UNIVERSITÀ DI PISA

Corso di Laurea in Informatica

Progetto di Tesi

Analisi e Mitigazione Di Scan e Abusi Nel Traffico Web

Relatore:
Luca Deri

Candidato:
Andrea Luchetti

ANNO ACCADEMICO 2024/2025

Abstract

La tesi nasce da un'esigenza concreta rilevata all'interno dell'azienda Genesys Informatica s.r.l. che commercializza servizi di web hosting con il brand Hosting Solutions. L'azienda, oltre che all'hosting, offre servizi come e il mantenimento di siti internet, e per questo motivo osserva quotidianamente un flusso elevato di connessioni da parte di client che visitano i siti web ospitati. Durante il monitoraggio del traffico, è emersa una significativa presenza di client automatici, come crawler e bot, che eseguono scansioni sistematiche dei siti. Sebbene alcuni di essi operino in modo legittimo, rispettando standard e limiti previsti, altri adottano comportamenti aggressivi che compromettono la qualità del servizio, generando un carico anomalo di traffico che porta a saturare le risorse, negato l'accesso agli utenti umani o legittimi.

L'obiettivo di questa tesi è stato confrontare vari strumenti per il riconoscimento e l'identificazione di Crawler, una volta stabilito l'algoritmo è stata implementata la propria versione prendendo come riferimento i vari studi condotti. Lo scopo del codice è quello di riconoscere il traffico generato da Crawler automatizzati e successivamente mitigare tale traffico non desiderato, cercando di preservare al contempo l'accesso legittimo. A tal fine, è stato condotto uno studio approfondito sui log di accesso del server web Apache, con particolare attenzione alla frequenza di connessioni, ai pattern di navigazione e ad altri parametri utili alla classificazione del traffico. L'analisi ha permesso di sviluppare una metodologia, basata su riconoscimento euristico, in grado di distinguere tra richieste legittime e potenzialmente dannose, implementando una strategia di traffic shaping e conseguente limitazione basata sulla categorizzazione dei client.

Tutti gli indirizzi IP o domini inseriti all'interno della tesi sono stati modificati per mantenere la privacy dei clienti e dell'azienda (di cui sopra citata) che ha fornito i dati.

Indice

Elenco delle figure

Listings

Introduzione

Negli ultimi anni, il traffico su Internet è cresciuto significativamente, parallelamente all'aumento del numero di utenti e organizzazioni che pubblicano contenuti sul Web. A questo fenomeno si è affiancata la diffusione di sistemi automatici, come i *crawler*, spesso basati su tecnologie di intelligenza artificiale. Questi software sono progettati per esplorare e indicizzare grandi porzioni della rete, raccogliendo automaticamente i contenuti testuali presenti nelle pagine web, al fine di costruire archivi consultabili per l'addestramento di modelli AI o per i motori di ricerca.

Sebbene i *crawler* svolgano un ruolo utile nell'ecosistema di Internet, il loro comportamento può talvolta generare un carico eccessivo sulle infrastrutture. In particolare, un numero elevato di richieste in un breve intervallo di tempo può compromettere le risorse computazionali dei server, causando un degrado delle prestazioni o, nei casi più estremi, l'interruzione del servizio.

Come riportato dall'indagine condotta dal GLAM-E Lab (2025) [1], non solo gli enti privati subiscono questi fenomeni bensì molte istituzioni come Gallerie, musei, archivi o biblioteche hanno rilevato questo fenomeno, che ha portato alla saturazione delle risorse.

La presente tesi nasce dall'osservazione diretta di questo tipo di problema in un'infrastruttura reale, basata sul server web Apache. È stato infatti rilevato un sovraccarico causato da client automatici – legittimi o meno – in grado di monopolizzare la banda disponibile, a scapito degli utenti umani.

Sono state prese in considerazione diverse implementazioni come l'introduzione di un *reverse proxy*, nello specifico `HAProxy`, posto davanti al server Apache, moduli di apache, utilizzo di firewall e tante altre soluzioni. Tra le varie proposte è stata scelta quella del bridge, il quale agisce come punto di ingresso al traffico web, riceve le richieste in arrivo e, sulla base di regole definite (ACL, Access Control List), classifica il traffico in quattro principali code:

- **Legittimi**

- **Crawler**
- **Malevoli**
- **Altro**

Ad ogni categoria viene assegnato un limite massimo di traffico espresso in pacchetti al secondo. L'obiettivo non è quello di bloccare le richieste, ma di regolarne il numero di connessioni in base alla categoria, implementando una forma di *shaping* del traffico. Ad esempio, si può assegnare:

- 80% della banda disponibile al traffico **legittimo**
- 9% ai **crawler**
- 7% al traffico **non identificato**
- 4% al traffico **malevolo**

Questo approccio si basa sul principio di non escludere completamente nessuna classe di traffico, per evitare falsi positivi, ma di limitarne la capacità di saturare la rete.

Tuttavia, l'adozione di HAProxy in infrastrutture esistenti presenta alcune criticità: richiede modifiche significative nella configurazione dei server, con impatti sia tecnici che organizzativi, specialmente quando si gestiscono molti clienti o domini.

Per questo motivo, si propone un'architettura alternativa basata su un *bridge trasparente*, implementato tramite una macchina Linux con due interfacce di rete. Questo bridge agisce come punto di passaggio tra la rete interna e Internet, inoltrando i pacchetti tra le interfacce senza modificare la logica del server web esistente. Il sistema di analisi istruisce dinamicamente il bridge tramite chiamate HTTP POST per applicare filtri in base all'indirizzo IP o alla classe di traffico rilevata, con azioni che includono: `DROP`, limitazione del traffico, oppure nessuna restrizione.

Questa tesi si inserisce in un contesto applicativo reale, con l'intento di contribuire alla protezione e all'efficienza delle infrastrutture web moderne.

Capitolo 1

Fondamenti Teorici

In questo capitolo verranno discussi in maniera teorica gli argomenti di cui tratta la tesi. Non verranno trattati in maniera verticale ma verranno analizzati orizzontalmente, cercando di coprire il più possibile dei concetti trattati.

1.1 Web scraping e web crawling

Come accennato precedentemente i bot possono categorizzarsi in legittimi o illegittimi, ma tale categorizzazione non è propriamente corretta poichè accorpa insieme due categorie di tecniche che permettono di automatizzare la raccolta e l'analisi di informazioni dal web e ritengo sia importante fare una distinzione:

- `Web Crawling`,
- `Web Scraping`

1.1.1 Web Crawling

Immaginiamo il web come un immenso labirinto di pagine interconnesse. Il web crawling, noto anche come web spidering, si avvale di software automatizzati, i crawler, che esplorano questo labirinto seguendo i collegamenti ipertestuali tra le pagine. Come un ragno tesse la sua ragnatela, il crawler mappa il web, scoprendo nuove pagine e ampliando la sua conoscenza della rete [2].

1.1.1.1 Crawler web

I *crawler* web, anche detti *spider* o *bot*, sono software automatici progettati per navigare sistematicamente il Web [3], raccogliendo contenuti da pagine e siti internet. Il

loro scopo principale è costruire o aggiornare indici di ricerca, come avviene per i motori di ricerca tradizionali (es. Googlebot, Bingbot), o per alimentare database utilizzati nell'addestramento di modelli di intelligenza artificiale.

Un **crawler legittimo** rispetta generalmente il file `robots.txt` di un sito, limita la velocità delle sue richieste (rate limiting) e si identifica chiaramente attraverso l'header `User-Agent`. Questi bot sono generalmente riconoscibili e benigni.

Al contrario, esistono anche **crawler malevoli** o non collaborativi, che:

- ignorano le direttive di `robots.txt`,
- effettuano richieste ad alta frequenza, causando carico eccessivo,
- mascherano il proprio User-Agent o usano indirizzi IP dinamici,
- sono utilizzati per scraping aggressivo, spam, analisi di vulnerabilità o furto di dati.

La distinzione tra crawler legittimi e malevoli può non essere immediata e spesso richiede un'analisi comportamentale basata su parametri come frequenza di accesso, URL richiesti, header HTTP, origine geografica, ASN, e pattern temporali.

Molte aziende che possiedono un sito web impediscono ai crawler di accedere alle proprie pagine per i seguenti motivi:

1. I web crawler possono compromettere la disponibilità dei web server;
2. I contenuti dei web server sono considerati proprietà intellettuale delle aziende;
3. Il sistema di determinazione dei prezzi di alcuni prodotti specifici potrebbe aiutare le aziende a mantenere la posizione di leadership sul mercato (ad esempio, la fluttuazione dei prezzi dei biglietti aerei e il prezzo più basso degli articoli più venduti).

Se aziende concorrenti o altri operatori rubano e utilizzano i dati per i propri scopi, ciò avrà un impatto negativo sull'azienda e, a lungo termine, rappresenterà una minaccia per il suo modello di business [4].

1.1.2 Web Scraping

Mentre il web crawling si concentra sulla scoperta e mappatura del web, il web scraping si focalizza sull'estrazione di dati specifici da pagine web. Come un minatore che cerca gemme preziose, il web scraper estrae informazioni rilevanti da siti web, organizzandole in formati strutturati come database o fogli di calcolo [3].

Lo scraper solitamente esegue il seguente processo:

- Identificazione del sito web;
- Richiesta della pagina web;
- Analisi del codice HTML;
- Estrazione dei dati;
- Formattazione e salvataggio;

Il web crawling viene utilizzato per il monitoraggio dei prezzi, ricerche di mercato, Creazione di database, Analisi dei dati.

1.2 Traffic shaping

Il *traffic shaping*, o modellamento del traffico, è una tecnica di gestione della rete che consiste nel controllare il flusso dei pacchetti al fine di:

- evitare congestioni,
- garantire qualità del servizio (QoS),
- limitare il consumo di banda da parte di client o applicazioni specifiche.

In ambito web, lo shaping è utilizzato per regolare la quantità di traffico generato da determinati client, specialmente quando si verificano comportamenti aggressivi o non bilanciati. Anziché bloccare completamente il traffico, il sistema può rallentare la velocità con cui un client può inviare richieste, usando algoritmi di rate limiting.

L'obiettivo dello shaping, in questo progetto, è mitigare il comportamento aggressivo dei crawler senza compromettere l'accesso degli utenti legittimi.

1.3 Server Proxy

Quando esponiamo sulla rete un servizio, come un web Server HTTP/HTTPS o un mail Server si pongono sempre notevoli problemi di sicurezza, in quanto ogni servizio esposto sulla rete pubblica è potenzialmente soggetto a tentativi di intrusione e attacco. Allo scopo di aumentare la sicurezza dei sistemi molto spesso si utilizzano dei componenti software chiamati Proxy.[5].

I proxy possono essere classificati in due categorie principali:

- **Forward proxy**: agisce per conto del client, tipicamente usato per filtrare l'accesso a Internet o per motivi di privacy.
- **Reverse proxy**: è un server proxy che trasmette le informazioni da uno o più server Web e le inoltra ai browser [5].

Nel contesto di questa tesi, ci si concentra sul **reverse proxy**: come riportato nell'articolo *"Forward Proxy vs. Reverse Proxy: The Difference Explained"* [6] la differenza sostanziale tra proxy tradizionale (Forward proxy) e un Reverse proxy sta nel fatto che, il primo, viene utilizzato per proteggere i client, il secondo invece viene utilizzato per proteggere i server. Esso riceve una richiesta da un client, la inoltra a un altro server tra i tanti disponibili e restituisce al client i risultati del server che ha effettivamente elaborato la richiesta, come se fosse stato il server proxy stesso a elaborarla. Il client comunica direttamente solo con il reverse proxy server e non sa che la sua richiesta è stata effettivamente elaborata da un altro server.

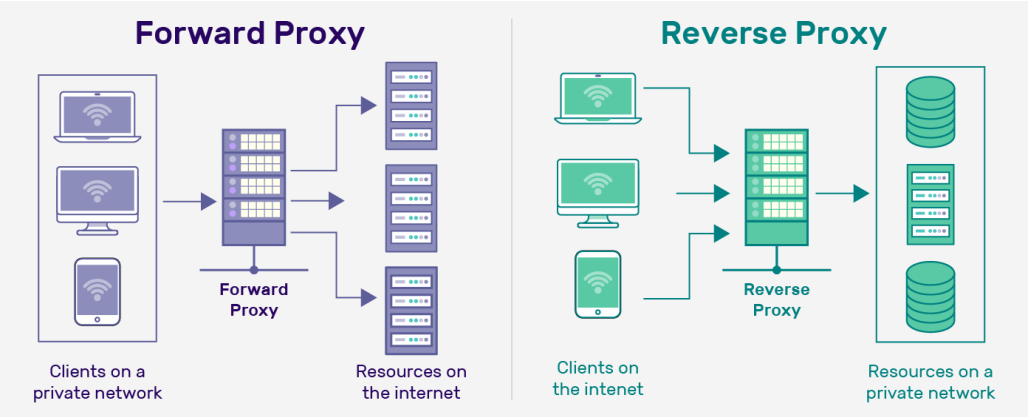

Figura 1.1: Reverse Proxy vs. Forward Proxy

1.3.1 HAProxy

HAProxy (High Availability Proxy) è un reverse proxy open source ad alte prestazioni, ampiamente utilizzato per bilanciare il carico e gestire grandi volumi di connessioni TCP e HTTP. Si tratta quindi di un bilanciatore di carico, grazie a cui è possibile smistare verso vari server le connessioni in entrata [7]. Supporta meccanismi di:

- **ACL** (*Access Control List*) per filtrare le richieste;
- **stick-tables** per il tracciamento degli indirizzi IP;
- **rate limiting e Traffic shaping**, per limitare il flusso di dati;
- **logging e metriche dettagliate**, per gestire eventuali errori;

Grazie alla sua configurabilità e scalabilità, HAProxy è spesso utilizzato come punto di ingresso per implementare logiche di sicurezza, mitigazione DDoS e classificazione del traffico.

1.4 Bridge

Il bridge (letteralmente ponte) è un dispositivo di rete che si colloca al livello datalink del modello ISO/OSI che connette e gestisce il flusso di dati tra due rete locali o tra due segmenti di una stessa rete locale, come illustrato nella figura 1.2 [8].

Come spiegato in [9], i bridge svolgono tre funzioni principali:

1. Creare una tabella di bridging per tenere traccia dei dispositivi su ciascun segmento;
2. Filtrare i pacchetti in base ai loro indirizzi MAC, ovvero inoltrare i pacchetti il cui indirizzo MAC di destinazione si trova su un segmento di rete diverso da quello di origine e rimuovere i pacchetti che non devono essere inoltrati ad altri segmenti;
3. Suddividere una singola rete in più domini di collisione, riducendo così il numero di collisioni su ciascun segmento e aumentandone efficacemente la larghezza di banda.

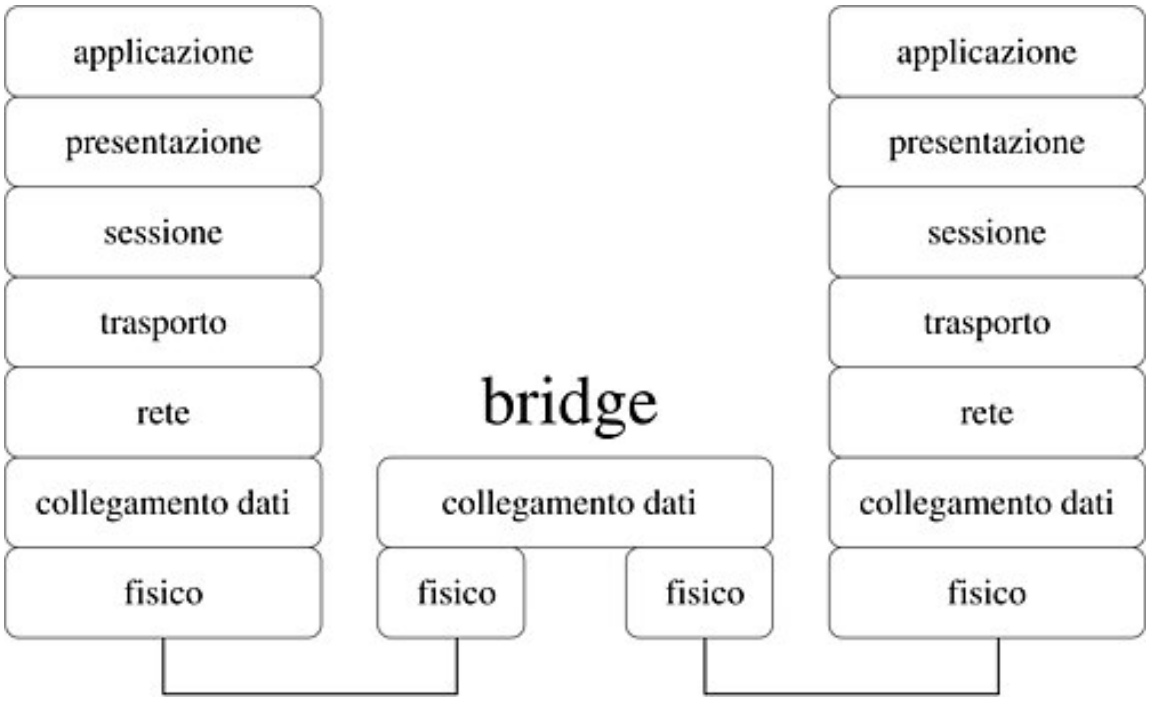

Figura 1.2: Schema Bridge

1.5 Endpoint

Un endpoint API è un luogo digitale in cui un'application programming interface (API) riceve chiamate API, note anche come richieste API, per le risorse sul suo server. E' un insieme di regole e protocolli che definiscono il modo con cui client e server possono cominucare. Gli endpoint funzionano un po' come i numeri di telefono: proprio come un

utente compone un numero di telefono per contattare una determinata persona o azienda, i client API (il software che effettua una chiamata API) forniscono un URL dell'endpoint per raggiungere una risorsa specifica. L'URL di un endpoint fornisce la posizione di una risorsa su un server API e aiuta a connettere il client API con la risorsa richiesta. In sostanza, dice al server: "La risorsa di cui ho bisogno si trova qui" [10]. Tale servizio è strutturato secondo i principi dell'architettura *RESTfull*, cioè un insieme di linee guida per la realizzazione di una "architettura di sistema"[11]:

- Identificazione delle risorse
- Utilizzo esplicito dei metodi HTTP
- Risorse autodescrittive
- Collegamenti tra risorse
- Comunicazione senza stato

Capitolo 2

Lavori Correlati

Per affrontare il problema, sono state esplorate diverse soluzioni, ognuna con vantaggi e limitazioni specifiche. La tesi si propone di esplorare diverse soluzioni che potrebbero essere spunti per miglioramenti futuri o possibili soluzioni alternative.

