UNIVERSITÀ DI PISA

DIPARTIMENTO DI INFORMATICA

Laurea Triennale in Informatica

Generazione semi-automatica di regole per la classificazione del traffico di rete

Relatore:

Prof: Luca Deri

Candidato:

Nicolò Fontanarosa

ANNO ACCADEMICO 2024/2025

Abstract

Il monitoraggio del traffico di rete per l'identificazione delle applicazioni rappresenta oggi una sfida complessa. La crescente diffusione della cifratura, che interessa ormai la quasi totalità del traffico Internet, rende estremamente difficile analizzare in modo diretto i contenuti delle comunicazioni e distinguere tra i diversi servizi o applicazioni coinvolti, limitando l'efficacia degli strumenti di analisi tradizionali.

Le tecniche basate su DPI (*Deep Packet Inspection*), un tempo efficaci per il riconoscimento dei protocolli, risultano oggi inadeguate a causa dell'uso estensivo di protocolli cifrati come TLS e QUIC, che nascondono gran parte delle informazioni utili all'identificazione del traffico.

Questo lavoro si concentra quindi sull'identificazione di applicazioni a partire da sessioni TLS, QUIC e HTTPS, sfruttando caratteristiche residue nei flussi cifrati. L'obiettivo è fornire ad un operatore di rete uno strumento *plug and play* in grado di supportare il processo di riconoscimento delle applicazioni, generando automaticamente regole interpretabili che mirano a definire l'univocità di ciascuna applicazione.

Il sistema sviluppato consiste in un parser che, a partire da una cattura di traffico generata da un'applicazione specifica, filtra i flussi irrilevanti e quelli prodotti da servizi ausiliari esterni, raggruppando quelli significativi sulla base di metadati estratti in precedenza. Il risultato finale è un insieme di regole interpretabili che provano a identificare univocamente quell'applicazione e che possono essere utilizzate per definire un protocollo custom utile all'aggiornamento di librerie di *Deep Packet Inspection* open source come nDPI.

Per validare l'efficacia del metodo proposto, il sistema è stato testato su un dataset composto da 30 applicazioni, includendo applicazioni mobili, VPN e servizi web. I risultati sperimentali mostrano che il sistema, nel caso standard A, raggiunge un F1-score del 95.6%. Nel caso C, invece, la variazione di un parametro coinvolto nel processo di identificazione dell'applicazione altera precisione e sensibilità, confermando l'importanza di una corretta configurazione dei parametri per ottenere una copertura ottimale del traffico analizzato.

Indice

Capitolo 1

Introduzione

1.1 Contesto e problema

Il monitoraggio del traffico di rete per la rilevazione delle applicazioni e la gestione delle prestazioni, rappresenta oggi una sfida complessa e in continua evoluzione. La crescente diffusione della cifratura, che coinvolge ormai la quasi totalità del traffico Internet, rende estremamente difficile identificare in modo diretto applicazioni e servizi, limitando l'efficacia degli strumenti di analisi tradizionali. Di conseguenza, gli amministratori di rete faticano ad avere visibilità sull'attività dei sistemi, ostacolando l'individuazione di anomalie o minacce, da parte di operatori, ricercatori e strumenti di sicurezza.

Le tecniche basate su DPI (*Deep Packet Inspection*), un tempo efficaci per il riconoscimento dei protocolli, risultano oggi inadeguate a causa dell'uso estensivo di protocolli cifrati come TLS e QUIC, che nascondono gran parte delle informazioni utili all'identificazione del traffico. Tuttavia, anche nei flussi cifrati rimangono accessibili alcune caratteristiche residue come il campo SNI (*Server Name Indication*), le *fingerprint* TCP e le estensioni TLS, che possono essere sfruttate per distinguere tra applicazioni e servizi differenti.

Queste limitazioni hanno motivato la ricerca di nuove strategie di analisi, capaci di sfruttare le poche informazioni residue ancora osservabili nei flussi cifrati per dedurre la natura dell'applicazione, dando vita ad approcci basati su *Machine Learning* e *Deep Learning*. Tuttavia, questi metodi presentano limiti significativi: i modelli tendono a degradare rapidamente con l'evoluzione dei protocolli e delle applicazioni e dipendono fortemente da dataset etichettati che, pur essendo assunti come corretti, possono introdurre bias e imprecisioni. Inoltre, l'SNI stesso è suscettibile a tecniche di offuscamento e cifratura (tramite *Encrypted Client Hello*), riducendo ulteriormente l'affidabilità delle classificazioni.

1.2 Obiettivo della tesi

L'obiettivo di questo lavoro è sviluppare un sistema in grado di analizzare il traffico di rete cifrato, esaminando le sessioni TLS, QUIC e HTTPS, per identificare automa-

ticamente l'applicazione o il servizio responsabile della comunicazione. Il sistema è progettato per assistere l'operatore di rete nella fase di riconoscimento, fornendo un metodo *plug and play* che genera regole interpretabili utili a definire univocamente ciascuna applicazione ed utilizzate per definire un protocollo custom integrabile all'interno della libreria nDPI.

Il sistema quindi, mira a ridurre il tempo necessario all'analisi manuale del traffico grezzo. Diversamente da un approccio diretto con `nDPI`, che richiederebbe l'ispezione di tutti i flussi generati, il parser sviluppato elabora i log generati da `nDPI` e ne estrae informazioni rilevanti, aggregandole e reinterpretandole in modo da produrre regole candidate al riconoscimento dell'applicazione. Tali regole non sono pensate per identificare l'intero traffico di rete, ma per isolare e descrivere in maniera interpretabile i flussi più rappresentativi dell'applicazione analizzata.

1.3 Caso d'uso e vincoli della soluzione proposta

Il sistema proposto opera in un contesto controllato e si basa su alcune precise assunzioni che ne delimitano il campo di applicazione. In particolare, esso analizza il traffico di rete su IPv4 e IPv6, concentrandosi esclusivamente sui flussi TLS, QUIC e HTTPS che presentano almeno uno tra i seguenti elementi identificativi: il campo SNI (*Server Name Indication*), la fingerprint JA4 o le informazioni contenute nel certificato del server (in particolare, il campo CN del subject). Tali caratteristiche consentono, anche in presenza di cifratura end-to-end, di capire a quale applicazione appartiene il traffico e di definire un protocollo custom su `nDPI` per il suo riconoscimento.

In scenari reali, un'applicazione può generare decine o centinaia di flussi, molti dei quali relativi a librerie comuni, CDN (*Content Delivery Network*) o servizi di terze parti condivisi da più applicazioni, che non contribuiscono all'identificazione univoca dell'app stessa. Il *parser*, pertanto, si focalizza sull'estrazione e la selezione dei flussi significativi, fornendo una stima delle regole che possono identificare l'applicazione in modo univoco. Ogni regola è accompagnata da una giustificazione basata sui metadati analizzati e sulle scelte effettuate durante l'elaborazione. Le regole ottenute possono essere:

- **Corrette**, quando l'identificazione è univoca e affidabile;
- **Parzialmente corrette**, quando la regola è valida ma comprende un eccesso o difetto di informazioni;
- **Non valide**, quando i flussi disponibili non contengono elementi sufficienti all'identificazione.

È importante sottolineare che la correttezza delle analisi dipende fortemente dalla qualità della cattura del traffico iniziale. L'utente deve assicurarsi di acquisire esclusivamente il traffico generato dall'applicazione di interesse, evitando il più possibile interferenze

o connessioni parallele. Nel presente studio, le catture sono state effettuate in contesti reali utilizzando PCAPdroid su dispositivi mobili e Wireshark su sistemi desktop, garantendo la riproducibilità e la coerenza dei dati raccolti.

1.4 Sfide e approccio proposto

Lo sviluppo del sistema ha richiesto di affrontare una serie di sfide dovute alla complessità e alla variabilità del traffico cifrato, nonché alla necessità di ottenere risultati robusti e riproducibili anche in presenza di rumore o comportamenti non deterministici. Le principali difficoltà individuate durante la progettazione possono essere suddivise in tre categorie:

- **Filtraggio del traffico**: isolare i flussi effettivamente pertinenti all'applicazione in analisi, rimuovendo connessioni di sistema, librerie comuni o flussi condivisi tra più applicazioni. In contesti reali, le catture di traffico contengono infatti un'elevata quantità di pacchetti estranei (richieste DNS, aggiornamenti di sistema o servizi in background), che rischiano di compromettere la qualità dell'analisi.
- **Raggruppamento dei flussi**: correlare le diverse connessioni generate da un singolo accesso a un servizio, al fine di ottenere una rappresentazione coerente e univoca del comportamento dell'applicazione. Poiché molte applicazioni stabiliscono più sessioni parallele verso domini differenti, è necessario un meccanismo in grado di riconoscere quali flussi appartengono realmente allo stesso servizio.
- **Generazione automatica di regole**: produrre in maniera automatica regole interpretabili e riutilizzabili, da integrare direttamente in `nDPI` come nuovi protocolli. Ciò richiede di convertire i risultati delle analisi in firme sintetiche ma rappresentative, capaci di discriminare in modo efficace le applicazioni.

L'obiettivo del lavoro è stato quindi quello di progettare un sistema in grado di affrontare tali problematiche combinando più livelli di analisi: dal filtraggio dei flussi non pertinenti, al raggruppamento logico delle connessioni correlate, fino alla generazione automatica di regole pronte per essere integrate in `nDPI`.

Nei paragrafi successivi, ciascuna di queste sfide verrà analizzata in modo dettagliato, illustrando sia le scelte architetturali adottate per affrontarle, sia le soluzioni implementative sviluppate per ottenere un sistema efficiente, scalabile e adattabile a scenari di traffico reale. Prima di procedere, verrà fornito un breve ripasso della teoria necessaria per comprendere al meglio il contenuto della tesi.

1.5 Evoluzione dell'analisi del traffico di rete

Il *World Wide Web* offre una vasta gamma di servizi, dalla ricerca di contenuti allo streaming video. L'identificazione dei servizi a partire dai flussi osservati in un elemento di rete (come un computer, uno smartphone o un router) è un compito cruciale per scopi quali la gestione della qualità del servizio (*Quality of Service, QoS*), l'analisi

delle prestazioni e la prevenzione di minacce alla sicurezza. Tuttavia, negli ultimi anni, tale attività è diventata sempre più complessa: in passato, quando gran parte delle comunicazioni avveniva in chiaro, era relativamente semplice per un analista o un sistema automatico determinare quale applicazione stesse generando un determinato flusso. Oggi, la diffusione massiccia di protocolli cifrati come TLS e QUIC ha reso tale operazione estremamente difficile. Questo perché la tendenza alla cifratura *end-to-end*, pur garantendo riservatezza e integrità, ostacola profondamente le tecniche tradizionali di classificazione. Le informazioni applicative, un tempo leggibili direttamente nel payload, risultano ora cifrate.

Storicamente, la classificazione del traffico era basata sulla coppia protocollo/porta (TCP 80 per HTTP, UDP 53 per DNS), ma l'adozione pervasiva della porta TCP 443 ha reso questo approccio obsoleto. Anche l'identificazione tramite indirizzo IP ha perso affidabilità a causa di fenomeni come:

- il dinamismo degli IP nelle infrastrutture cloud nelle quali gli indirizzi sono assegnati dinamicamente e possono cambiare frequentemente;
- la condivisione degli IP tra applicazioni differenti soprattuto quando vengono utilizzati CDN (*Content Delivery Network*) come Cloudflare o Akamai.

Di conseguenza, l'indirizzo IP non costituisce più una caratteristica discriminante affidabile. Le applicazioni mobili, in particolare, instaurano connessioni verso server multipli distribuiti su infrastrutture cloud, rendendo inefficaci euristiche basate su porte o indirizzi IP.

Oltre ai metodi tradizionali basati su indirizzo IP, porta o DPI, negli ultimi anni sono state sviluppate diverse tecniche per il riconoscimento dei servizi nel traffico cifrato, che sfruttano metadati e caratteristiche residue dei protocolli TLS e QUIC. Tra le principali, si possono individuare le seguenti:

- **TLS Fingerprinting**: Questa tecnica si basa sull'osservazione di pattern unici presenti nelle fasi di negoziazione TLS. Tali caratteristiche consentono di associare i flussi a comportamenti noti di specifiche applicazioni o librerie. Tuttavia, il *fingerprinting* TLS (esempi sono JA3, JA3S e la più recente JA4) presenta limiti significativi nell'identificare applicazioni nuove o aggiornate, poiché modifiche anche minime nella configurazione o nell'implementazione possono alterare i pattern precedentemente osservati e non identificare più univocamente un'applicazione.
- **SNI**: L'identificazione basata su SNI [14] utilizza il nome del server, incluso nel messaggio di handshake TLS (*ClientHello*), per determinare l'applicazione o il servizio coinvolto nella comunicazione. Nel caso di TLS 1.2 e delle versioni precedenti, il campo SNI è trasmesso in chiaro all'interno del messaggio *ClientHello*. Con l'introduzione di TLS 1.3, tuttavia, il campo SNI può essere cifrato: la recente estensione denominata ECH (*Encrypted Client Hello*), evoluzione del precedente ESNI (*Encrypted Server Name Indication*), mira infatti a proteggere la privacy dell'utente cifrando le informazioni sensibili contenute nella fase di handshake. Di

conseguenza, i metodi di identificazione basati su SNI rimangono applicabili solo in contesti in cui tale campo non risulti cifrato.

- **Machine Learning**: I classificatori basati su apprendimento automatico utilizzano modelli statistici addestrati su caratteristiche osservabili del traffico cifrato, come i tempi di handshake, la lunghezza dei pacchetti, la distribuzione temporale o i parametri crittografici negoziati, per identificare l'applicazione sottostante. Questi metodi offrono maggiore flessibilità e capacità di adattamento a nuovi schemi di traffico, ma richiedono grandi quantità di dati etichettati per l'addestramento e risorse computazionali significative per l'elaborazione e l'aggiornamento continuo dei modelli.

Alla luce dei limiti dei metodi tradizionali di classificazione, questa tesi propone una soluzione ibrida che combina il campo SNI, le fingerprint TLS (JA3S e JA4) e l'informazione CN del certificato TLS, con l'obiettivo di generare regole personalizzate per la libreria `nDPI`. Queste regole permettono di associare in modo coerente i flussi TLS, QUIC e HTTPS all'applicazione o al servizio che li ha generati, definendo così un protocollo custom interpretabile e integrabile all'interno di `nDPI`. In questo contesto, un protocollo non rappresenta più soltanto un'entità "tecnica" (come HTTP o DNS), ma un insieme di comportamenti osservabili a livello di rete che caratterizzano un'applicazione specifica. In questo modo, la soluzione proposta estende le capacità di riconoscimento di `nDPI` fornendo all'operatore di rete uno strumento in grado di automatizzare e semplificare il processo di identificazione.

Capitolo 2

Fondamenti Teorici

Cercherò in questa sezione di spiegare in maniera teorica gli argomenti utili all'apprendimento della mia tesi.

2.1 Crittografia TLS

Il **TLS** (*Transport Layer Security*) [11], [27] è un protocollo collocato al di sopra del livello di trasporto, progettato per garantire **confidenzialità**, **integrità** e **autenticazione** nelle comunicazioni dei protocolli applicativi. Nella maggior parte dei casi, TLS viene eseguito sopra il protocollo **TCP** (*Transmission Control Protocol*).

Il protocollo TLS mette a disposizione un'ampia gamma di valori configurabili. La loro combinazione costituisce una sorta di **impronta digitale** o **fingerprint** che permette di distinguere un particolare client o server, e che può essere sfruttata per il riconoscimento delle applicazioni. Questi possono essere TLS protocol versions, cipher suites, compression algorithms ...

Una volta instaurata la connessione TCP, il client e il server avviano la procedura di **handshake TLS**, durante la quale vengono negoziati i parametri crittografici necessari a stabilire un **canale sicuro**. Questa fase iniziale è fondamentale per il **TLS fingerprinting**, descritto più avanti, in quanto i messaggi *ClientHello* e *ServerHello* includono l'elenco delle **cipher suites**, le **estensioni** e ulteriori parametri (cfr. Figura 2.1).

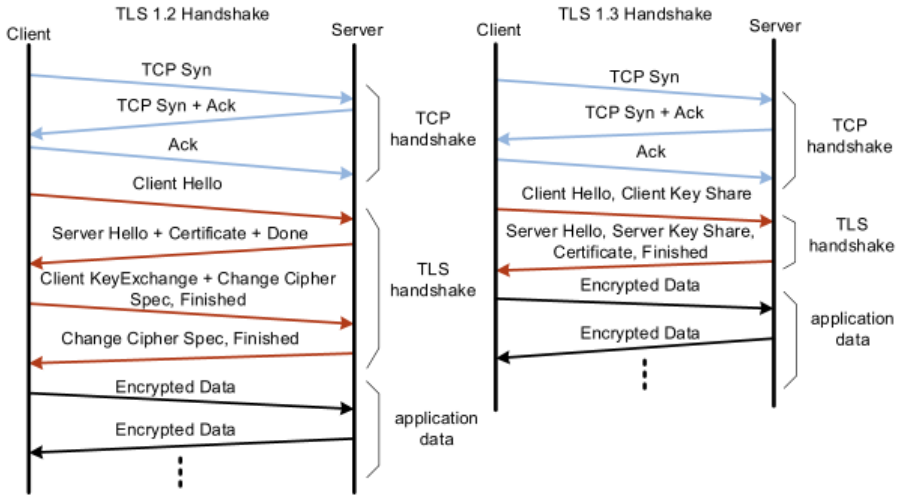

Figura 2.1: handshake TLS versione 1.2 e 1.3

Durante la fase di **handshake TLS**, il *client* e il *server* scambiano una sequenza di messaggi che permette di negoziare i parametri crittografici e stabilire un canale sicuro. In particolare:

- **ClientHello**: inviato dal client al server, contiene la lista delle versioni TLS supportate, le **cipher suites** disponibili, gli algoritmi di compressione utilizzabili e un numero casuale impiegato per la generazione delle chiavi.
- **ServerHello + Certificate + Done**: il messaggio *ServerHello* specifica la versione TLS scelta, la cipher suite e l'algoritmo di compressione da utilizzare nella sessione cifrata. Il messaggio *Certificate* trasmette il certificato del server (ed eventuali certificati intermedi), mentre *ServerHelloDone* indica la conclusione della fase di handshake dal lato server.
- **ClientKeyExchange + ChangeCipherSpec + Finished**: il messaggio *ClientKeyExchange* trasmette la chiave di sessione, cifrata con la chiave pubblica del server. Successivamente, tramite *ChangeCipherSpec* e *Finished*, il client segnala l'attivazione dei nuovi parametri crittografici e conferma l'avvenuta transizione.
- **ChangeCipherSpec + Finished**: analogamente, il server risponde con *ChangeCipherSpec* e *Finished*, confermando l'adozione della nuova specifica crittografica e la corretta conclusione dell'handshake.

Nei protocolli TLS fino alla versione 1.2, i messaggi di handshake non sono cifrati. Pertanto, le **caratteristiche distintive dei servizi** possono essere estratte analizzando direttamente il *payload* di tali messaggi mediante tecniche, descritte più avanti, di DPI (*Deep Packet Inspection*).

È importante sottolineare che i dati scambiati durante l'**handshake TLS non risultano cifrati**. Solo dopo il completamento di questa fase, tutto il traffico successivo tra client e server viene reso **confidenziale** tramite cifratura.

2.1.1 Funzionalità TLS

Per il **riconoscimento delle applicazioni cifrate** e la creazione di un'**impronta digitale TLS**, la letteratura individua tre principali categorie di **feature** utilizzabili nei modelli di classificazione:

1. **Attributi TLS** estratti durante la fase di *handshake*;
2. **Metadati del flusso TLS** come il numero di byte e pacchetti scambiati o la durata complessiva della connessione;
3. **Contenuto completo dei pacchetti**.

