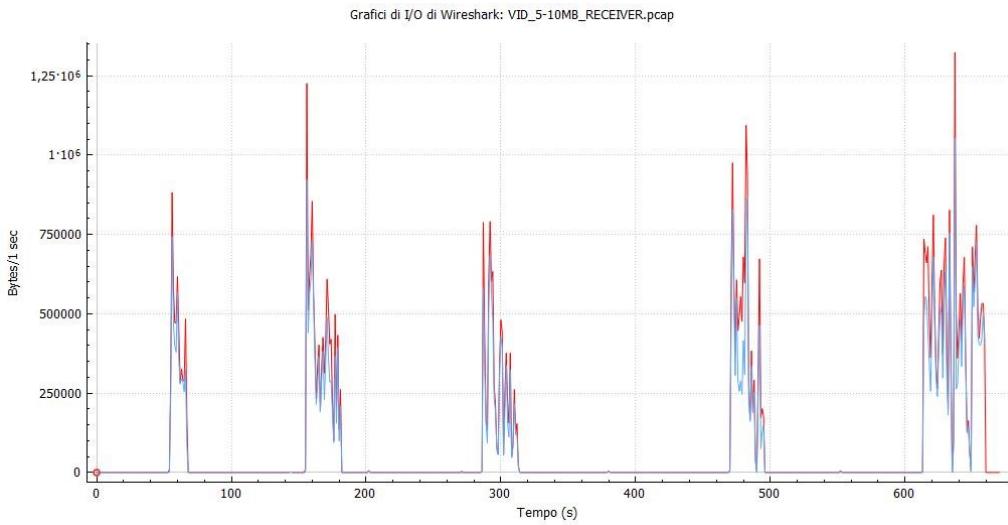


Figura 7.17 - Grafico I/O Ricezione Video 5-10MB - In rosso TCP, in blu TLS (Y=Bytes)

Di grafici si evincono i picchi relativi alla ricezione dei video. Come si può notare, la ricezione di un video è rappresentata da picchi TCP e TLS alti.

Dal lato del ricevente sono state individuati i seguenti server WhatsApp:

- 157.240.193.50 – (whatsapp-cdn-shv-01-fco1.fbcdn.net) USA
- 157.240.193.55 – (whatsapp-chatd-edge-shv-01-fco1.facebook.com) USA

7.3.2.2

Test Case: VID_20MB

Prendiamo in esame l'invio/ricezione di video di peso 20MB (VID_20MB), andando ad analizzare i rispettivi file PCAP generati dal lato inviante e dal lato ricevente.

Invio Video 20MB

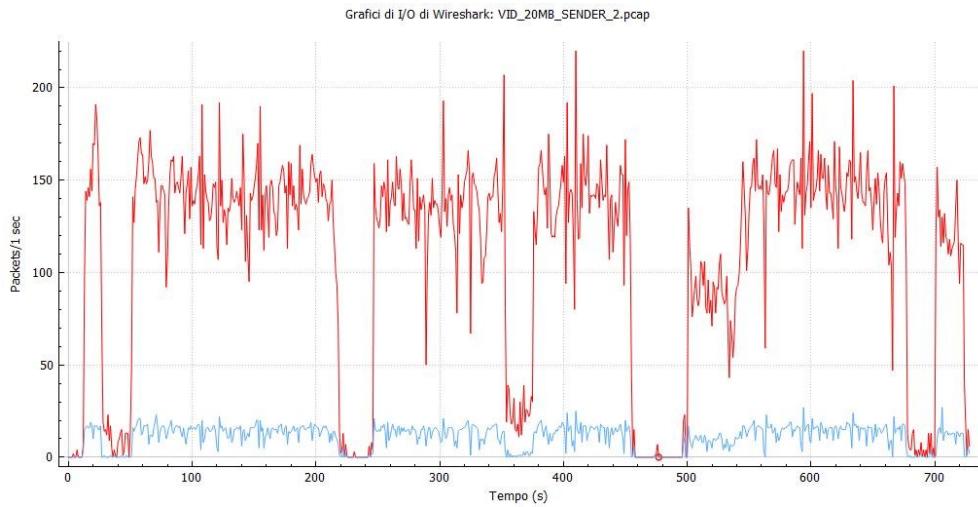


Figura 7.18 - Grafico I/O Invio Video 20MB - In rosso TCP, in blu TLS (Y=Pacchetti)

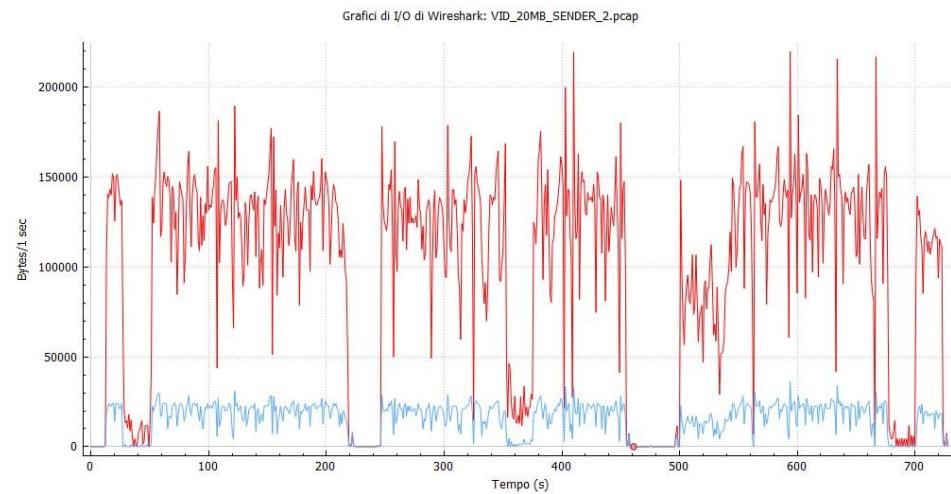


Figura 7.19 - Grafico I/O Invio Video 20MB - In rosso TCP, in blu TLS (Y=Bytes)

Dai grafici si evincono i picchi relativi all'invio dei video. Come si può notare, l'invio di un video è rappresentato da picchi TCP alti e TLS bassi.

Dal lato dell'inviatore sono state individuati i seguenti server WhatsApp:

- 157.240.193.55 – (whatsapp-chatd-edge-shv-01-fco1.facebook.com) USA
- 157.240.193.50 – (whatsapp-cdn-shv-01-fco1.fbcdn.net) USA
- 157.240.193.34 – (edge-mqtt-mini-shv-01-fco1.facebook.com) USA

Ricezione Video 20MB

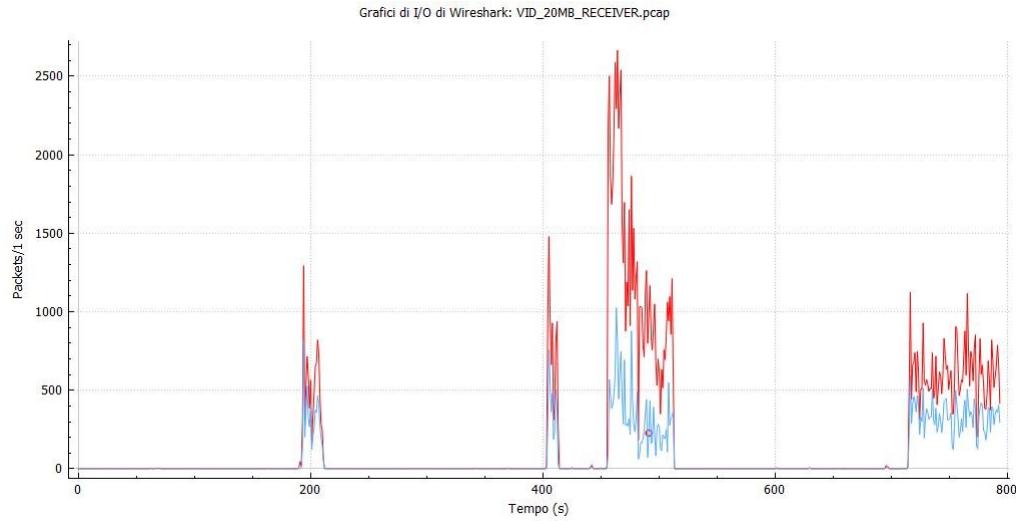


Figura 7.20 - Grafico I/O Ricezione Video 20MB - In rosso TCP, in blu TLS (Y=Packetti)