La gestione del traffico automatizzato indesiderato è una sfida complessa che ha portato allo sviluppo svariate soluzioni. Il panorama della sicurezza web offre alternative che spaziano dai moduli specifici per server web, ai firewall applicativi, fino a tecniche avanzate di inganno e apprendimento automatico.

2.1 Utilizzo di Apache

Apache HTTP Server, o più comunemente Apache, è un server web libero sviluppato dalla *Apache Software Foundation* [12].

È un software che realizza le funzioni di trasporto delle informazioni e di collegamento. Questo strumento ha il vantaggio di offrire funzioni di controllo per la sicurezza come quelle effettuate da un proxy.

Apache presenta una architettura **modulare**, cioè composto da più moduli, ognuno dei quali lavora indipendentemente per ogni richiesta del client [2.3]. I vari moduli sono coordinati dal **Core**, il quale interroga ogni modulo in maniera sequenziale usando i parametri di output di un modulo come parametri di input per il successivo.

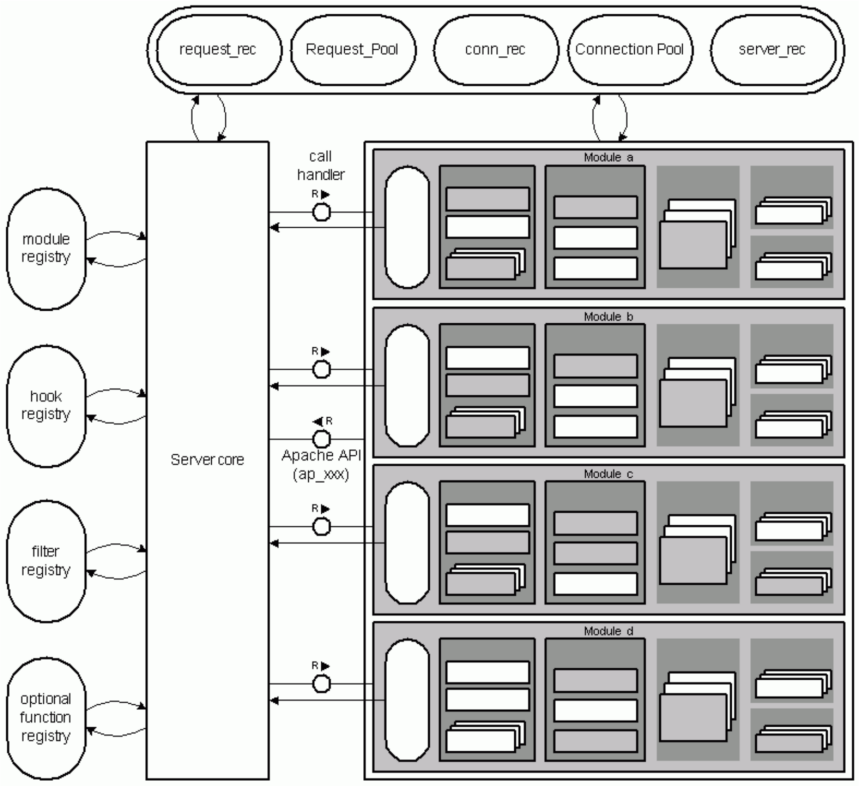

Figura 2.3: Apache Modules

Grazie alla natura di **Apache** di essere un server web libero è possibile integrare una serie di moduli per aggiungere funzionalità, nel nostro caso esistono vari moduli che hanno il compito di analizzare le richieste e filtrarle.

Di seguito vengono presentate due tipologie di moduli utili per tale scopo.

2.1.1 Mod evasive

Il modulo di apache **ModEvasive**, fornisce un'azione di mitigazione in caso di attacco HTTP Distributed Denial of Service (DDoS/DoS) o attacco brute force. È progettato per essere uno strumento di rilevamento e gestione della rete, è anche in grado di sfruttare tecniche di IPchain, firewall, router e altro ancora. **ModEvasive** attualmente segnala gli abusi tramite email e syslog [13].

La logica che utilizza questo modulo è simile all'algoritmo implementato nel progetto di riconoscimento di Crawel come verrà spiegato nel capitolo successivo. La differenza sta

nel fatto che **Mod_evasive** categorizza le connessioni per combinazione di nome utente e password. A seguito della categorizzazione il modulo salva indirizzo IP e URI in una tabella Hash, la quale viene aggiornata periodicamente con l'integrazione di liste note di indirizzi IP ritenuti malevoli. Se l'IP è presente nell'elenco degli indirizzi bloccati, restituisce un codice di errore 403 in risposta alla richiesta.

2.1.2 Mod security

Il modulo **ModSecurity** di Apache è un strumento che offre funzionalità avanzate di monitoraggio, gestione degli accessi e filtraggio delle connessioni HTTP e HTTPS in tempo reale. E' pensato per proteggere i servizi web da attacchi informatici come DOS, DDos o SQL injection. Il modulo è configurato basandosi su regole chiamate *SecRules*, che consentono di applicare regole a discrezione delle necessità dell'amministratore. Nel contesto di traffic shaping, è possibile implementare delle politiche di monitoraggio basate su contatori che vengono incrementati ogni qual volta che un indirizzo IP crea traffico anomalo. Quando il contatore raggiunge una determinata soglia viene applicata una limitazione. Questa configurazione consente di gestire picchi di traffico iniziali e di applicare limitazioni dinamiche in base al comportamento dell'utente [14].

Un esempio di questo approccio, basato su regole e sull'analisi dei log di accesso del server Apache, è stato proposto da Bas Seyyar [15] . Il metodo illustrato si concentra sul rilevamento di scansioni di vulnerabilità e attacchi come SQL Injection (SQLi) e Cross-Site Scripting (XSS). Gli autori dello studio citato sottolineano come questo approccio basato su regole sia efficiente in termini di consumo di memoria e rapido nell'esecuzione, a differenza dei modelli di machine learning che richiedono grandi quantità di dati di training e tempo. Questo studio enfatizza l'importanza di analizzare i dati passati per creare politiche di sicurezza efficaci nel riconoscimento di bot. Le regole scelte per il riconoscimento derivano da osservazioni sul comportamento dei robot web, come la richiesta del file robots.txt, un'alta percentuale di richieste 4xx, e un'elevata percentuale di referrer non assegnati.

2.2 Utilizzo di WAF o Firewall

Web Application Firewall, (WAF), e il **Network Firewall** sono due strumenti di difesa, che proteggono la rete privata dalla rete esterna (internet). Entrambi hanno lo scopo di analizzare e filtrare le connessioni in entrata, hanno però una differenza sostanziale, che risiede nel livello del modello OSI a cui operano:

- Un **Network Firewall** si concentra a livelli più bassi dello stack, come Layer 3 (di rete) e Layer 4 (di trasporto). Questa soluzione però non è pensata per proteggere direttamente un'applicazione web, bensì è uno strumento che protegge la rete da attacchi esterni;

- Un **WAF**, invece, nasce dall'esigenza di proteggere un servizio specifico, quale l'applicazione web. Per tale motivo lavora a Layer 7 (applicativo), filtrando le richieste specificatamente HTTP e HTTPS [2.4] .

La Tecnica del WAF consente di esaminare il contenuto delle richieste web e di bloccare tutte coloro che corrispondono a pattern di attacco noti o comportamenti anomali. L'utilizzo di regole di filtraggio unita alla capacità di analizzare in profondità il traffico HTTP li rendono particolarmente efficaci contro attacchi specifici per le applicazioni web, come SQLi e XSS.

Questi sistemi possono utilizzare sia approcci basati su blacklist (blocchi di indirizzi IP noti malevoli) che su whitelist (blocchi di indirizzi IP noti legittimi). L'efficacia dei WAF e dei Firewall nel bloccare e mitigare gli attacchi è legata alla qualità delle regole utilizzate e dalla capacità di adattarsi alle nuove minacce [16] [17].

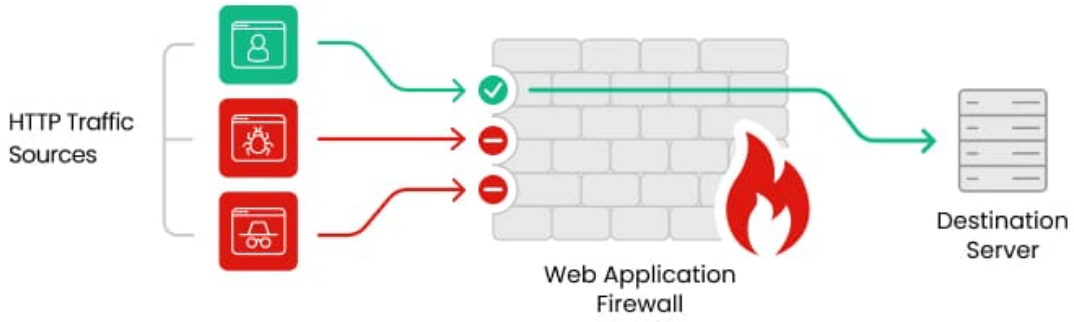

Figura 2.4: WAF

2.3 Utilizzo di CloudFlare

CloudFlare è una rete globale di server che fornisce servizi di sicurezza, prestazioni e affidabilità per siti web e applicazioni online.

CloudFlare fa parte delle **CDN**, (Content Delivery Network), cioè una rete di server tra loro collegati in maniera tale da trasmettere informazioni e contenuti Internet nel modo più veloce, sicuro e affidabile. Per raggiungere questi obiettivi, la CDN è strutturata secondo la presenza di server presso i punti di scambio tra le diverse reti, i cosiddetti IXP (Internet Exchange Point), i quali non sono altro che le sedi da cui i portali internet si connettono per offrire l'accesso reciproco al traffico generato dalle reti online [18].

Gli studi condotti negli articoli "*Somesite I Used To Crawl: Awareness, Agency and Efficacy in Protecting Content Creators From AI Crawlers*" [19] e "*Web Runner 2049: Evaluating Third-Party Anti-bot Services*" [20] creano una panoramica di sviluppo per questo strumento.

La sua posizione di intermediario gli consente di rilevare e bloccare il traffico direttamente sui propri server, senza richiedere interazione diretta con i clienti. **CloudFlare** si distingue per le sue metodologie per la mitigazione dei bot:

- **Rilevamento basato sulla reputazione IP**: Cloudflare si affida principalmente alla reputazione degli indirizzi IP, grazie e blacklist che uniscono indirizzi IP noti, basandosi su attività malevole passate, e whitelist, indirizzi IP resi noti dagli enti stessi;

- **Regole firewall**: Cloudflare sfrutta anche regole firewall basate sulle richieste dei client per prevenire comportamenti automatizzati e dannosi;

Nonostante la sua facilità d'uso e la capacità di bloccare attivamente i contenuti, la sua adozione non è ancora diffusa.

2.4 Utilizzo di HoneyPot o Tarpits

In informatica, un **Honeypot** (letteralmente: "barattolo del miele") è un sistema o componente hardware o software usato come "trappola" o "esca" a fini di protezione contro gli attacchi informatici [21].

Nel nostro contesto creare delle *trappole* in cui i bot potrebbero cadere è un progetto ancora in fase di sperimentazione.

Un esempio di **Honeypot** potrebbe essere un link nascosto, reso invisibile agli utenti umani, con cui gli utenti illegittimi potrebbero interagire. Una volta identificati, è possibile bloccarli.

Una ricerca in questo contesto è stata condotta da, Elisa Chiapponi [22] la quale ha approfondito l'uso di tecniche di inganno ("deception") per mitigare il traffico di scraping senza che i crawler ne fossero consapevoli.

La ricerca da lei condotta, ha dimostrato che è possibile indurre gli scraper a credere di aver raggiunto il loro obiettivo, fornendo loro informazioni modificate ma plausibili. Questo "avvelena" il loro processo di raccolta dati, rendendo inutili le informazioni acquisite. L'analisi dei dati raccolti tramite honeypot ha fornito preziose informazioni sull'ecosistema dei bot di scraping, inclusa la loro distribuzione e i pattern di ripetizione degli IP, suggerendo il coinvolgimento di servizi di Residential IP Proxies (resips).

Un altro approccio più aggressivo è possibile implementarlo tramite un **Tarpit**. Un **Tarpit** è un servizio su un sistema informatico che ritarda intenzionalmente le connessioni in entrata. Questo strumento è stato sperimentato anche per limitare i crawler e i bot più aggressivi, servendo una pagina infinita e dispendiosa in termini di risorse ai bot sospetti. Strumenti come **Nepenthes** generano dinamicamente un labirinto infinito di link indesiderati che un bot seguirà all'infinito. L'obiettivo è sprecare tempo e risorse del crawler e

possibilmente inquinare il suo dataset di intelligenza artificiale con spazzatura. Lo studio condotto da Griffioen e Doerr [23] dimostra la fattibilità di tale soluzione, sebbene venga utilizzata in ambito di difesa da Botnet questo metodo può essere applicato al nostro contesto.

Le due soluzioni prese in considerazione hanno dei limiti che possono ritorcersi contro. Sebbene l'obiettivo sia quello di far consumare risorse ai crawler, conseguentemente consumano risorse anche i server che ospitano il servizio. I Tarpit sono una tecnica sperimentale, che funge da forma di protesta tanto quanto di protezione, e la loro efficacia è discutibile, poiché è stato riportato che GPTBot di OpenAI ha imparato a sfuggire ad alcune trappole di Nepenthes [24].

2.5 Rilevamento avanzato e apprendimento automatico

Come anticipato precedentemente, i crawler cercano costantemente nuovi modi per eludere il rilevamento. Questo ha spinto la ricerca verso l'adozione di tecniche di apprendimento automatico.

2.5.1 Apprendimento non supervisionato

Lo studio condotto da Stevanovic [25] ha esplorato l'uso di algoritmi di reti neurali non supervisionate come le Self-Organizing Map (SOM) e la Modified Adaptive Resonance Theory 2 (Modified ART2) per l'analisi dei web log. Questo approccio è ritenuto innovativo, in quanto è in grado di comprendere le tipologie dei visitatori senza l'utilizzo di dati pre-etichettati. Questo vantaggio è fondamentale in quanto non necessita dell'analisi e lo studio dei dati per addestrare l'algoritmo, andando a diminuire gli errori di categorizzazione.

Lo studio condotto ha rivelato che, sebbene vi sia una netta separazione tra crawler malevoli e tutti gli altri visitatori, il 52% dei crawler malevoli mostra un comportamento di navigazione molto "simile a quello umano", rendendoli una sfida particolare per i sistemi di sicurezza. Le funzionalità utilizzate per l'analisi includono il numero di click, il rapporto HTML-immagine, la percentuale di richieste HEAD, le richieste di file PDF/PS,

e la percentuale di referrer non assegnati. È interessante notare come crawler noti e "ben educati" come GoogleBot, MSNBot e YahooBot mostrino stili di scansione distinti e ben riconoscibili. E' stato notato però che spesso alcuni bot non legittimi sfruttano gli stili di scansione noti per eludere i sistemi di difesa.