Nel mio lavoro, il **fingerprint TLS** è stato realizzato adottando il primo approccio, basato sull'analisi degli **attributi TLS** estratti dall'*handshake*.

2.2 JA Fingerprinting: JA3, JA3S, JA4 e JA4S

Il fingerprinting TLS è una tecnica che permette di identificare applicazioni, client o librerie TLS basandosi sui metadati dell'handshake anziché sul contenuto cifrato. Una delle metodologie più note è **JA3**, proposta da Salesforce e successivamente adottata dalla comunità open source, che costruisce un hash `MD5` derivato dalla concatenazione di specifici campi del messaggio `Client Hello`. L'obiettivo è generare un'impronta univoca per ogni configurazione TLS, indipendentemente dal contenuto cifrato del flusso. L'hash risultante rappresenta la firma TLS del client, utile per riconoscere applicazioni, librerie o malware. **JA3S** è l'equivalente per il `Server Hello`.

Tra gli attributi comunemente considerati per la costruzione dei fingerprint JA3, aggiornato poi in JA4, ampiamente impiegati in letteratura e utilizzati come riferimento nella mia tesi, ricordiamo i seguenti:

- **Versione**: indica la versione del protocollo TLS utilizzata durante l'*handshake*.
- **Cipher Suites**: rappresentano l'insieme delle possibili combinazioni di algoritmi per lo **scambio di chiavi**, l'**autenticazione**, la **cifratura** e l'**integrità dei dati**. L'attuale lista mantenuta da *IANA TLS Parameters* comprende oltre 350 combinazioni. La lista può includere valori casuali **GREASE** [7], inseriti per testare la compatibilità client/server; tali valori, poiché introducono instabilità, vengono esclusi dal calcolo del fingerprint.
- **Estensioni TLS**: definiscono funzionalità aggiuntive del protocollo. Ad oggi, sono state standardizzate circa 60 estensioni differenti.
- **Supported Groups (SG)**: estensione che specifica i gruppi crittografici supportati dal client per lo scambio di chiavi, ordinati dal più al meno preferito.
- **Elliptic Curve Point Format (ECFormat)**: descrive i formati di codifica accettati dal client per la trasmissione dei punti di curva ellittica.

- **Application Layer Protocol Negotiation (ALPN)**: utilizzata quando un server supporta più protocolli applicativi; client e server negoziano quale protocollo utilizzare per la connessione [28].
- **Supported Versions**: contiene l'elenco delle versioni TLS supportate dal client, ordinate in base alla preferenza [27].
- **Signature Algorithms**: elenca gli algoritmi di hash supportati per le firme digitali.

La Figura 2.2 illustra come differenti insiemi di attributi TLS, relativi sia al **client** (*JA3*, *JA4*) sia al **server** (*JA3S*, *JA4S*), vengano combinati per generare il **fingerprint TLS**. In particolare, viene evidenziato come vari attributi del protocollo contribuiscano alla costruzione di tale *impronta digitale*.

TLS Attribute	JA3	JA3S	JA4	JA4S
TLS/QUIC protocol			x	x
Handshake Version	x	x	x	x
Cipher Suites	x	x	x	x
Extensions	x	x	x	x
Supported Groups	x			
EC Format	x			
SNI				
ALPN			x	x
Supported Versions			x	x
Signature Algorithms			x	

Figura 2.2: attributi TLS usati nelle fingerprint JA3 e JA4

Tuttavia, con l'introduzione di *TLS 1.3* e l'uso sempre più diffuso di tecniche di randomizzazione e di evasione del fingerprint, *JA3* ha mostrato limiti nel rappresentare in modo coerente la varietà degli handshake moderni.

2.2.1 JA4 e JA4S

Per superare tali limiti, è stata introdotta una nuova generazione di fingerprint, **JA4** e **JA4S**. Queste adottano un approccio più robusto e, a differenza di JA3, tengono conto delle variazioni strutturali e semantiche dell'handshake, includendo:

- L'ordine e la presenza delle estensioni TLS;
- La tipologia del protocollo;
- Indicatori di negoziazione.

La struttura di una JA4 è illustrata in Figura 2.3.

Questo approccio rende JA4 più resistente ai tentativi di camuffamento da parte di malware o VPN, risultando più adatto per applicazioni di *threat intelligence*, *anomaly*

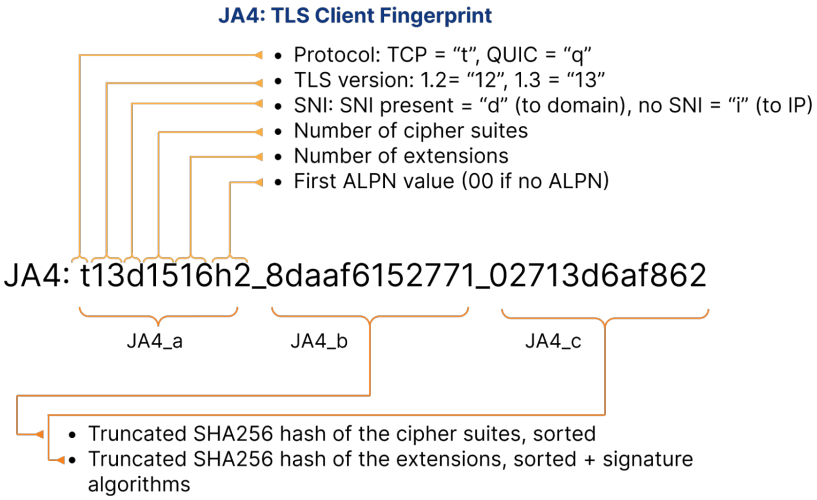

Figura 2.3: struttura del JA4

detection e *traffic classification*. JA4 e JA4S rappresentano quindi un'evoluzione significativa verso una forma più stabile e duratura di fingerprinting di rete, orientata non solo all'identificazione statica, ma anche all'analisi comportamentale dei client e dei server TLS. Come mostrato in Figura 2.3, la prima parte del JA4 contiene informazioni troppo generiche e condivise tra più fingerprint; per questo motivo, in questo lavoro sono state considerate solo la seconda e la terza parte del JA4. Il fingerprint JA4S, invece, non è stato utilizzato poiché è ancora in fase di studio; di conseguenza, in questa ricerca sono stati impiegati JA4 e JA3S, per motivi che verranno approfonditi nei capitoli successivi.

2.3 Certificati digitali

I certificati digitali rappresentano un elemento fondamentale nei protocolli di sicurezza di rete, in particolare all'interno del protocollo TLS, poiché garantiscono l'autenticità dell'identità di un server o di un client e abilitano la cifratura end-to-end. Un certificato `X.509` contiene una serie di attributi che ne determinano la validità e l'identità. Tra gli attributi principali troviamo:

- **Subject**: identifica l'entità cui il certificato appartiene;
- **Issuer**: l'autorità di certificazione (CA) che ha emesso il certificato;
- **Serial Number**: numero univoco che identifica il certificato all'interno della CA;
- **Validity**: periodo di validità del certificato;
- **Subject Public Key Info**: contiene l'algoritmo e la chiave pubblica associata al certificato;

- **Signature Algorithm**: indica l'algoritmo di firma utilizzato per validare il certificato.

I campi **Subject** e **Issuer** includono a loro volta diverse informazioni, tra cui:

- **C (Country Name)**: paese dell'entità (es. `IT`, `US`);
- **ST (State or Province Name)**: stato o provincia di registrazione;
- **L (Locality Name)**: città o località;
- **O (Organization Name)**: nome dell'organizzazione;
- **OU (Organizational Unit Name)**: reparto o unità organizzativa;
- **CN (Common Name)**: nome comune, spesso corrispondente al dominio (es. `www.example.com`).

Il lavoro proposto utilizza il campo `CN` del *Subject* come principale metadato per la generazione delle regole. Questo campo risulta molto simile all'*SNI* del TLS, in quanto entrambi forniscono un identificativo dell'entità o del dominio a cui la sessione è destinata, rendendoli strumenti particolarmente adatti per distinguere le applicazioni o i servizi durante l'analisi del traffico cifrato.

2.4 QUIC

QUIC è un protocollo di trasporto sviluppato inizialmente da Google e successivamente standardizzato. È progettato per combinare le funzionalità di trasporto di TCP e la cifratura di TLS in un unico livello, operando sopra UDP. L'obiettivo principale di QUIC è ridurre la latenza e migliorare l'efficienza delle connessioni, specialmente su reti mobili o con alta perdita di pacchetti, grazie all'integrazione nativa del *0-RTT handshake* e alla gestione indipendente dei flussi multipli all'interno di una stessa connessione.

Tuttavia, l'integrazione stretta tra trasporto e sicurezza comporta una minore visibilità del traffico per i sistemi di analisi di rete. La maggior parte delle informazioni utili alla classificazione, come i dettagli del protocollo applicativo o gli identificatori dei flussi, risulta infatti cifrata all'interno del pacchetto QUIC. Nonostante ciò, alcune parti dell'header rimangono leggibili e possono essere sfruttate per l'identificazione:

- **Version field**: specifica la versione del protocollo QUIC utilizzata e può essere utile per distinguere tra implementazioni differenti;
- **Server Name Indication**: presente nel Client Hello, trasmesso in chiaro prima dell'attivazione della cifratura completa;
- **Fingerprint TLS**: calcolata sui campi del Client Hello e Server Hello, consente di caratterizzare l'implementazione del client.

Queste informazioni, sebbene limitate, permettono di ottenere una firma parziale del flusso e di distinguere tra diverse applicazioni o servizi che adottano QUIC. La sfida principale rimane la progressiva adozione dell'estensione ECH (*Encrypted Client Hello*), che in futuro renderà completamente cifrato anche l'SNI, riducendo ulteriormente la visibilità esterna del protocollo.

2.5 HTTPS

HTTPS (*HyperText Transfer Protocol Secure*) è l'evoluzione sicura di HTTP e rappresenta oggi lo standard dominante per le comunicazioni web. Basato su TLS, garantisce confidenzialità, integrità e autenticazione dei dati scambiati tra client e server. Basandosi su TLS, nonostante la cifratura end-to-end, alcune informazioni del handshake TLS restano accessibili in chiaro e possono essere utilizzate per scopi di classificazione o identificazione.

2.6 DPI

La **DPI** (*Deep Packet Inspection*) è una tecnica che permette di effettuare operazioni sui pacchetti ed è eseguita a livello di un elemento di rete (un computer, uno smartphone o un router). Il suo principio si basa sull'analisi del contenuto dei pacchetti catturati, con l'obiettivo di discriminare in modo **accurato** e **tempestivo** il tipo di traffico.

La DPI permette di esaminare il *payload* dei pacchetti per rilevare fenomeni indesiderati come **spam**, **software malevolo** o altri tipi di attacchi. Un elemento di rete che esegue DPI raccoglie inoltre **informazioni statistiche** e decide se un pacchetto possa essere inoltrato o meno, sulla base dei risultati dell'analisi.

Le informazioni vengono ricercate nel payload dei pacchetti piuttosto che nelle intestazioni. Le firme vengono spesso ricercate per rilevare i tipi di flusso e agire di conseguenza. Questo può portare a svantaggi principalmente legati al rallentamento del traffico analizzato, dovuto all'aumento della potenza di calcolo e di RAM necessaria per trattare grandi quantità di pacchetti.

2.7 VPN

Le VPN (*Virtual Private Network*) [15] [21] rappresentano una delle tecnologie più diffuse per garantire privacy, anonimato e sicurezza durante la comunicazione in rete. Esse creano un canale cifrato tra il dispositivo dell'utente e un server remoto, instradando tutto il traffico attraverso tale tunnel. In questo modo, l'indirizzo IP reale dell'utente viene mascherato, e il contenuto dei pacchetti diventa illeggibile per soggetti esterni. Le VPN possono basarsi su diversi protocolli di trasporto e cifratura, tra cui OpenVPN, WireGuard, IKEv2 o protocolli proprietari. Tuttavia, proprio questa capacità di offuscamento rappresenta una sfida per i sistemi di monitoraggio del traffico, che

devono distinguere tra l'uso legittimo della VPN e potenziali abusi legati ad attività di evasione dei controlli o esfiltrazione di dati.

2.8 CDN

Le CDN (*Content Delivery Network*) sono infrastrutture distribuite progettate per ottimizzare la distribuzione di contenuti digitali su Internet, riducendo la latenza e migliorando l'esperienza utente. Una CDN è costituita da una rete di server dislocati geograficamente che replicano e servono contenuti statici o dinamici, come immagini, video, file di aggiornamento o pagine web, in base alla prossimità geografica dell'utente. Tra le più note si trovano *Cloudflare*, *Akamai*, *Fastly* e *Amazon CloudFront*.

Dal punto di vista dell'analisi del traffico, l'uso delle CDN introduce una complessità significativa. I flussi di rete non sono più diretti al server originario dell'applicazione, ma verso nodi di caching (o proxy) distribuiti appartenenti alla rete della CDN. Questo comportamento rende difficile associare un flusso a un'applicazione o servizio specifico, poiché indirizzi IP e certificati TLS possono essere condivisi tra molteplici domini e applicazioni. Inoltre, le CDN spesso implementano tecniche di cifratura avanzate e aggiornamenti frequenti delle chiavi o delle configurazioni TLS, rendendo meno efficaci gli approcci di riconoscimento.

Per un sistema di classificazione del traffico, come quello proposto in questo lavoro, le CDN rappresentano quindi un ostacolo naturale: la loro natura distribuita e la condivisione delle risorse tra diversi client, possono ridurre la capacità di identificare univocamente l'origine del traffico. Tuttavia, l'analisi combinata di più indicatori, come SNI, fingerprint TLS e metadati associati al certificato, può fornire informazioni sufficienti per discriminare in modo affidabile il comportamento di determinate applicazioni anche in presenza di CDN.

2.9 Strumenti di analisi del traffico usati

L'analisi del traffico di rete rappresenta un'attività fondamentale per la comprensione del comportamento delle applicazioni, l'identificazione di anomalie e la classificazione dei protocolli. Negli ultimi anni, l'aumento del traffico cifrato, in particolare con la diffusione di TLS e QUIC, ha reso necessario lo sviluppo di strumenti in grado di estrarre metadati significativi anche quando il contenuto dei pacchetti non è più direttamente ispezionabile. In questo contesto, esistono diversi strumenti, ciascuno con caratteristiche e finalità differenti, che vengono comunemente utilizzati nell'ambito della ricerca e dell'amministrazione di rete:

- **Wireshark**: Wireshark[1] è lo strumento di analisi dei pacchetti più diffuso e rappresenta lo standard per l'ispezione del traffico di rete. Wireshark consente

[1] https://www.wireshark.org

di catturare, filtrare e visualizzare in tempo reale i pacchetti trasmessi su una rete, fornendo informazioni dettagliate per ogni pacchetto. Supporta l'analisi dei protocolli di handshake TLS e permette di visualizzare parametri come versioni, suite crittografiche negoziate, estensioni TLS, certificati e campi SNI. Wireshark è quindi uno strumento ideale per l'analisi *fine-grained* del traffico, anche in ambito sperimentale o accademico.

- **nDPI**: nDPI [10] è una libreria open source di *Deep Packet Inspection* sviluppata da NTOP, concepita per l'identificazione automatica dei protocolli e delle applicazioni. A differenza di strumenti *general-purpose* come Wireshark, nDPI è pensata per l'integrazione diretta all'interno di applicazioni di monitoraggio, firewall e sistemi di *network management*. È in grado di riconoscere oltre 400 protocolli attraverso tecniche di DPI. La libreria supporta inoltre regole personalizzate basate su indirizzi IP, porte, domini, e parametri TLS, consentendo di estendere facilmente la capacità di riconoscimento. nDPI rappresenta quindi una base ideale per sviluppare sistemi di classificazione automatica e per la generazione dinamica di regole, come quello proposto nel presente lavoro di tesi.

- **PCAPdroid**: PCAPdroid[2] è un'applicazione Android open source che consente di catturare il traffico generato dalle applicazioni mobili direttamente dal dispositivo, senza necessità di accesso root. I pacchetti vengono esportati in formato PCAP (leggibili poi con nDPI o Wireshark) o analizzati in tempo reale tramite un server remoto, rendendo lo strumento particolarmente utile per studi di comportamento delle app mobili. PCAPdroid rappresenta una risorsa preziosa per estendere l'analisi del traffico cifrato anche in contesti mobili, spesso difficili da monitorare con strumenti tradizionali.

2.10 nDPI

nDPI è una libreria open-source dedicata al *Deep Packet Inspection*, progettata per fornire un'analisi dettagliata e avanzata del traffico di rete. Tra le sue funzionalità principali vi sono il riconoscimento del protocollo applicativo di ciascun flusso, la classificazione dei flussi cifrati e l'estrazione di metadati e metriche utili per sistemi di monitoraggio e raccolta dati.

Sul mercato dei DPI esistono numerosi strumenti, sia open-source che commerciali. Alcuni prodotti proprietari, come **SolarWinds**, offrono capacità di analisi del traffico simili a quelle di nDPI, ma includono anche funzionalità aggiuntive per la gestione delle infrastrutture IT, consentendo visibilità su server, dispositivi di rete e servizi cloud. Altri strumenti, come **Netify**, si concentrano principalmente sulla raccolta di statistiche di utilizzo e sul profiling del comportamento applicativo, pur supportando un numero ridotto di protocolli rispetto a nDPI.

[2] https://github.com/emanuele-f/PCAPdroid

La nascita di nDPI risponde alla necessità di ottenere visibilità in contesti in cui la maggior parte del traffico Internet è cifrata, rendendo inefficaci le soluzioni basate esclusivamente su firme o regole statiche. Tali approcci tradizionali mostrano limiti evidenti, specialmente quando la crittografia *end-to-end* viene sfruttata per nascondere il contenuto delle comunicazioni.

In conclusione, nDPI si distingue come soluzione open-source flessibile, performante e scalabile per l'analisi dei flussi di rete. La sua architettura modulare e la capacità di riconoscere centinaia di protocolli lo rendono uno strumento consolidato in ambito di ricerca, sicurezza informatica e monitoraggio di rete, offrendo una copertura più ampia rispetto a molti DPI commerciali.

2.10.1 Protocolli Custom in nDPI

Nel contesto di `nDPI`, il termine *protocollo* assume un significato più ampio rispetto alla definizione tecnica tradizionale:

- **Struttura Gerarchica:** Ogni protocollo è rappresentato tramite una struttura gerarchica `<maggiore>.<minore>`, ad esempio `DNS.Facebook`, `QUIC.YouTube` o `QUIC.YouTubeUpload`.
- **Protocolli Applicativi:** Applicazioni come Facebook o Skype vengono trattate da nDPI come protocolli applicativi, pur non essendo definite tali nelle specifiche **IETF** (*Internet Engineering Task Force*).
- **Supporto Esteso:** nDPI riconosce oltre **400 protocolli**, tra cui:
  - P2P (es. BitTorrent)
  - Messaging (es. WhatsApp, Telegram, Facebook, Viber)
  - Conferencing (es. Skype, Webex, Teams, Meet, Zoom)
  - Streaming (es. Netflix, Disney+, Zattoo)
  - Business e Gaming (es. VNC, RDP, Citrix)

Rilevamento basato su stringhe

Con la diffusione della cifratura end-to-end, l'identificazione corretta dei protocolli è diventata più complessa. nDPI integra quindi un sistema in grado di estrarre informazioni visibili, come il nome dell'host presente nel campo SNI, utili sia per l'analisi del traffico cifrato sia per la definizione di protocolli *custom*. Alcune stringhe che nDPI estrae e che possono essere utilizzate per la creazione di un protocollo sono:

- Campo SNI nei protocolli TLS / QUIC;
- Fingerprint TLS JA4;
- Informazioni dal certificato TLS, come il campo CN.

Ad esempio, il traffico verso domini come `netflix.com` o `nflxext.com` viene classificato nella categoria `NDPI_PROTOCOL_CATEGORY_STREAMING`, permettendo di associare correttamente il flusso all'applicazione Netflix.