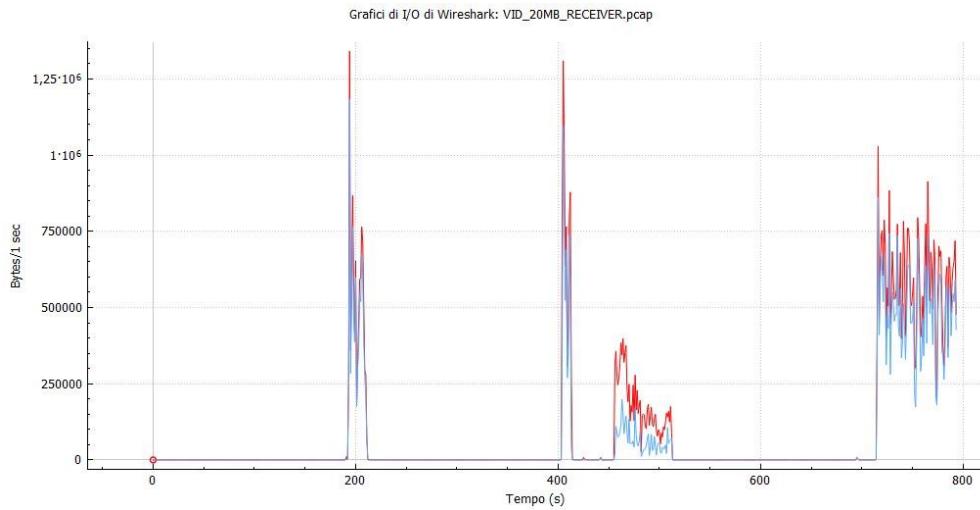


Figura 7.21 - Grafico I/O Ricezione Video 20MB - In rosso TCP, in blu TLS (Y=Bytes)

Di grafici, si evincono i picchi relativi alla ricezione dei video. Come si può notare, la ricezione di un video è rappresentata da picchi TCP e TLS alti.

Dal lato del ricevente sono state individuati i seguenti server WhatsApp:

- 157.240.193.50 – (whatsapp-cdn-shv-01-fco1.fbcdn.net) USA
- 157.240.193.55 – (whatsapp-chatd-edge-shv-01-fco1.facebook.com) USA

7.3.3 Invio/Ricezione degli audio

7.3.3.1 *Test Case: AUD_1MB*

Prendiamo in esame l'invio/ricezione di audio di peso 1.3MB (AUD_1MB), andando ad analizzare i rispettivi file PCAP generati dal lato inviante e dal lato ricevente.

Invio Audio 1MB

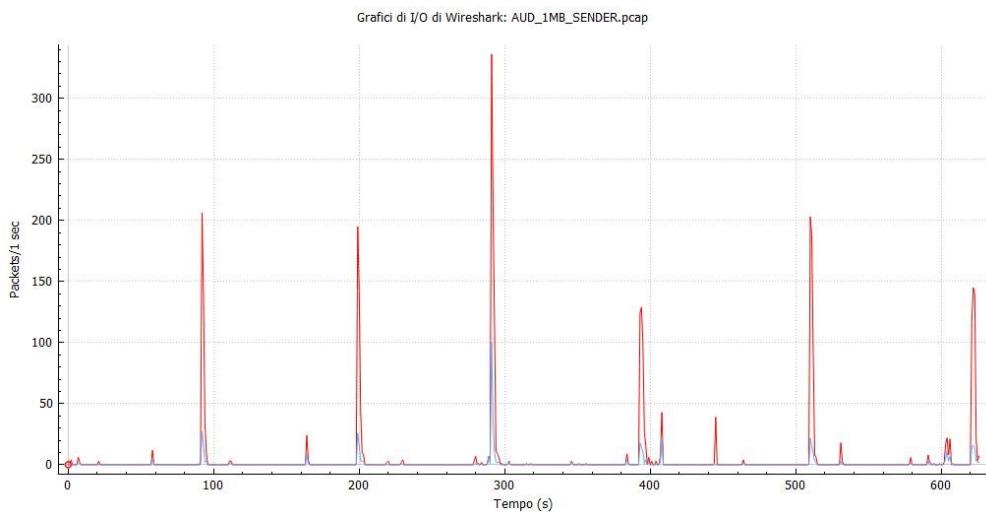


Figura 7.22 - Grafico I/O Invio Audio 1MB - In rosso TCP, in blu TLS (Y=Pacchetti)

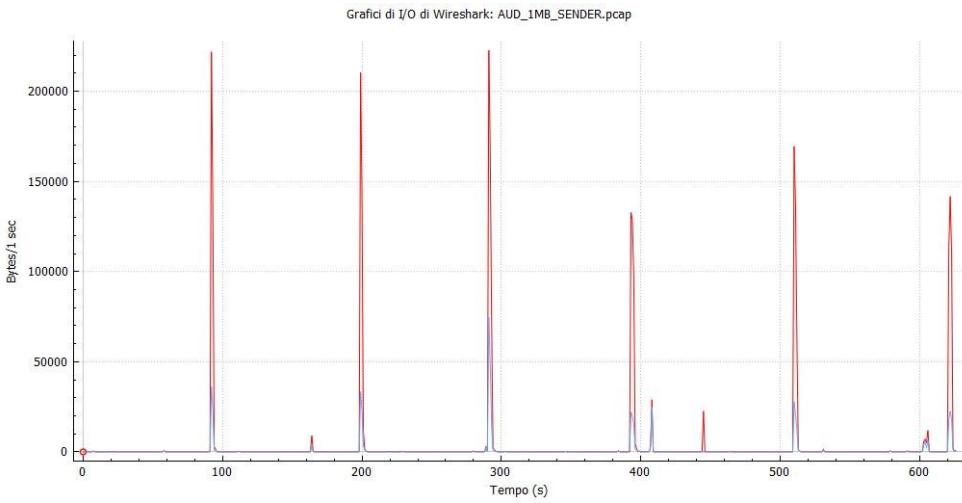


Figura 7.23 - Grafico I/O Invio Audio 1MB - In rosso TCP, in blu TLS (Y=Bytes)

Dai grafici si evincono i picchi relativi all'invio degli audio. Come si può notare, l'invio di un audio è rappresentato da picchi TCP alti e TLS bassi.

Dal lato dell'invitante sono state individuati i seguenti server WhatsApp:

- 157.240.193.55 – (whatsapp-chatd-edge-shv-01-fco1.facebook.com) USA
- 157.240.193.50 – (whatsapp-cdn-shv-01-fco1.fbcdn.net) USA
- 157.240.193.34 – (edge-mqtt-mini-shv-01-fco1.facebook.com) USA

Ricezione Audio 1MB

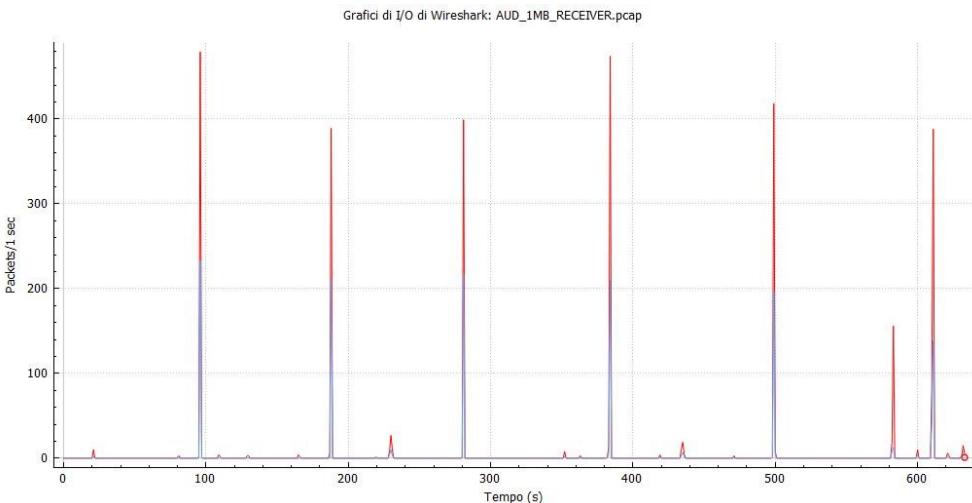


Figura 7.24 - Grafico I/O Ricezione Audio 1MB - In rosso TCP, in blu TLS (Y=Pacchetti)

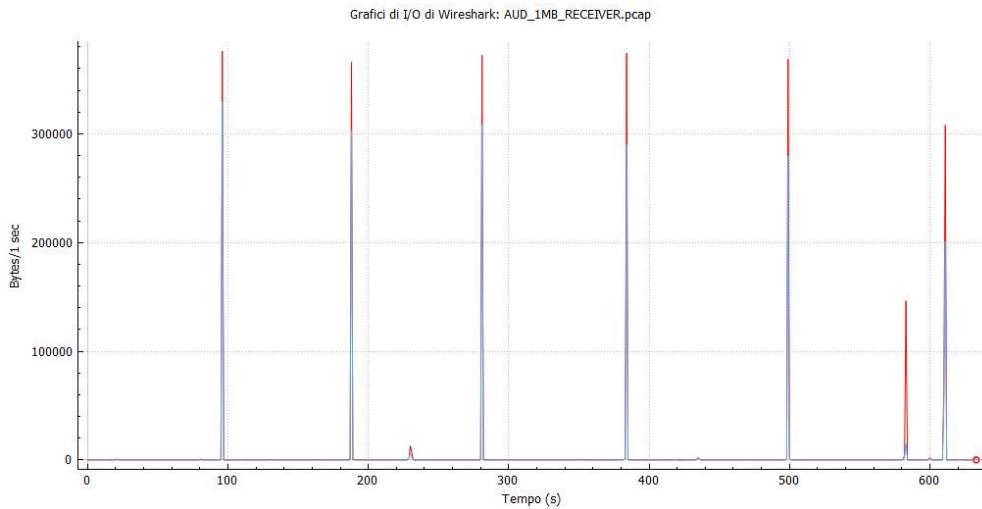


Figura 7.25 - Grafico I/O Ricezione Audio 1MB - In rosso TCP, in blu TLS (Y=Bytes)

Di grafici si evincono i picchi relativi alla ricezione degli audio. Come si può notare, la ricezione di un audio è rappresentata da picchi TCP e TLS alti. Da notare il picco più basso è in realtà un messaggio testuale, in quanto durante l'esperimento il ricevente ha ricevuto un messaggio testuale da parte di un altro utente.