2.5.2 Apprendimento supervisionato

Stevanovic, in altri lavori da quelli precedentemente citati, ha anche studiato algoritmi supervisionati come C4.5, RIPPER, k-Nearest Neighbours, Naïve Bayesian Learning e Support Vector Machine per la classificazione degli utenti web [26]. L'efficacia di questi algoritmi, a differenza dell'apprendimento non supervisionato, dipende fortemente da un processo di etichettatura dei dati affidabile, il quale deve essere condotto manualmente da un tecnico, portando a inevitabili errori umani . In questo studio sono stati introdotti due nuove e importanti comportamenti noti per la rilevazione dei crawler: il rapporto di richieste sequenziali consecutive e la deviazione standard della profondità delle pagine richieste. Queste metriche si sono dimostrate molto utili nella distinzione tra utenti umani e crawler.

2.6 Modelli avanzati

Altri studi più recenti, mostrano strumenti come TS-Finder [27] e Attributed Action Net (AANet) [4], i quali impiegano complesse tecniche di deep learning per il rilevamento dei crawler. TS-Finder, ad esempio, utilizza i pattern di accesso noti uniti alle caratteristiche spazio-temporali dei siti visitati, andando a creare un modello combinato di serie temporali con rete neurale residua di whitening per apprendere i comportamenti. Un altro tipo di algoritmo è AANet, utilizzato per i Location-Based Services (LBS), il quale accetta sequenze di richieste URL come input e apprende simultaneamente le sequenze di azioni, i pattern temporali-spaziali e le informazioni di contesto. Questi modelli utilizzano architetture complesse come le unità ricorrenti (GRU), le funzioni di attivazione tanh e meccanismi di self-attention per affrontare la variabilità della lunghezza delle sequenze di input e per dare peso dinamico a diverse informazioni. Xia in "Crawler De-

tection in Location-Based Services Using Attributed Action Net" ha proposto amche un framework di apprendimento a tre fasi per gestire grandi quantità di dati e il problema dei dati etichettati limitati.

2.7 Rilevamento basato su dati di stato TCP

Lo studio condotto sul sistema Canopy [28] rappresenta una nuova frontiera nel rilevamento degli attacchi informatici, incapsulando i dati di stato TCP in una forma che permette previsioni rapide e accurate del comportamento malevolo. Il lavoro condotto sul nuovo strumento, ha dimostrato un'elevata precisione (99%) nell'identificare tentativi di intrusione, rilevando attacchi mai riconosciuti prima entro 750 millisecondi. Sebbene si concentri su attacchi DDoS a livello applicativo, è possibile sfruttare i principi di stato di rete per il rilevamento dei crawler.

2.8 Geolocation di scraper

Una nuova tipologia di riconoscemento in via di sviluppo, viene illustrata da Chiapponi [22], la quale sfrutta la possibilità di geolocalizzare i client originali che inviano richieste tramite i Residential IP Proxies (resips). La tecnica rttlocator sfrutta le differenze di Round Trip Time (RTT) nelle connessioni e le molteplici velocità dei pacchetti per triangolare la posizione dello scraper. Questo approccio può fornire nuovi dati aggiuntivi sulla provenienza delle attività di scraping, nonostante le sfide poste dalla natura 2D della mappa utilizzata e dalla vicinanza dei gateway resip.

2.9 Modello soglia a coda lunga

Lo studio "Detection Method for Distributed Web-Crawlers: A Long-Tail Threshold Model" [29], mostra un nuovo modello chiamato **Modello soglia a coda lunga**. Funziona sfruttando una caratteristica specifica del traffico web: la "coda lunga". In pratica, se si ordinano gli elementi di un sito in base alla loro frequenza di accesso, la maggior parte

del traffico si concentra su pochi elementi molto richiesti, mentre una vasta gamma di elementi riceve pochissime richieste, formando appunto una "coda lunga". Il modello opera in questo modo:

- Identifica gli indirizzi IP che accedono con una frequenza insolita agli elementi meno richiesti (quelli nella "coda lunga"). L'idea è che un attaccante, per copiare l'intero contenuto di un server, deve necessariamente accedere anche a queste parti meno popolari, cosa che un utente legittimo raramente farebbe.
- Quando un IP accede a un elemento nella regione della coda lunga, il suo contatore di accesso viene incrementato.
- Gli indirizzi IP identificati come crawler vengono gradualmente aggiunti a una lista di blocco.

Questo metodo si è dimostrato efficace nel rilevare i crawler distribuiti e nel mantenere un tasso di falsi positivi significativamente basso (ad esempio, 0.0275% nei test) rispetto ai metodi convenzionali basati sulla frequenza.

2.10 HAProxy

Come illustrato della sezione dedicata al reverse proxy (HAProxy), questa soluzione consente di filtrare, limitare o reindirizzare il traffico in ingresso sulla base di regole personalizzate. Tale soluzione è in grado di fornire svariati strumenti ben strutturati che rendono sicuro il sistema. L'azienda ha però notato due grandi criticità nell'adozione di questo strumento: la prima criticità risiede sul fatto che per adottare questo strumento, il tecnico che gestisce l'infrastruttura, ha bisogno di effettuare una migrazione del sito, ha bisogno di modificare i protocolli e le regole.

Queste operazione hanno un effetto sulle configurazioni del sito che devono essere comunicate al proprietario.

Gli effetti potrebbero causare il cambio degli indirizzi, la modifica del server in cui è posto il sito, o altri cambiamenti che potrebbero rendere insoddisfatto il cliente.

La seconda criticità è legata anche alla mole di lavoro che dovrebbe essere impiegata per effettuare la migrazione per tutti i siti che ha in gestione l'azienda. Tali criticità rendono, la soluzione del Reverse Proxy, attualmente inappropriata.

La seguente immagine (2.5) mostra concettualmente la configurazione della rete con HAProxy.

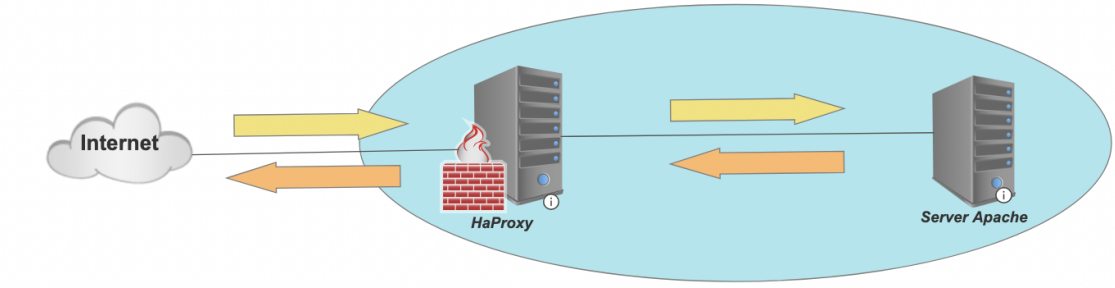

Figura 2.5: Configurazione Con Proxy

2.11 Bridge trasparente

Un'altra soluzione proposta è legata all'uso di un **bridge trasparente**, che funge da punto di passaggio tra la rete esterna e la rete privata.

In tale configurazione il **bridge** ha il compito di filtrare e poi inoltrare tutte le connessioni che provengono dall'interfaccia collegata con la rete globale, attraverso l'interfaccia posta all'interno della rete aziendale, dove saranno posti i vari server di apache dedicati alla gestione delle connessioni.

Questa architettura non richiede modifiche al server web, ma consente un'analisi completa del traffico in ingresso. I principali vantaggi sono:

- Non interferisce con l'architettura esistente, permettendo di mantenere inalterati i server e i domini ospitati, il che è un vantaggio poichè non serve comunicare al cliente le modifiche effettuate;
- Fornisce una panoramica dettagliata delle connessioni, con la possibilità di classificarle in base a parametri comportamentali;

- Consente di adottare politiche di limitazione e di Traffic shaping in modo estremamente flessibile.

La seguente immagine (2.6), mostra la configurazione concettuale della soluzione tramite il **Bridge**.

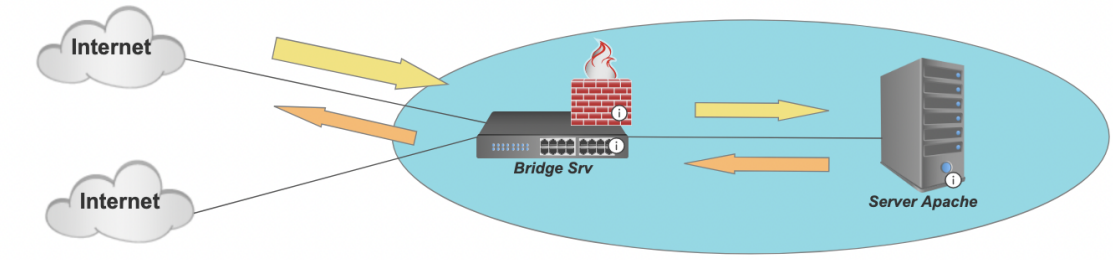

Figura 2.6: Configurazione Con Bridge Trasparente

2.12 Confronto delle soluzioni

La scelta dell'approccio più adatto al riconoscimento dei Crawler non dipende solamente dall'implementazione dell'algoritmo ma dipende soprattutto dal tipo dall'infrastruttura a cui è destinato il progetto. Potrebbe dipendere quindi da molteplici fattori come la dimensione del traffico web, la natura della minaccia a cui sono esposti, le risorse disponibili. In questa sezione verranno presentati i principali vantaggi e svantaggi di tutte le soluzioni proposte.

2.12.1 Utilizzo di Apache, WAF e Firewall

- **Pro:** Il modulo ModEvasive è efficace nel rilevare e bloccare attacchi noti come DDoS e brute force. ModSecurity, operando come WAF a livello 7 OSI, consente il filtraggio approfondito delle richieste HTTP/HTTPS e l'applicazione di *rate limiting* dinamici;

- **Contro:** I WAF e i firewall di rete richiedono una configurazione accurata e una manutenzione continua delle regole. I firewall di rete, operando a livelli inferiori, non sono specificamente progettati per le minacce a livello applicativo;

2.12.2 Utilizzo di CloudFlare

- **Pro:** Agisce come intermediario fornendo servizi di sicurezza, prestazioni e affidabilità. Utilizza regole firewall basate su richieste HTTP per prevenire comportamenti automatizzati e dannosi.
- **Contro:** Nonostante la facilità d'uso e la capacità di blocco, la sua adozione non è ancora universalmente diffusa;

2.12.3 Utilizzo di Honeypot o Tarpit

2.12.3.1 Honeypot

- **Pro:** Ha fornito preziose informazioni sull'ecosistema dei bot di scraping, inclusa la distribuzione, i pattern più frequenti, metodologie utilizzate e molto altro;
- **Contro:** Gli scraper più sofisticati possono rilevare le ridirezioni e adattare il loro comportamento, eludendo il sistema. La dimensione del pool di IP degli honeypot potrebbe non riflettere accuratamente quella dei fornitori di resip reali;

2.12.3.2 Tarpit

- **Pro:** Grazie alla possibilità di servire pagine infinite o dispendiose sono in grado di intrappolare una buona parte dei bot. Mirano a "inquinare" i dataset di intelligenza artificiale dei crawler;
- **Contro:** Richiedono risorse anche dai server che li ospitano. La loro efficacia è stata messa in discussione, con bot avanzati (come GPTBot di OpenAI) che hanno dimostrato di poterli eludere;

2.12.4 Rilevamento Avanzato e Apprendimento Automatico

2.12.4.1 Apprendimento non supervisionato

- **Pro:** Analizza i log web per comprendere le tipologie e la distribuzione dei visitatori basandosi sul comportamento di navigazione, senza la necessità di dati pre-etichettati, il che è un vantaggio significativo data la difficoltà di etichettare accuratamente i dati. Distingue stili di scansione distinti per crawler noti;
- **Contro:** Il 52% dei crawler malevoli mostra un comportamento molto "simile a quello umano", rendendo la loro identificazione una sfida particolare per i sistemi di sicurezza;

2.12.4.2 Apprendimento supervisionato

- **Pro:** L'introduzione di feature come il rapporto di richieste sequenziali consecutive e la deviazione standard della profondità delle pagine richieste si è dimostrata utile nel distinguere utenti umani e crawler;
- **Contro:**L'efficacia di questi algoritmi dipende fortemente da un processo di etichettatura dei dati affidabile;

2.12.4.3 Modelli avanzati

- **Pro:** Utilizzano tecniche di deep learning per apprendere complessi comportamenti e informazioni di contesto.
- **Contro:** La complessità di questi modelli può renderne difficile l'implementazione e la modifica;

2.12.4.4 Rilevamento basato su dati di stato TCP

- **Pro:** Un approccio innovativo per identificare attacchi low-and-slow (DDoS), incapsulando i dati di stato TCP per previsioni rapide e accurate del comportamento malevolo. Ha dimostrato un'elevata precisione (99%) nell'identificare attacchi;

- **Contro:** Sebbene rilevante per i crawler aggressivi, il suo focus primario è sugli attacchi DDoS;

2.12.5 Geolocation di Scraper

- **Pro:** Permette di geolocalizzare i client originali che inviano richieste tramite Residential IP Proxies (resips) sfruttando le differenze di Round Trip Time (RTT) e le velocità dei pacchetti;

- **Contro:** L'uso di una rappresentazione 2D del mondo introduce limitazioni intrinseche nella precisione della geolocalizzazione. La vicinanza dei gateway resip e la difficoltà nel trovare la combinazione ottimale di velocità rendono la geolocalizzazione una sfida complessa e ancora oggetto di ricerca;

2.12.6 Modello Soglia a Coda Lunga

- **Pro:**È efficace nel rilevare i crawler distribuiti e mantiene un tasso di falsi positivi significativamente basso (0.0275% nei test) rispetto ai metodi convenzionali basati sulla frequenza;

2.12.7 HAProxy

- **Pro:** Il tool è ampiamente utilizzato per il bilanciamento del carico e la gestione di grandi volumi di connessioni TCP e HTTP. Supporta Access Control List (ACL) per filtrare, stick-tables per il tracciamento degli IP, rate limiting e shaping, logging e metriche dettagliate;

- **Contro:** La sua adozione in infrastrutture esistenti richiede modifiche significative nella configurazione dei server. Potrebbe causare modifiche agli indirizzi o alla posizione del server, generando insoddisfazione del cliente. Richiede un'ingente mole di lavoro per la migrazione;

2.12.8 Bridge Trasparente

- **Pro:** Agisce come punto di passaggio tra la rete esterna e quella interna senza richiedere modifiche alla logica esistente del server web. È una soluzione meno invasiva sull'infrastruttura esistente;
- **Contro:**La fase di valutazione dei risultati è complessa a causa dell'enorme quantità di dati generati. Il riconoscimento euristico, sebbene flessibile, è meno autonomo e preciso di un algoritmo basato sull'IA;

Capitolo 3

Analisi del problema

In questa sezione verrà presentata un'analisi dettagliata del contesto aziendale e tecnico in cui è stato sviluppato il progetto. Verranno descritti il problema legato al traffico automatizzato, le soluzioni esplorate e le motivazioni che hanno portato alla scelta finale di implementare un bridge trasparente.

3.1 Il problema

Hosting Solutions è un'azienda fiorentina specializzata in servizi di web hosting, cloud computing e gestione di data center. La missione principale dell'azienda è quella di offrire soluzioni scalabili per l'hosting di siti web e applicazioni aziendali, garantendo altissimi standard di affidabilità, continuità del servizio e sicurezza informatica. Poichè l'azienda gestisce un elevato flusso di dati, si trova quotidianamente a confronto con attacchi informatici, tentativi di scraping e attività di bot,che possono compromettere la stabilità della rete e la qualità del servizio offerto. Si trovano costretti dunque, a mettersi al riparo da tali esigenze.

Durante le attività di monitoraggio dell'infrastruttura, è emerso un problema significativo legato alla presenza di numerosi client automatizzati, come crawler e bot, che generano un elevato volume di richieste verso i server web ospitati dall'azienda. Sebbene alcuni di questi client possono essere legittimi, come ad esempio Googlebot, che rispettano i pattern di accesso o le richeste del file `robots.txt`. Altri invece presentano comportamenti aggressivi che rischiano di compromettere le risorse del sistema.

Le immagini 3.8 e 3.7 mostrano il traffico giornaliero prodotto su uno dei server utilizzati dall'azienda per gestire il carico di lavoro, evidenziando la discrepanza tra il numero di connessioni legittime generate da utenti umani e quelle provenienti da traffico automatizzato. In particolare, è possibile osservare che, mentre il numero di connessioni da parte

degli utenti umani legittimi rappresenta il 54,5%, la percentuale di righe nel file di log corrispondenti a queste connessioni è solo del 5,74%. La differenza risiede nel fatto che ogni connessione umana si articola in un numero "ragionevole" di interazioni con il server, mentre i crawler generano flussi massivi di connessioni.