Capitolo 3

Stato dell'arte

La classificazione del traffico cifrato e l'identificazione delle applicazioni di rete rappresentano un campo di ricerca attivo sin dalla diffusione dei protocolli di comunicazione sicura. In questo capitolo viene presentata una panoramica dei principali studi relativi ai metodi di identificazione delle applicazioni, con particolare attenzione al *TLS fingerprinting* e con cenni sull'apprendimento automatico e l'impiego di reti neurali, che negli ultimi anni hanno ricevuto un crescente interesse da parte della comunità scientifica.

In numerosi casi, i tipi di servizio e i numeri di porta risultano fortemente correlati [5]. L'identificazione di un servizio basata sull'indirizzo IP o sulla porta rappresenta il metodo più elementare, ma anche il meno affidabile. Tale approccio mostra infatti scarsa accuratezza e risulta limitato in presenza di tecniche come il cambio dinamico degli indirizzi IP [23].

In diversi scenari, uno stesso indirizzo IP può ospitare servizi eterogenei o cambiare dinamicamente. Inoltre, la maggior parte dei servizi web utilizza le porte 80 e 443, rendendo inefficace la sola analisi delle porte per distinguere tra protocolli o applicazioni. Sebbene l'indirizzo IP e il numero di porta possano fornire indicazioni sul protocollo di trasporto o sull'host di destinazione, essi non sono sufficienti per identificare un servizio specifico. Per questo motivo, nel sistema sviluppato in questa tesi non vengono considerati né indirizzi IP né numeri di porta per la definizione del protocollo anche se nDPI mette a disposizione questa funzionalità.

Velan et al. [32] hanno condotto un'analisi approfondita di lavori passati relativi all'estrazione di informazioni dal traffico cifrato. È emerso che, sebbene i protocolli di rete garantiscano la riservatezza dei dati, le fasi iniziali non cifrate delle comunicazioni possono comunque fornire preziose informazioni utili al monitoraggio e alla classificazione del traffico.

Gigi et al. [16] hanno analizzato i certificati TLS nel traffico Internet, evidenziando l'ampia distribuzione geografica dei server dei principali content provider. Sebbene il loro lavoro non affronti direttamente l'identificazione dei servizi, esso mostra come tali provider abbiano contribuito alla centralizzazione e privatizzazione dell'infrastrut-

tura di rete globale. Questo aspetto è rilevante per il presente studio, poiché motiva l'esclusione dei grandi *CDN* o dei server condivisi che, pur essendo utilizzati da un'applicazione, non permettono di identificarla in modo univoco.

Shbair et al. [29] hanno analizzato differenti approcci per l'identificazione di servizi in traffico HTTPS, basandosi sulla struttura del protocollo e sull'uso del campo SNI. Tuttavia, i dataset impiegati nel loro lavoro risultano datati, con l'ultimo risalente al 2015, riducendo la validità dei risultati rispetto allo scenario attuale.

Bortolameotti et al. [8] hanno proposto una tecnica per rilevare connessioni TLS malevole sfruttando SNI e certificati SSL. Gli autori calcolano la distanza di Levenshtein tra i valori SNI e i domini dei 100 siti più popolari, analizzando la struttura del nome del server. Tuttavia, a distanza di dieci anni, questo approccio risulta parzialmente efficace: oggi molti client TLS omettono il campo SNI o utilizzano valori casuali o non coerenti con il servizio reale. In questo senso, il presente lavoro può essere considerato un aggiornamento di tali studi, verificando l'effettivo utilizzo e affidabilità del campo SNI nell'era del TLS 1.3.

Anderson et al. [3] hanno esplorato l'uso delle impronte TLS per il rilevamento di malware nel traffico cifrato, estraendo *cipher suites*, estensioni TLS e lunghezze delle chiavi pubbliche dai messaggi *Client Hello* e *Server Hello*. Considerando anche statistiche di flusso (lunghezze dei pacchetti, tempi di inter-arrivo, distribuzione dei byte), hanno raggiunto un'accuratezza del 99.6% nella classificazione del malware. Sebbene l'approccio si basi su dataset privati, limitandone la riproducibilità, evidenzia l'importanza di analizzare, oltre al campo SNI, anche le estensioni TLS. Nel sistema proposto, queste informazioni vengono infatti integrate nella generazione delle regole di identificazione.

Shbair et al. [30] si sono inoltre concentrati sul filtraggio del traffico HTTPS basato su SNI, valutando l'affidabilità del campo per l'identificazione e il controllo del traffico. Pur dimostrando la possibilità di aggirare un firewall sfruttando variazioni nel campo SNI, il loro studio considera solo singole connessioni, senza analizzare casi in cui un'applicazione stabilisce sessioni multiple con SNI differenti, un comportamento comune nelle applicazioni mobili.

Anderson e McGrew [2] hanno affrontato il problema della sovrapposizione tra fingerprint, proponendo di includere, oltre agli attributi TLS, anche indirizzo di destinazione, porta e SNI, per ottenere impronte più contestualizzate. Questo approccio richiama le tecniche JA3S e JA4S. Analogamente, nel mio lavoro vengono impiegati attributi del server e valori SNI per affrontare l'identificazione. Gli autori valutano la similarità tra fingerprint tramite distanza di Levenshtein, pesando gli attributi in base al *information gain ratio*. Mentre il loro obiettivo è la classificazione di famiglie di applicazioni, il presente studio si focalizza, ove possibile, sull'individuazione di singole applicazioni.

Zhang e Chi-Jiun [35] propongono un nuovo framework basato su regole per analizzare il traffico QUIC crittografato utilizzando esclusivamente le informazioni non crittografate disponibili a livello di rete e trasporto, in particolare la dimensione , la tempistica e la

direzione dei pacchetti QUIC. Questo metodo presenta ottimi risultati, basandosi però sull'ambiente di rete e il tempo di risposta di ogni singolo pacchetto.

Kim et al. [19] hanno introdotto un metodo per classificare il traffico in base ai servizi applicativi, sfruttando le informazioni di pubblicazione del certificato TLS, non cifrate. Sebbene simile a studi precedenti, il loro approccio si distingue per l'uso dell'ID di sessione. Tuttavia, non considera l'analisi di connessioni multiple o l'impiego congiunto del campo SNI.

Ci sono numerosi articoli in letteratura (es. [20], [25], [36]) che trattano l'identificazione delle VPN utilizzando statistiche relative alle dimensioni dei pacchetti, ai tempi di arrivo, e ad altre caratteristiche del traffico. Questi studi impiegano sia tecniche di *machine learning*, sia modelli a stati finiti. Altri lavori invece (es. [18]) si concentrano sull'analisi di caratteristiche legate al *traffic tunneling*; ad esempio, la presenza di un pacchetto ACK generato entro 500 ms può rappresentare una caratteristica distintiva del protocollo TCP.

Il limite comune a questi studi è che essi si focalizzano sull'identificazione dell'uso di una VPN o delle applicazioni cifrate al suo interno, ma non sul riconoscimento del tipo specifico di VPN utilizzata. Oggi esistono infatti numerose VPN, o meglio, applicazioni diverse che offrono servizi di VPN, le quali possono differire notevolmente tra loro.

Questa tesi non ha come obiettivo principale lo studio o l'identificazione delle VPN, ma come effetto collaterale, e mira ad individuare alcune caratteristiche che le contraddistinguono. Come accennato nella parte introduttiva, il traffico odierno è prevalentemente cifrato e, per stabilire tale cifratura, è necessario un *handshake* iniziale. In ogni applicazione che offre un servizio VPN, questo *handshake* avviene e presenta gli stessi metadati tipici di tutte le altre applicazioni.

Abbiamo già osservato che il campo SNI può non essere presente o può essere camuffato; tuttavia, questa tesi si basa proprio sull'assunzione che tale campo sia disponibile. Pertanto, per le VPN in cui l'SNI è presente e non è generato dinamicamente, è possibile identificare con certezza l'applicazione VPN allo stesso modo delle altre applicazioni analizzate.

Un grosso contributo va a Burgetová, Ryšavy e Matoušek [9] che hanno sottolineato l'efficacia combinata del campo SNI e delle impronte JA nella corretta identificazione delle applicazioni, mostrando come questa unione garantisca il più alto tasso di riconoscimento univoco (cfr. Figura 3.1)

L'articolo ha analizzato l'**unicità** e la **rilevanza** dei principali attributi TLS, evidenziando come alcuni di essi contribuiscano in misura maggiore alla capacità di distinguere le applicazioni. In particolare, lo studio mostra che l'attributo che fornisce il contributo più significativo all'unicità del **fingerprint TLS**, e quindi al riconoscimento delle applicazioni, è l'**SNI**.

Fingerprint type	Total	Uniqueness	Covered apps	Efficiency
JA3	8208	99.5%	67.5%	1.02
JA4	111	54.1%	41.6%	3.36
JA3S	77	44.2%	20.8%	4.53
JA4S	97	48.5%	27.3%	4.06
JA3+JA3S	8330	99.2%	80.5%	1.02
JA4+JA4S	264	66.7%	70.1%	2.16
JA3+JA4+JA3S+JA4S	8349	99.2%	84.4%	1.02
SNI	728	88.0%	89.6%	1.27

Figura 3.1: Fingerprint utilizzate per l'identificazione univoca di un'applicazione

Una seconda linea di ricerca si concentra invece sull'impiego del *machine learning*. Sebbene non sia il focus principale di questo lavoro, è utile menzionare alcuni contributi rilevanti.

Gli studi svolti da Draper-Gil at al. [13] e Miller et al. [22] analizzano l'efficacia delle **caratteristiche temporali di flusso** in due contesti principali:

- **Rilevamento del traffico VPN:** valutare l'efficacia di tali caratteristiche nel distinguere il traffico proveniente da reti private virtuali;
- **Classificazione del traffico cifrato:** identificare diverse categorie di traffico basandosi su attributi temporali;
- **Contributo metodologico:** proporre un metodo di classificazione basato esclusivamente su caratteristiche temporali per ridurre la complessità computazionale e rendere il classificatore indipendente dalla cifratura.

Nel contesto del mio lavoro, si dimostra come, con un approccio *plug and play*, senza addestramento né grandi quantità di dati, sia comunque possibile identificare in modo efficace molte VPN attraverso l'analisi congiunta di SNI, JA4 e certificato.

Nonostante i risultati incoraggianti infatti, tali studi presentano limiti legati alla generalizzabilità e alla natura dei dataset utilizzati, spesso ristretti ad un numero limitato di categorie o applicazioni. Inoltre, il traffico VPN è stato catturato utilizzando un singolo provider e il metodo si basa unicamente su caratteristiche temporali, riducendo la robustezza rispetto a classificatori ibridi.

Nakao et al. [24] hanno proposto un metodo di identificazione basato su *machine learning* che, tuttavia, richiede un lungo tempo di addestramento e un volume consistente di dati, rendendo difficile l'applicazione in scenari reali con traffico limitato.

Ulteriori lavori, tra cui Ding et al. [12], Barut et al. [6] e Yang et al. [34], hanno esplorato metodi basati su *machine learning* e *deep learning*, utilizzando metadati TLS, lunghezze e tempi di inter-arrivo dei pacchetti. Shbair [31], Zou [37] e Wang [33] et al. hanno esteso questo approccio introducendo ulteriori caratteristiche statistiche, confermando come l'apprendimento automatico possa offrire buone prestazioni, seppur a costo di un'elevata complessità e dipendenza dai dataset.

Akbari et al. [1] propongono un approccio più semplice, concentrato sui parametri estraibili dal *TLS handshake*, ritenuti le informazioni più rilevanti per l'identificazione del servizio. Gli autori evidenziano l'importanza del campo SNI, pur rimuovendolo in alcuni casi per evitare ambiguità dovute a nomi condivisi. A tali metadati affiancano statistiche ricavate da strumenti come *nDPI* o flowmeter, basate su deviazione standard, dimensione dei pacchetti, inter-arrival time e flag TCP. Il presente studio approfondisce proprio la questione di quanto l'SNI possa effettivamente essere considerato discriminante, mettendone in luce i limiti pratici.

In questo contesto, un metodo privo di addestramento, come quello proposto in questa tesi, si presenta come un complemento essenziale ai modelli basati su *machine learning*. Questi ultimi dipendono infatti da condizioni di rete variabili e da dataset etichettati, mentre il metodo proposto opera in modo indipendente dalla qualità della rete.

Inoltre, sebbene molti autori dichiarino elevate accuratezze, tali risultati spesso trascurano le peculiarità del traffico cifrato: i modelli trattano i dati come sequenze non interpretate e introducendo distorsioni legate agli algoritmi di cifratura o al tipo di applicazione. Ciò porta a dataset sbilanciati e difficili da gestire.

Infine, Burgetová, Ryšavy e Matoušek [9] confermano i limiti degli approcci basati su *Machine Learning* e *Deep Learning* affermando che tali modelli tendono a degradare rapidamente con l'evoluzione dei protocolli e delle applicazioni, e dipendono fortemente da dataset etichettati che possono introdurre *bias* e ridurre la precisione complessiva.

Alla luce di queste considerazioni e studi svolti, l'obiettivo di questa tesi può essere interpretato sia come una tecnica per il riconoscimento del traffico di rete basata sull'analisi delle *caratteristiche TLS*, sia come una metodologia atta a facilitare l'operatore di rete nel processo di etichettatura del traffico.

Secondo quanto riportato nello studio di Guerra, Catania e Veas [17], esistono diverse metodologie di etichettatura attualmente impiegate, che possono essere suddivise principalmente in due categorie:

1. **Etichettatura Automatica:** queste strategie sono ampiamente utilizzate per generare grandi volumi di traffico etichettato grazie alla loro rapidità e al ridotto livello di competenza richiesto. Queste tecniche però presentano una bassa rappresentatività del traffico reale, accuratezza moderata, difficoltà di riproduzione e scarsa attenzione ai dettagli basati sul'ambiente;

2. **Etichettatura Guidata dall'Uomo:** queste metodologie si basano sull'esperienza di analisti esperti operanti in ambienti di rete non controllati. Possono essere ulteriormente suddivise in approcci *manuali* e *assistiti*, ad esempio mediante tecniche di AL (*Active Learning*). Queste tecniche presentano un'elevata rappresentatività e alta accuratezza dei risultati ma anche lentezza del processo, necessità di competenze elevate (nel caso manuale) e difficoltà di riproduzione. I metodi assistiti migliorano la velocità, ma richiedono comunque l'intervento di esperti umani.

La difficoltà risiede nel fatto che molte fingerprint non sono univoche: lo stesso JA4 può essere condiviso da applicazioni differenti, così come uno stesso SNI può rappresentare più servizi ospitati dietro un CDN. Per questo, il processo di etichettatura richiede una combinazione di conoscenza esperta, dati esterni e tecniche di inferenza. Alla luce di queste considerazioni, emerge la necessità di un *trade-off* tra qualità e velocità del processo di etichettatura. Il sistema proposto in questa tesi mira proprio a colmare tale divario, fornendo una soluzione che combina l'efficienza dei metodi automatici con l'affidabilità delle strategie guidate dall'uomo.

Uno studio di riferimento su cui si basa questa tesi è quello svolto da Asaoka et al. [4], i quali propongono un metodo per identificare i servizi all'interno di flussi TLS 1.2 (e precedenti) sfruttando il campo SNI (*Server Name Indication*), non cifrato nella fase di *handshake*. Gli approcci, denominati *SNI Occurrence Method* e *First SNI Occurrence*, analizzano rispettivamente la frequenza dei valori SNI osservati in più sessioni TLS relative a un singolo accesso e la prima occorrenza di un SNI nelle varie sessioni TLS.

La presente tesi si ispira a tale lavoro, rielaborandone i principi in un contesto differente: non per individuare un singolo servizio, come definito dagli autori, ma per identificare l'applicazione corrispondente e il relativo protocollo, secondo la terminologia adottata in *nDPI*. I metodi presentati nello studio verranno quindi ripresi e adattati nelle sezioni successive, all'interno di un approccio più generale volto all'etichettatura automatica e semantica del traffico cifrato.

Il flusso metodologico seguito in questa tesi si ispira inoltre al lavoro presentato da Park et al. [26], i quali propongono un metodo di generazione automatica delle regole per il rilevamento delle *azioni utente* nel traffico di rete. Tale studio nasce dall'esigenza di ottimizzare il monitoraggio delle attività in ambienti cloud, dove è necessario tracciare le operazioni degli utenti per motivi di sicurezza e di gestione delle licenze.

Gli autori evidenziano come i metodi tradizionali, basati sulla definizione manuale delle regole dopo una pre-analisi dell'applicazione target, risultino estremamente dispendiosi in termini di tempo e risorse. Per ovviare a ciò, propongono un sistema automatico che, attraverso un processo di analisi delle sequenze di traffico, consente la generazione automatica delle regole di rilevamento.

Il processo proposto segue una pipeline chiara: dalla definizione e raccolta del traffico relativo alle azioni utente, alla sua pre-elaborazione e conversione in flussi a 5-tuple, fino alla generazione e verifica delle regole. In sintesi, lo studio [26] dimostra che la generazione automatica delle regole può ridurre significativamente il carico manuale senza compromettere l'accuratezza. Il presente lavoro adotta un flusso analogo, ma con un obiettivo differente: anziché rilevare azioni utente, mira alla generazione automatica di regole utili all'identificazione di protocolli e applicazioni all'interno di flussi TLS, QUIC e HTTPS cifrati, ma sempre mirando a ridurre significativamente il carico manuale dell'operatore di rete.

Capitolo 4

Architettura

In questo capitolo viene descritta la struttura generale del sistema e le principali fasi di sviluppo. L'obiettivo è quello di identificare in modo univoco un'applicazione a partire dal traffico cifrato che essa genera, al fine di definire automaticamente una regola che possa essere integrata in *nDPI* per il riconoscimento del protocollo applicativo corrispondente.

A tal fine, vengono analizzate e confrontate diverse tecniche presenti in letteratura per la classificazione del traffico TLS, evidenziandone punti di forza e limiti. L'approccio proposto in questa tesi rappresenta una rielaborazione dei metodi esistenti, con l'obiettivo di renderli applicabili in scenari reali e non controllati, dove il traffico può includere rumore, sessioni parallele e connessioni di background.

4.1 Raggruppamento delle sessioni TLS secondo Asaoka et al.

Lo studio [4] ha evidenziato che il campo `SNI`, contenuto nel messaggio `ClientHello`, consente di associare i flussi TLS a specifici servizi in modo da distinguere efficacemente le diverse applicazioni. Sono stati proposti due approcci principali per identificare i servizi basandosi sull'analisi del campo SNI:

1. **Identificazione basata sul nome host contenuto nello SNI della prima sessione TLS**;
2. **Identificazione basata sulla frequenza di occorrenza dello SNI**.

Il secondo metodo rappresenta un'evoluzione più complessa e accurata, mentre il primo può essere considerato una versione semplificata e leggera. Quest'ultimo costituisce il punto di partenza concettuale per il presente lavoro.

4.1.1 Identificazione basata sull'SNI della prima sessione

1. Si accede una volta a ciascun servizio candidato e si cattura il relativo traffico TLS.
2. Si analizzano i campi SNI di tutte le sessioni TLS generate e si considera il primo valore rilevato.
3. Si costruisce quindi un database che memorizza la relazione tra ciascun servizio e il relativo vettore di occorrenza (di dimensione unitaria), denominato *SNI occurrence database*.

Questo metodo riduce la dimensionalità del vettore di occorrenza a uno, basandosi esclusivamente sulla prima sessione TLS di ciascun accesso. Il campo SNI della prima sessione TLS solitamente descrive il nome host incluso nell'URL del servizio visitato, permettendo quindi un'associazione diretta e intuitiva.