Dal lato del ricevente sono state individuati i seguenti server WhatsApp:

- 157.240.193.50 – (whatsapp-cdn-shv-01-fco1.fbcdn.net) USA
- 157.240.193.55 – (whatsapp-chatd-edge-shv-01-fco1.facebook.com) USA

7.3.3.2

Test Case: AUD_6MB

Prendiamo in esame l'invio/ricezione di audio di peso 6MB (AUD_6MB), andando ad analizzare i rispettivi file PCAP generati dal lato inviante e dal lato ricevente.

Invio Audio 6MB

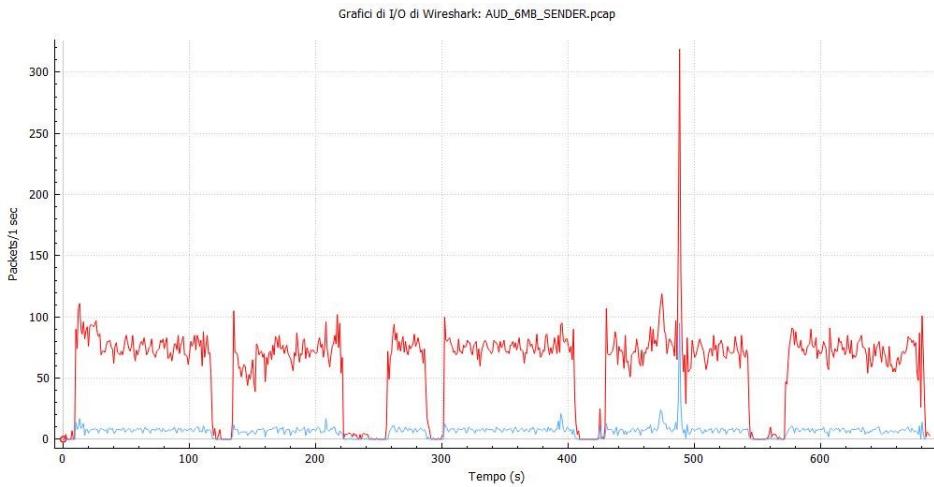


Figura 7.26 - Grafico I/O Invio Audio 6MB - In rosso TCP, in blu TLS (Y=Pacchetti)

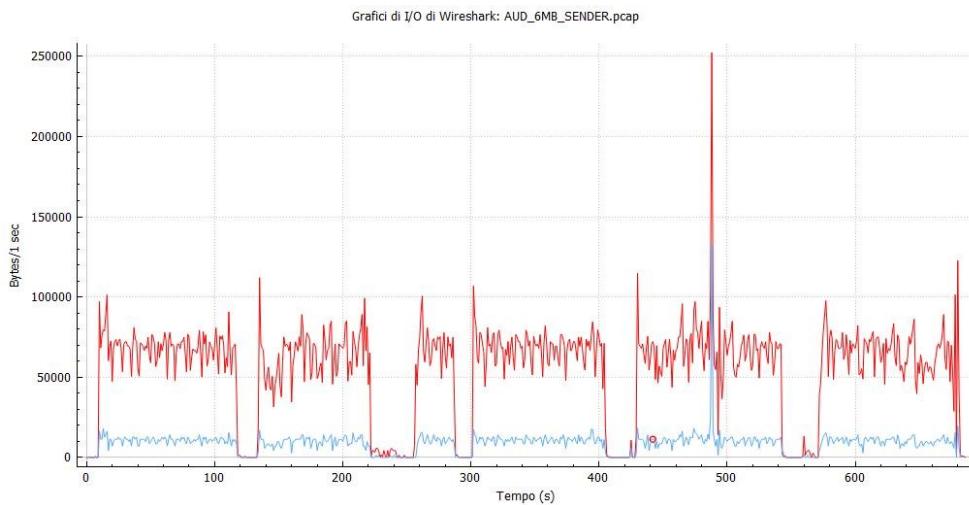


Figura 7.27 - Grafico I/O Invio Audio 6MB - In rosso TCP, in blu TLS (Y=Bytes)

Dai grafici si evincono i picchi relativi all'invio degli audio. Come si può notare, l'invio di un audio è rappresentato da picchi TCP alti e TLS bassi.

Dal lato dell'invitante sono state individuati i seguenti server WhatsApp:

- 157.240.193.55 – (whatsapp-chatd-edge-shv-01-fco1.facebook.com) USA
- 157.240.193.50 – (whatsapp-cdn-shv-01-fco1.fbcdn.net) USA
- 157.240.193.34 – (edge-mqtt-mini-shv-01-fco1.facebook.com) USA

Ricezione Audio 6MB

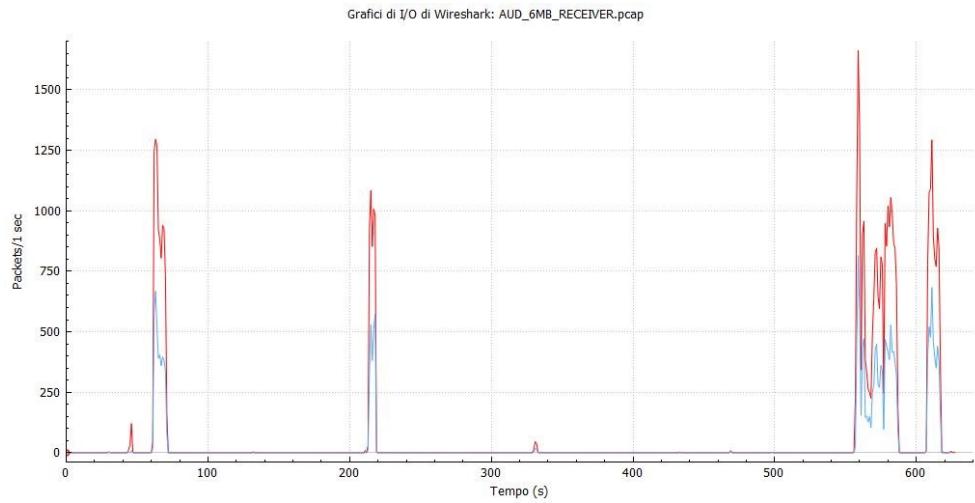


Figura 7.28 - Grafico I/O Ricezione Audio 6MB - In rosso TCP, in blu TLS (Y=Pacchetti)

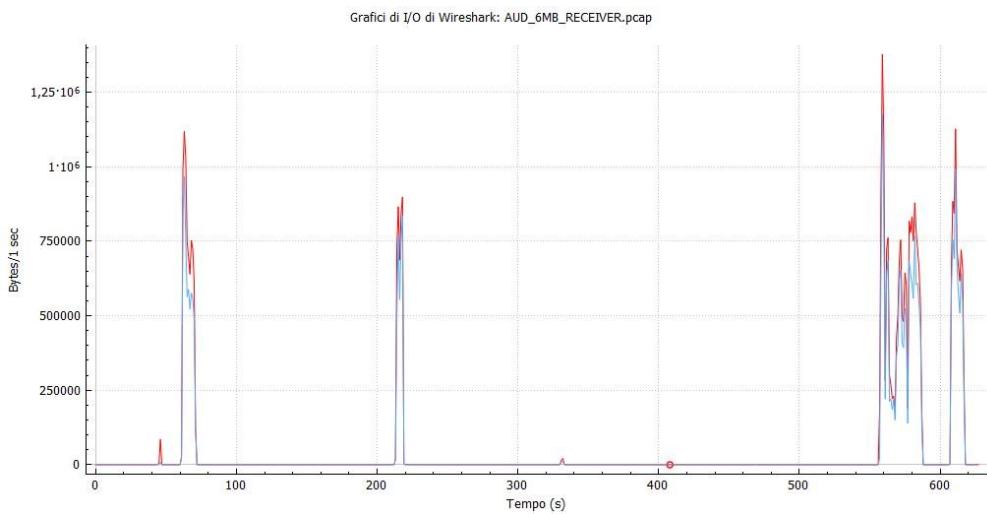


Figura 7.29 - Grafico I/O Ricezione Audio 6MB - In rosso TCP, in blu TLS (Y=Bytes)

Di grafici, si evincono i picchi relativi alla ricezione degli audio. Come si può notare, la ricezione di un audio è rappresentata da picchi TCP e TLS alti.