Questo esempio evidenzia la necessità di limitare il numero connessioni effettuate da utenti, umani e non, che creano traffico "superfluo" in modo da garantire che le connessioni legittime possano accedere senza ostacoli. L'obiettivo principale della tesi è proprio quello di riuscire a riconoscere il traffico cosiddetto "superfluo".

```
=== Statistiche globali ===
→ Percentuale connessioni umane legittime: 54.5%
→ Percentuale crawler noti benigni: 27.77%
→ Percentuale crawler sospetti (non noti o aggressivi): 2.98%
→ Percentuale crawler illegittimi: 3.78%
→ Percentuale di ip riconosciuti di CloudFlare: 10.97%
→ Totale IP analizzati: 40247 (somma categorie: 40247)
```

Figura 3.7: Statistiche connessioni globali

```
=== Percentuali su base numero di richieste (righe) ===
→ Totale righe classificate per categoria: 1108525
→ 5.74% delle righe appartengono a connessioni legittime
→ 59.95% delle righe appartengono a crawler benigni
→ 7.08% delle righe appartengono a attività sospette
→ 27.23% delle righe appartengono a crawler illegittimi
```

Figura 3.8: Statistiche globali su righe

3.1.1 Cause

Il problema deriva principalmente dal fatto che quasi il 90% delle connessioni in ingresso sono effettuate da scan, ricerche automatiche e abusi. Le seguenti immagini (3.1 e 3.2) mostrano il confronto tra una connessione legittima di un utente e una connessione proveniente da un crawler.

```
1 www.mywebsite.it:443 192.168.131.7 - - [24/May/2025:11:23:19 +0200] "GET /faq/
      foto_normativa.html HTTP/2.0" 200 3760 "-" "Mozilla/5.0 (iPhone; CPU iPhone
      OS 18_4_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Mobile/15
      E148" 2449 -
2 www.mywebsite.it:443 192.168.131.7 - - [24/May/2025:11:23:19 +0200] "GET /text.
      css HTTP/2.0" 200 368 "https://mywebsite.it/..." "Mozilla/5.0 (iPhone; CPU
      iPhone OS 18_4_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko)
      Mobile/15E148" 1136 -
3 www.mywebsite.it:443 192.168.131.7 - - [24/May/2025:11:23:19 +0200] "GET /index
      /flag-eng2.gif HTTP/2.0" 200 688 "https://mywebsite.it/..." "Mozilla/5.0 (
      iPhone; CPU iPhone OS 18_4_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML,
      like Gecko) Mobile/15E148" 1362 -
4 www.mywebsite.it:443 192.168.131.7 - - [24/May/2025:11:23:19 +0200] "GET /arrow
      -back-bl.gif HTTP/2.0" 200 266 "https://mywebsite.it/..." "Mozilla/5.0 (
      iPhone; CPU iPhone OS 18_4_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML,
      like Gecko) Mobile/15E148" 1928 -
5 www.mywebsite.it:443 192.168.131.7 - - [24/May/2025:11:23:19 +0200] "GET /page-
      title-it.gif HTTP/2.0" 200 1776 "https://mywebsite.it/..." "Mozilla/5.0 (
      iPhone; CPU iPhone OS 18_4_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML,
      like Gecko) Mobile/15E148" 1251 -
```

Code 3.1: Esempio di connessione di un crawler

```
1  www.myWebSite.it:80 192.168.77.202 - - [24/May/2025:11:24:15 +0200] "GET /sites
       /default/files/css/... HTTP/1.1" 200 4165 "http://myWebSite.it/result?page
       =1" "Mozilla/5.0 (Linux; Android 6.0.1; Nexus 5X Build/MMB29P) AppleWebKit
       /537.36 (KHTML, like Gecko) Chrome/136.0.7103.92 Mobile Safari/537.36 (
       compatible; Googlebot/2.1; +http://www.google.com/bot.html)" 3692 -
2  www.myWebSite.it:80 192.168.77.202 - - [24/May/2025:11:54:13 +0200] "GET /
       examples HTTP/1.1" 200 7724 "-" "Mozilla/5.0 (Linux; Android 6.0.1; Nexus 5X
        Build/MMB29P) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.7103.113
       Mobile Safari/537.36 (compatible; GoogleOther)" 99099 -
3  www.myWebSite.it:80 192.168.77.202 - - [24/May/2025:11:54:15 +0200] "GET /2023-
       g7-battagliola-fortini-giovanetti-mancarella-marti-ruvioli HTTP/1.1" 403
       26664 "-" "Mozilla/5.0 (Linux; Android 6.0.1; Nexus 5X Build/MMB29P)
       AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.7103.113 Mobile Safari
       /537.36 (compatible; GoogleOther)" 759048 -
4  www.myWebSite.it:80 192.168.77.202 - - [24/May/2025:11:54:18 +0200] "GET /all
       HTTP/1.1" 200 8147 "-" "Mozilla/5.0 (Linux; Android 6.0.1; Nexus 5X Build/
       MMB29P) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.7103.113 Mobile
       Safari/537.36 (compatible; GoogleOther)" 40104 -
5  www.myWebSite.it:80 192.168.77.202 - - [24/May/2025:11:54:20 +0200] "GET /
       taxonomy/term/4 HTTP/1.1" 200 8965 "-" "Mozilla/5.0 (Linux; Android 6.0.1;
       Nexus 5X Build/MMB29P) AppleWebKit/537.36 (KHTML, like Gecko) Chrome
       /136.0.7103.113 Mobile Safari/537.36 (compatible; GoogleOther)" 36452 -
6  www.myWebSite.it:80 192.168.77.202 - - [24/May/2025:11:54:22 +0200] "GET /
       result?page=0 HTTP/1.1" 200 10644 "-" "Mozilla/5.0 (Linux; Android 6.0.1;
       Nexus 5X Build/MMB29P) AppleWebKit/537.36 (KHTML, like Gecko) Chrome
       /136.0.7103.113 Mobile Safari/537.36 (compatible; GoogleOther)" 37164 -
7  www.myWebSite.it:80 192.168.77.202 - - [24/May/2025:11:54:25 +0200] "GET /2023-
       gulcehre HTTP/1.1" 403 26664 "-" "Mozilla/5.0 (Linux; Android 6.0.1; Nexus 5
       X Build/MMB29P) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.7103.113
        Mobile Safari/537.36 (compatible; GoogleOther)" 232206 -
8  www.myWebSite.it:80 192.168.77.202 - - [24/May/2025:11:54:27 +0200] "GET /
       taxonomy/term/18 HTTP/1.1" 200 8970 "-" "Mozilla/5.0 (Linux; Android 6.0.1;
       Nexus 5X Build/MMB29P) AppleWebKit/537.36 (KHTML, like Gecko) Chrome
       /136.0.7103.113 Mobile Safari/537.36 (compatible; GoogleOther)" 40014 -
9  www.WebFinto.it:443 192.168.77.202 - - [24/May/2025:20:00:28 +0200] "POST /wp-
       admin/admin-ajax.php HTTP/1.1" 200 4402 "https://....." "Mozilla/5.0 (Linux;
        Android 6.0.1; Nexus 5X Build/MMB29P) AppleWebKit/537.36 (KHTML, like Gecko
       ) Chrome/136.0.7103.92 Mobile Safari/537.36 (compatible; Googlebot/2.1; +
       http://www.google.com/bot.html)" 309627 -
10
11 ...
```

Code 3.2: Esempio di connessione di un crawler

La prima immagine mostra una connessione tipica di un utente umano, con una serie di richieste limitate e logiche, tutte relative a file legittimi come immagini, CSS e altre risorse statiche. La seconda immagine, invece, mostra il comportamento di un crawler, in cui si osserva un numero significativo di richieste in tempi molto brevi, spesso con URL generici o anomali, tipici di attività di scraping.

3.1.2 Effetti

L'impatto principale di questo tipo di traffico è la saturazione delle risorse di rete e di calcolo:

- Aumento del tempo di risposta per gli utenti reali;
- Sovraccarico dei server web;
- Blocco temporaneo dei servizi, con conseguente disservizio per i clienti legittimi;
- Difficoltà di gestione a livello di supporto tecnico.

Se non gestito adeguatamente, questo problema può compromettere gravemente la qualità complessiva del servizio e danneggiare la reputazione dell'azienda.

Capitolo 4

Architettura

In questo capitolo descriviamo la struttura del progetto e le fasi attraverso cui è stato sviluppato. L'obiettivo principale è quello di analizzare i file di log generati da un server web, al fine di stabilire quali connessioni possano essere considerate legittime e quali invece debbano essere soggette a limitazione o monitoraggio. L'analisi si concentra su parametri estratti direttamente dalle richieste HTTP e punta a riconoscere comportamenti anomali o automatizzati che potrebbero compromettere le risorse del sistema.

4.1 Analisi del file di log

Il file di log del server Apache contiene informazioni fondamentali su ogni interazione avvenuta tra un client e il server. Ogni riga del log rappresenta una richiesta HTTP e segue un formato esteso personalizzato, arricchito con metadati utili per il monitoraggio delle performance e l'analisi del traffico.

Una tipica riga di log contiene i seguenti elementi:

- **Nome host e porta**: il dominio e la porta su cui è arrivata la richiesta (es. `www.mywebsite:443`).
- **IP originario**: l'indirizzo IP che ha inoltrato la richiesta verso il server.
- **Ident e user**: spesso rappresentati come `- -`, poichè non specificati.
- **Timestamp**: data e ora della richiesta, incluso il fuso orario (es. `[24/May/2025:00:00:54 +0200]`).
- **Richiesta HTTP**: contiene il metodo (GET, POST, HEAD), il percorso e la versione del protocollo.

- **Codice di stato HTTP**: ad esempio `200` per successo, `404` per risorsa non trovata.
- **Dimensione della risposta**: in byte.
- **Referer**: indica da quale URL proviene l'utente (può essere `"-"` se assente).
- **User-Agent**: identifica il software del client, utile per rilevare browser o bot.
- **Durata della richiesta**: tempo in microsecondi impiegato dal server per rispondere.
- **IP originario effettivo (X-Forwarded-For)**: in presenza di reverse proxy o CDN come Cloudflare.

4.1.1 Parsing e normalizzazione

Per rendere l'analisi efficace, il file viene letto e analizzato riga per riga tramite un'espressione regolare appositamente progettata per identificare ciascun campo anche in presenza di valori facoltativi o anomali. I dati vengono normalizzati e arricchiti con valori di default (es. `"-"` per referer o durata assente), per permettere l'elaborazione uniforme da parte del sistema.

Una volta estratti i dati, tutte le richieste vengono aggregate per indirizzo IP, permettendo così di analizzare il comportamento complessivo di ciascun client.

4.1.2 Esempio pratico

La seguente riga è un esempio reale di richiesta HTTP:

```
1 www.mywebsite.it:443 100.15.20.30 - - [24/May/2025:00:01:58 +0200] "GET / HTTP
      /1.1" 200 159021 "-" Uptime-Kuma/... 1145867 -
```

Code 4.3: HTTP Example

Dalla precedente stringa si estraggono le seguenti informazioni:

- **Host:** `www.mywebsite.com`
- **Porta:** `443`

- **IP client visibile:** `100.15.20.30`
- **Ident/User:** `- -`
- **Timestamp:** `24/May/2025 00:01:58 +0200`
- **Richiesta:** `GET HTTP/1.1`
- **Codice di stato:** `200`
- **Dimensione della risposta:** `159021 bytes`
- **Referer:** `-`
- **User-Agent:** `Uptime-Kuma/...`
- **Durata:** `1145867 microsecondi`
- **IP originario reale:** `-` (non specificato, presumibilmente coincidente con il primo IP)

Attualmente il codice raggruppa le richieste provenienti dallo stesso indirizzo in una unica connessione da analizzare. In uno sviluppo successivo potrebbe essere integrata una analisi più dettagliata andando a creare degli intervalli di tempo e categorizzando le richieste sia per indirizzo che per lo slot di tempo in cui è arrivato il pacchetto. Tale implementazione è stata sviluppata da Lu e Yu [30].

L'analisi del traffico generato da questo IP ha restituito le seguenti metriche:

```
IP: 100.15.20.30
    Numero richieste: 5 in delta tempo: 1h 31min 0s
    Frequenza richieste: < 0.01 req/sec
    Identificazione: Crawler identificato (uptime)
    Clicks: 5
    Html img ratio: inf
    Pdf ps percent: 0.0
    Error 4xx percent: 0.0
    Head percent: 0.0
    Referer null percent: 100.0
    Robots requested: 0
    Unique url count: 1
    Login attempts: 0
```

```
14          Post count: 0
15          Env file requested: 0
16          Country: DE
17          ASN: AS14061 DIGITALOCEAN-ASN
18          PTR: monitor.wau73.cloud
19              Probabile provider cloud
20          Crawler score: 4 (solo HTML, nessuna immagine, referer assenti (>75%))
21          Classificazione euristica: Crawler noto (comportamento benigno)
22          [AbuseIPDB] Abuse Score: 0, Whitelisted: None
```

Classificazione finale: Il programma ha riconosciuto la firma `uptime` nel campo `User-Agent`, indicando la natura automatica della connessione identificandolo quindi come crawler. Tuttavia, il comportamento complessivo è stato considerato benigno: basso volume di richieste, nessuna attività sospetta o malevola, rispetto dei pattern tipici di un crawler standard.

Come enunciato nell'articolo "Web robot detection based on hidden Markov model" [30], un crawler è identificabile mediante l'`User-Agent`, il quale contiene parole chiavi note come "crawler" o "spider" o "uptime".

Classificazione euristica assegnata: *Crawler noto (comportamento benigno.)* Questa classificazione viene attribuita in modo automatico da uno scoring che valuta la frequenza, il numero di risorse esplorate, la qualità delle richieste (assenza di errori o POST sospetti), e l'intenzione apparente del bot. In particolare, uno score pari a 4 non è sufficiente per indicare aggressività o pericolosità, infatti è possibile notare che si contraddistingue da altri Crawler per una frequenza di richieste su unità di tempo, estremamente bassa. Il quale non lo rende un obiettivo su cui prestare attenzione.

I campi *Country*, *ASN*, *PTR* e *AbuseIPDB*, sono campi che non fanno parte della classificazione e non contribuiscono allo score bensì sono stati utilizzati in fase di Analisi dei risultati e sono serviti per identificare falsi positivo o falsi negativi. Verranno trattati nel dettaglio in seguito.

4.2 Crawler Score

Il `Crawler Score` rappresenta una misura euristica che sintetizza, in un unico valore numerico, il comportamento osservato di ogni client. Il punteggio è calcolato sommando penalità associate a determinati pattern considerati sospetti o anomali. I criteri scelti sono frutto di studi passati, in particolare mi riferisco agli articoli *"Feature evaluation for web crawler detection with data mining techniques"*[31] e *"Web robot detection-preprocessing web logfiles for robot detection"* [32],i quali hanno racchiuso i comportamenti di un crawelr generico a 9 tipiche interazioni. A tali interazioni ho ritenuto opportuno aggiungere altri 3 comportamenti che a mio avviso rendono ancora più evidente il comportamento automatizzato. Di seguito una spiegazione dettagliata dei punteggi assegnati:

4.2.1 Numero di richieste o di click:

Una volta raggruppate le connessioni per indirizzi è possibile contare precisamente le richieste e stabilire la frequenza con cui ha effettuato tali connessioni. Con numero di click si intende un attributo numerico calcolato come il numero di richieste HTTP inviate da un utente in una singola sessione. Se le connessioni sono inferiori a 5 richieste per secondo (req/sec) non vengono analizzare nel dettaglio ma vengono considerate a prescindere legittime, poichè l'obiettivo finale è quello di rallentare tutti gli indirizzi che potrebbero congestionare il servizio e poichè nel tempo analizzato ha effettuato meno di 5 connessioni ritengo che non possano creare problemi di questo tipo. Dopo di che viene assegnato un punteggio per ogni range di frequenza di richieste per unità di tempo:

- **frequenza compresa tra 5 e 10:** viene assegnato un punteggio di +1 con un tag assegnato `"frequenza sospetta (≥5 req/sec)"`
- **frequenza compresa tra 10 e 20:** viene assegnato un punteggio di +3 con un tag assegnato `"frequenza alta (≥10 req/sec)"`
- **frequenza maggiore di 20:** vviene assegnato un punteggio di +5 con un tag assegnato `"frequenza estrema (≥20 req/sec)"`

4.2.2 Rapporto HTML / Immagini (html img ratio):

Tale rapporto viene calcolato come numero di richieste di pagine HTML rispetto al numero di immagini richiesti (JPEG e PNG) inviate in una singola sessione. Il parametro **HtmlImgRatio** è fondamentale per distinguere le richieste umane da quelle automatizzate, come spiegato nell'articolo "Efficiency Analysis Of Resource Request Patterns In Classification Of Web Robots And Humans" [33], un utente umano ogni qual volta che visita una pagina via brawser, scarica sia il contenuto testuale che le immagini correllate. I punteggi assegnati sono quindi due:

- **html_img_ratio infinito:** equivale a richiedere solamente HTML e nessuna immagine, viene quindi assegnato un punteggio di +2 con il tag `"solo HTML, nessuna immagine"`
- **html_img_ratio**$\geq$**5:** viene assegnato un punteggio di +1 con un tag assegnato `"HTML/img alto"`

4.2.3 Referer percent:

Il referer indica l'URL della pagina web da cui si è arrivati alla pagina corrente. Il valore `Referer percent` è un attributo calcolato come percentuale di campi vuoti o non assegnati in una singola sessione. La maggior parte dei web crawler avvia richieste HTTP con un campo referrer vuoto, mentre la maggior parte dei browser fornisce di default le informazioni sul referrer.