L'identificazione del servizio è possibile solo se il nome host è univoco all'interno del set di servizi considerato. Se invece un dominio è condiviso da più servizi (n), il metodo restringe i candidati a tali n elementi, ma non è in grado di determinare univocamente il servizio.

Questo approccio risulta estremamente efficiente, richiedendo tempi di elaborazione ridotti e un basso consumo di risorse computazionali, rendendolo adatto per implementazioni leggere. Tuttavia, nel traffico reale generato da utenti comuni, la prima connessione TLS potrebbe non corrispondere al servizio principale, ma a richieste di background o a flussi di terze parti.

Per questo motivo, il presente lavoro si ispira a tale metodologia, ma la estende considerando un insieme più ampio di valori SNI osservati durante la cattura.

4.2 Tecniche di classificazione del traffico secondo Burgetová et al.

L'articolo [9] distingue tre principali categorie di tecniche per la classificazione del traffico TLS:

1. **Metodi basati sulla struttura del protocollo**: sfruttano campi in chiaro come *SNI* e *Certificate Information*. Sono rapidi ma vulnerabili a tecniche di cifratura più recenti;
2. **Metodi di fingerprinting**: generano firme deterministiche, come *JA4* e *JA3S*, basate sulla combinazione e sull'ordine dei parametri TLS nel messaggio di handshake;
3. **Metodi statistici o di Machine Learning**: analizzano caratteristiche dei flussi, come dimensione dei pacchetti e tempo di interarrivo, per addestrare modelli predittivi. Questi metodi raggiungono un'elevata accuratezza ma richiedono dataset di addestramento ampi e risorse computazionali elevate.

4.2.1 Identificazione basata su SNI e TLS Handshake

Le tecniche avanzate basate sull'analisi dello SNI mostrano un'elevata accuratezza nell'identificazione di servizi specifici:

- **Identificazione basata su SNI Occurrence**: rappresenta gli SNI osservati durante un singolo accesso come un vettore binario. Applicando inferenza bayesiana, è possibile identificare con buona precisione il servizio corrispondente. Tali metodi hanno mostrato un'accuratezza media del 90.39% per Google e dell'81.62% per Yahoo;
- **Fingerprint JA4 e JA4S**: la combinazione di *JA4* e *JA4S* ha raggiunto un'accuratezza del 97,1% nella classificazione dei flussi, dimostrandosi superiore a *JA3* grazie alla gestione dell'ordine dei parametri TLS, ma comunque solo di poco migliore rispetto a *JA3* e *JA3S*.

Da questi lavori emerge che l'unione delle informazioni derivate da `SNI` e dalle fingerprint TLS può essere sufficiente per identificare correttamente un'applicazione. Nel presente studio vengono utilizzati `JA4` e `JA3S` per raggruppare i flussi TLS, poiché `JA4S` è ancora in fase di sviluppo e standardizzazione. Tuttavia, gli studi citati non considerano la complessità del traffico reale, generato da utenti e dispositivi eterogenei, dove possono coesistere flussi paralleli o rumore di fondo.

4.3 Correlazione dei flussi secondo Park et al.

L'articolo [26] propone tecniche di correlazione per migliorare la classificazione in assenza di SNI. In particolare, l'analisi della PSD (*Packet Size Distribution*) consente di estrarre informazioni statistiche sulla distribuzione delle dimensioni dei pacchetti all'interno di un flusso TLS.

- **Firme multi-modali (SNI + PSD)**: l'integrazione di firme *SNI* e *PSD* riduce la dipendenza dal campo SNI e aumenta la robustezza del riconoscimento. I metodi che combinano informazioni di header, SNI e PSD hanno raggiunto un'accuratezza del 97–100% nell'identificazione di azioni specifiche su Microsoft Office 365.

Da questo studio emerge l'importanza di integrare, oltre ai metadati TLS, anche caratteristiche statistiche dei flussi; nel mio caso, il numero di pacchetti per flusso e il numero totale di flussi generati dall'applicazione.

4.4 Limitazioni dell'approccio proposto

Nonostante l'efficacia dei metodi analizzati, esistono alcune limitazioni notevoli che si riflettono anche in questo studio:

- **Crittografia dell'SNI**: l'introduzione di TLS 1.3 e dell'ECH (*Encrypted Client Hello*) rende progressivamente inaccessibile il campo SNI, limitando l'efficacia dei metodi basati su di esso;

- **Bypass dei filtri SNI**: i client TLS 1.2 possono manipolare o omettere l'SNI, inviando nomi di dominio alternativi per aggirare sistemi di filtraggio.
- **Ambiguità nelle fingerprint**: firme TLS come *JA3* o *JA4* possono essere condivise da più applicazioni, riducendo la capacità discriminante del metodo.

La motivazione di questo lavoro è quindi quella di progettare un approccio automatico, veloce e robusto, basato sull'analisi del traffico TLS, QUIC e HTTPS riguardante le applicazioni che non presentano i punti appena descritti, al fine di generare firme DPI affidabili, riutilizzabili e direttamente integrabili in *nDPI* per il riconoscimento in tempo reale del traffico, anche in presenza di rumore di fondo o connessioni spurie.

4.5 Approccio proposto

Unendo i risultati dei lavori analizzati, ognuno con i propri limiti, si propone un sistema ibrido capace di:

- analizzare tutti i flussi TLS osservati in una cattura, filtrandone una parte, e considerando anche gli `SNI` successivi al primo;
- filtrare gli `SNI` irrilevanti, escludendo quelli già noti a nDPI, associati a CDN o riconducibili a siti pubblicitari;
- raggruppare i flussi in base alle fingerprint `JA4`, `JA3S` e ai certificati, selezionando solo quelli che superano una soglia minima di pacchetti generati nel traffico catturato, parametro configurabile dall'utente (attualmente impostato a 5%).

La figura seguente (cfr. Figura 4.1) offre una panoramica sintetica dell'architettura del sistema, evidenziando le principali fasi che compongono il flusso di elaborazione.

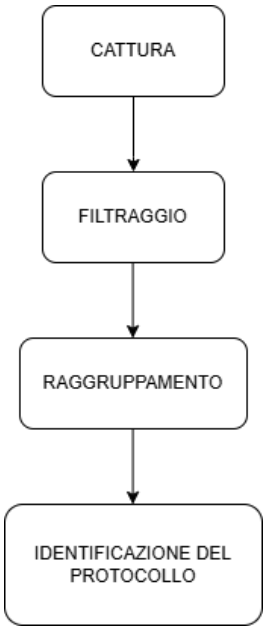

Figura 4.1: Diagramma di flusso dell'architettura proposta

Fase di cattura

Nella fase di indagine preliminare vengono eseguiti i seguenti passaggi:

1. Si effettua l'accesso a ciascuna applicazione e si cattura il relativo traffico, composto da sessioni varie, utilizzando strumenti quali **Wireshark** e **PCAPdroid**, rispettivamente per applicazioni desktop e mobile.

2. Si analizzano le parti non cifrate dei messaggi di handshake, grazie all'uso di **nDPI**, che consente di aggregare i pacchetti in flussi. In questo lavoro, tuttavia, vengono considerate solo le sessioni dei seguenti protocolli:

   `"TLS", "HTTP", "HTTP_Connect", "HTTP.WebSocket", "QUIC"`

Fase di filtraggio

Nella fase di filtraggio vengono applicati diversi criteri per eliminare il traffico di disturbo e ottenere gruppi di flussi potenzialmente utili al riconoscimento dei protocolli secondo **nDPI** e alla generazione di regole custom. Questo metodo si articola in più passaggi:

1. Si filtrano i flussi nei log di **nDPI** che non soddisfano determinati requisiti.

2. Si raggruppano i flussi rimanenti secondo chiavi specifiche, dipendenti dal protocollo analizzato, per eliminare ridondanze dovute a differenze in statistiche come numero di pacchetti, IAT, ecc., aggregandoli in un unico flusso.

3. Si esegue la fase di filtraggio degli `SNI`, rimuovendo tutti i flussi con SNI presenti in dataset di riferimento, tra cui domini pubblicitari, domini riconosciuti da **nDPI**, CDN note, e SNI generati dinamicamente.

La letteratura non mostra molti approcci per affrontare il problema descritto nel punto tre: alcuni lavori sfruttano euristiche basate sugli indirizzi IP conosciuti dei servizi non desiderati o altri applicano regole di filtraggio basate su porte o protocolli. La difficoltà maggiore risiede nel bilanciare il rischio di eliminare traffico realmente utile con quello di mantenere troppo rumore che riduce l'accuratezza dei passaggi successivi.

Fase di raggruppamento

In questa fase vengono raggruppati i flussi rimanenti dalla fase di filtraggio. La procedura comprende:

1. Raggruppamento dei flussi in base a una chiave composta da `JA4`, `JA3S` e certificato.

2. Successivamente, per ogni gruppo vengono estratti tutti gli `SNI`, che vengono aggregati tramite l'algoritmo *Longest Common Suffix Grouping*. Questa scelta è vantaggiosa per raggruppare `SNI` con molte parti non significative comuni (ad esempio: `image1.vinted.com`, `image2.vinted.com`, `www.vinted.com`), ma può risultare problematica per domini diversi (ad esempio: `glovo.api.com` e `justit.api.com`). In ogni caso, i domini esclusi a seguito di questa aggregazione vengono mostrati in output, così da consentire all'operatore di rete di comprendere meglio la situazione.

3. Vengono selezionati solo i gruppi che contribuiscono alla generazione di almeno X% del traffico complessivo, dove X è un parametro configurabile dall'utente (attualmente impostato al 5%).

Fase di identificazione del protocollo

Nella fase di identificazione si procede all'estrazione di una regola per il riconoscimento del protocollo:

1. Se il gruppo contiene almeno un `SNI`, questo viene indicato come possibile identificatore, insieme agli altri `SNI` presenti nella lista. Ricordiamo che tali `SNI` sono stati raggruppati in base a `JA4`, `JA3S` e certificato, e che vengono mostrati anche gli `SNI` scartati durante la fase di raggruppamento.

2. Se non sono presenti `SNI`, viene estratto dal campo `Subject` del certificato l'attributo `CN`, che contiene il nome di dominio del proprietario del certificato.

3. Se non è disponibile il certificato, viene mostrato il `JA4` parziale utilizzato. Ricordiamo che sono considerate solo le ultime due parti del `JA4`, e quindi nel risultato vengono mostrati tutti i possibili `JA4` completi osservati che potrebbero identificare l'applicazione.

Capitolo 5

Implementazione

In questa sezione viene descritta l'implementazione del parser, illustrando i metodi proposti precedentemente. Il sistema è stato realizzato in *Python*. È stata inoltre sviluppata una versione con interfaccia grafica, basata su Textual[1], ma poiché non è rilevante ai fini dell'analisi del parser, non ne verranno approfonditi i dettagli implementativi. Come illustrato nel capitolo precedente, il parser esegue diverse fasi per giungere al risultato finale. Tali fasi sono descritte di seguito:

5.1 Fase di cattura

In questa fase viene acquisito il traffico generato da diverse applicazioni, che verranno descritte nel capitolo successivo. Il traffico raccolto viene successivamente elaborato da *nDPI*, che produce un file di log successivamente interpretato dal parser sviluppato.

Questo file contiene una serie di statistiche generate da *nDPI* e l'elenco completo dei flussi catturati. Ogni flusso è rappresentato in formato testuale, riportando informazioni come indirizzi IP e porte di origine / destinazione, protocollo, categoria, durata, dimensione dei pacchetti, SNI, versioni TLS supportate, fingerprint `JA3S` / `JA4`, dettagli del certificato, e altre metriche di rete che non verranno considerate in questo studio.

Un esempio semplificato di flusso registrato nel log è il seguente:

```
TCP 192.168.80.96:63666 <-> 108.138.192.75:443 [proto: 91/TLS]
[IP: 265/AmazonAWS][Encrypted][Confidence: DPI]
[Hostname/SNI: glovo.dhmedia.io][TLSv1.3]
[JA4: t13d1513h2_8daaf6152771_eca864cca44a]
[JA3S: f4febc55ea12b31ae17cfb7e614afda8]
[Cipher: TLS_AES_128_GCM_SHA256]
[329 pkts/31037 bytes <-> 584 pkts/796459 bytes]
```

[1] https://textual.textualize.io

```
[Plen Bins: 5,1,0,0,0,3,0,0,...,0,0]
[....]
```

Il campo `Plen Bins` rappresenta una distribuzione statistica delle dimensioni dei pacchetti osservati in un flusso. *nDPI* suddivide l'intervallo possibile delle lunghezze dei pacchetti in un certo numero di "bin" e conta quanti pacchetti rientrano in ciascun intervallo. In questo modo si ottiene una sorta di firma del traffico basata sulla variazione delle dimensioni dei pacchetti, indipendentemente dal loro contenuto.

Questa informazione è particolarmente utile per distinguere applicazioni che utilizzano lo stesso protocollo cifrato, ma che generano pattern di traffico diversi. Infatti, anche se il contenuto dei pacchetti è criptato, la loro dimensione e frequenza possono variare significativamente a seconda dell'applicazione o del servizio utilizzato. Tale campo verrà sfruttato nella fase di filtraggio descritta nella sezione successiva.

Ogni voce del log rappresenta quindi un flusso univoco, che sarà poi oggetto delle successive fasi di filtraggio, raggruppamento e identificazione del protocollo.

5.2 Fase di filtraggio

Nella fase di filtraggio vengono applicati diversi criteri per eliminare il traffico di disturbo e ottenere gruppi di flussi potenzialmente utili al riconoscimento dei protocolli secondo **nDPI**. Questo processo si articola in più passaggi consecutivi, descritti di seguito.

Pulizia iniziale dei flussi

Il primo passo consiste nell'eliminazione dei flussi che non soddisfano determinati requisiti. In particolare, vengono scartati:

- i flussi con il campo `Plen Bins` composto interamente da zeri, in quanto indicano flussi incompleti o appena creati, privi di informazioni significative poiché flussi appartenenti all'applicazione monitorata ma catturati solo parzialmente, cioè prima che il traffico assumesse caratteristiche identificabili;
- i flussi con handshake `TLS` non completati;
- i flussi non appartenenti ai protocolli *TLS*, *QUIC*, *HTTP*, *HTTP_Connect* e *HTTP.WebSocket*.
- i flussi già riconosciuti da *nDPI* come protocolli specifici, che non richiedono ulteriori analisi. Vedremo poi come questo filtro può non essere vero in base ad un parametro configurabile dall'utente.

Aggregazione dei flussi

Successivamente, i flussi rimanenti vengono raggruppati in base a specifiche chiavi di aggregazione, che variano a seconda del protocollo analizzato. In particolare:

- **Per i flussi TLS**: la chiave utilizzata è composta da: `ip_source`, `tcp_fingerprint`, `ip_destination`, `sni`, `ja3s`, `ja4`, `tls_version`, `proto_field` e `transport_protocol`;
- **Per i flussi HTTP**: viene impiegata la chiave composta da: `ip_source`, `tcp_fingerprint`, `ip_destination`, `sni`, `url`, `user_agent`, `proto_field` e `transport_protocol`;
- **Per i flussi QUIC**: la chiave di aggregazione include: `ip_source`, `tcp_fingerprint`, `ip_destination`, `sni`, `ja3s`, `ja4`, `quic_version`, `proto_field` e `transport_protocol`.

Durante il raggruppamento, il numero totale di pacchetti e flussi appartenenti a ciascun gruppo viene sommato, in modo da preservare le informazioni quantitative necessarie alle analisi successive.

Filtro sugli SNI

L'ultima fase riguarda il filtraggio degli `SNI`, durante la quale vengono applicati diversi criteri di esclusione:

- Vengono rimossi i flussi associati a SNI troppo lunghi, applicando la tecnica *TLD+4*, poiché dai test è emerso che tali domini sono spesso dinamici o appartenenti a Content Delivery Network;
- Infine, vengono eliminati i flussi i cui SNI sono presenti in specifici dataset di esclusione:
  - un dataset di SNI già riconosciuti da *nDPI*;
  - un dataset di domini pubblicitari costruito appositamente;
  - un dataset di domini associati a CDN, anch'esso realizzato dal sottoscritto;
  - un dataset di domini associati a servizi di monitoraggio comportamentale, protezione e tracciamento.

Questo filtraggio finale consente di ottenere un insieme di flussi più pulito e rappresentativo del traffico realmente utile per le successive fasi di analisi e classificazione.

5.3 Fase di raggruppamento

In questa fase vengono raggruppati i flussi rimanenti dalla fase di filtraggio. La procedura comprende:

1. Raggruppamento dei flussi in base a una chiave composta da `JA4`, `JA3S` e certificato. I flussi privi di `JA4`, certificato o `SNI` vengono scartati e non raggruppati poiché non è possibile estrarre informazioni utili alla generazione di regole custom per nDPI.

2. Per ogni gruppo, vengono estratti tutti gli `SNI` e aggregati tramite l'algoritmo *Longest Common Suffix Grouping*. Questa scelta è utile per raggruppare `SNI` con molte parti non significative comuni (ad esempio: `image1.vinted.com`, `image2.vinted.com`, `www.vinted.com`), ma può risultare problematica per domini diversi (ad esempio: `glovo.api.com` e `justit.api.com`).

3. Dopo l'aggregazione degli `SNI` all'interno di ciascun gruppo, vengono selezionati solo i gruppi che contribuiscono alla generazione di almeno X% del traffico complessivo, dove X è un parametro configurabile dall'utente (attualmente impostato al 5%). All'interno di questi gruppi, vengono poi selezionati i soli flussi, con campi `SNI`, `JA4` o certificato, che hanno generato il maggior volume di traffico, considerando sempre la soglia definita X.

5.4 Fase di identificazione del protocollo

Una volta completata la fase di raggruppamento, vengono generate le regole finalizzate all'identificazione del protocollo. Queste regole possono essere classificate come:

- **Corrette**, quando l'identificazione è univoca e affidabile;
- **Parzialmente corrette**, quando la regola è valida ma contiene informazioni in eccesso o in difetto;
- **Non valide**, quando i flussi disponibili non contengono elementi sufficienti per l'identificazione.

1. Se il gruppo contiene almeno un `SNI`, questo viene indicato come possibile identificatore, insieme agli altri `SNI` presenti nella lista. Ricordiamo che tali `SNI` sono stati raggruppati in base a `JA4`, `JA3S` e certificato, e che vengono mostrati anche gli `SNI` scartati durante la fase di raggruppamento. Esempio di regola è il seguente:

   ```
   host:"expedia.com"@customproto        oppure
   host:"expedia.com",host:"expe.it"@customproto
   ```

   In questi esempi di regole, il singolo `SNI` è il campo risultante dalle fasi precedenti, mentre `customproto` rappresenta il nome del protocollo passato in input.

2. Se non sono presenti `SNI`, viene estratto dal campo `Subject` del certificato l'attributo `CN`, che contiene il nome di dominio del proprietario del certificato. Esempio di regola è il seguente:

```
trusted_subject_dn:"CN=google.com"@customproto
```

In questa regola, `CN` corrisponde al campo `CN` del `Subject` del certificato, mentre `customproto` rappresenta il nome del protocollo passato in input. L'output include anche le istruzioni per inserire correttamente il certificato in *nDPI*, pertanto la regola non può essere applicata direttamente così com'è.