Dal lato del ricevente sono state individuati i seguenti server WhatsApp:

- 157.240.193.50 – (whatsapp-cdn-shv-01-fco1.fbcdn.net) USA
- 157.240.193.55 – (whatsapp-chatd-edge-shv-01-fco1.facebook.com) USA

7.3.4 Invio/Ricezione dei file

7.3.4.1

Test Case: FIL_1-2MB

Prendiamo in esame l'invio/ricezione di file di peso tra 1-2MB (FIL_1-2MB), andando ad analizzare i rispettivi file PCAP generati dal lato inviante e dal lato ricevente.

Invio File 1-2MB

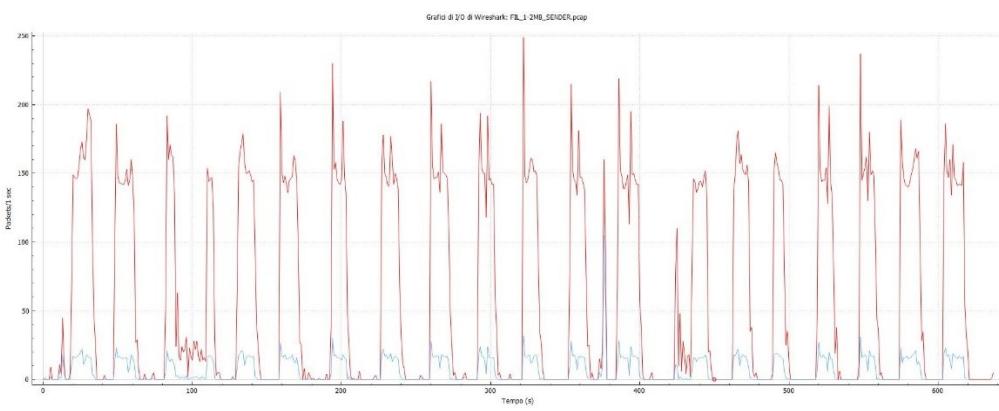


Figura 7.30 - Grafico I/O Invio File 1-2MB - In rosso TCP, in blu TLS (Y=Pacchetti)

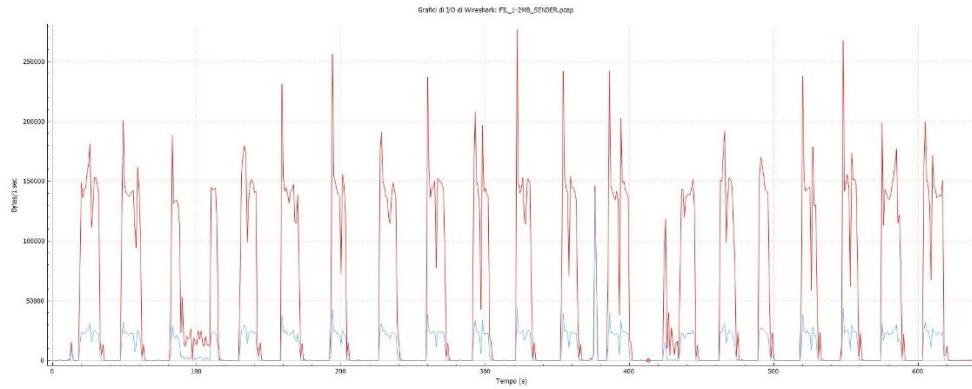


Figura 7.31 - Grafico I/O Invio File 1-2MB - In rosso TCP, in blu TLS (Y=Bytes)

Dai grafici si evincono i picchi relativi all'invio dei file. Come si può notare, l'invio di un file è rappresentato da picchi TCP alti e TLS bassi.

Dal lato dell'inviante sono state individuati i seguenti server WhatsApp:

- 157.240.193.55 – (whatsapp-chatd- edge-shv-01-fco1.facebook.com) USA
- 157.240.193.50 – (whatsapp-cdn-shv-01-fco1.fbcdn.net) USA
- 157.240.193.34 – (edge-mqtt-mini-shv-01-fco1.facebook.com) USA
- 157.240.193.18 – (edge-star-shv-01-fco1.facebook.com) USA
- 157.240.193.63 – (instagram-p3-shv-01-fco1.fbcdn.net) USA

Ricezione File 1-2MB

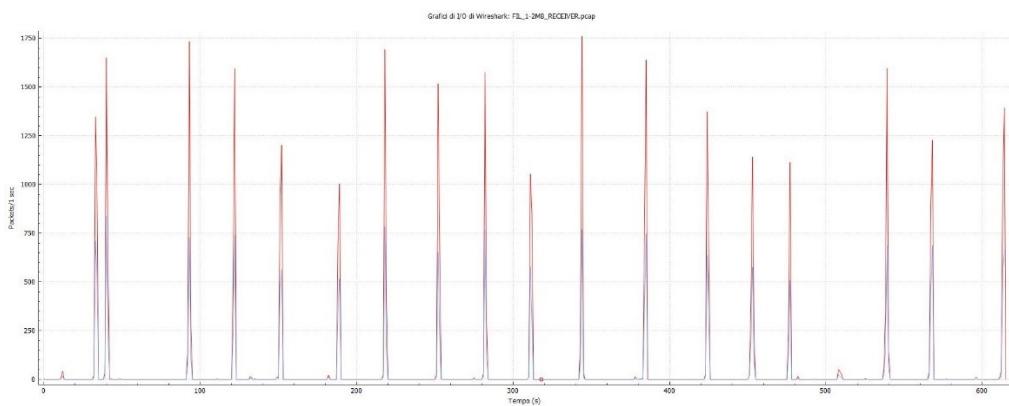


Figura 7.32 - Grafico I/O Ricezione File 1-2MB - In rosso TCP, in blu TLS (Y=Pacchetti)

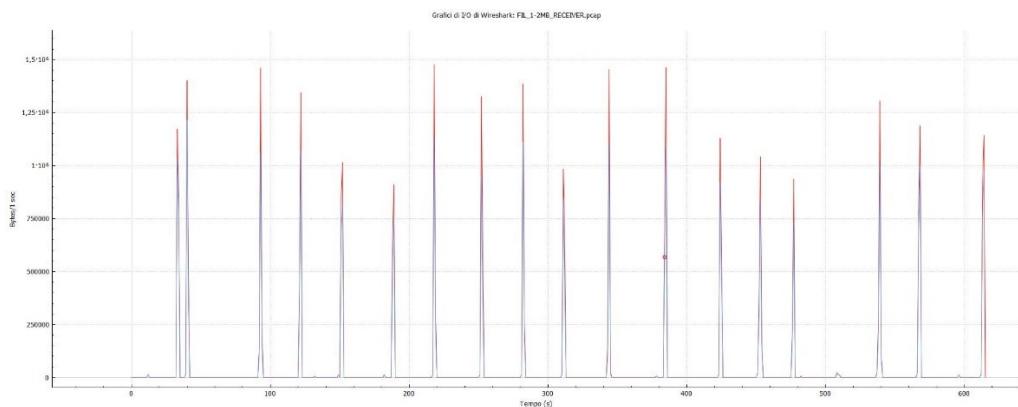


Figura 7.33 - Grafico I/O Ricezione File 1-2MB - In rosso TCP, in blu TLS (Y=Bytes)

Di grafici si evincono i picchi relativi alla ricezione dei file. Come si può notare, la ricezione di un file è rappresentata da picchi TCP e TLS alti.

Dal lato del ricevente sono state individuati i seguenti server WhatsApp:

- 157.240.193.50 – (whatsapp-cdn-shv-01-fco1.fbcdn.net) USA

- 157.240.193.55 – (whatsapp-chatd-edge-shv-01-fco1.facebook.com) USA
- 157.240.193.18 – (edge-star-shv-01-fco1.facebook.com) USA
- 157.240.193.63 – (instagram-p3-shv-01-fco1.fcdn.net) USA

7.3.4.2 *Test Case: FIL_10MB*

Prendiamo in esame l'invio/ricezione di audio di peso tra 10MB (FIL_10MB), andando ad analizzare i rispettivi file PCAP generati dal lato inviante e dal lato ricevente.