- **Referer_percent** $\geq$50**:** viene assegnato un punteggio di +1 con tag `"referer spesso assente (`$\geq$`50%)"`
- **Referer_percent** $\geq$75**:** viene assegnato un punteggio di +2 con tag `"referer assenti (`$\geq$`75%)"`

4.2.4 Percentuale di errori 4xx:

ErrorPercent è un attributo numerico che viene calcolato come la percentuale di richieste HTTP errate inviate in un singola sessione. Un esempio potrebbe essere `error 404`, il quale suggerisce che il client sta cercando pagine inesistenti o sta analizzando il sito.

- **error_4xx_percent** $\geq$10**:** viene assegnato un punteggio di +1 con tag `"errori 4xx frequenti (`$\geq$`10%)"`
- **error_4xx_percent** $\geq$20**:** viene assegnato un punteggio di +2 con tag `"molti errori 4xx (`$\geq$`20%)"`.

4.2.5 Intensità e varietà delle richieste:

Una penalità aggiuntiva è prevista in caso di combinazione tra:

- più di 50 richieste
- rapporto HTML/img superiore a 2
- frequenza superiore a 0.01 req/sec

se la condizione fosse verificata viene aggiunto 1 al punteggio con tag `"molte richieste (senza asset) con frequenza sostenuta"`.

4.2.6 Richieste PDF o PostScript:

Indicatore di un bot Scraping che cerca documenti PDF o PS nel sito. In genere un crawler che attraversa un sito, tenta di recuperare tutti i file PDF/PS incontrati, mentre un visitatore umano è molto più selettivo su ciò che sceglie di recuperare. Se verificato viene aggiunto un punto con tag "PDF/PS richiesti".

4.2.7 Uso del metodo HEAD:

Un attributo numerico calcolato come percentuale di richieste HTTP di tipo HEAD inviate in una singola sessione. Nel caso di una richiesta HTTP HEAD, il server restituisce

solo l'intestazione della risposta e non tutta la risposta effettiva. Il metodo HEAD come enunciato nell'articolo "Securing Web Service by Automatic Robot Detection" [34],è tipico di tool di monitoraggio e scanner automatici poichè tentano di ridurre la quantità di dati richiesti ad un sito. In generale le richieste provenienti da un utente umano che naviga su un sito web tramite browser sono, di default, di tipo GET.

- **head_percent** $\geq$90% **e frequenza** $\geq$0.01**:** viene assegnato un punteggio di +1 con tag "HEAD frequenti e ritmo sostenuto".

4.2.8 Richieste di accesso al file `robots.txt`:

Attributo booleano che ha come valori 0 o 1, l'attributo indica se il file è stato richiesto o meno. Gli amministratori del sito Web utilizzano un file in formato speciale denominato `robots.txt` per indicare ai robot in visita quali parti dei loro siti non dovrebbero essere visitate. È improbabile che un utente umano controlli questo file, poiché non esiste alcun collegamento ipertestuale che porti a questo file, né (la maggior parte) degli utenti ne conoscono l'esistenza [35]. Viene assegnato un punteggio di +0.5 con tag "robots.txt richiesto".

4.2.9 Richieste di accesso al file `Env`:

Il file `Env` viene letto da utenti illegittimi poichè contine al suo interno credenziali sensibili. Viene assegnato un punteggio di +1 con tag "tentativo accesso a file .env".

4.2.10 URL unici richiesti:

Il parametro **Unique url count** rappresenta il numero di URL distinti richiesti da un determinato indirizzo IP. Tale parametro è un indicatore fondamentale per identificare una scansione in atto. Un utente umano ogni qual volta che visita un sito web tende a visitare un numero limitato di pagine, a differenza di un bot, benigno o meno, che scansiona tutti i link in maniera sistematica senza seguire una vera logica tra i vari link.

- ≥50% **URL diversi:** viene assegnato un punteggio di +1 con tag `"molti URL unici"`.

4.2.11 Tentativi di login (login_attempts):

L'attributo rappresenta il numero di tentativi di accesso che vengono eseguiti. Tale valore, se molto alto, può indicare un attacco in atto. Attualmente non viene fatto un controllo se il tentativo di accesso ha avuto successo o meno, e nemmeno se gli accessi sono eseguiti su siti distinti. In uno sviluppo futuro, tale parametro, potrebbe essere migliorato e reso più selettivo. Lo studio "Good Bot, Bad Bot: Characterizing Automated Browsing Activity" [36] ha mostrato che il 96,6% dei bot effettua meno di 10 tentativi di login, e solo lo 0,3% effettuano più di 100 tentativi. Per tale motivo ho impostato le seguenti soglie:

- **login_attempts**≥1**:** viene assegnato un punteggio di +1 con tag `"alcuni tentativi di login"`.
- **login_attempts**≥10**:** viene assegnato un punteggio di +2 con tag `"tentativi login sospetti"`.

4.2.12 Numero di richieste POST (post_count):

Un uso eccessivo del metodo POST può essere segnale di upload, attacchi o simulazioni di interazione.

- **post_count**≥20**:** viene assegnato un punteggio di +1 con tag `"molti POST (possibile attacco)"`.

Conclusione: Ogni penalità riflette un possibile comportamento anomalo, ma non è sufficiente da sola a classificare negativamente un client. Il sistema valuta l'insieme delle caratteristiche e, tramite la somma dello score, è in grado di dare un giudizio oggettivo sulle connessioni effettuate da un determinato IP. Questo approccio è abbastanza accurato ma non può essere utilizzato in autonomia poichè ci sono molti casi limite che potrebbero eludere il sistema. E' per questo motivo, come spiegato nella sezione successiva,

che verranno integrate delle soglie stabilite a priori per minimizzare gli errori e cercare di categorizzare più correttamente quei casi incerti che lo score potrebbe andare a creare.

4.3 Classificazione euristica

Una volta calcolato il *Crawler Score* per ciascun IP, l'algoritmo procede con una classificazione finale del comportamento, assegnando ad ogni IP una e una sola categoria tra quelle previste.

La classificazione segue una logica a cascata, basata su soglie predefinite e condizioni prioritarie, che tengono conto sia del punteggio ottenuto, sia della frequenza temporale delle richieste, sia di eventi specifici considerati critici. Rintengo indispensabile soffermarmi ed entrare nel dettaglio di questo capitolo, poichè rappresenta il cuore del codice:

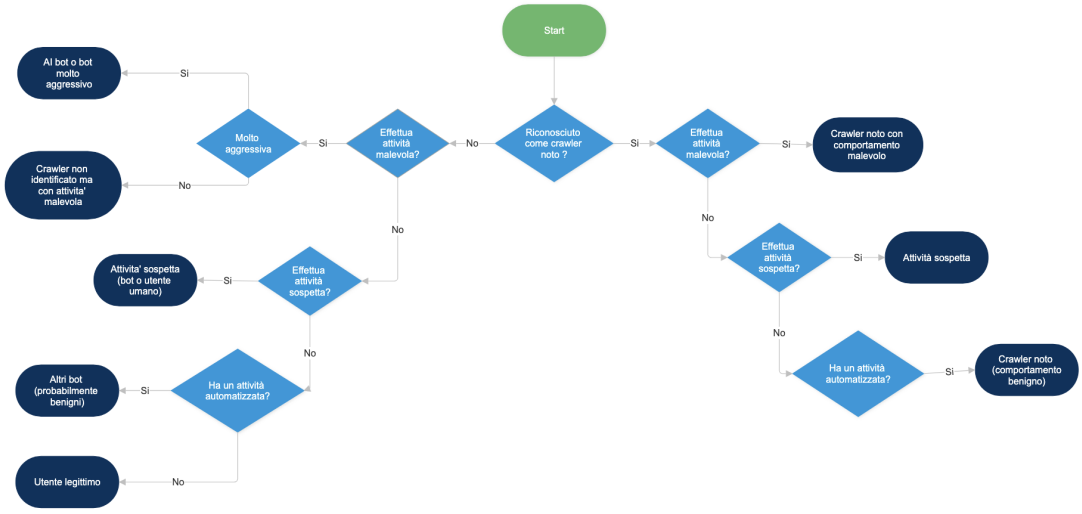

Figura 4.9: Diagramma di Flusso

L'immagine 4.9 mostra in maniera molto semplificata la logica a cascata che è stata implementata. Per ogni condizione è stato scelto di inserire sia la valutazione grezza del score, come per esempio uno score $\geq$3, che potrebbe rappresentare un craweler *malevolo*, ma anche dei controlli sulle soglie. Le soglie sono delle variabili modificabili a discrezione

dell'utente in base alla dimensione della rete e in base alle proprie esigenze. Un esempio di logica è la seguente:

```
1 elif (features["login_attempts"] >= LOGIN_SOGLIA_MALEVOLO or
2      (req_per_sec >= FREQ_SOGLIA_ILLEGITTIMO and delta.total_seconds() >= 10)
     or
3      features["post_count"] >= POST_SOGLIA_MALEVOLO or features["
     error_4xx_percent"] >= ERROR_4XX_SOGLIA_MALEVOLO or features["
     env_file_requested"] > 0) and score >= 3:
4         classificazione_finale = "Crawler non identificato ma con attivita'
     malevola"
5         illegittimi += 1
```

Code 4.4: Examples Code Flow Diagram

Le categorie previste sono le seguenti:

1. Crawler noto con comportamento malevolo Assegnata se il client è presente nella lista dei crawler noti ma mostra segnali di attività pericolosa, come tentativi di login elevati, molte POST, errori 4xx significativi, richiesta del file `.env`, o una frequenza di richieste eccessiva su intervallo consistente. È inoltre richiesto uno score di almeno 4.

2. Attività sospetta (da crawler noto) Client noto con score almeno 3 e comportamenti sospetti ma non pienamente malevoli, come accessi a risorse sensibili, errori 4xx elevati, POST intermedie o frequenza oltre soglia legittima. Anche la richiesta di file come `.env` può contribuire.

3. Crawler noto (comportamento benigno) Assegnato se il crawler è noto e non mostra attività aggressive. Include score bassi e traffico distribuito o moderato.

4. AI bot o bot molto aggressivo Client non noto con punteggio elevato ($\geq$3), frequenza molto alta e contemporanea presenza di tentativi di login e POST malevoli. Tipico di strumenti automatizzati ad alta intensità.

5. Crawler non identificato ma con attività malevola Client non noto che mostra segni chiari di attività ostile: login, POST, errori 4xx o richiesta del file `.env`, con score almeno 3.

6. Attività sospetta (bot o utente umano) Client non noto, score $\geq$3, e comportamento ambiguo (POST o login sospetti, errori 4xx, richiesta file `.env`), ma sotto soglie critiche. Potrebbe trattarsi di un bot a bassa intensità o di un utente dubbio.

7. Altri bot (probabilmente benigni) Client con score $\geq$1 ma senza frequenza o eventi anomali rilevanti. Include strumenti di monitoraggio o crawler specializzati a bassa intensità.

8. Utente legittimo Tutti gli altri casi. Include traffico distribuito, score basso e assenza di segnali sospetti.

4.4 Contributo apportato

Nel codice sopra descritto, rispetto agli algoritmi presenti in letteratura, ha apportato alcuni miglioramenti.

Rispetto ai pattern di accesso dei Crawler Web ritengo di aver dato il mio contributo aggiungendo il riconsicmento di alcuni particolari comportamenti quali il riconoscimento delle richieste di file speciali (capitoli 4.2.8 e 4.2.9) e il riconoscimento della varietà delle richieste (capitolo 4.2.5).

Rispetto all'algoritmo di classificazione, ritengo di aver contributo modificando le tecniche di categorizzazione presenti, creando il sistema di classificazione basato sul punteggio (score) e soglie prestabilite.

Il codice proposto non è ancora completo e presenta alcune lacune, che potrebbero essere migliorate con il tempo e con l'esperienza.

Capitolo 5

Validazione

In questa sezione vengono presentati sia i risultati ottenuti dal sistema di classificazione euristica, che l'analisi effettuata per indentificare la correttezza dei risultati. L'analisi si articola in due livelli: uno `macroscopico`, volto a fornire una visione aggregata e statistica delle connessioni classificate, e uno `microscopico` finalizzato alla valutazione puntuale di singoli casi e al confronto con fonti esterne, come AbuseIPDB, per l'individuazione di eventuali falsi positivi o falsi negativi.

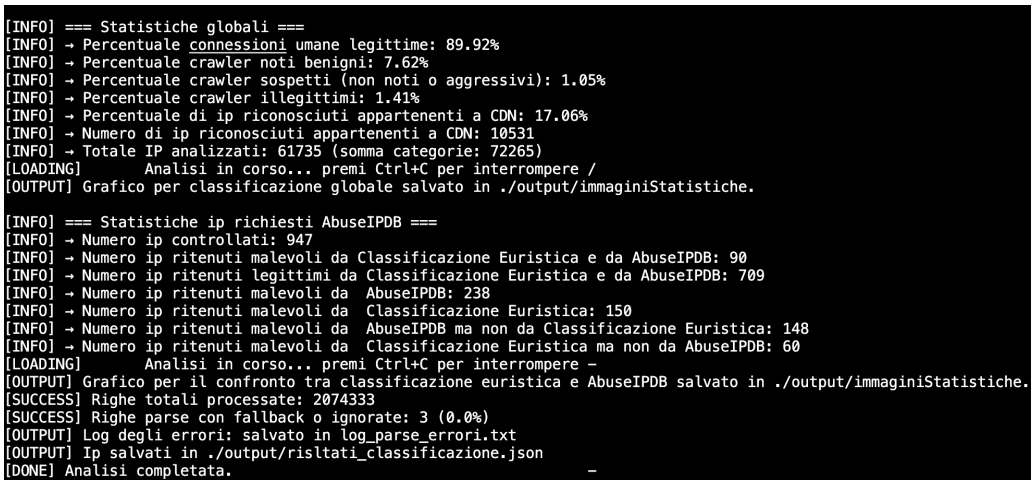

Figura 5.10: Output del sistema di classificazione (estratto)

5.1 Analisi macroscopica

Per effettuare un'analisi su scala globale, sono stati integrati nel codice diversi contatori statistici in grado di tracciare: il numero di righe di log lette, le righe effettivamente analizzate, quelle contenenti anomalie di parsing, il numero complessivo di connessioni, gli indirizzi IP unici osservati e molte altre metriche. Questi strumenti hanno permesso di costruire una panoramica quantitativa utile a garantire che nessuna connessione significativa venisse trascurata.

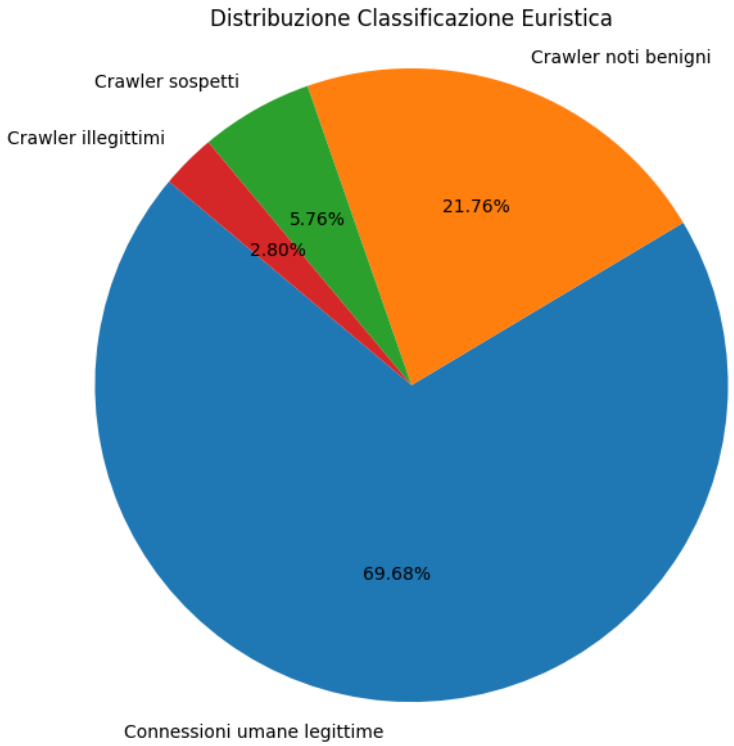

Figura 5.11: Distribuzione globale delle classificazioni

La figura 5.11 mostra la distribuzione percentuale delle categorie di traffico rilevate in una tipica giornata, su una delle macchine messe a disposizione dall'azienda. Nello specifico:

- **Connessioni umane legittime**: 68,68%.
- **Crawler benigni**: 21,76%.
- **Crawler sospetti**: 6,76%.
- **Crawler illegittimi**: 2,80%.