3. Se non è disponibile il certificato, viene mostrato il `JA4` parziale utilizzato. Si considerano solo le ultime due parti del `JA4`, e nel risultato vengono riportati tutti i possibili `JA4` completi che potrebbero identificare l'applicazione. Esempio di regola è il seguente:

```
ja4:c866b44c5a26_b39be8c56777@customproto
```

In questa regola, il campo `JA4` rappresenta le ultime due parti della fingerprint TLS, mentre `customproto` è il nome del protocollo passato in input. È importante notare che questa regola, così com'è, non è utilizzabile direttamente: bisogna applicare le regole generate dal sistema, che includono tutti i `JA4` univoci le cui ultime due parti coincidono con quelle riportate. Esempio:

```
ja4:t12d1409h1_c866b44c5a26_b39be8c56777@customproto
ja4:t12d1409h2_c866b44c5a26_b39be8c56777@customproto
```

5.5 Commenti sui parametri utilizzati

All'interno del parser è presente un file `constants` contenente i parametri utilizzati durante le fasi descritte precedentemente. Di seguito vengono commentati i parametri più importanti, essenziali per la fase di test:

- **SHOW_NDPI_PROTOCOL**: Questo parametro permette di decidere se scartare in partenza i flussi già riconosciuti da *nDPI*. Tuttavia, alcune applicazioni o funzionalità particolari possono definire protocolli differenti rispetto a quello principale. Ad esempio, *Google Maps* è riconosciuto da *nDPI* come traffico Google, ma è comunque possibile identificare il traffico relativo alle mappe offline grazie a un `SNI` unico. Nel capitolo successivo, nella fase di test e validazione, questo parametro potrà essere modificato da `true` a `false` e viceversa, a seconda dell'applicazione analizzata.

- **TOP_PERCENT**: Questa soglia consente di selezionare solo i gruppi di flussi, contenenti `SNI`, `JA4` e certificati, che superano una percentuale minima di traffico. Durante la fase di test e validazione, questo parametro può variare. Rappresenta la porzione di traffico totale che si intende considerare.

- **PROTOCOLS**: Elenca i protocolli che il parser deve considerare durante l'analisi.
- **KEYS_BY_PROTOCOL**: Specifica le chiavi utilizzate per il raggruppamento dei flussi, differenziate in base al protocollo.
- **SNI_MAX_DOMAIN**: Indica il numero massimo di domini supportati per ciascun `SNI`. Se un `SNI` supera questo numero, viene scartato. Questo parametro è impiegato nell'implementazione della tecnica *TLD+4*, utile per filtrare `SNI` dinamici o appartenenti a CDN.

Capitolo 6

Validazione

In questo capitolo vengono approfonditi i processi di validazione e i test effettuati sul sistema, con l'obiettivo di valutarne l'efficacia.

6.1 Ambiente di test e dataset

L'ambiente di test utilizzato comprende catture effettuate da diversi dispositivi mobili mediante l'applicazione PcapAndroid per la raccolta del traffico generato dalle applicazioni mobili e dalle VPN, e Wireshark per le catture HTTPS su sistemi Windows e macOS. Entrambi gli strumenti hanno prodotto file in formato `pcap` o `pcapng`, annotando le connessioni in base ai processi applicativi.

Mi sono concentrato principalmente sulle applicazioni Android, data la loro maggiore diffusione sul mercato, anche se applicazioni appartenenti ad altri ambienti possono essere analizzate con un approccio analogo. Le catture non sono state eseguite in un ambiente isolato, ma deliberatamente in condizioni il più possibile realistiche, poiché il tool sviluppato è progettato per operare in presenza di traffico "rumoroso" e in contesti reali.

Le applicazioni oggetto di analisi sono state selezionate in base a suggerimenti forniti da un gruppo di utenti scelti casualmente, al fine di mantenere un contesto d'uso realistico e rappresentativo del traffico effettivamente generato da utenti comuni. Per ciascuna applicazione è stata effettuata una singola cattura del traffico, ritenuta sufficiente per testare la capacità del sistema di riconoscere pattern distintivi anche a partire da campioni limitati ma reali.

Nel caso delle applicazioni Android, grazie alle funzionalità di *PcapAndroid*, vengono monitorate esclusivamente le comunicazioni generate dall'applicazione di interesse. Al contrario, nelle catture effettuate tramite *Wireshark*, vengono registrate tutte le comunicazioni di rete provenienti dall'host.

È stato creato un dataset contenente 30 applicazioni, suddivise in: 19 mobili, 4 web e 7 VPN. Tra le applicazioni mobili, 3 sono state classificate come sottoprotocollo di un protocollo già presente in *nDPI*. Come già descritto, *nDPI* consente di definire

protocolli in base a determinate caratteristiche, tra cui, nel contesto di questo lavoro, l'*SNI*, la fingerprint *JA4* e il *CN* del certificato. Un sottoprotocollo rappresenta quindi una versione più specifica di un protocollo già noto a *nDPI*. Nei paragrafi successivi verrà illustrata l'analisi condotta su tali applicazioni.

Durante la fase di cattura è stato osservato come molte applicazioni desktop e mobili, disponibili sul *Microsoft Store* e sul *Play Store*, siano distribuite come servizi *SaaS* (*Software as a Service*). Queste applicazioni, essendo ospitate su infrastrutture cloud condivise, presentano spesso limitazioni nell'identificazione precisa dell'applicazione stessa: in particolare, non sempre sono presenti *SNI* unici o fingerprint *JA4* distintivi che permettano di riconoscere in modo univoco il servizio. Ciò accade perché il fornitore cloud può ospitare più applicazioni o domini sullo stesso server, rendendo difficile distinguere i flussi di rete appartenenti a una specifica applicazione.

Di conseguenza, in questo studio l'analisi si è concentrata su applicazioni che non appartengono a queste categorie, riservando lo studio delle applicazioni hostate su servizi SaaS a futuri sviluppi del lavoro.

6.2 Metriche di valutazione

Per valutare correttamente le prestazioni del sistema, introduciamo in questa sezione alcune semplici metriche di riferimento. La metrica principale è la *coverage*, calcolata come percentuale applicata ai flussi. Essa indica la quota di flussi riconosciuti correttamente dal sistema rispetto al totale osservato.

Ad esempio, se su 100 flussi totali il sistema ne riconosce 80, la coverage sarà:

$$\text{Coverage} = \frac{80}{100} \times 100 = 80\%.$$

Lo stesso principio si applica anche al numero di pacchetti.

Questa metrica fornisce un'indicazione diretta della capacità del sistema di identificare correttamente il traffico di rete. Tuttavia, le regole generate dal sistema e utilizzate per ottenere tali risultati possono introdurre *falsi positivi* e *falsi negativi*. Nelle varie applicazioni analizzate, verranno discussi nel dettaglio i risultati, individuando e quantificando tali errori.

A partire da queste osservazioni, verrà costruita una **matrice di confusione** per valutare la correttezza complessiva del sistema, stimando la frequenza degli errori e quindi la percentuale di accuratezza e di errore. Da tale matrice sarà inoltre possibile calcolare diverse metriche di valutazione standard:

- **Accuratezza**: indica quanto spesso il sistema ha classificato correttamente, cioè la percentuale di casi in cui ha dato la risposta giusta (sia veri positivi che veri negativi) rispetto al totale dei casi analizzati;
- **Precisione**: misura quante volte, tra tutti i flussi che il sistema ha riconosciuto come positivi, essi lo erano davvero. Diminuisce se ci sono molti **falsi positivi**;

- **Sensibilità**: misura quante volte, tra tutti i flussi che dovevano essere positivi, il sistema li ha riconosciuti correttamente. Diminuisce se ci sono molti **falsi negativi**;
- **F1-Score**: combina precisione, recall e sensibilità in un unico valore, calcolando la loro media armonica. È utile per valutare il bilanciamento tra falsi positivi e falsi negativi.

Queste misure permettono di valutare in modo più approfondito il comportamento del sistema, evidenziando non solo la sua correttezza globale ma anche la qualità delle decisioni nei singoli casi.

Oltre alla matrice di confusione, verranno presentati quattro grafici di sintesi, di cui uno rappresenta la combinazione di tre misure fondamentali. Tali grafici rappresentano:

- la coverage ottenuta da **nDPI** sulle applicazioni analizzate;
- la coverage di **nDPI** + **GT** (*Ground Truth*), dove la GT rappresenta le regole corrette per identificare le applicazioni;
- la coverage di **nDPI** + **regole generate dal sistema** sviluppato.

È utile chiarire cosa si intende per **GT** (*Ground Truth*) in questo contesto: rappresenta l'insieme di regole selezionate manualmente per identificare in modo univoco ciascuna applicazione e costituisce il riferimento per valutare le prestazioni del sistema. Un presupposto fondamentale in questa fase di validazione è che la **Ground Truth** sia corretta. In questo lavoro, la GT è stata definita manualmente dal sottoscritto per descrivere in modo univoco ogni applicazione, diventando così un elemento centrale e un punto di riferimento imprescindibile per la valutazione del sistema.

6.3 Casi di test e configurazione dei parametri

In questa sezione vengono descritti i principali parametri configurabili dall'utente, utilizzati durante la fase di test. Sono stati definiti due casi di test, denominati **Caso A** e **Caso B**, ognuno caratterizzato da una diversa configurazione dei parametri per analizzare l'impatto delle scelte sul comportamento del sistema.

Caso A: Configurazione standard
Questo caso rappresenta il comportamento considerato "normale" o di riferimento. I parametri impostati sono i seguenti:

- **SHOW_NDPI_PROTOCOL**: questo parametro consente di decidere se scartare o meno, fin dall'inizio, i flussi già riconosciuti da *nDPI*. Nel set di 30 applicazioni catturate, è stato impostato su `true` solo in 3 casi (corrispondenti ai sottoprotocoli analizzati), mentre in tutti gli altri è rimasto su `false`.
- **TOP_PERCENT**: rappresenta la soglia che permette di selezionare solo i gruppi di flussi, `SNI`, `JA4` e certificati che superano una percentuale minima di traffico. Nel caso standard, la soglia è stata fissata a `5%`.

Caso B – Variazione della soglia di selezione
In quest'altro caso invece, si è voluto osservare l'effetto dell'aumento della soglia di selezione:

- **SHOW_NDPI_PROTOCOL**: invariato rispetto al Caso A.
- **TOP_PERCENT**: aumentata a `20%`, per verificare come una maggiore selettività influisca sulla qualità e quantità delle regole generate.

Per un utilizzo pratico e futuro del sistema, si consiglia di sperimentare diverse combinazioni di questi due parametri. In particolare, è utile variare la soglia `TOP_PERCENT` e alternare l'attivazione o la disattivazione del parametro `SHOW_NDPI_PROTOCOL`, tenendo presente che valori molto bassi della soglia permettono il passaggio di un numero maggiore di flussi attraverso i filtri, incrementando così sia il numero di regole potenzialmente corrette, con la possibilità di introdurre falsi positivi.

6.3.1 Applicazioni analizzate

Le applicazioni analizzate, come spiegato in precedenza, sono state scelte da un gruppo di utenti selezionati casualmente, con l'obiettivo di mantenere un contesto realistico e non artificiale. Questa scelta ha permesso di ottenere un insieme di dati rappresentativo dell'utilizzo reale del sistema, evitando ambienti isolati o scenari di laboratorio.

Le applicazioni considerate coprono diverse categorie, tra cui shopping, alimentari, viaggi, giochi, VPN, strumenti di utilità e servizi web. Di seguito è riportato l'elenco completo delle applicazioni analizzate:

Tabella 6.1: Elenco delle applicazioni analizzate

Applicazioni generali	**VPN**	**Strumenti di supporto**	**Servizi web**
Doctor_App	AviraVPN	NotebookLLM	Warframe
Trenitalia	FireNetVPN	Maps	Expedia
Subito	HideVPN	MapsOffline	Austrian
Strava	HoxxVPN		Hostelworld
Notion	PlanetVPN		
MarioKart	UltrasurfVPN		
MarinoBus	XrpTunnelVPN		
Ryanair			
Crunchyroll			
Mapy			
LidlPlus			
Klarna			
JustEat			
Glovo			
Alza			
Vinted			

6.4 Risultati caso A

La valutazione delle applicazioni nel caso A, così come negli altri due casi, è stata effettuata suddividendo i risultati in base alle categorie descritte in precedenza: applicazioni VPN, sottoprotocolli, servizi web e applicazioni generali.

Come anticipato, verranno presentati quattro grafici principali. Il primo mostra la percentuale di riconoscimento di *nDPI* sull'applicazione analizzata, prima dell'applicazione di qualsiasi regola generata dal sistema. Le immagini 6.1 - 6.2 riportano i risultati ottenuti per ciascuna categoria di applicazioni.

Alcuni commenti sui risultati: per tutte le applicazioni testate, le barre blu indicano la percentuale di flussi riconosciuti rispetto al totale dei flussi generati durante il monitoraggio dell'applicazione.

Per le applicazioni generali, le barre mostrano una percentuale di coverage compresa tra il 14% e il 45%. Questo conferma che, durante la cattura del traffico di un'applicazione, sono presenti numerosi servizi e protocolli di supporto o di rumore che operano in background e che *nDPI* riesce a riconoscere. Tuttavia, nel complesso, le applicazioni analizzate non vengono identificate direttamente da *nDPI*. Simile risultato per il caso dei servizi web

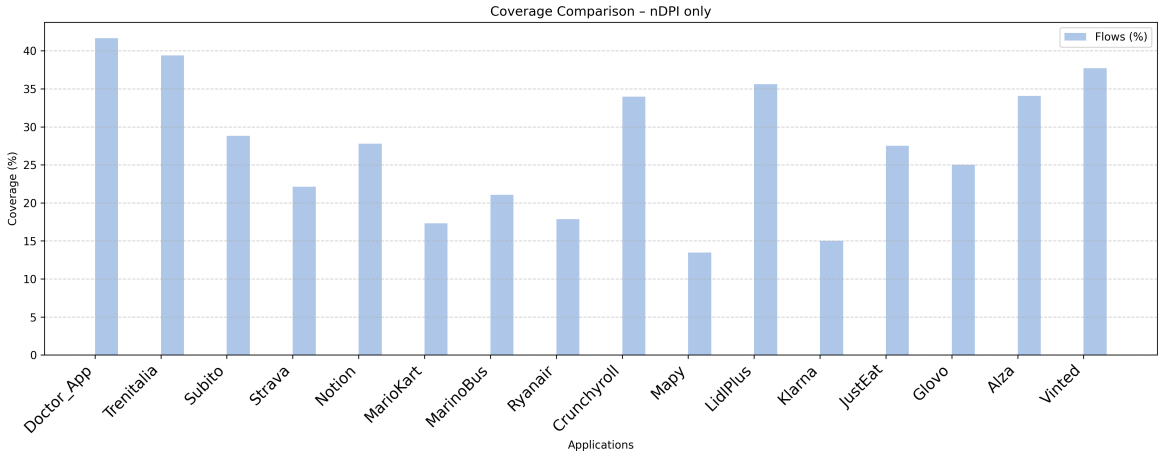

Figura 6.1: Applicazioni generali

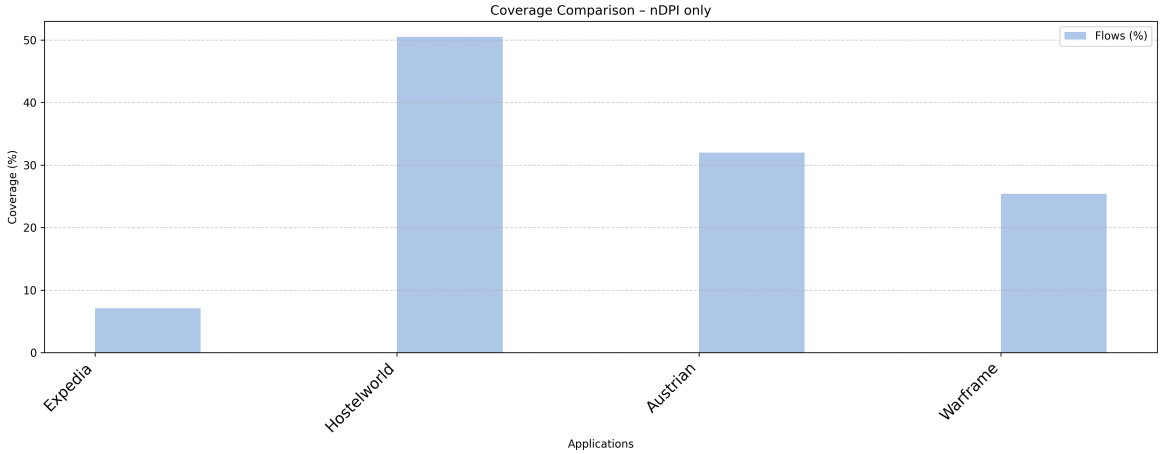

Figura 6.2: Servizi web

Per le sottoapplicazioni si osserva come *nDPI* riesca a riconoscere una percentuale minima del 52% (per *NotebookLLM*) fino a un massimo del 79% (per *Maps*).

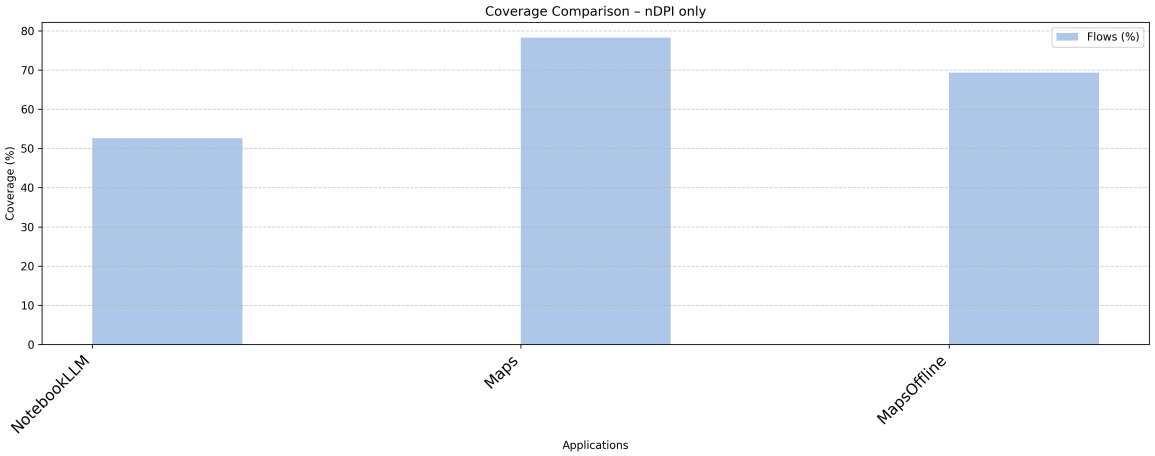

Figura 6.3: Sottoprotocolli

Nel caso delle VPN, si noti che *nDPI* riconosce come *VPN* la sola *AviraVPN*, con una coverage di circa il 50% dei flussi analizzati. Per VPN come *Hide.me*, *HoxxVPN* e *PlanetVPN*, invece, il sistema non riesce quasi mai a identificarne correttamente il traffico. Al contrario, per *XrpTunnelVPN*, *UltraSurfVPN* e *FireNetVPN*, la percentuale di riconoscimento è sensibilmente più alta, raggiungendo nel caso di *FireNetVPN* circa il 73%.

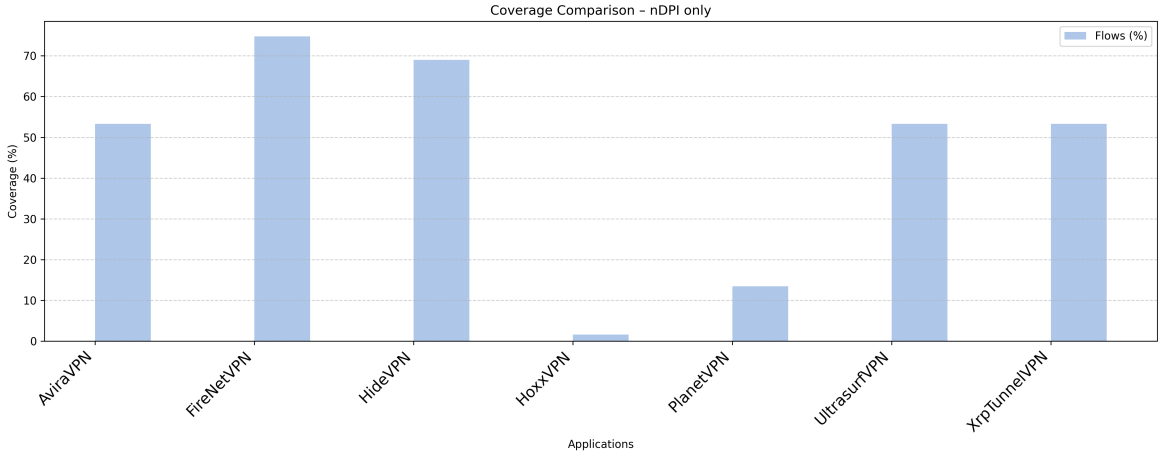

Figura 6.4: Applicazioni VPN

È importante sottolineare che tali differenze dipendono fortemente sia dalla qualità della cattura effettuata dall'utente, sia dal tipo di tecnologia VPN utilizzata. Ad esempio, nel caso di *FireNetVPN*, la maggior parte dei flussi catturati, appartiene a servizi come *YouTube*, mentre solo pochi flussi corrispondono effettivamente alla VPN. Analizzando i pacchetti riconosciuti da *nDPI* per Firenet VPN infatti (Figura Figura 6.5), si nota come questi ultimi risultino molto bassi rispetto ai flussi. Lo stesso principio vale anche per *UltraSurfVPN* e *XrpTunnelVPN*.