Invio File 10MB

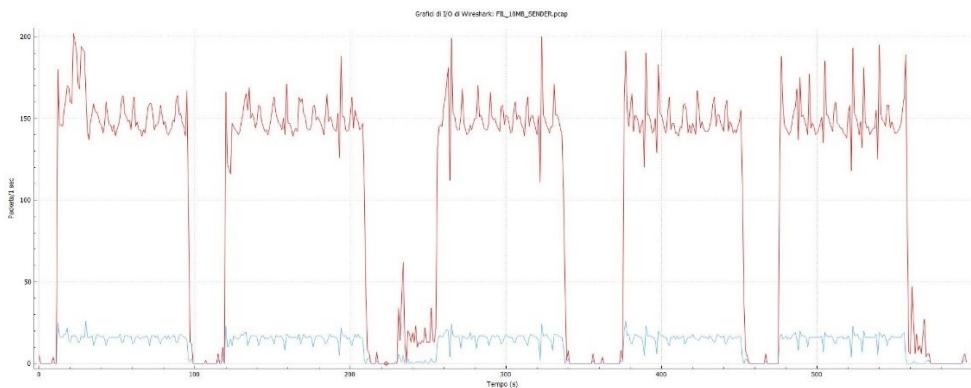


Figura 7.34 - Grafico I/O Invio File 5-10MB - In rosso TCP, in blu TLS (Y=Packetti)

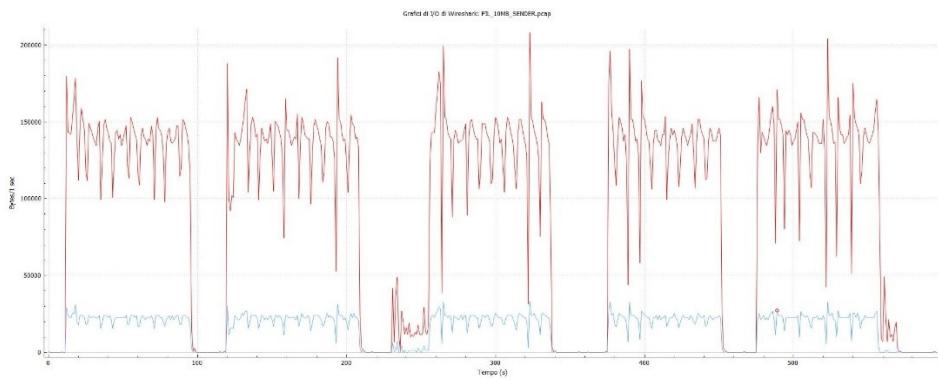


Figura 7.35 - Grafico I/O Invio File 5-10MB - In rosso TCP, in blu TLS (Y=Bytes)

Dai grafici si evincono i picchi relativi all'invio dei file. Come si può notare, l'invio di un file è rappresentato da picchi TCP alti e TLS bassi.

Dal lato dell'inviaente sono state individuati i seguenti server WhatsApp:

- 157.240.193.55 – (whatsapp-chatd-edge-shv-01-fco1.facebook.com) USA
- 157.240.193.50 – (whatsapp-cdn-shv-01-fco1.fbcdn.net) USA
- 157.240.193.18 – (edge-star-shv-01-fco1.facebook.com) USA
- 157.240.193.63 – (instagram-p3-shv-01-fco1.fbcdn.net) USA

Ricezione File 10MB

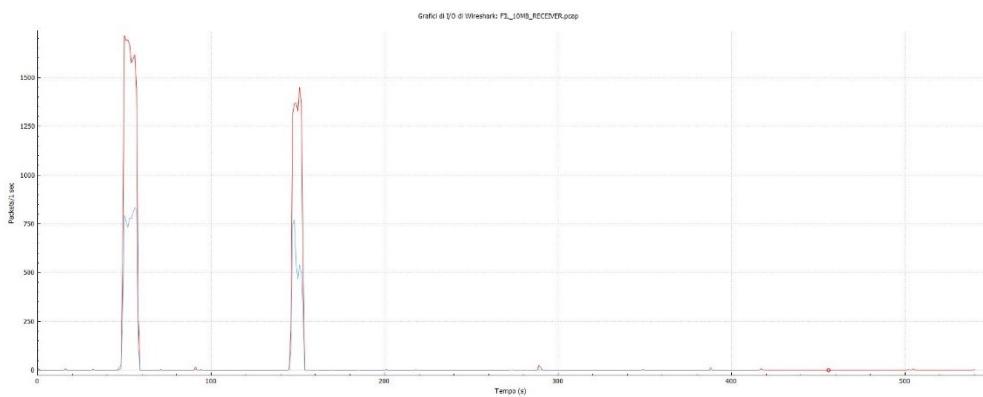


Figura 7.36 - Grafico I/O Ricezione File 10MB - In rosso TCP, in blu TLS (Y=Pacchetti)

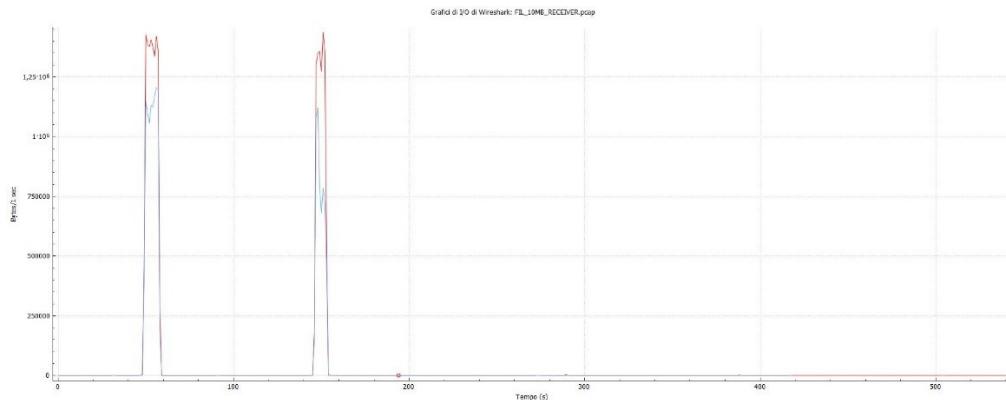


Figura 7.37 - Grafico I/O Ricezione File 10MB - In rosso TCP, in blu TLS (Y=Bytes)

Di grafici, si evincono i picchi relativi alla ricezione dei file. Come si può notare, la ricezione di un file è rappresentata da picchi TCP e TLS alti.

Dal lato del ricevente sono state individuati i seguenti server WhatsApp:

- 157.240.193.50 – (whatsapp-cdn-shv-01-fco1.fbcdn.net) USA
- 157.240.193.55 – (whatsapp-chatd-edge-shv-01-fco1.facebook.com) USA

7.3.5 Conclusioni

In questo paragrafo si raccolgono le evidenze individuate a seguito di una analisi forense dello scambio di oggetti tramite l'app WhatsApp. Il traffico dell'invio/ricezione di immagini, audio e file di dimensioni sul MB è rappresentato da un picco TCP, le cui caratteristiche sono quelle di essere breve e alto; mentre il traffico relativo all'invio/ricezione di video, file e audio di dimensioni sui 5MB in su è rappresentato da picchi TCP stretti, lunghi e consequenziali.

Di seguito, nella tabella, sono riportati i server contatti in tutti i test case (occorre sottolineare che WhatsApp è di proprietà di Facebook, così come Instagram).

Server	Resolve Host	Inviante	Ricevente
185.60.216.54	whatsapp-chatd-edge-shv-01-frx5.facebook.com	IMG_1MB	
157.240.20.15	edge-star-shv-02-frt3.facebook.com	IMG_1MB	
157.240.193.50	whatsapp-cdn-shv-01-fco1.fcdn.net	IMG_1MB IMG_4-6MB VID_5-10MB VID_20MB AUD_1MB AUD_6MB FIL_1-2MB FIL_10MB	IMG_1MB IMG_4-6MB VID_5-10MB VID_20MB AUD_1MB AUD_6MB FIL_1-2MB FIL_10MB
157.240.193.55	whatsapp-chatd-edge-shv-01-fco1.facebook.com	IMG_4-6MB VID_5-10MB VID_20MB AUD_1MB AUD_6MB FIL_1-2MB FIL_10MB	IMG_1MB IMG_4-6MB VID_5-10MB VID_20MB AUD_1MB AUD_6MB FIL_1-2MB FIL_10MB
157.240.193.34	edge-mqtt-mini-shv-01-fco1.facebook.com	VID_20MB AUD_1MB AUD_6MB FIL_1-2MB	
157.240.193.63	instagram-p3-shv-01-fco1.fcdn.net	FIL_1-2MB FIL_10MB	FIL_1-2MB
157.240.193.18	edge-star-shv-01-fco1.facebook.com	FIL_1-2MB FIL_10MB	FIL_1-2MB

Tabella 7.4 - Server di WhatsApp individuati nei test case

Il server che viene contatto in entrambi i lati in ogni test case è il server 157.240.193.50; Anche il server 157.240.193.55 viene contatto in entrambi i lati, tranne che per il test case relativo all'invio dell'immagine di 1MB. Tuttavia,

analizzando i flussi (vedi sezione 6.4) delle connessioni si può notare che sono quelli maggiormente coinvolti nell’interazione con i client, cioè generano maggior traffico di rete. Soffermandoci sempre sui flussi si può notare che vengono utilizzate soltanto due porte per il traffico in uscita: 5222 e 443.