Va sottolineato che tali percentuali si riferiscono a una specifica giornata di osservazione e possono variare sensibilmente nel tempo. In particolare, picchi anomali nella percentuale di crawler sospetti o illegittimi possono rappresentare un potenziale indicatore di attacchi automatizzati in corso.

5.1.1 Confronto con AbuseIPDB

Per affinare ulteriormente la validazione del sistema e stimare la presenza di falsi positivi o negativi, è stato integrato un confronto con le risposte ottenute dalla piattaforma *AbuseIPDB*, tramite interrogazioni API automatizzate.

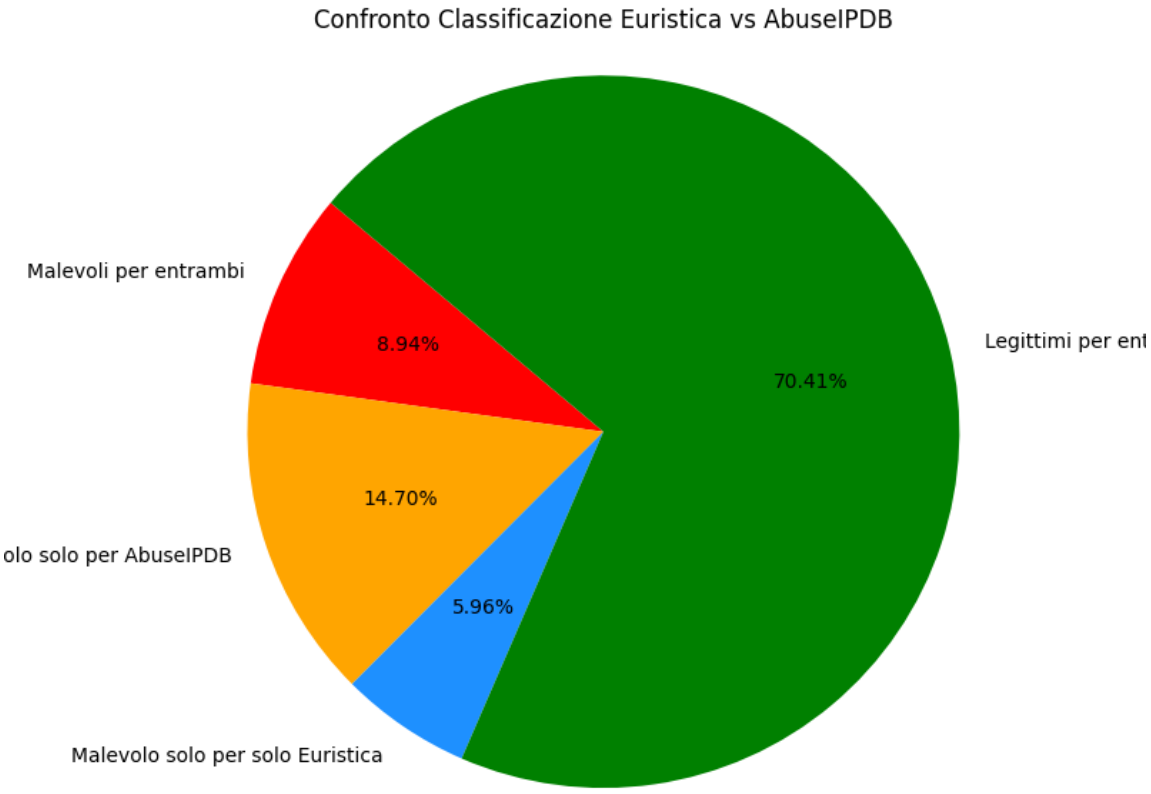

Figura 5.12: Confronto tra classificazione euristica e AbuseIPDB

AbuseIPDB è un servizio che raccoglie segnalazioni da parte degli utenti su indirizzi IP ritenuti malevoli. Sebbene sia uno strumento utile, presenta dei limiti: un indirizzo che non è stato ancora segnalato potrebbe risultare "legittimo" nella loro piattaforma, pur manifestando comportamenti sospetti nel nostro sistema.

Conclusione. L'integrazione con AbuseIPDB ha fornito un importante punto di confronto per validare i risultati ottenuti. Tuttavia, va ricordato che tale servizio si basa su segnalazioni volontarie, e non rappresenta una verità assoluta. Il sistema di classificazione

euristica proposto si è dimostrato in grado di identificare comportamenti sospetti anche in assenza di segnalazioni esterne, evidenziando la sua utilità come strumento proattivo di monitoraggio.

5.2 Analisi microscopica

Una volta verificato il corretto funzionamento dei contatori globali e la coerenza statistica generale, è stata condotta un'analisi più approfondita su singole connessioni, al fine di esaminare casi limite o particolarmente significativi. Questa fase si è basata su un set di informazioni aggiuntive, tra cui:

- **Crawler Score** e i relativi motivi del punteggio
- **Paese di provenienza (Country)**
- **Autonomous System Number (ASN)**
- **PTR record**
- **Verifica AbuseIPDB**

Va sottolineato che nessuna di queste informazioni modifica direttamente la classificazione finale. Tuttavia, esse forniscono un contesto utile per l'analisi qualitativa e possono essere determinanti nell'identificazione di falsi positivi o negativi. Alcuni elementi, come il numero di tentativi di login o la richiesta del file `.env`, mantengono invece un ruolo oggettivo e deterministico nella classificazione.

Nelle sezioni seguenti, i riferimenti saranno fatti all'elenco 5.6.

5.2.1 Crawler Score con motivazioni dettagliate

Il campo *Crawler Score* sintetizza, tramite un punteggio numerico, il comportamento anomalo rilevato per ogni client. Come mostrato nel seguente estratto, il punteggio totale rispecchia i comportamenti di uno o più pattern considerati sospetti:

```
1 Crawler score: 11 (HTML/img alto, referer assenti (>75%), molte richieste (
      senza asset) con frequenza sostenuta, molti URL unici, tentativi login
      sospetti, molti POST (possibile attacco), tentativo accesso a file .env)
```

Code 5.5: Crawler Score

Un punteggio così elevato indica un comportamento marcatamente aggressivo, riconducibile a un bot malevolo.

5.2.2 Country

Il campo Country indica la nazione di provenienza di un determinato indirizzo IP, tale informazione può essere rilevante in determinati contesti. In particolare, il sistema permette di bloccare il traffico proveniente da paesi considerati a rischio o inaccessibili per motivi geopolitici. Tali restrizioni possono essere configurate manualmente tramite il file */list/nazionibloccate.txt*.

5.2.3 ASN

Il campo ASN (Autonomous System Number) identifica il numero del sistema autonomo a cui appartiene l'IP. Questo valore può essere utile per riconoscere traffico proveniente da infrastrutture cloud, provider noti o entità sospette. Un ASN associato a Google, AWS o altri grandi operatori può indicare l'origine di un bot automatizzato legittimo, ma anche di uno malevolo.

5.2.4 PTR

Il campo PTR (Pointer Record) rappresenta il nome a dominio inverso associato all'indirizzo IP. Lo studio di questo campo può fornire ulteriori indizi sull'origine del traffico. In particolare, la presenza di riferimenti a domini come `googlebot.com` o `amazonaws.com` può aiutare a distinguere tra bot noti e sconosciuti.

5.2.5 AbuseIPDB

Come già anticipato, il sistema effettua interrogazioni a AbuseIPDB per ogni IP selezionato. Il seguente esempio (elenco 5.6) mostra un caso in cui un indirizzo è stato classificato come malevolo dal sistema euristico, ma non risulta segnalato nella piattaforma esterna:

```
IP: -.-.-.-
   IP origine CDN multipla: ...
   Numero richieste: 62239 in delta tempo: 23h 59min 57s
   Frequenza richieste: 0.72 req/sec
   Identificazione: Crawler identificato (bot, googlebot)
   Clicks: 62239
   Html img ratio: 30.13
   Pdf ps percent: 0.02
   Error 4xx percent: 0.22
   Head percent: 0.77
   Referer null percent: 97.52
   Robots requested: 0
   Unique url count: 54822
   Login attempts: 2391
   Post count: 54526
   Env file requested: 5
   Country: IT
   ASN: AS15169 GOOGLE
   PTR: crawl-66-249-68-72.googlebot.com
         Probabile provider cloud
   Crawler score: 11 (HTML/img alto, referer assenti (>75%), molte richieste (
    senza asset) con frequenza sostenuta, molti URL unici, tentativi login
    sospetti, molti POST (possibile attacco), tentativo accesso a file .env)
   Classificazione euristica: Crawler noto con comportamento malevolo
   [AbuseIPDB] Abuse Score: 0, Whitelisted: None
```

Code 5.6: Esempio di output dettagliato per un IP classificato come crawler malevolo

Nella conclusione del precedente estratto è possibile notare che lo score della connessione è molto alto (oltre 10), con una serie di comportamenti che rendono la connessione un tentativo di eccesso all'infrastruttura. E' anche possibile notare lo score assegnato a tale indirizzo da AbuseIPDB. Questo è un esempio in cui AbuseIPDB non ha ancora ricevuto segnalazioni da utenti per questo indirizzo. In una fase successiva potrebbe essere

integrata anche una segnalazione automatica al tool utilizzato per dare un contributo alla mappatura degli indirizzi malevoli.

5.3 Accuratezza dei risultati

La valutazione delle prestazioni del sistema di classificazione è stata condotta utilizzando una **matrice di confusione**, da cui sono state calcolate le principali metriche: *accuratezza*, *precisione*, *recall*, *F1-score*, *specificità* e *tasso di errore*.

5.3.1 Dati utilizzati

Per construire la matrice di confusione 5.13 sono stati utilizzati un set di 1000 dati estratti in maniera casuale da un file di log generato in una giornata. Tali classificazioni sono state estratte da varie fasi della giornata, poichè tra le ore notturne e le ore giornaliere potrebbe modificarsi la tipologia e l'intensità delle scansioni. Tale matrice è stata costruita analizzando i vari parametri elencati precedentemente ed, eventualmente non fossero bastati per una classifcazione accurata, è stato analizzato il file di Log di Apache per analizzare il Log grezzo.

5.3.2 Matrice di Confusione

La matrice di confusione è uno strumento che permette di confrontare le classi *predette* dal modello con le classi *reali*. La matrice creata è riportata nella Figura 5.13, è composta da quattro classi:

- **Legittimo** – traffico umano regolare,
- **Benigno** – bot noti e corretti,
- **Sospetto** – traffico ambiguo o opaco,
- **Malevolo** – crawler con comportamento dannoso.

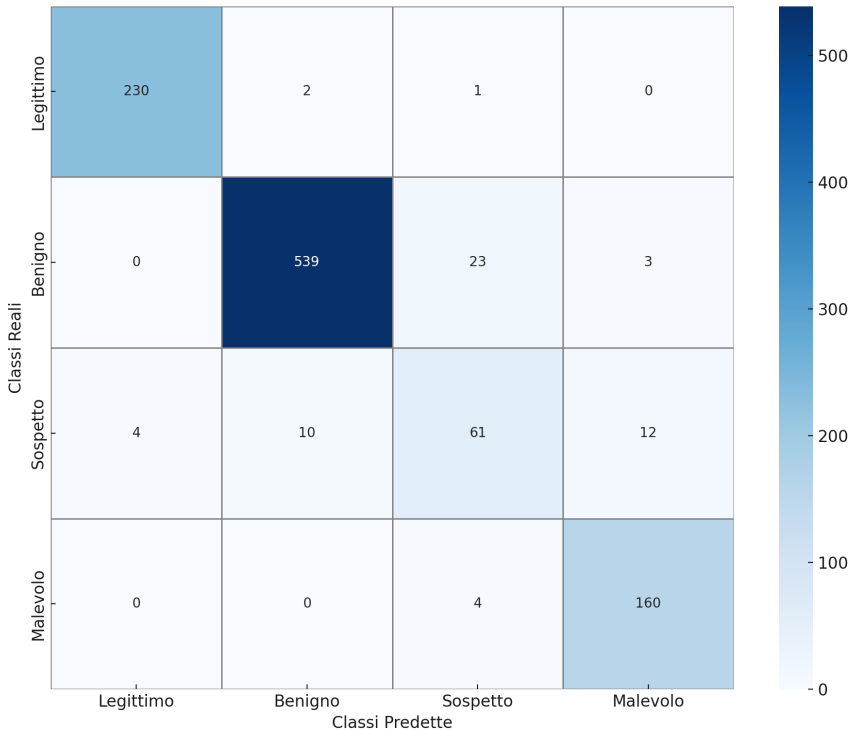

Figura 5.13: Matrice di confusione del sistema di classificazione

5.3.3 Metriche utilizzate

Le seguenti metriche sono state calcolate per ciascuna classe, interpretando ogni classe a turno come positiva e le altre come negative (approccio one-vs-all).

TP (True Positive) : casi positivi correttamente classificati

FP (False Positive) : casi negativi erroneamente classificati come positivi

FN (False Negative) : casi positivi classificati come negativi

TN (True Negative) : casi negativi correttamente esclusi

Le formule usate sono le seguenti:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$
$$\text{Precisione} = \frac{TP}{TP + FP}$$
$$\text{Recall} = \frac{TP}{TP + FN}$$
$$\text{F1-score} = \frac{2 \cdot \text{Precisione} \cdot \text{Recall}}{\text{Precisione} + \text{Recall}}$$
$$\text{Specificità} = \frac{TN}{TN + FP}$$
$$\text{Errore} = 1 - \text{Accuracy}$$

5.3.4 Metriche per Classe

La Tabella 5.1 riassume i risultati ottenuti per ciascuna classe:

Classe	**TP**	**FP**	**FN**	**TN**	**Accuracy**	**Precisione**	**Recall**	**F1-score**	**Specificità**	**Errore**
Legittimo	230	4	3	812	0.993	0.983	0.987	0.985	0.995	0.007
Benigno	539	12	26	472	0.964	0.978	0.954	0.966	0.975	0.036
Sospetto	61	28	26	934	0.949	0.685	0.701	0.693	0.971	0.051
Malevolo	160	15	4	870	0.982	0.914	0.976	0.944	0.983	0.018

Tabella 5.1: Metriche di classificazione per ciascuna classe

5.3.5 Visualizzazione delle Metriche

La Figura 5.14 mostra, tramite istogrammi comparativi, le performance aggiornate in precisione, recall e F1-score per ciascuna classe.

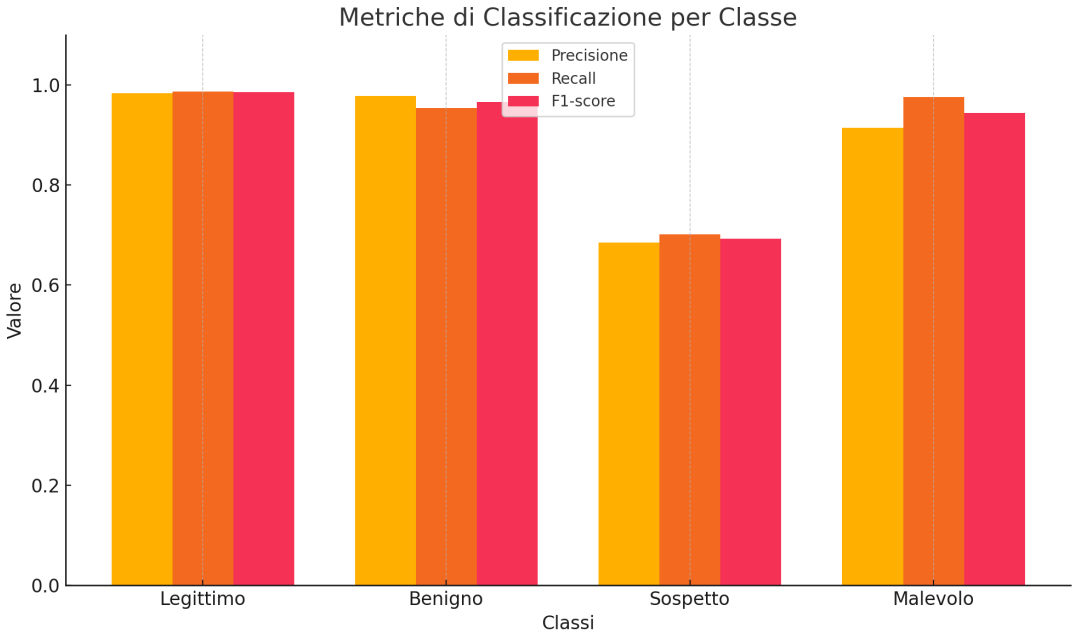

Figura 5.14: Confronto tra Precisione, Recall e F1-score per ogni classe.