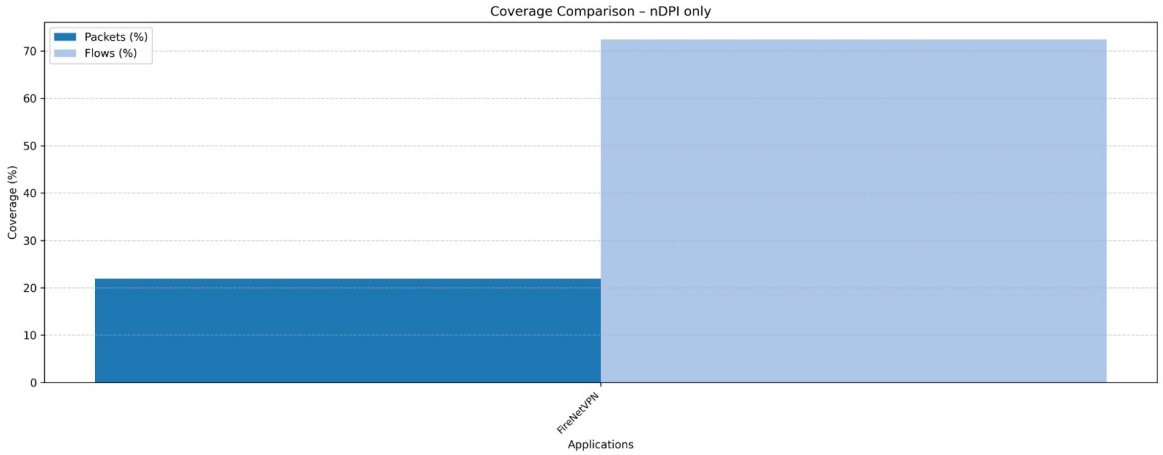

Figura 6.5: Flussi e pacchetti di *FireNetVPN*

Se confrontiamo ora queste immagini con i grafici ottenuti dall'unione di *nDPI* con le regole reali basate sulla *Ground Truth* (Figura 6.6 - 6.9), possiamo osservare un incremento generale del numero di flussi riconosciuti. In particolare, per *FireNet VPN* (Figura 6.10) si nota un notevole aumento dei pacchetti identificati, con un incremento di circa il 10% nel numero complessivo di flussi riconosciuti.

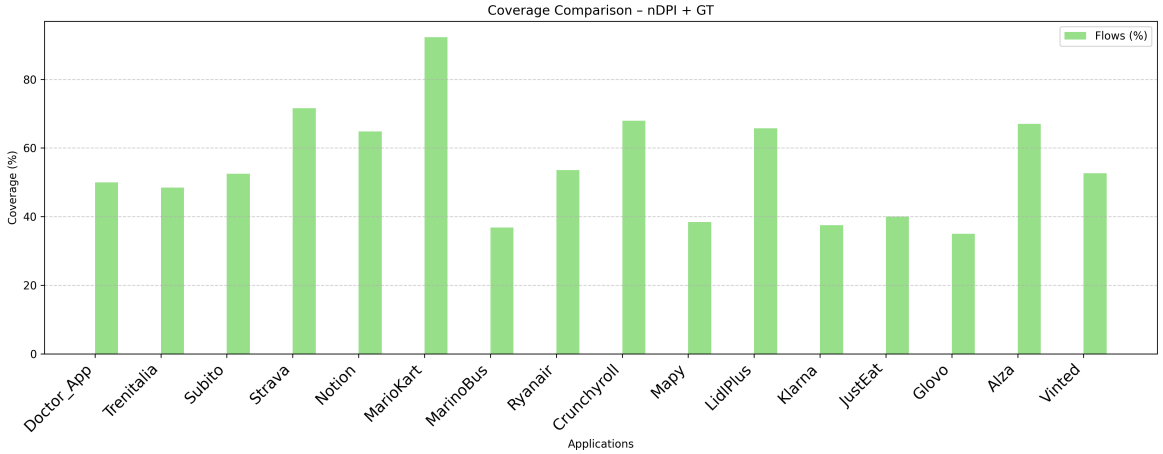

Figura 6.6: Applicazioni generali

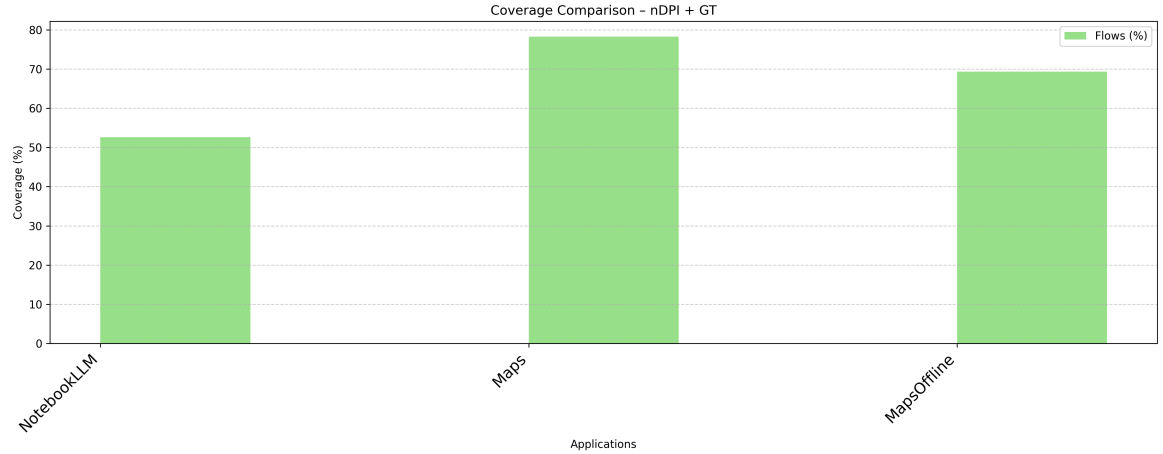

Figura 6.7: Sottoprotocolli

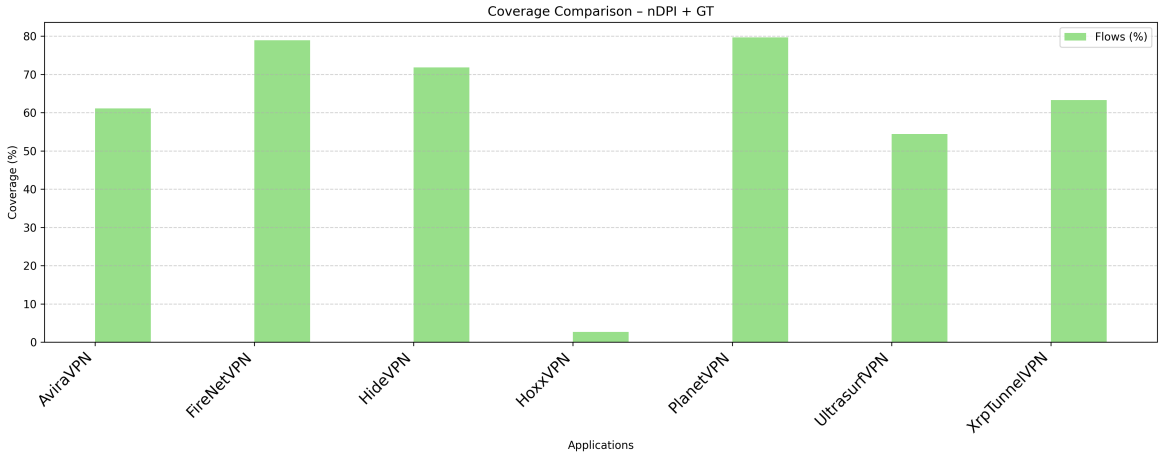

Figura 6.8: Applicazioni VPN

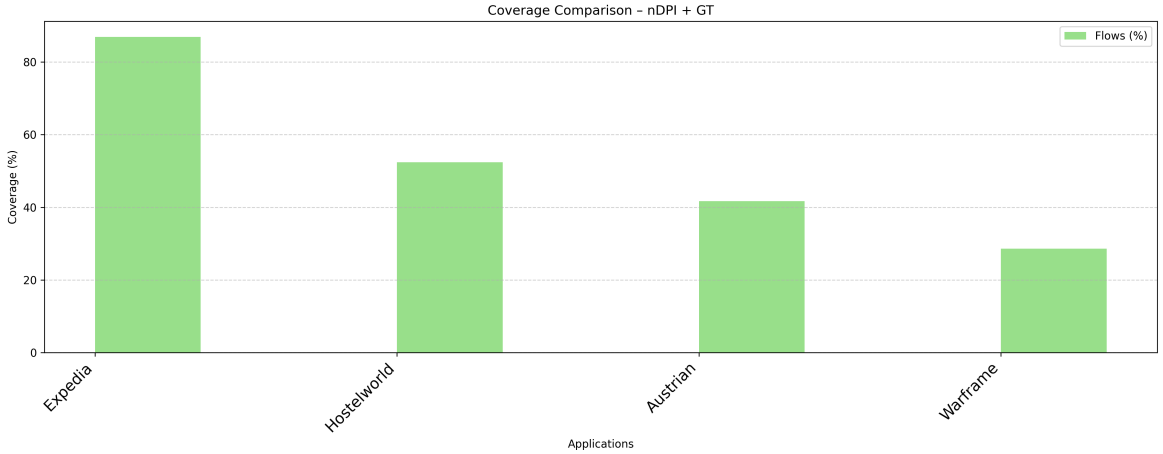

Figura 6.9: Servizi web

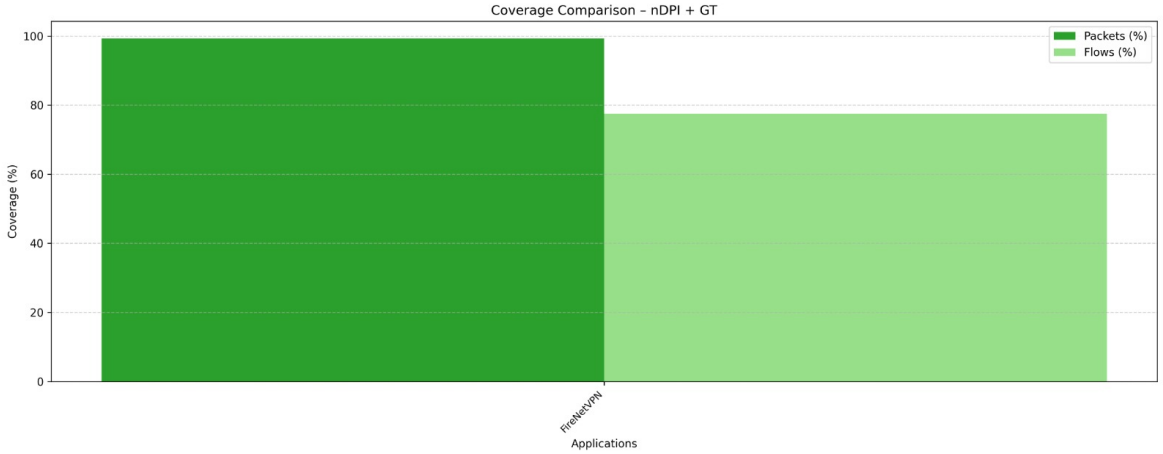

Figura 6.10: Flussi e pacchetti di *FireNet VPN*

A questo punto, ci si aspetta che il sistema sviluppato generi delle regole in grado di avvicinarsi alle percentuali di *coverage* ottenute da *nDPI + Ground Truth*. Di seguito sono riportati i risultati ottenuti (Figura 6.11 - 6.14):

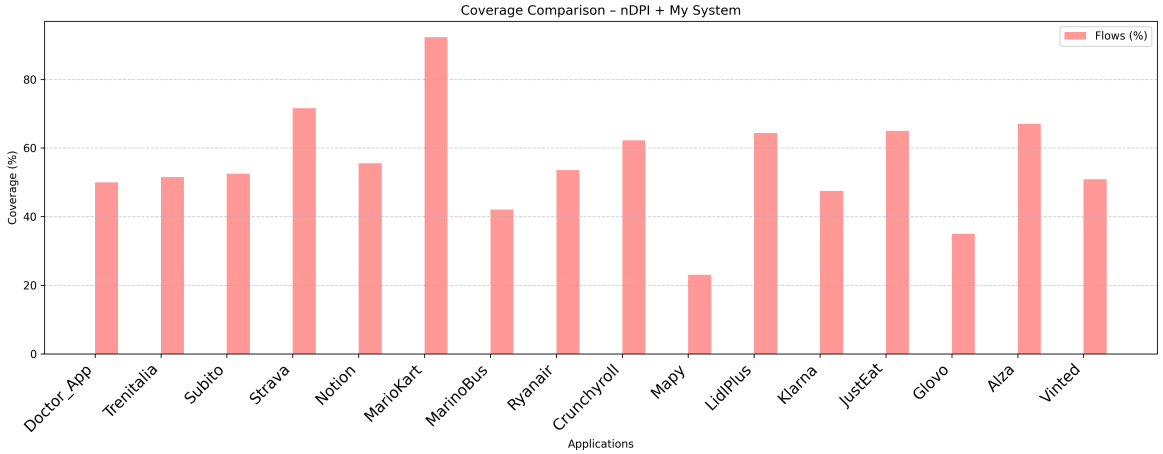

Figura 6.11: Applicazioni generali

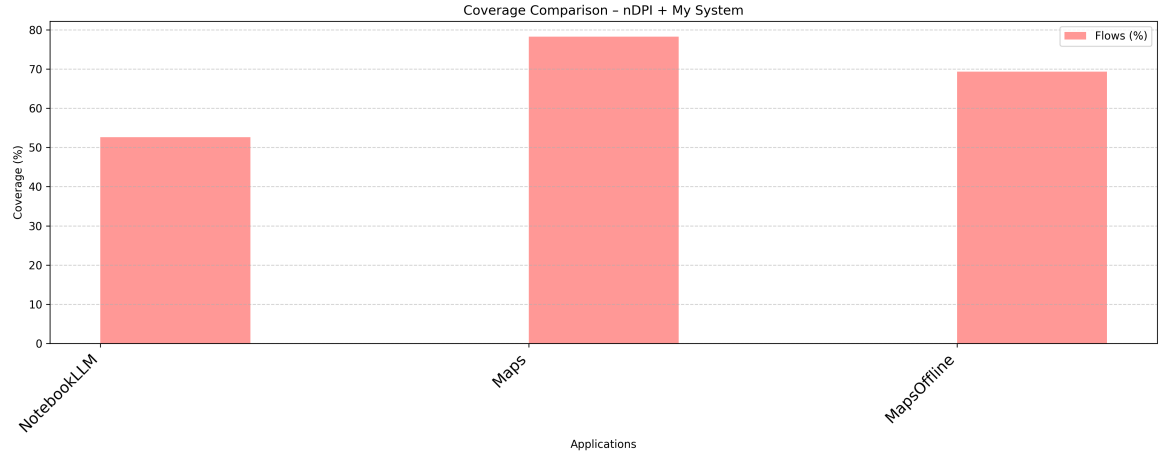

Figura 6.12: Sottoprotocolli

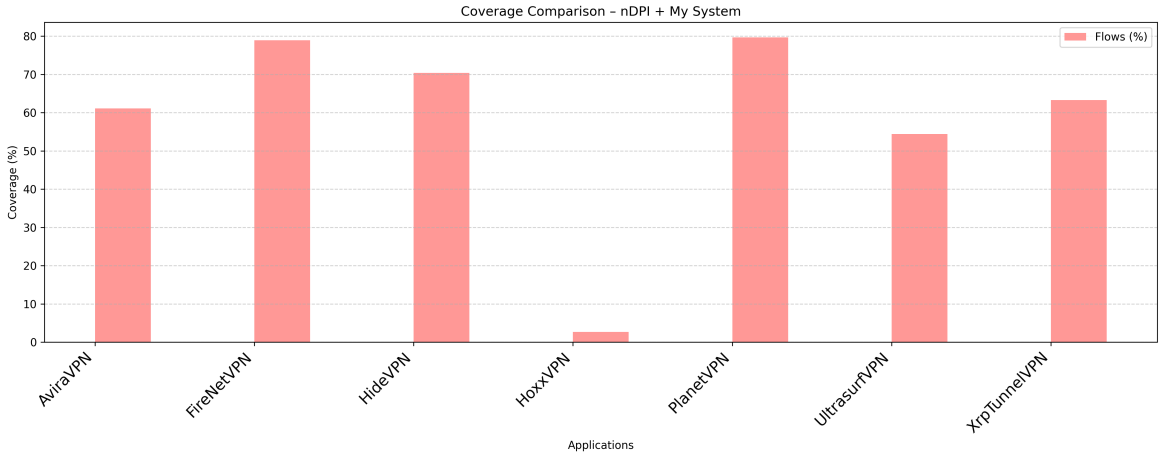

Figura 6.13: Applicazioni VPN

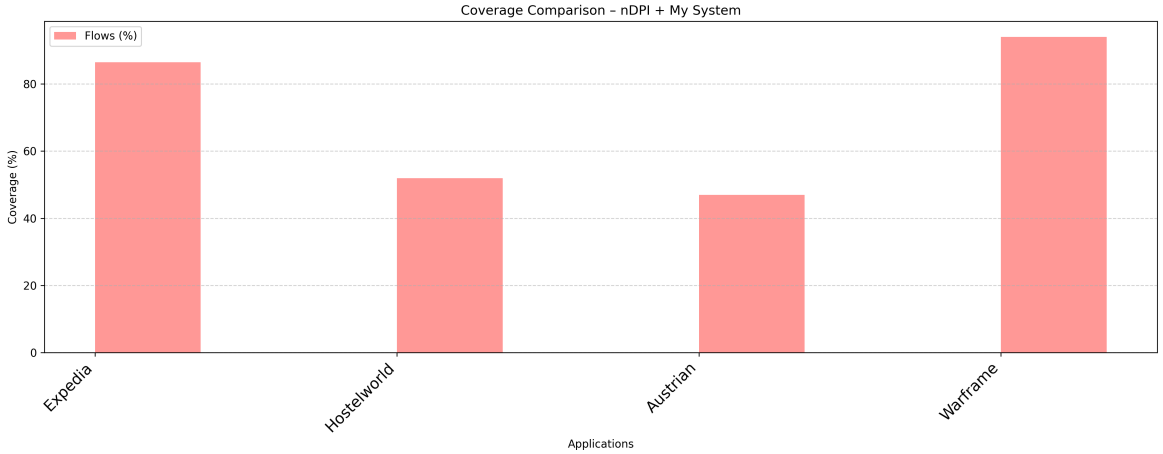

Figura 6.14: Servizi web

Per rendere più chiaro il confronto tra i diversi approcci, è utile affiancare i risultati dei vari sistemi (Figura 6.15 - 6.18):