7.4 Generazione del Dataset tramite IMSD Parser

In questa sezione si presenterà la generazione di vari dataset (file CSV) utilizzando il tool IMSD Parser. Ogni dataset ha una modalità differente di frequenza di campionamento e, inoltre, ciascuno di essi conterrà le features relativi ai flussi dei Test Case (descritti in 7.1.1), considerando soltanto i file PCAP del lato inviante.

Le features vengono estrapolate da tutti i flussi di tutte le connessioni individuate tra il client e i server IM, ovvero eseguendo il comando `!EXTRACT_FEATURE_CONN`. Ad ogni analisi di un nuovo test case viene aggiornato il rispettivo dataset (Il tool richiede del tempo, non del tutto trascurabile, per la preparazione e l’elaborazione).

Nelle tabelle di seguito sono definiti i flussi rilevati all’interno delle catture PCAP dei vari Test Case.

	Flussi Lato Inviaente	Flussi Lato Ricevente
IMG_1MB	192.168.137.141 : 37300 - 185.60.216.54 : 443 192.168.137.141 : 37704 - 157.240.20.15 : 443 192.168.137.141 : 41863 - 157.240.193.50: 443 192.168.137.141 : 41864 - 157.240.193.50: 443	192.168.137.5 : 44699 - 157.240.193.55 : 443 192.168.137.5 : 44695 - 157.240.193.55 : 443 192.168.137.5 : 44697 - 157.240.193.55 : 443 192.168.137.5 : 44700 - 157.240.193.55 : 443 192.168.137.5 : 44691 - 157.240.193.55 : 443 192.168.137.5 : 38121 - 157.240.193.50 : 443 192.168.137.5 : 38119 - 157.240.193.50 : 443 192.168.137.5 : 38124 - 157.240.193.50 : 443
IMG_4-6MB	192.168.137.141 : 58955 - 157.240.193.55 : 5222 192.168.137.141 : 41885 - 157.240.193.50 : 443 192.168.137.141 : 41884 - 157.240.193.50 : 443	192.168.137.5 : 44715 - 157.240.193.55 : 443 192.168.137.5 : 43152 - 157.240.193.55 : 5222 192.168.137.5 : 39213 - 157.240.193.50 : 443 192.168.137.5 : 39215 - 157.240.193.50 : 443
VID_5-10MB	192.168.137.141 : 33284 - 157.240.193.55 : 5222 192.168.137.141 : 44447 - 157.240.193.50 : 443 192.168.137.141 : 44446 - 157.240.193.50 : 443	192.168.137.5 : 44229 - 157.240.193.55 : 5222 192.168.137.5 : 44236 - 157.240.193.55 : 5222 192.168.137.5 : 45793 - 157.240.193.55 : 443 192.168.137.5 : 39213 - 157.240.193.50 : 443 192.168.137.5 : 39215 - 157.240.193.50 : 443 192.168.137.5 : 39221 - 157.240.193.50 : 443 192.168.137.5 : 39219 - 157.240.193.50 : 443
VID_20MB	192.168.137.141 : 53441 - 157.240.193.55 : 443 192.168.137.141 : 44924 - 157.240.193.50 : 443 192.168.137.141 : 44923 - 157.240.193.50 : 443 192.168.137.141 : 60261 - 157.240.193.34 : 443	192.168.137.5 : 45182 - 157.240.193.55 : 5222 192.168.137.5 : 46747 - 157.240.193.55 : 443 192.168.137.5 : 45184 - 157.240.193.55 : 5222 192.168.137.5 : 46745 - 157.240.193.55 : 443 192.168.137.5 : 46738 - 157.240.193.55 : 443 192.168.137.5 : 40169 - 157.240.193.50 : 443 192.168.137.5 : 40171 - 157.240.193.50 : 443 192.168.137.5 : 40167 - 157.240.193.50 : 443 192.168.137.5 : 40165 - 157.240.193.50 : 443

Tabella 7.5 - flussi tra client e WhatsApp individuati nei test case

7.4.1 Generazione Dataset: frequenza diversa per ogni oggetto

Per la generazione del seguente dataset viene definita una determinata frequenza di campionamento per ogni oggetto e lo stride viene fissato ad 1.

Di seguito riportiamo in forma tabellare le frequenze di campionamento, le quali sono state determinate effettuando una media sul numero di pacchetti al secondo di ogni connessione di ciascuna categoria di oggetti (sono stati considerati soltanto i file PCAP lato inviante).

Inviante			
	Connessione	# Pacchetti al sec	Media
IMG_1MB	192.168.137.141 - 185.60.216.54	1.069	8.520 = 9
	192.168.137.141 - 157.240.20.15	2.0	
	192.168.137.141 - 157.240.193.50	16.213	
IMG_4-6MB	192.168.137.141 - 157.240.193.55	0.896	47.897 = 48
	192.168.137.141 - 157.240.193.50	22.424	
VID_5-10MB	192.168.137.141 - 157.240.193.55	0.299	11.459 = 11
	192.168.137.141 - 157.240.193.50	108.347	
VID_20MB	192.168.137.141 - 157.240.193.55	0.193	24.504 = 25
	192.168.137.141 - 157.240.193.50	109.814	
	192.168.137.141 - 157.240.193.34	20.832	
AUD_1MB	192.168.137.141 - 157.240.193.34	0.137	11.459 = 11
	192.168.137.141 - 157.240.193.55	0.199	
	192.168.137.141 - 157.240.193.50	4.767	
AUD_6MB	192.168.137.141 - 157.240.193.34	3.0	24.504 = 25
	192.168.137.141 - 157.240.193.55	0.170	
	192.168.137.141 - 157.240.193.50	60.480	
FIL_1-2MB	192.168.137.193 - 157.240.193.18	0.052	24.504 = 25
	192.168.137.193 - 157.240.193.34	0.179	
	192.168.137.193 - 157.240.193.55	0.180	
	192.168.137.193 - 157.240.193.50	70.074	
	192.168.137.193 - 157.240.193.63	34.272	
FIL_1-2MB	192.168.137.193 - 157.240.193.55	0.261	24.504 = 25
	192.168.137.193 - 157.240.193.63	0.476	
	192.168.137.193 - 157.240.193.50	113.046	

Tabella 7.6 - contenente la frequenza di campionamento calcolata per ogni oggetto

Il nome del dataset in formato CSV contenente tutte le features è “DATASET_SENDER_MUL_FREQ”, il quale viene aggiornato ad ogni nuova analisi di un nuovo test case.

Di seguito, nella tabella, riportiamo per ogni Test Case: i flussi individuati, la frequenza di campionamento scelta, lo stride e il numero di finestre temporali in cui viene suddiviso ogni flusso per estrarre le features.

Il file CSV “DATASET_SENDER_MUL_FREQ” contiene 312412 finestre temporali da cui sono state estratte le features.

	Inviaente		
	Flussi	Frequenza / Stride	N° Finestre Temporali
IMG_1MB	192.168.137.141 : 37300 - 185.60.216.54 : 443	9/1	629
	192.168.137.141 : 37704 - 157.240.20.15 : 443	9/1	2
	192.168.137.141 : 41863 - 157.240.193.50: 443	9/1	7
	192.168.137.141 : 41864 - 157.240.193.50: 443	9/1	9459
IMG_4-6MB	192.168.137.141 : 58955 - 157.240.193.55 : 5222	9/1	527
	192.168.137.141 : 41885 - 157.240.193.50 : 443	9/1	13090
	192.168.137.141 : 41884 - 157.240.193.50 : 443	9/1	9
VID_5-10MB	192.168.137.141 : 33284 - 157.240.193.55 : 5222	9/1	185
	192.168.137.141 : 44447 - 157.240.193.50 : 443	9/1	66285
	192.168.137.141 : 44446 - 157.240.193.50 : 443	9/1	6
VID_20MB	192.168.137.141 : 53441 - 157.240.193.55 : 443	48/1	139
	192.168.137.141 : 44924 - 157.240.193.50 : 443	48/1	78350
	192.168.137.141 : 44923 - 157.240.193.50 : 443	48/1	6
	192.168.137.141 : 60261 - 157.240.193.34 : 443	48/1	5
AUD_1MB	192.168.137.141 : 60165 - 157.240.193.34 : 443	11/1	3
	192.168.137.141 : 60159 - 157.240.193.34 : 443	11/1	3
	192.168.137.141 : 53354 - 157.240.193.55: 443	11/1	124
	192.168.137.141 : 44838 - 157.240.193.50 : 443	11/1	768
	192.168.137.141 : 44836 - 157.240.193.50 : 443	11/1	6
	192.168.137.141 : 44841 - 157.240.193.50 : 443	11/1	867
AUD_6MB	192.168.137.141 : 44850 - 157.240.193.50 : 443	11/1	899
	192.168.137.141 : 60260 - 157.240.193.34 : 443	11/1	3
	192.168.137.141 : 33802 - 157.240.193.55: 443	11/1	116
	192.168.137.141 : 44965 - 157.240.193.50 : 443	11/1	40743

Tabella 7.8 - contenente per ogni flusso il numero di finestre temporali calcolate in ogni test case

7.4.2 Generazione del Dataset: frequenza uguale per tutti gli oggetti

In questo paragrafo descriviamo un altro modo di generazione di un dataset denominato “DATASET_SENDER_FREQ_23”, il quale viene aggiornato ad ogni nuova analisi di un nuovo test case.