5.4 Analisi conclusiva

I risultati evidenziano un'elevata accuratezza globale del sistema, con performance ottime nelle classi *Legittimo*, *Benigno* e *Malevolo*. La classe *Sospetto* risulta più difficile da distinguere, mostrando una precisione inferiore (0.656), ma un buon recall (0.833), coerente con la natura ambigua della categoria stessa.

Capitolo 6

Implementazione Bridge Trasparente

Questo capitolo descriverà l'implementazione della fase dello shaping degli indirizzi IP identificati come crawler sospetti o malevoli.

Tale capitolo ha lo scopo di integrare la funzionalità della *limitazione* al fine di rendere il progetto completo e funzionante. Tale sezione ritengo sia un capitolo aggiuntivo e separato del riconoscimento dei crawler web, e per tale motivo non contribuisce creando nuove funzionalità, bensì rende il progetto funzionante.

6.1 Configurazione Di Rete

Per l'implementazione del sistema di limitazione, mi sono state fornite tre macchine distinte:

- **Web Server**
- **Bridge**
- **Testlab**

L'immagine 2.6 fornisce una rappresentazione concettuale dello scenario di una connessione reale, mentre l'immagine 6.15 mostra la configurazione reale della rete di test. La macchina **Testlab** simula un normale client esterno alla rete che effettua connessioni a un sito web ospitato sul **Web Server**. Nello scenario ricreato, le macchine **Web Server** e **Bridge srv**, simulano una rete privata. Il **Bridge srv** funge da firewall che filtra le connessioni in entrata, mentre sul **Web Server** è in esecuzione il programma responsabile del riconoscimento dei crawler. Una volta identificati gli IP da limitare, il programma invia delle richieste API al **Bridge srv** tramite chiamate REST, utilizzando un endpoint dedicato, che gestisce l'aggiunta o la rimozione delle limitazioni per ciascun indirizzo IP. Ogni IP viene classificato come **Malevolo** oppure **Sospetto**. Ad ogni categoria è associato

un limite di connessioni per secondo chiamato *Rate* e un *Burst*, che rappresenta un picco considerato legittimo per un breve periodo di tempo.

Ad entrambe le categorie è anche associato un tempo di permanenza, dopo il quale ogni limitazione viene rimossa in maniera automatica. Entrambi valori, Limiti e Tempi di permanenza, è possibile modificarli tramite il file di configurazione.

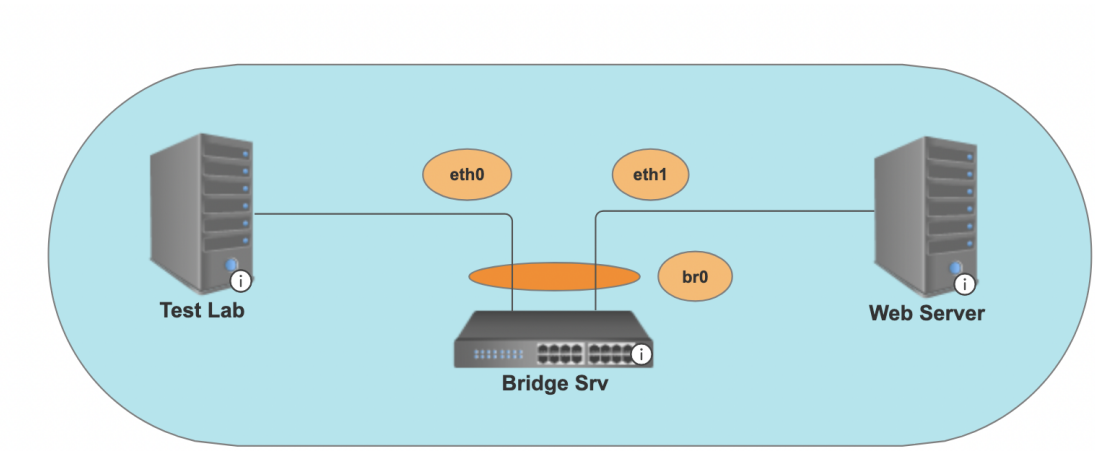

Figura 6.15: Configurazione di rete

6.2 Configurazione del Bridge

Il bridge, nello scenario che abbiamo ricreato, agisce come punto di passaggio tra la rete esterna e quella interna. Il suo compito è trasmettere i pacchetti dalla rete esterna verso quella interna, applicando le limitazioni del traffico impostate tramite iptables e trasmettere invece tutti i pacchetti da quella interna a quella esterna.

In pratica **Bridge Srv** ha due interfacce unite in bridge, una collegata allo switch della rete pubblica (nel nostro esempio Test Lab), l'altra collegata al **Webserver**. In questo modo il traffico diretto al **Webserver** passa dal'interfaccia esterna a quella interna e viceversa. Operando a livello2 il **Bridge Srv** è come se fosse uno switch e risulta trasparente al webserver.

Le limitazioni di traffico, come ad esempio i limiti di banda per determinati indirizzi IP o classi (come sospetto e malevolo), vengono applicate direttamente su **Bridge Srv**. Le

limitazioni vengono gestite a livello di tabella FORWARD di iptables, che è utilizzata per applicare regole di filtraggio sulle connessioni tra una rete e l'altra, cioè tra bridge e Web Server. Questo permette di garantire che solo il traffico che soddisfa determinati criteri di velocità e accesso possa raggiungere il Web Server. Le interfacce di rete presenti sono le seguenti:

```
br: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::e48b:e8ff:fed3:526c  prefixlen 64  scopeid 0x20<link>
        ...

enp5s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        ether 00:16:3e:9b:f8:cb  txqueuelen 1000  (Ethernet)
        ...

enp6s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        ether 00:16:3e:7d:0b:9f  txqueuelen 1000  (Ethernet)
        ...

enp7s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet -.-.-.-  netmask 255.255.254.0  broadcast 151.11.53.255
        inet6 fe80::216:3eff:fe77:5065  prefixlen 64  scopeid 0x20<link>
        ether 00:16:3e:77:50:65  txqueuelen 1000  (Ethernet)
        ...

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        ...
```

Code 6.7: Interfacce di Rete Bridge

6.3 Endpoint

L'endpoint è stato implementato tramite **Uvicorn** , un server HTTP ASGI ad alte prestazioni, utilizzato per gestire le chiamate REST in entrata. Grazie a tale servizio, il programma che si occupa di riconoscere i crawler, una volta identificato come tale, invia una chiamata HTTP all'Endpoint di tipo POST:

```
1 http POST http://151.11.53.1:8000/limit ip_class=151.11.53.3 queue=malevolo
```

Code 6.8: HTTP POST

Quando il programma riceve la richiesta HTTP POST esegue un Thread che segue il seguente flusso: 6.16.

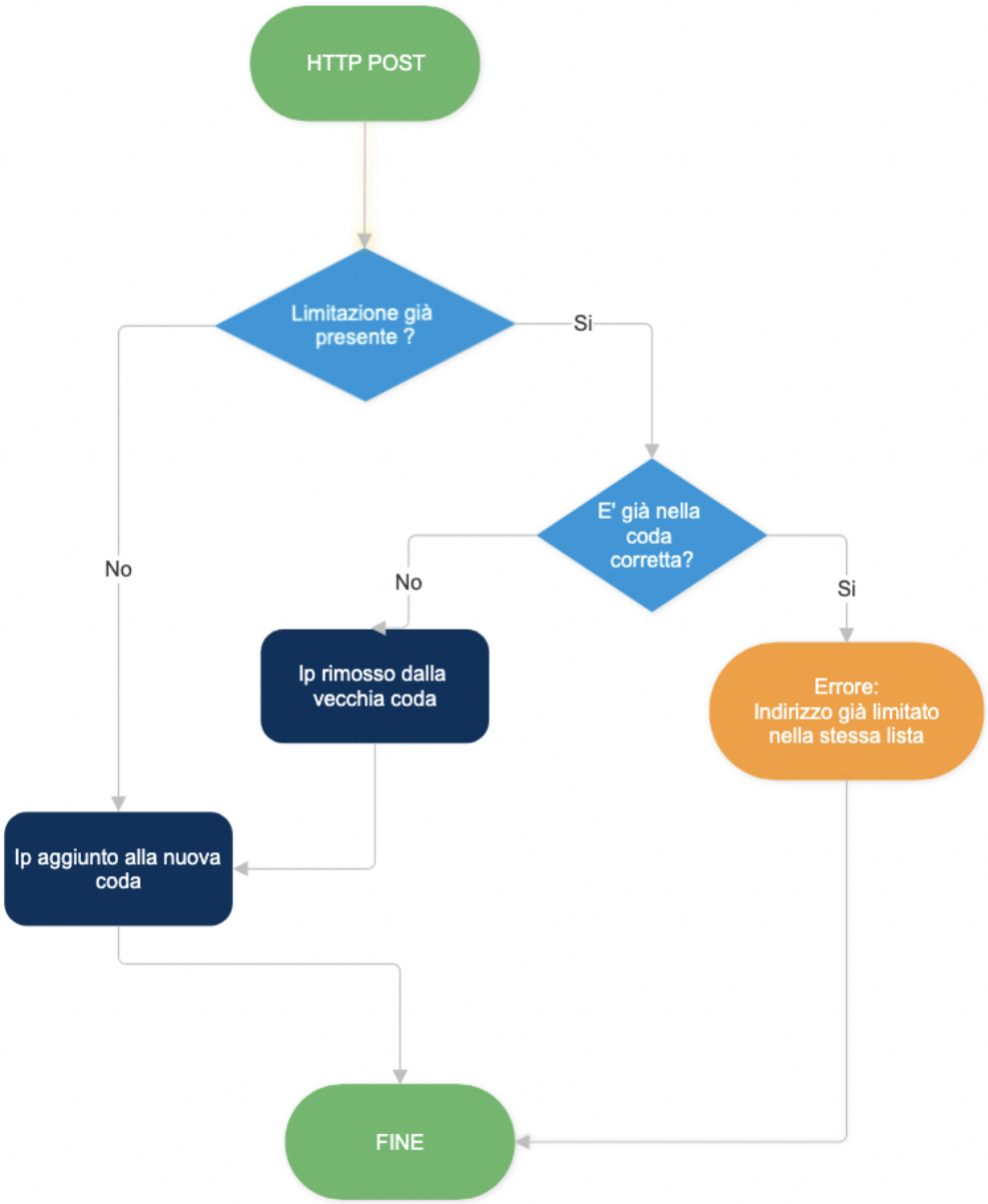

Figura 6.16: Flusso logico Post EndPoint

6.3.1 Regole di limitazione

Le regole di limitazione sono gestite tramite il comando **iptables** e sono applicate al traffico TCP proveniente da IP identificati come crawler. In particolare, la configurazione prevede l'uso del modulo **hashlimit**, che consente di impostare limiti di traffico basati sulla frequenza delle richieste da parte di un singolo IP.

Di seguito viene mostrato il codice che implementa le regole di limitazione:

```
# Regola con hashlimit per TCP
        subprocess.run([
            "iptables", "-A", "FORWARD", "-s", ip,
            "-p", "tcp",
            "-m", "hashlimit",
            "--hashlimit", f"{rate}/sec",
            "--hashlimit-burst", burst,
            "--hashlimit-mode", "srcip",
            "--hashlimit-name", label,
            "-j", "ACCEPT"
        ], check=True)

        # Regola di fallback DROP per TCP
        subprocess.run([
            "iptables", "-A", "FORWARD", "-s", ip,
            "-p", "tcp",
            "-j", "DROP"
        ], check=True)
```

Code 6.9: Codice dell'Endpoint

Il codice sopra riportato definisce una regola di limitazione utilizzando il modulo **hashlimit**, che limita la frequenza delle richieste per ogni IP. La regola di fallback applica invece una limitazione totale (DROP) per gli IP che superano i criteri di limitazione. Concettualmente le due regole di limitazione citate vengono applicate in sequenza:. la prima regola viene applicata a tutte le nuove connessioni finché rientrano nei limiti. Una volta superati i limiti, tutte le nuove connessioni passano automaticamente alla seconda regola, la quale applicherà l'azione di DROP per tutti i nuovi pacchetti.

Nel codice descritto, sono state implementate due diverse classi di limitazione:

- limitazione per IP **Malevoli**
- limitazione per IP **Sospetti**

Ad ogni limitazione è stato associato sia un valore medio di pacchetti al secondo (**Rate**), sia un parametro di **burst**, ovvero un picco temporaneo di traffico che può essere tollerato per brevi intervalli di tempo. L'immagine 6.17 illustra i limiti.

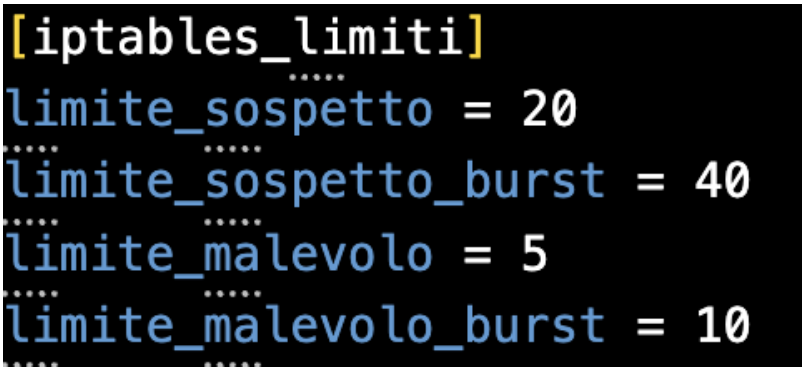

Figura 6.17: Limiti medio e burst associati alle classi

Ad ogni classe di limitazione è stato anche associato un tempo di permanenza [6.18]. Ogni qualvolta che una limitazione viene aggiunta, viene contemporaneamente salvato il *timestamp* relativo all'aggiunta della limitazione. Al termine del periodo di **Timeout** la limitazione viene rimossa tramite un thread separato, in moda da non bloccare l'intero processo.

Figura 6.18: Tempi di permanenza delle classi di limitazione

E' importante notare che ogni valore mostrato dei seguenti dati è a scopo illustrativo, e grazie alla natura di adattabilità del codice è possibile modificare ogni valore a seconda della necessità.

Capitolo 7

Sperimentazione conclusiva

La fase di Sperimentazione finale è stata condotta mediante un complesso esperimento, in quanto non è stato possibile simulare un vero e proprio flusso di dati in tempo reale.

Come mostrato nell'immagine 2.6 la macchina **TestLab** rappresenta un client illegittimo che tenta di scansionare un sito web situato sul **Server Apache**. Per replicare lo streaming è stato creato, tramite uno script python, un file di log che viene popolato prelevando le righe file di log reale giornaliero e stampate dinamicamente ad una velocità variabile. Uno degli indirizzi IP presenti nel file è stato modificato sostituendolo con l'indirizzo della macchina **TestLab**.

Simultaneamente dalla macchina **TestLab** è stato lanciato il comando:

```
1 sudo hping3 -S 150.11.53.1 -p 80 -c 100-i u1000
```

Code 7.10: Ping di Test

Il comando 7.10 nel dettaglio:

- **hping3**:Strumento di generazione di pacchetti per test di rete e diagnostica;
- **-S**: Specifica il tipo di pacchetto, in questo caso **TCP SYN**;
- **150.11.53.1**: Indirizzo destinatario;
- **-p 80**: Specifica la porta da di detinazione
- **-c 100**: Specifica il numero di pacchetti da inviare;
- **-i u1000**: Imposta l'intervallo tra l'invio di pacchetti successivi a 1 millisecondo (1000 microsecondi);

Nel momento in cui il codice adibito al riconoscimento dei crawler riconosce l'indirizzo della macchina come tale, invia il comando POST e il comando di hping3 passa da

avere lo 0% di pacchetti persi al 96% di perdita. Le immagini 7.11 e 7.12 mostrano tale cambiamento.

```
HPING 151.11.53.65 (enp5s0 151.11.53.65): S set, 40 headers + 0 data bytes
len=40 ip=150.11.53.1 ttl=64 DF id=0 sport=80 flags=RA seq=0 win=0 rtt=0.7 ms
len=40 ip=150.11.53.1 ttl=64 DF id=0 sport=80 flags=RA seq=1 win=0 rtt=0.5 ms
len=40 ip=150.11.53.1 ttl=64 DF id=0 sport=80 flags=RA seq=2 win=0 rtt=0.4 ms
len=40 ip=150.11.53.1 ttl=64 DF id=0 sport=80 flags=RA seq=3 win=0 rtt=1.3 ms
...
len=40 ip=150.11.53.1 ttl=64 DF id=0 sport=80 flags=RA seq=99 win=0 rtt=1.3 ms
--- 150.11.53.1 hping statistic ---
100 packets transmitted, 100 packets received, 0% packet loss
round-trip min/avg/max = 0.3/1.4/2.9 ms
```

Code 7.11: Output di Hping3 precedente a limitazione

```
HPING 150.11.53.1 (enp5s0 150.11.53.1): S set, 40 headers + 0 data bytes
len=40 ip=150.11.53.1 ttl=64 DF id=0 sport=80 flags=RA seq=11 win=0 rtt=1.0 ms
len=40 ip=150.11.53.1 ttl=64 DF id=0 sport=80 flags=RA seq=31 win=0 rtt=1.4 ms
len=40 ip=150.11.53.1 ttl=64 DF id=0 sport=80 flags=RA seq=51 win=0 rtt=2.9 ms
len=40 ip=150.11.53.1 ttl=64 DF id=0 sport=80 flags=RA seq=90 win=0 rtt=2.0 ms
--- 150.11.53.1 hping statistic ---
100 packets transmitted, 4 packets received, 96% packet loss
round-trip min/avg/max = 1.0/1.8/2.9 ms
```

Code 7.12: Output di Hping3 successivo a limitazione

7.1 Conclusione

L'implementazione del sistema di limitazione tramite il bridge trasparente ha rappresentato una soluzione efficiente per gestire il traffico web automatizzato, riducendo al minimo l'invasività sull'infrastruttura esistente. Grazie a questa soluzione, è stato possibile migliorare le performance dei server, garantendo al contempo la protezione contro il traffico indesiderato generato dai crawler e dai bot malevoli. La scelta di un'architettura modulare e scalabile consente una gestione flessibile e dinamica delle connessioni in ingresso, senza compromettere la qualità del servizio.