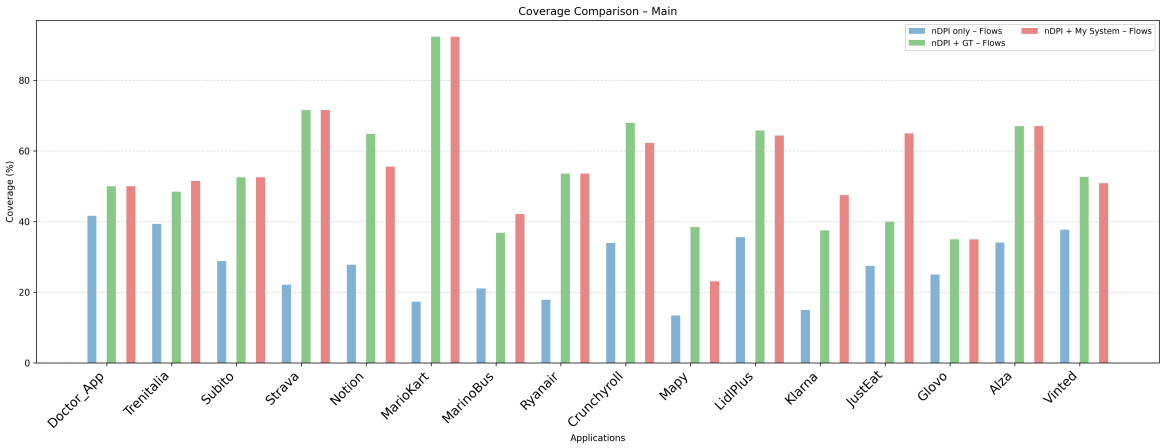

Figura 6.15: Applicazioni generali

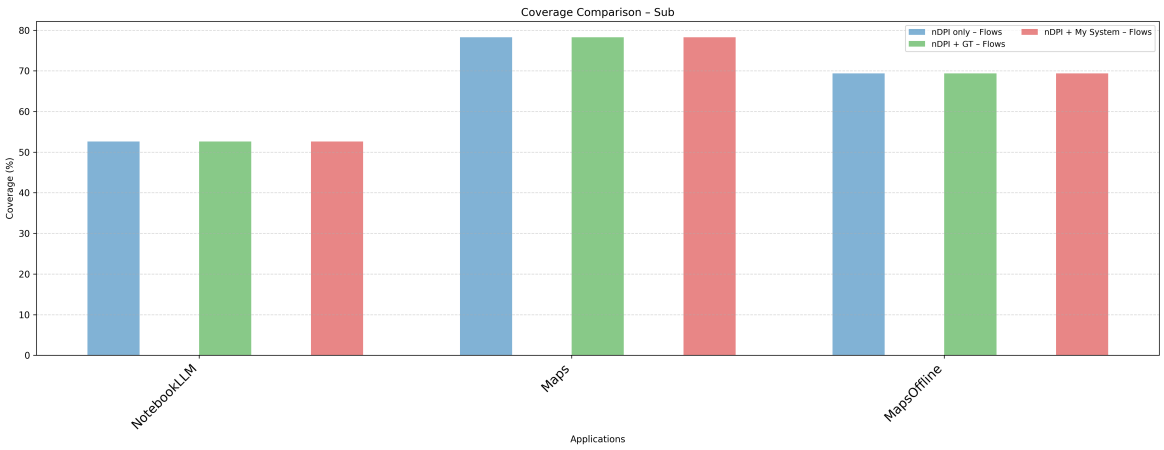

Figura 6.16: Sottoprotocolli

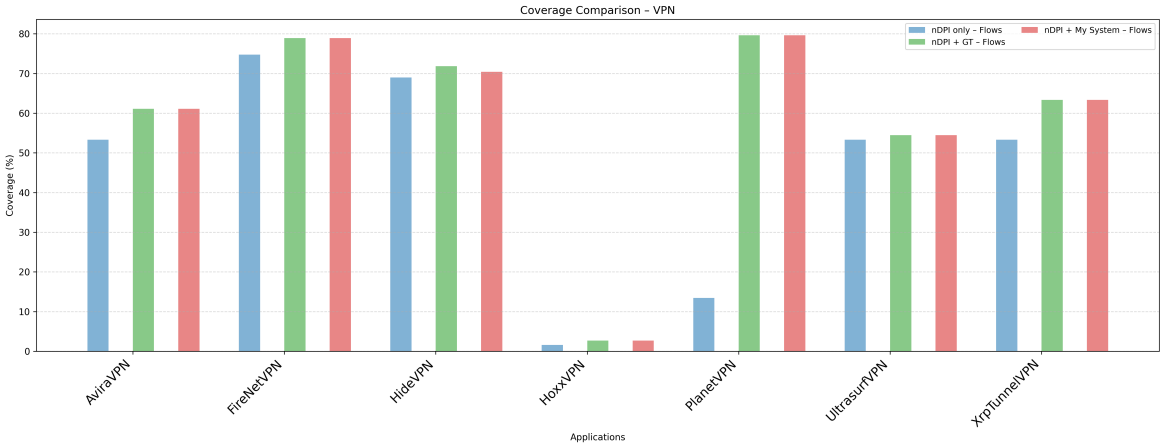

Figura 6.17: Applicazioni VPN

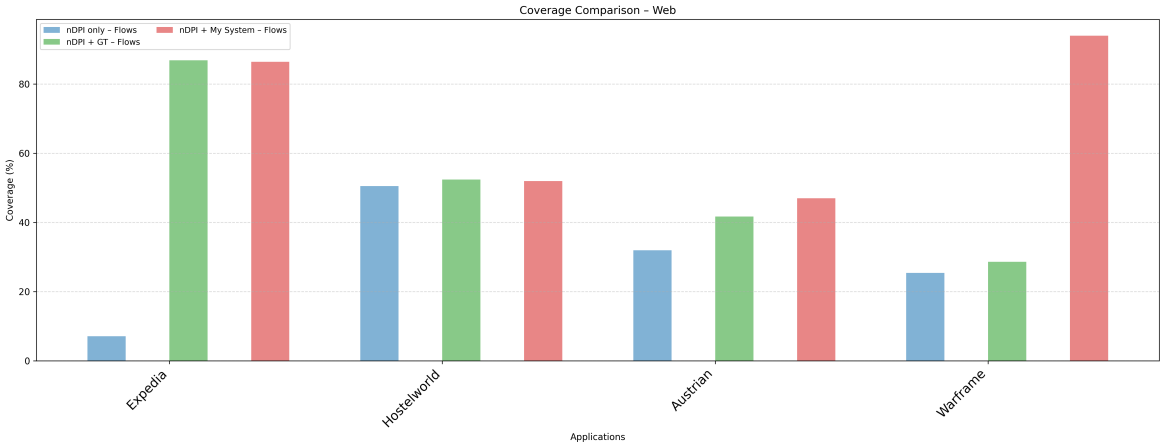

Figura 6.18: Servizi web

Come si può notare dal confronto, per le applicazioni generali, il mio sistema genera una regola che, nella maggior parte dei casi, rispecchia i risultati ideali, ossia quelli ottenuti applicando la regola *GT* al sistema *nDPI*. Per le applicazioni appartenenti alla categoria dei sottoprotocolli, ovvero *Notebook LLM*, *Maps* e *Maps Offline*, il sistema è riuscito a generare esattamente la stessa regola della *GT*. Lo stesso risultato si osserva per le VPN, mentre per le applicazioni web, *Warframe* non viene riconosciuto correttamente generando molti falsi positivi.

Questi grafici, presi e interpretati così come sono, non forniscono una valutazione completamente accurata. Infatti, anche se la barra del mio sistema coincide con quella della Ground Truth, è possibile che il risultato sia influenzato da un numero di falsi positivi e falsi negativi che si compensano a vicenda. In altre parole, potrei aver generato una regola errata che aumenta i falsi positivi, ma allo stesso tempo aver omesso la regola corretta che avrebbe ridotto i falsi negativi. Per questo motivo la matrice di confusione (Figura 6.19) aiuta a comprendere meglio i risultati ottenuti: sulla diagonale si trovano i valori corretti, ovvero i flussi correttamente identificati (**TP**) e quelli correttamente scartati (**TN**); nella parte in alto a destra sono presenti i **falsi positivi** (**FP**

), ossia i flussi classificati erroneamente come appartenenti a un'applicazione, mentre in basso a sinistra i **falsi negativi** (**FP**), cioè i flussi appartenenti all'applicazione ma non riconosciuti come tali.

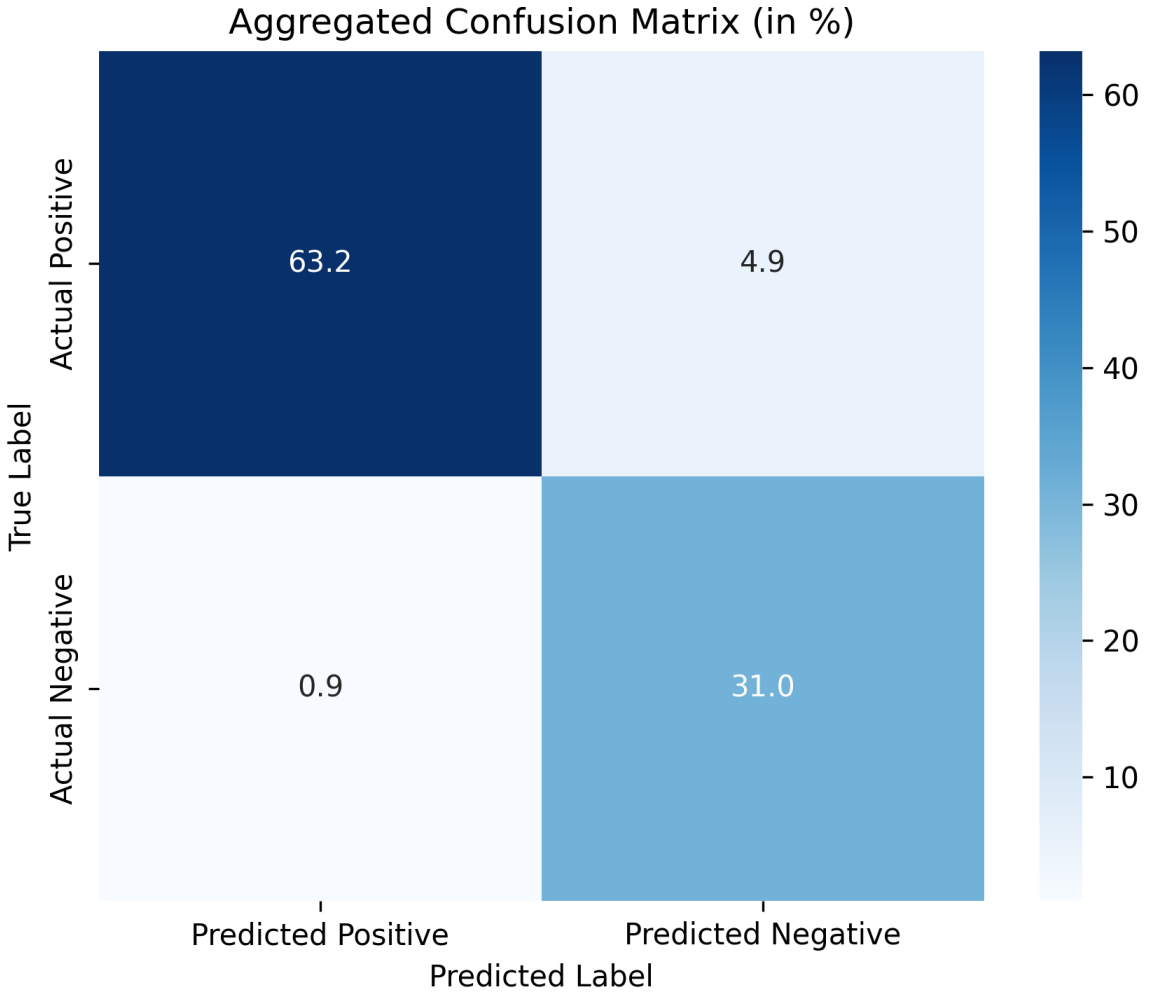

Figura 6.19: Matrice di confusione

A partire da questa matrice, sono state calcolate le metriche (Figura 6.20) di **accuratezza**, **precisione**, **sensibilità** e **F1-score**, ottenendo un valore di **F1 pari al 95.6%**, che conferma l'elevata affidabilità del sistema.

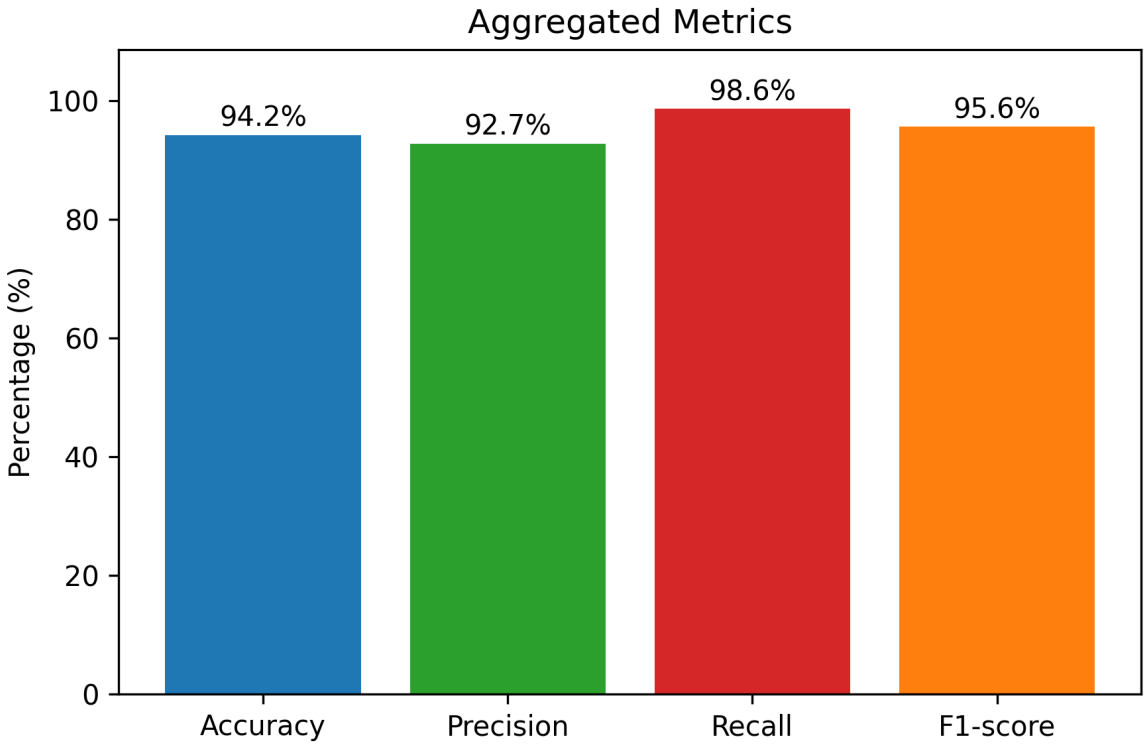

Figura 6.20: Metriche di valutazione

6.5 Risultati caso B

Per l'ultimo caso, impostiamo la variabile **TOP_PERCENT** al 20%, rispetto al 5% utilizzato nei due casi precedenti. Ricordiamo che questa variabile rappresenta la soglia che permette di selezionare solo i gruppi di flussi, `SNI`, `JA4` e certificati che superano una percentuale minima di traffico. Possiamo quindi aspettarci che, aumentando questa soglia, il sistema produca risultati più precisi, ma in numero inferiore. In altre parole, la soglia è direttamente collegata alla *precisione* del sistema: incrementandola, diminuiscono drasticamente i FP *falsi positivi*, ma aumenta anche il numero di FN *falsi negativi* generati. Mostriamo di seguito i grafici ottenuti (Figura 6.21 - 6.24 con relativa matrice di confusione 6.25 e metrice 6.26):