La frequenza di campionamento, questa volta, è la stessa per tutti gli oggetti. Il valore scelto è una media di tutte le frequenze di ciascun oggetto il cui valore è riportato nella casella “Media TOT” nella tabella precedente (pari a 23) . Lo stride viene fissato ad 1.

Ovviamente, il numero di finestre temporali per ogni flusso resta invariato, in quanto il numero delle finestre temporali dipende dal parametro stride.

Per completezza , riportiamo nella tabella per ogni Test Case: la frequenza di campionamento scelta (23), lo stride (1) e il numero di finestre temporali in cui viene suddiviso ogni flusso per estrapolare le features.

Il file CSV “DATASET_SENDER_FREQ_23” contiene 312412 finestre temporali (come il dataset precedente) da cui sono state estrapolate le features.

	Inviaente		
	Flussi	Frequenza / Stride	N° Finestre Temporali
IMG_1MB	192.168.137.141 : 37300 - 185.60.216.54 : 443	23/1	629
	192.168.137.141 : 37704 - 157.240.20.15 : 443	23/1	2
	192.168.137.141 : 41863 - 157.240.193.50: 443	23/1	7
	192.168.137.141 : 41864 - 157.240.193.50: 443	23/1	9459
IMG_4-6MB	192.168.137.141 : 58955 - 157.240.193.55 : 5222	23/1	527
	192.168.137.141 : 41885 - 157.240.193.50 : 443	23/1	13090
	192.168.137.141 : 41884 - 157.240.193.50 : 443	23/1	9
VID_5-10MB	192.168.137.141 : 33284 - 157.240.193.55 : 5222	23/1	185
	192.168.137.141 : 44447 - 157.240.193.50 : 443	23/1	66285
	192.168.137.141 : 44446 - 157.240.193.50 : 443	23/1	6
VID_20MB	192.168.137.141 : 53441 - 157.240.193.55 : 443	23/1	139
	192.168.137.141 : 44924 - 157.240.193.50 : 443	23/1	78350
	192.168.137.141 : 44923 - 157.240.193.50 : 443	23/1	6
	192.168.137.141 : 60261 - 157.240.193.34 : 443	23/1	5
AUD_1MB	192.168.137.141 : 60165 - 157.240.193.34 : 443	23/1	3
	192.168.137.141 : 60159 - 157.240.193.34 : 443	23/1	3
	192.168.137.141 : 53354 - 157.240.193.55: 443	23/1	124
	192.168.137.141 : 44838 - 157.240.193.50 : 443	23/1	768
	192.168.137.141 : 44836 - 157.240.193.50 : 443	23/1	6
	192.168.137.141 : 44841 - 157.240.193.50 : 443	23/1	867
AUD_6MB	192.168.137.141 : 44850 - 157.240.193.50 : 443	23/1	899
	192.168.137.141 : 60260 - 157.240.193.34 : 443	23/1	3
	192.168.137.141 : 33802 - 157.240.193.55: 443	23/1	116
	192.168.137.141 : 44965 - 157.240.193.50 : 443	23/1	40743

Tabella 7.9 - contenente per ogni flusso il numero di finestre temporali calcolate in ogni test case

8. Conclusione

Le applicazioni di Instant Messaging lasciano poche informazioni a disposizione da poter analizzare, in quanto il traffico di rete è cifrato.

L'unica soluzione adottabile, come visto, è applicare un'analisi statistica ispirata alla Side Channel Analysis. Applicandolo sugli esperimenti presentati è stato possibile individuare pattern ed informazioni:

- Per il traffico UDP, risulta facile individuare le fasi e lo svolgimento di chiamate VOIP;
- Per il traffico TCP/TLS, invece, relativo a stream di oggetti (immagini, video, audio e file) risulta più articolato determinare la tipologia del flusso di rete che si sta osservando.

Lo svantaggio principale è che un lavoro del genere se svolto in maniera manuale richiede tempo e un notevole sforzo, per cui non risulta fattibile se si vuole analizzare un traffico di rete voluminoso. Per tale motivo è necessario affidarsi a quelle che sono le tecniche di Intelligenza Artificiale per automatizzare tale processo.

L'obiettivo principale del lavoro di tesi è stato quello di sviluppare un tool denominato Instant Messaging Stream Detector Parser (IMSD Parser) per l'estrapolazione delle features da stream TCP di oggetti (immagini, video, audio e file) nei servizi di Instant Messaging.

È stato, inoltre, effettuato un esperimento per la generazione di un dataset di features utilizzando il tool IMSD Parser, il quale può essere utilizzato, ad esempio, come modello di addestramento per il machine learning. Tale esperimento si basa sull'analisi forense di alcune acquisizioni di rete di stream di oggetti tra due utenti WhatsApp.

Da tale analisi si evince che il traffico dell'invio/ricezione di immagini, audio e file di dimensioni sul MB è rappresentato da un picco TCP, le cui caratteristiche sono

quelle di essere breve e alto; mentre il traffico relativo all'invio/ricezione di video, file e audio di dimensioni sui 5MB in su è rappresentato da picchi TCP stretti, lunghi e consequenziali.

Le difficoltà maggiore riscontrata è la necessità di dovere effettuare manualmente l'invio di oggetti tra due utenti in uno scenario di rete. Inoltre, per non compromettere i risultati, occorre che il traffico generato avvenga solo tra i due utenti, cioè non deve esserci un terzo utente ad ostacolare l'esperimento.

Gli esempi e i casi di studio analizzati hanno mostrato la validità del processo e delle linee guida da seguire per ulteriori sviluppi futuri, i quali sono elencati di seguito:

- Analisi di altri servizi di Instant Messaging;
- Utilizzare e sperimentare altre metodologie di acquisizione
 - Comportamento dell'invio/ricezione di più volte di uno stesso oggetto (caso particolare presentato nella sezione 7.2);
- Analisi approfondita sulla correlazione dei dati/informazioni tra lato inviante e lato ricevente;
- Implementare una rete neurale per classificare il traffico
 - Addestrare la rete neurale con i dataset di features di stream di oggetti generati dal tool IMSD Parser (sezione 4.1.3);
 - data un'acquisizione PCAP di un invio/ricezione di un oggetto tramite un servizio IM, determinare a quale categoria di oggetti appartiene tale traffico;

Bibliografia

Capitolo 2

- [1] R. Pizzolante, Slide del Corso di Digital Forensics, 2018/2019
- [2] George M. Mohay, Computer and intrusion forensics, Artechhouse, 2003, p. 395
- [3] Peter Sommer, The future for the policing of cybercrime, in Computer Fraud & Security, 2004, pp. 8–12
- [4] D. Parker, Crime by Computer, 1976
- [5] C. Stoll, The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage, 1990
- [6] Peter Sommer, The future for the policing of cybercrime, 2004, pp. 8–12
- [7] Eoghan Casey, Digital Evidence and Computer Crime, 2004
- [8] SWGDE, Best practices for Computer Forensics (PDF), 2010
- [9] ISO, ISO/IEC 17025:2005, 2010
- [10] Peter Sommer, The future for the policing of cybercrime, in Computer Fraud & Security, 2004, pp. 8–12
- [11] SG Punja, Mobile device analysis (PDF), in Small Scale Digital Device Forensics Journal, 2008
- [12] Rizwan Ahmed, Mobile forensics: an overview, tools, future trends and challenges from law enforcement perspective (PDF), in 6th International Conference on E-Governance, 2008
- [13] Peterson, Gilbert e Shenoi, Sujeet, Digital Forensic Research: The Good, the Bad and the Unaddressed, in Advances in Digital Forensics V, IFIP Advances in Information and Communication Technology, 2009, pp. 17–36
- [14] Simson L. Garfinkel, Digital forensics research: The next 10 years, in Digital Investigation, 2010, p. S64–S73
- [15] C. Maioli, Introduzione all'informatica forense, in La sicurezza preventiva della comunicazione
- [16] F. Insa, The Admissibility of Electronic Evidence in Court (A.E.E.C.): Fighting against High-Tech Crime-Results of a European Study, 2006, p. 285.

- [17] P. Tonini, Nuovi profili processuali del documento informatico, in Scienza e processo penale: linee guida per l'acquisizione della prova scientifica, 2010, p. 427
- [18] SWGDE, Digital Evidence: Standards and Principles, 1999
- [19] S. Mason, Electronic Evidence. Discovery & Admissibility, 2007, par. 2.03
- [20] Palmer G Digital Forensic Science in Networked Environments (Network Forensics), 2001, pp 27–30
- [21] Broucek V, Turner P, Forensic computing: developing a conceptual approach for an emerging academic discipline, 2001, pp 55–68
- [22] Berghel H, The discipline of internet forensics, 2003
- [23] Ren W, Jin H, Modeling the network forensics behaviors, 2005, pp 1–8
- [24] Garfinkel S, Network forensics: tapping the internet, 2002
- [25] Forte D, The future of computer and network forensics, 2002
- [26] Raynal F, Berthier Y, Biondi P, Kaminsky D, Honeytrap forensics, Part 1: analyzing the network. IEEE Secur Priv 2(4):72–78
- [27] Raynal F, Berthier Y, Biondi P, Kaminsky D, Honeytrap forensics, Part II: analyzing the compromised host, 2004
- [28] Nikkel BJ, An introduction to investigating IPv6 networks, 2007
- [29] Govil J, Govil J, Kaur N, Kaur H, An examination of IPv4 and IPv6 networks: constraints and various transition mechanisms, 2008, pp 178–185
- [30] Vural I, Venter H, Mobile botnet detection using network forensics, 2010, pp 57–67
- [31] Vural I, Venter HS, Using network forensics and artificial intelligence techniques to detect bot-nets on an organizational network, 2010, pp 725–731
- [32] Guo R, Cao T, Luo X, Application layer information forensics based on packet analysis, 2010, pp 206–209
- [33] Nikkel BJ, Domain name forensics: a systematic approach to investigating an internet presence, 2004
- [34] Naqvi S, Massonet P, Arenas A, Scope of forensics in grid computing – vision and perspective, 2006, pp 964–970
- [35] Lillard TV, Garrison CP, Schiller CA, Steele J, What is network forensics?, 2010, pp 3–20

[36] Saad S, Traore I, Method ontology for intelligent network forensics analysis, 2010, pp 7–14

Capitolo 3

- [1] Mannan, M. & van Oorschot, P. C. , “A Protocol for Secure Public Instant Messaging,” 2006, 20-35.
- [2] Mannan, M., “Secure Public Instant Messaging,” master's thesis, School of Computer Science, 2005
- [3] Huang, A. H. , “Theoretical Foundations and a Framework for Instant Messaging Research, 2007
- [4] Messenger Wars: How Facebook lost its lead by OnDevice Research, a mobile market research company
- [5] Church; de Oliveira, Karen, "What's up with WhatsApp? Comparing Mobile Instant Messaging Behaviors with Traditional SMS" (PDF), 2013
- [6] Ling, Rich; Lai, Chih-Hui, "Microcoordination 2.0: Social Coordination in the Age of Smartphones and Messaging Apps", 2016
- [7] Rouse, M., Learn IT: Instant Messaging in the Workplace, 2003
- [8] Jeff, T., & Cooper, A., HowStuffWorks, 2001
- [9] Jennings, R. B., Nahum, E. M., Olshefski, D. P., Saha, D., Shae, Z. & Waters, C., “A Study of Internet Instant Messaging and Chat Protocols, 1996
- [10] <https://blog.mindorks.com/how-voice-and-video-call-works-b0896aa0a630>
- [11] Janczukowicz, E. C., QoS management for WebRTC: loose coupling strategies (Doctoral dissertation, Ecole nationale supérieure Mines-Télécom Atlantique), 2017
- [12] Day, M., Aggarwal, S., Mohr, G. & Vincent, J., “Instant Messaging / Presence Protocol Requirements,” RFC 2779, 2000
- [13] Peterson, J., “Common Profile for Instant Messaging (CPIM),” RFC 3860, 2004
- [14] Saint-Andre, P., ”Extensible Messaging and Presence Protocol (XMPP): Core,” RFC 3920, 2004
- [15] Jabber Inc. - About Us Archiviato il 3 luglio 2007 in Internet Archive
- [16] Jabber Instant Messaging User Base Surpasses ICQ, xmpp.org, 2003

- [17] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. & Schooler, E., "SIP: Session Initiation Protocol," RFC 3261, 2002
- [18] Campbell, B; Rosenberg, J; Schulzrinne, H; Huitema, C; Gurle, D., "Session Initiation Protocol (SIP) Extension for Instant Messaging," RFC 3428, 2002
- [19] Unger, Nik; Dechand, Sergej; Bonneau, Joseph; Fahl, Sascha; Perl, Henning; Goldberg, Ian Avrum; Smith, Matthew, SoK: Secure Messaging (PDF), 2015, pp. 232–249
- [20] Shirey, R., "Internet Security Glossary," RFC 2828, 2000
- [21] Pashalidis, A. & Mitchell, C. J., "A Taxonomy of Single Sign-on Systems," 2003, 249-264.
- [22] Dierks, T. & Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246, 2008.
- [23] John, R., Practical Internet Security, 2007
- [24] Salin, P., "Mobile Instant Messaging Systems - A Comparative Study and Implementation," master's thesis, 2004
- [Figura 3.3, 3.4, 3.5] <https://support.medialooks.com/hc/en-us/articles/360000213312-%D0%95nvironment-signaling-STUN-and-TURN-servers>
- [Figura 3.8] G. De Luca Fiscone, "Network Forensics di WhatsApp e Instagram: Profiling delle principali attività dei Cybercriminali", 2019

Capitolo 4

- [1] D. Floreano, Manuale sulle reti neurali, 1996
- [2] Russell, Stuart; Norvig, Peter, Artificial Intelligence: A Modern Approach (2nd ed.), 2005
- [3] Brownlee, Jason, "What is the Difference Between Test and Validation Datasets?", 2017.
- [4] Neural Smithing, Supervised Learning in Feedforward Artificial Neural Networks, 1999
- [5] Miljanovic, Milos, Comparative analysis of Recurrent and Finite Impulse Response Neural Networks in Time Series Prediction, 2012

[Figura 4.1]

https://upload.wikimedia.org/wikipedia/commons/thumb/a/a8/Neuron_-_annotated-it.svg/440px-Neuron_-_annotated-it.svg.png

[Figura 4.4] <https://mathworld.wolfram.com/SigmoidFunction.html>

[Figura 4.5] <https://mathworld.wolfram.com/HyperbolicTangent.html>

[Figura 4.6] http://jdcruzan.tripod.com/DCWeb_LinearFunctions.html

[Figura 4.7, 4.8]

https://www.researchgate.net/profile/Murat_Sazli/publication/228394623/figure/fig2/AS:302007991717904@1449015724170/A-multi-layer-feed-forward-neural-network.png

[Figura 4.9] <https://www.okpedia.it/data/okpedia/rete-neurale-ricorrente.gif>

Capitolo 5

[1] Casey, E., Digital Evidence and Computer Crime, 2011, pp. 727-735

[2] EC-Council, Computer Forensics: Investigating Network Intrusions and Cyber Crime, 2010, pp.11-4.

[3] Kizza, J. M., A Guide to Computer Network Security (3rd Edition), 2009, pp. 1-60.

[4] Sanders, C., Practical Packet Analysis_ Using Wireshark to Solve Real-World Network Problems, 2011, pp. 2

[5] G. De Luca Fiscone, “Network Forensics di WhatsApp e Instagram: Profiling delle principali attività dei Cybercriminali”, 2019

[6] <https://techcrunch.com/2018/01/31/whatsapp-hits-1-5-billion-monthlyusers-19b-not-so-bad/?ncid=rss>

[7] <https://www.socialmediaeasy.it/statistiche-instagram-marketing>

[Figura 5.2-23] G. De Luca Fiscone, “Network Forensics di WhatsApp e Instagram: Profiling delle principali attività dei Cybercriminali”, 2019

Capitolo 6

[1] Lescisin, Michael, Mahmoud, Qusay, Tools for Active and Passive Network Side-Channel Detection for Web Applications, 2018

[2] Kadloor, S, Gong, X, Kiyavash, N, Tezcan, T, Borisov, Nikita, LowCost Side Channel Remote Traffic Analysis Attack in Packet Networks, 2010

[3] Rosner, N., Kadron, I. B., Bang, L., Bultan, T. , Profit: Detecting and Quantifying Side Channels in Networked Applications, 2018

Capitolo 7

- [1] <http://lear.inrialpes.fr/~jegou/data.php#holidays>
- [2] <https://file-examples.com/>
- [3] <https://www.philadelphiafed.org/research-and-data/real-time-center/greenbook-data/pdf-data-set>
- [4] <http://marsyas.info/downloads/datasets.html>
- [5] <https://www.musicradar.com/news/tech/sampleradar-502-free-80s-samples-233696>
- [6] <https://www.appsloveworld.com/download-sample-mp4-video-mp4-test-videos/>