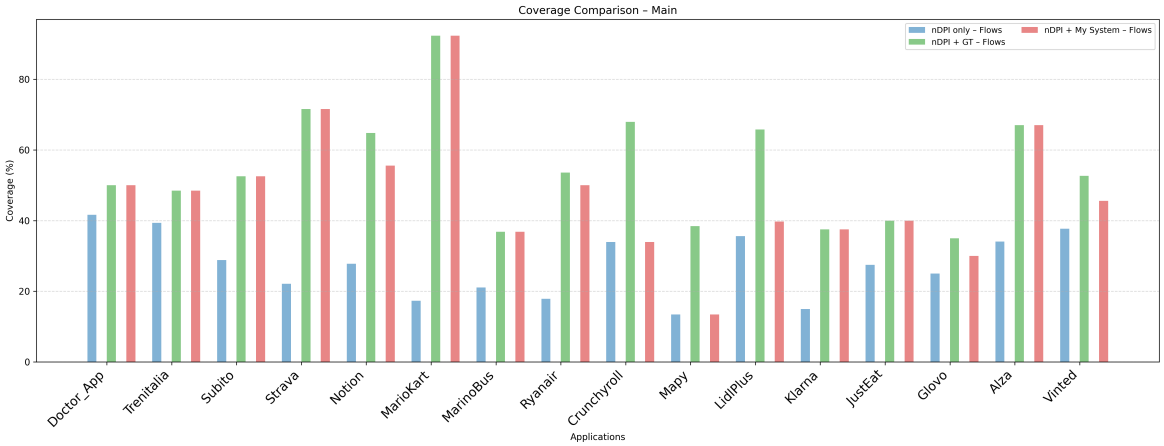

Figura 6.21: Applicazioni generali

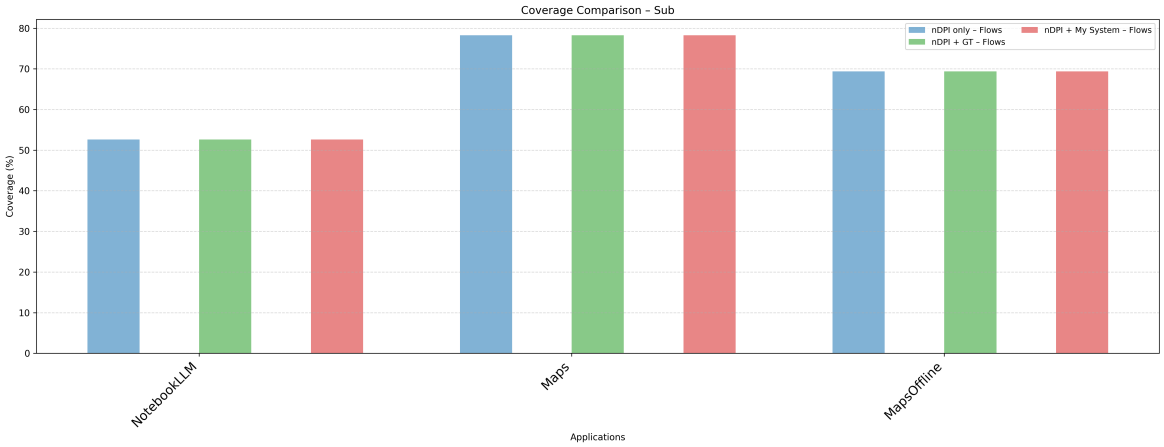

Figura 6.22: Sottoprotocolli

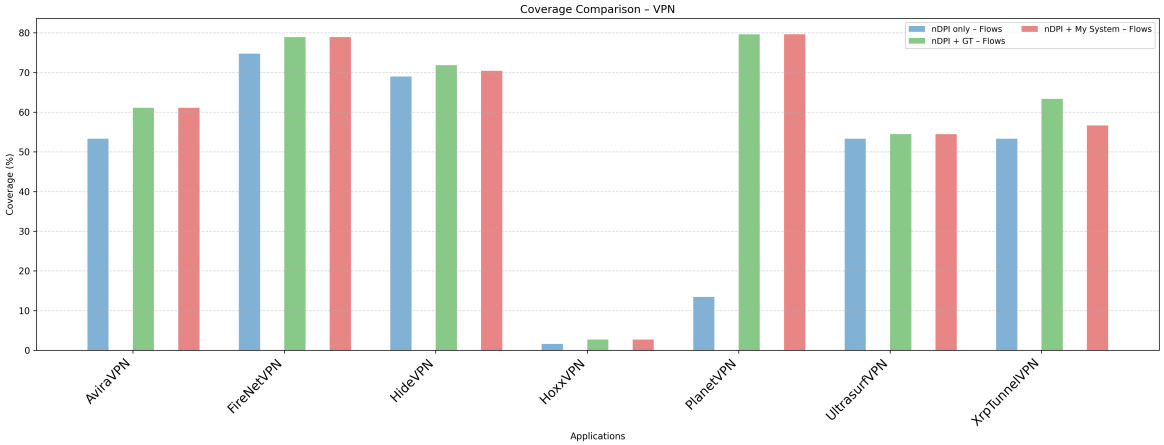

Figura 6.23: Applicazioni VPN

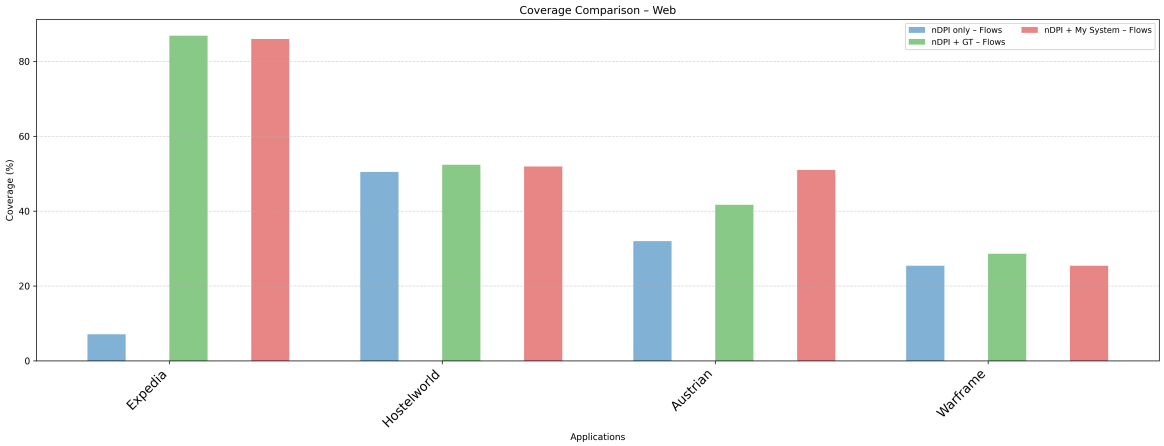

Figura 6.24: Servizi web

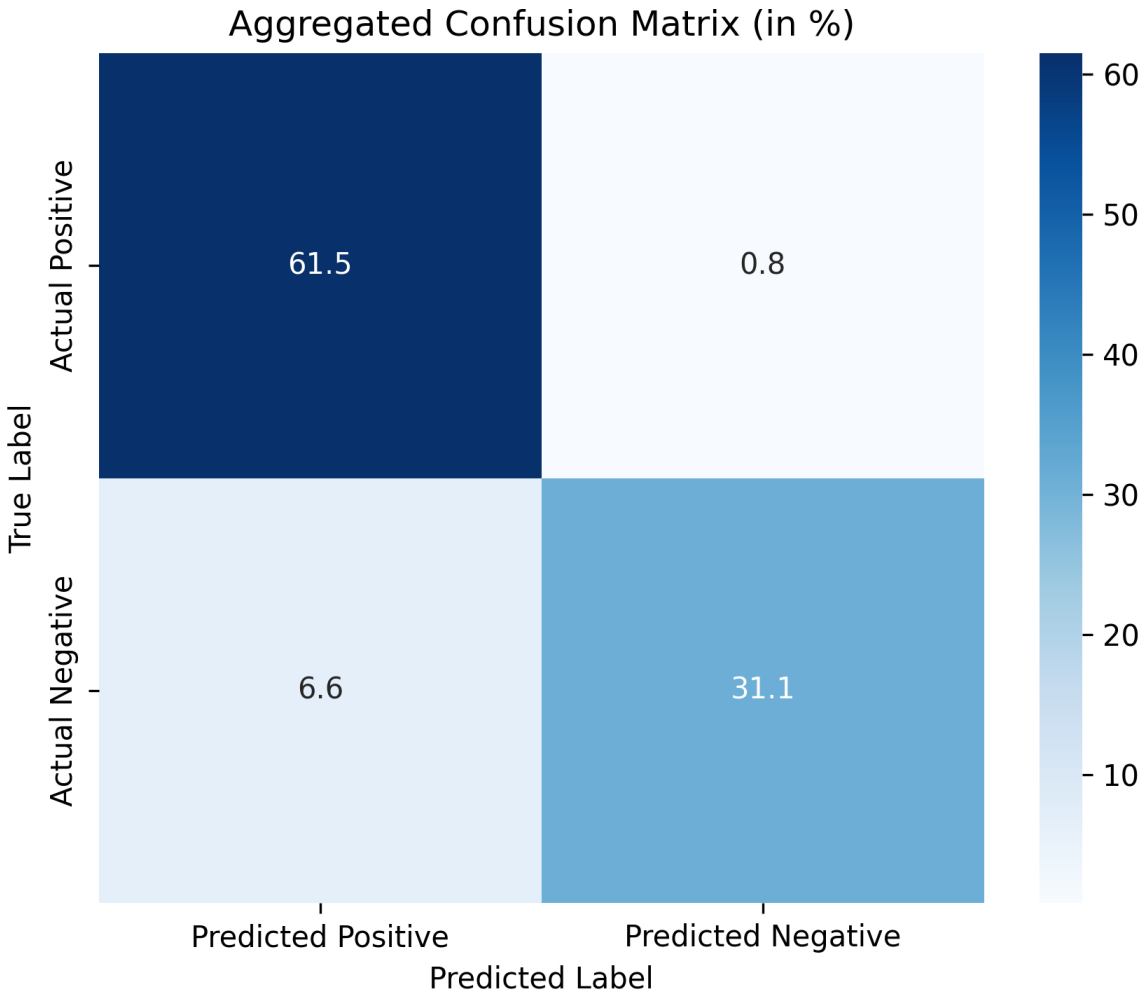

Figura 6.25: Matrice di confusione

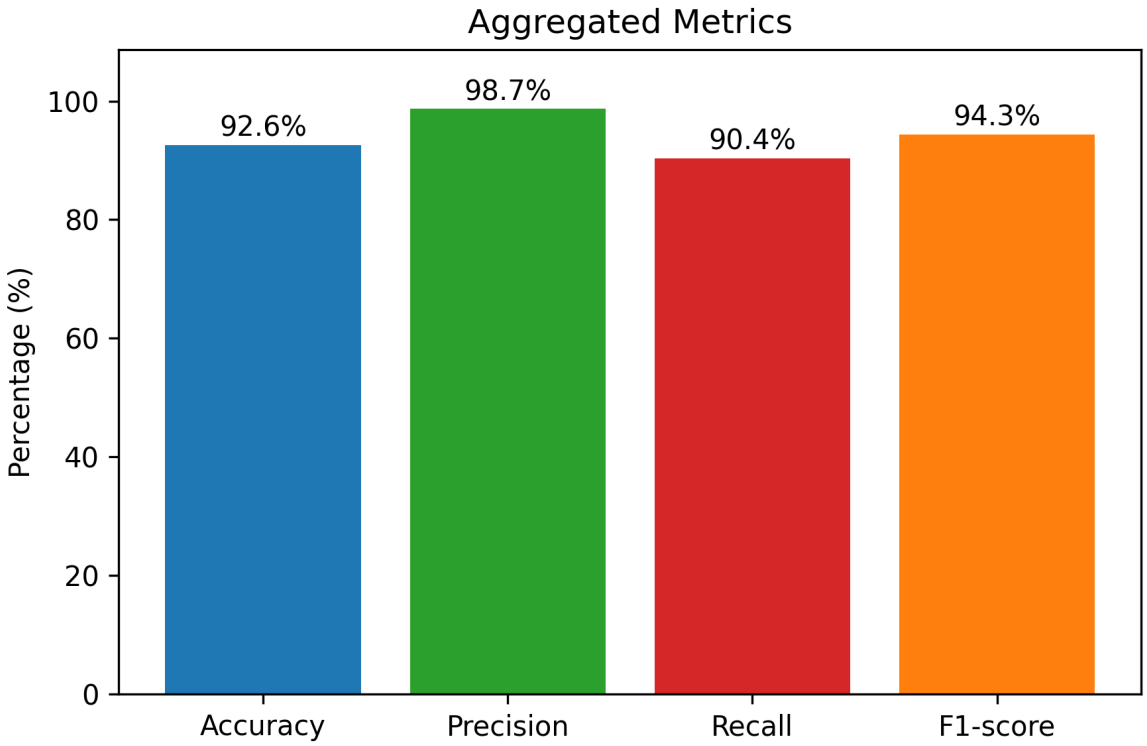

Figura 6.26: Metriche di valutazione

Come previsto, il sistema mostra un incremento dei *falsi negativi* e una drastica riduzione dei *falsi positivi*. Tuttavia, pur aumentando la *precisione*, si osserva una diminuzione sia della *sensibilità* che dell'*accuratezza* complessiva. Per questo motivo, è consigliabile mantenere questa soglia bassa, idealmente non superiore al 10%.

Capitolo 7

Conclusioni

In questo lavoro ho progettato, implementato e validato un sistema in grado di analizzare il traffico cifrato e generare automaticamente regole interpretabili per l'identificazione di applicazioni e protocolli. Il sistema combina le funzionalità di *nDPI* per protocolli custom con un parser che filtra i flussi irrilevanti e aggrega le informazioni più significative. L'obiettivo principale è migliorare la copertura e l'affidabilità del riconoscimento delle applicazioni, offrendo agli operatori di rete uno strumento efficiente e pratico per l'analisi dei flussi cifrati.

Il sistema è stato testato su un dataset di 30 applicazioni, comprendente applicazioni mobili, VPN e servizi web. Le applicazioni sono state selezionate in modo casuale da utenti reali per garantire un contesto realistico. Le catture inoltre, sono state effettuate una sola volta per ogni applicazione, sia in ambiente mobile con *PcapAndroid* sia in ambiente desktop con *Wireshark*, in sistemi non isolati, proprio per un uso reale e futuro del sistema.

Sono stati definiti due casi principali di test con parametri configurabili:

- **Caso A**: comportamento standard, con `SHOW_NDPI_PROTOCOL` variabile e `TOP_PERCENT = 5%`;
- **Caso B**: soglia `TOP_PERCENT` aumentata al 20%, per analizzare l'effetto della selezione più restrittiva dei flussi rilevanti.

Dall'analisi dei risultati emergono alcune considerazioni chiave:

- Nel **Caso A**, il sistema ha generato regole che si avvicinano molto alle percentuali di coverage ideali ottenute combinando *nDPI* con la *Grand Truth*, ottenendo un F1-score pari al 95.6%, con un buon equilibrio tra falsi positivi e falsi negativi.
- Nel **Caso B**, aumentando `TOP_PERCENT` al 20%, si osserva un netto calo dei falsi positivi ma un aumento significativo dei falsi negativi, con conseguente riduzione della sensibilità e dell'accuratezza complessiva. Questo conferma che la soglia deve essere mantenuta relativamente bassa (non superiore al 10%) per bilanciare precisione e recall, ottenendo un F1-score del 94.3%.

In conclusione, il sistema sviluppato mostra un buon potenziale per l'identificazione automatica di applicazioni e protocolli. I risultati confermano l'importanza della scelta dei parametri configurabili e della conoscenza preliminare delle applicazioni da monitorare. Il sistema quindi, rappresenta con ottimi risultati, un metodo plug-and-play per generare regole in grado di identificare un'applicazione ed aiutare l'operatore di rete in questo.

Le possibili estensioni future del lavoro includono:

- ulteriori test su un numero maggiore di applicazioni, per aumentarne la robustezza;
- integrazione con tecniche di fingerprinting avanzate, per migliorare ulteriormente l'accuratezza;
- ampliamento dei protocolli analizzati, estendendo l'analisi oltre HTTPS, TLS e QUIC.

Bibliografia

[1] Iman Akbari, Mohammad A. Salahuddin, Noura Limam, Raouf Boutaba, Bertrand Mathieu, Stephanie Moteau, Stephane Tuffin, and Leni Aniva. Traffic classification in an increasingly encrypted web. *Communications of the ACM Research and Advances: Artificial Intelligence and Machine Learning Research Highlights*, 2022.

[2] Blake Anderson and David A. McGrew. Accurate tls fingerprinting using destination context and knowledge bases. *CoRR*, abs/2009.01939, 2020.

[3] Blake Anderson, Subharthi Paul, and David McGrew. Deciphering malware's use of tls (without decryption). *Journal of Computer Virology and Hacking Techniques*, 2018.

[4] Ryo Asaoka, Yuto Soma, Hiroaki Yamauchi, Akihiro Nakao, Masato Oguchi, Saneyasu Yamaguchi, and Aki Kobayashi. Service identification of tls flows based on handshake analysis. *Journal of Information Processing*, 31:131–142, 2023.

[5] Internet Assigned Numbers Authority. Service name and transport protocol port number registry. *IANA — Internet Assigned Numbers Authority*, 2022. Available at: https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml.

[6] O. Barut, R. Zhu, Y. Luo, and T. Zhang. Tls encrypted application classification using machine learning with flow feature engineering. pages 32–41, 2020.

[7] D. Benjamin. Applying generate random extensions and sustain extensibility (grease) to tls extensibility. *IETF RFC 8701*, January 2020.

[8] R. Bortolameotti, A. Peter, M. H. Everts, and D. Bolzoni. Indicators of malicious ssl connections. pages 162–175, 2015.

[9] I. Burgetová, O. Ryšavy, and P. Matoušek. Towards identification of network applications in encrypted traffic. *Proceedings of the Cyber Security in Networking Conference*, pages 213–221, 2024.

[10] Luca Deri, Maurizio Martinelli, Tomasz Bujlow, and Alfredo Cardigliano. ndpi: Open-source high-speed deep packet inspection. pages 617–622, 2014.

[11] T. Dierks and E. Rescorla. The transport layer security (tls) protocol version 1.2. *IETF RFC 5246*, August 2008.

[12] R. Ding and W. Li. A hybrid method for service identification of ssl/tls encrypted traffic. pages 250–253, 2016.

[13] G. Draper-Gil, A. Habibi Lashkari, M. S. I. Mamun, and A. A. Ghorbani. Characterization of encrypted and vpn traffic using time-related features. pages 407–414, 2016.

[14] D. Eastlake. Transport layer security (tls) extensions: Extension definitions. *IETF RFC 6066*, January 2011.

[15] Perry B. Gentry. What is a vpn? *Information Security Technical Report*, 6(1):15, 2001.

[16] P. Gigis, M. Calder, L. Manassakis, G. Nomikos, V. Kotronis, X. Dimitropoulos, E. Katz-Bassett, and G. Smaragdakis. Seven years in the life of hypergiants' off-nets. pages 516–533, 2021.

[17] Jorge Luis Guerra, Carlos Catania, and Eduardo Veas. Datasets are not enough: Challenges in labeling network traffic. *Computer & Security*, 2022. Preprint.

[18] Michelina Hanlon, Gerry Wan, Anna Ascheman, and Zakir Durumeric. Detecting vpn traffic through encapsulated tcp behavior. *Free and Open Communications on the Internet*, 2:77–82, 2024.

[19] S.-M. Kim, Y.-H. Goo, M.-S. Kim, S.-G. Choi, and M.-J. Choi. A method for service identification of ssl/tls encrypted traffic with the relation of session id and server ip. pages 487–490, 2015.

[20] Sicai Lv, Chao Wang, Zibo Wang, Shuo Wang, Bailing Wang, and Yongzheng Zhang. Aae-dsvdd: A one-class classification model for vpn traffic identification. *Computer Networks*, 236, 2023.

[21] Aniss Maghsoudlou, Lukas Vermeulen, Ingmar Poese, and Oliver Gasser. Characterizing the vpn ecosystem in the wild. 2023.

[22] Shane Miller, Kevin Curran, and Tom Lunney. Multilayer perceptron neural network for detection of encrypted vpn network traffic. 2018.

[23] A. W. Moore and K. Papagianaki. Toward the accurate identification of network applications. *Passive and Active Network Measurement*, 3431:41–54, 2005.

[24] A. Nakao and P. Du. Toward in-network deep machine learning for identifying mobile applications and enabling application-specific network slicing. *IEICE Transactions on Communications*, E101-B(7):1519–1529, 2018.

[25] Sanghak Oh, Minwook Lee, Hyunwoo Lee, Elisa Bertino, and Hyoungshick Kim. Appsniffer: Towards robust mobile app fingerprinting against vpn. pages 2318–2328, 2023.

[26] Jee-Tae Park, Min-Seong Lee, Ui-Jun Baek, Jeong-Woo Choi, Chang-Yui Shin, and Myung-Sup Kim. Network user action detection based on psd signature through encrypted traffic analysis. 2023.

[27] E. Rescorla. The transport layer security (tls) protocol version 1.3. *IETF RFC 8446*, August 2018.

[28] A. Langley S. Friedl, A. Popov and E. Stephan. Transport layer security (tls) application-layer protocol negotiation extension. *IETF RFC 7301*, July 2014.

[29] W. M. Shbair, T. Cholez, J. François, and I. Chrisment. A survey of https traffic and services identification approaches. *arXiv preprint arXiv:2008.08339*, 2020.

[30] W. M. Shbair, T. Cholez, A. Goichot, and I. Chrisment. Efficiently bypassing sni-based https filtering. pages 990–995, 2015.

[31] Wazen M. Shbair, Thibault Cholez, Jerome Francois, and Isabelle Chrisment. A multi-level framework to identify https services. 2020.

[32] P. Velan, M. Čermák, P. Čeleda, and M. Drašar. A survey of methods for encrypted traffic classification and analysis. *International Journal of Network Management*, 25(5):355–374, 2015.

[33] Wei Wang, Ming Zhu, Jinlin Wang, Xuewen Zeng, and Zhongzhen Yang. End-to-end encrypted traffic classification with one-dimensional convolution neural networks. 2017.

[34] Y. Yang, C. Kang, G. Gou, Z. Li, and G. Xiong. Tls/ssl encrypted traffic classification with autoencoder and convolutional neural network. pages 362–369, 2018.

[35] Qianqian Zhang and Chi-Jiun Su. Application-layer characterization and traffic analysis for encrypted quic transport protocol. *arXiv preprint arXiv:2310.10676v1*, 2023.

[36] Di Zhao, Bo Jiang, Song Liu, Susu Cui, Meng Shen, Dongqi Han, Xingmao Guan, and Zhigang Lu. Language of network: A generative pre-trained model for encrypted traffic comprehension. *Journal of LaTeX Class Files*, 14(8), 2021.

[37] Zhuang Zou, Jingguo Ge, Hongbo Zheng, Yulei Wu, Chunjing Han, and Zhongjiang Yao. Encrypted traffic classification with a convolutional long short-term memory neural network. pages 362–369, 2018.

Appendice A

Appendice

Codice Sorgente e Risorse

Il codice sorgente sviluppato, comprensivo di script, test e documentazione, è disponibile al seguente indirizzo:
Repository GitHub della tesi:

https://github.com/Nicofontanarosa/Thesis

Il progetto *nDPI*, utilizzato come riferimento tecnico durante il lavoro di tesi, è disponibile al seguente indirizzo:
Repository GitHub nDPI:

https://github.com/ntop/nDPI