Capitolo 8

Conclusioni

Il progetto nasce dall'esigenza reale di limitare il traffico generato da Crawler, bot e utenti illegittimi , che vanno a saturare le risorse dell'infrastruttura, andando a penalizzare la qualità del servizio per tutti gli utenti legittimi (bot o umani). Il progetto si pone quindi come obiettivo il riconoscimento, mediante l'analisi delle interazioni, di tutte quelle connessioni che creano traffico anomalo, causando congestione della rete. Per tale riconoscimento è stato deciso di utilizzare un'euristica basata su alcuni pattern di comportamenti noti, prendendo come spunto l'articolo sopracitato. Una volta riconosciuto un IP, tramite l'uso di un endpoint, è possibile comunicare ad un Bridge che l'indirizzo IP in questione è stato riconosciuto come malevolo o sospetto. A ciascuna categoria viene applicata una limitazione per numero di pacchetti al secondo. L'approccio fondamentale scelto consiste nel non bloccare mai completamente nessun indirizzo, ma nel limitare drasticamente il numero di pacchetti al secondo che possono passare. Ogni limitazione, inoltre, è temporizzata, con un tempo di permanenza definito, dopo il quale la limitazione viene rimossa. Il progetto è stato sviluppato in collaborazione con l'azienda Hosting Solution, inizialmente concepito come argomento per un tirocinio sotto la supervisione di Ing. Francesco Leoncino. Per motivi pratici il progetto è passato da essere un tirocinio ad una tesi, ed è stato ridotto, portando l'attenzione alla fase del riconoscimento. Per tale motivo la parte dell'endpoint è stata implementata in maniera meno dettagliata e con uno studio meno approfondito, al fine di lasciare all'azienda un progetto concreto e funzionante, lasciando quindi spazio a sviluppi e miglioramenti futuri. Ing. Leoncino, Responsabile tecnico di Hosting Solutions, è stata la figura di riferimento all'interno dell'azienda.

8.1 Limiti del progetto

Il progetto presenta una serie di limiti basati sul tipo di algoritmo scelto per il riconoscimento delle connessioni. E' stato scelto un algortimo euristico piuttosto che un approccio basato su deep learnig, il quale, a seguito di un addestramento, sarebbe stato in grado di riconoscere le connessioni in autonomia. Questa scelta ha reso però il progetto più flessibile, poichè un algoritmo basato su IA è molto rigido e complesso da modificare per essere adattato a diversi contesti. Un altro limite del progetto è legato al volume connessioni da analizzare. L'enorme quantità di dati generati è risultata un ostacolo importante nella valutazione dei risultati. L'utilizzo di un'eventuale IA avrebbe potuto rendere tale fase automatica e più accurata.

8.2 Sviluppi futuri

Come anticipato nella sezione precedente, uno degli sviluppi futuri consiste nell'integrazione di algoritmi basati sull'intelligenza artificiale, sia nel riconoscimento di errori di valutazione, che algoritmi basati su Deep Learnig per riconoscimetno di Crawler. L'introduzione di tali algoritmi permetterebbe di rendere il sistema più autonomo e preciso nel distinguere tra traffico legittimo e traffico automatizzato. Un ulteriore sviluppo potrebbe riguradare la fase di shaping del traffico, estendendola a tutti i protocolli di rete, il miglioramento della sicurezza dell'endpoint e della sua usabilità.

Bibliografia

[1] Michael Weinberg e GLAM-E Lab. *Are AI Bots Knocking Cultural Heritage Offline?* Rapp. tecn. Online report, last accessed June 2025. GLAM-E Lab (New York University + University of Exeter), 2025. URL: https://www.glamelab.org/products/are-ai-bots-knocking-cultural-heritage-offline/.

[2] Vito La Vecchia. *Caratteristiche e differenza tra web crawling e web scraping*. Accesso il 4 giugno 2025. 2025. URL: https://vitolavecchia.altervista.org/caratteristiche-e-differenza-tra-web-crawling-e-web-scraping/.

[3] Trupti V Udapure, Ravindra D Kale e Rajesh C Dharmik. «Study of web crawler and its different types». In: *IOSR Journal of Computer Engineering* 16.1 (2014), pp. 01–05.

[4] Wei Xia, Fei Zhao, Haishuai Wang, Peng Zhang, Anhui Wang e Kang Li. «Crawler detection in location-based services using attributed action net». In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2021, pp. 4234–4242.

[5] Gm Informatica. *Server Proxy, Reverse Proxy, Forward Proxy: scopriamo cosa sono*. Accesso il 17 giugno 2025. 2024. URL: https://gmadv.it/it/sistemi-operativi/server-proxy-reverse-proxy-forward-proxy-scopriamo-cosa-sono/.

[6] StrongDM. *What's the Difference Between Proxy and Reverse Proxy?* Accesso il 17 giugno 2025. 2024. URL: https://www.strongdm.com/blog/difference-between-proxy-and-reverse-proxy.

[7] TN Solutions. *Che cos'è HAProxy e come funziona*. Accesso il 17 giugno 2025. 2024. URL: https://www.tnsolutions.it/it/che-cose-haproxy/.

[8] Wikipedia contributors. *Bridge (informatica) – Wikipedia*. Ultimo accesso il 17 giugno 2025. 2024. URL: https://it.wikipedia.org/wiki/Bridge_(informatica).

[9] The Linux Information Project. *Bridge Definition*. Accesso il 17 giugno 2025. 2024. URL: https://www.linfo.org/bridge.html.

[10] IBM. *Che cos'è un API Endpoint?* Accesso il 17 giugno 2025. 2024. URL: https://www.ibm.com/it-it/think/topics/api-endpoint.

[11] HTML.it. *I principi dell'architettura RESTful*. Accesso il 17 giugno 2025. 2024. URL: https://www.html.it/pag/19596/i-principi-dellarchitettura-restful/.

[12] Wikipedia contributors. *Apache HTTP Server — Wikipedia*. Ultimo accesso il 17 giugno 2025. 2024. URL: https://it.wikipedia.org/wiki/Apache_HTTP_Server.

[13] Linode. *How to Install and Configure Mod$_e$vasiveonApache*. Ultimo accesso il 17 giugno 2025. 2024. URL: https://www.linode.com/docs/guides/modevasive-on-apache/.

[14] John Leach. *Rate Limiting with Apache and Mod Security*. Ultimo accesso il 17 giugno 2025. 2012. URL: https://johnleach.co.uk/posts/2012/05/15/rate-limiting-with-apache-and-mod-security/.

[15] Merve Bas Seyyar, Ferhat Özgür Çatak e Ensar Gül. «Attack-oriented scan detection from Apache HTTP server access logs». In: *Applied Computing and Informatics* 14.1 (2018), pp. 28–36. DOI: 10.1016/j.aci.2017.04.002.

[16] M. Auxilia e D. Tamilselvan. «Anomaly detection using negative security model in web application». In: *2010 International Conference on Computer Information Systems and Industrial Management Applications (CISIM)*. IEEE, 2010, pp. 481–486. DOI: 10.1109/CISIM.2010.5643461.

[17] Merve Baş Seyyar, Ferhat Özgür Çatak e Ensar Gül. «Detection of attack-targeted scans from the Apache HTTP Server access logs». In: *Applied computing and informatics* 14.1 (2018), pp. 28–36.

[18] Aleide Web Agency. *Cloudflare: Come Funziona e i Vantaggi*. Ultimo accesso il 17 giugno 2025. 2024. URL: https://www.aleidewebagency.com/web-agency-milano/sicurezza/cloudflare-come-funziona-vantaggi.html.

[19] E. Liu, E. Luo, S. Shan, G. M. Voelker, B. Y. Zhao e S. Savage. «Somesite I Used To Crawl: Awareness, Agency and Efficacy in Protecting Content Creators From AI Crawlers». In: *IMC '25*. Madison, WI, USA, 2025.

[20] Bahar Amin Azad, Oliver Starov, Pierre Laperdrix e Nick Nikiforakis. «Web Runner 2049: Evaluating Third-Party Anti-bot Services». In: *NDSS Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb 2020)*. San Diego, United States, 2020, pp. 138–159.

[21] Wikipedia contributors. *Honeypot — Wikipedia*. Ultimo accesso il 17 giugno 2025. 2024. URL: https://it.wikipedia.org/wiki/Honeypot#:~:text=In%20informatica%2C%20un%20honeypot%20(letteralmente,gli%20attacchi%20di%20pirati%20informatici..

[22] Elisa Chiapponi. «Detecting and Mitigating the New Generation of Scraping Bots». Tesi di dott. Sorbonne Université, 2023.

[23] Harm Griffioen e Christian Doerr. «Could you clean up the Internet with a Pit of Tar? Investigating tarpit feasibility on Internet worms». In: *2023 IEEE Symposium on Security and Privacy (SP)*. 2023, pp. 2551–2565. DOI: 10.1109/SP46215.2023.10179467.

[24] Stytch. *How to Block AI Web Crawlers*. Ultimo accesso il 17 giugno 2025. 2024. URL: https://stytch.com/blog/how-to-block-ai-web-crawlers/#honeypotsandtarpits.

[25] Dusan Stevanovic, Natalija Vlajic e Aijun An. «Detection of malicious and non-malicious website visitors using unsupervised neural network learning». In: *Applied Soft Computing* 13.1 (2013), pp. 698–708. DOI: 10.1016/j.asoc.2012.08.028.

[26] Dusan Stevanovic, Aijun An e Natalija Vlajic. «Feature evaluation for web crawler detection with data mining techniques». In: *Expert Systems with Applications* 39.10 (2012), pp. 8707–8717. DOI: 10.1016/j.eswa.2012.01.210.

[27] Jing Zhao, Rui Chen e Pengcheng Fan. «TS-Finder: privacy enhanced web crawler detection model using temporal–spatial access behaviors». In: *The Journal of Supercomputing* 80 (2024), pp. 17400–17422. DOI: 10.1007/s11227-024-06133-6.

[28] Authors not specified in provided excerpt. «Canopy: A Distributed Solution for Low-and-Slow DDoS Attack Mitigation». In: *ICISSP 2021 - 7th International Conference on Information Systems Security and Privacy*. Authors not specified in the provided excerpt. 2021.

[29] Inwoo Ro, Joong Soo Han e Eul Gyu Im. «Detection Method for Distributed Web-Crawlers: A Long-Tail Threshold Model». In: *Security and Communication Networks* 2018.1 (2018), p. 9065424.

[30] Wei-Zhou Lu e Shun-Zheng Yu. «Web robot detection based on hidden Markov model». In: *2006 International Conference on Communications, Circuits and Systems*. Vol. 3. IEEE. 2006, pp. 1806–1810.

[31] Dusan Stevanovic, Aijun An e Natalija Vlajic. «Feature evaluation for web crawler detection with data mining techniques». In: *Expert Systems with Applications* 39.10 (2012), pp. 8707–8717.

[32] Christian Bomhardt, Wolfgang Gaul e Lars Schmidt-Thieme. «Web robot detection-preprocessing web logfiles for robot detection». In: *New Developments in Classification and Data Analysis: Proceedings of the Meeting of the Classification and Data Analysis Group (CLADAG) of the Italian Statistical*

Society, University of Bologna, September 22–24, 2003. Springer. 2005, pp. 113–124.

[33] Grazyna Suchacka e Igor Motyka. «Efficiency Analysis Of Resource Request Patterns In Classification Of Web Robots And Humans.» In: *ECMS*. 2018, pp. 475–481.

[34] KyoungSoo Park, Vivek S Pai, Kang-Won Lee e Seraphin B Calo. «Securing Web Service by Automatic Robot Detection.» In: *USENIX Annual Technical Conference, General Track*. 2006, pp. 255–260.

[35] Athena Stassopoulou e Marios D Dikaiakos. «Web robot detection: A probabilistic reasoning approach». In: *Computer Networks* 53.3 (2009), pp. 265–278.

[36] Xigao Li, Babak Amin Azad, Amir Rahmati e Nick Nikiforakis. «Good bot, bad bot: Characterizing automated browsing activity». In: *2021 IEEE symposium on security and privacy (sp)*. IEEE. 2021, pp. 1589–1605.

Appendice A

Codice Sorgente

Per completezza, in queste sezione verranno presentati alcuni estratti del codice da me scritto al fine di non appesantire la tesi.

Il codice è stato scritto in *Python* e si divide in due programmi differenti, la parte responsabile del riconoscimento di crawler, e la parte dedicata al Traffic Shaping.

In questa sezione verranno mostrati i riferimenti al codice sorgente di Github, i file di configurazione di entrambi i progetti ed i relativi Makefiles.

Ogni contesto in cui vengono applicati i codici devono essere adattati, modificando le soglie a seconda della dimensione della rete.

A.1 Codice sorgente

Di seguito i link per accedere ad entrambi i repository di Github.

- **ARSCrawler** (Analysis and recognition of suspicious crawlers):

  `https://github.com/andluk01/ARSCrawler.git`

- **Bridge-api**:

  `https://github.com/andluk01/Bridge-api.git`

Entrambi i repository contengono i seguenti file fondamentali :

- Il file *Makefile*.
- Il file *requirements.txt*.
- Il file *configurazione.conf*.
- Il file *main.py*.

A.2 File di configurazione di ARSCrawler

```
#sezione dedicata a tutti i percorsi per i file
[percorsi]
log_file = ./Log/F11/global.log.1
cdn_path = ./lists/cdn_ranges.txt
crawler_file = ./lists/Crawlernoti.txt
country_db = ./lists/GeoLite2-Country.mmdb
asn_db = ./lists/GeoLite2-ASN.mmdb
whitelist_db = ./lists/whitelists.ips
providers_keywords = ./lists/cloud_providers_keywords.txt
nazioni_bloccate = ./lists/nazioni_bloccate.txt

#sezione dedicata a tutte le soglie
[classificazione]
login_soglia_sospetto = 2
login_soglia_malevolo = 5
post_soglia_sospetto = 7
post_soglia_malevolo = 30
frequenza_legittimo = 1
frequenza_illegittimo = 12
error_4xx_soglia_sospetto = 50
error_4xx_soglia_malevolo = 70

#sezione dedicata alla grandezza della finestra di default
[generale]
finestra_default = 15
```

Code A.13: File configurazione ARSCrawler

A.3 File di configurazione di Endpoint

```
[iptables_limiti]
limite_sospetto = 20
limite_sospetto_burst = 40
```

```
limite_malevolo = 5
limite_malevolo_burst = 10

[ip_time]
IP_TIMEOUT_SOSPETTO=60
IP_TIMEOUT_MALEVOLO=90
```

Code A.14: File configurazione Endpoint

A.4 Makefile

Il make file è il medesimo per entrambi i progetti. Di seguito viene mostrato l'output:

```
Comandi disponibili:
  make venv      - Crea un ambiente virtuale in venv
  make install   - Installa le dipendenze da requirements.txt
  make run       - Esegue il file main.py usando ambiente virtuale
```

Code A.15: File configurazione Endpoint

Ringraziamenti

Giunto finalmente alla fine di questo percorso ritengo importante ringraziare tutti voi per avermi accompagnato e sostenuto.

Prima di tutto ringrazio i miei genitori e tutti i miei parenti, non solo coloro che hanno pagato questa università, ma coloro che l'hanno resa possibile. Coloro che mi hanno insegnato come vivere, il rispetto per gli altri e mi hanno reso ciò che sono adesso, nel bene e nel male.

Ringrazio poi i miei tre fratelli, Filippo Elisa e Giacomo. I fratelli "veri", Filippo ed Elisa, che ci sono sempre stati e come fratelli mi hanno sempre fatto aprire gli occhi a cose che non avrei mai visto da solo e nessun altro mi avrebbe mai fatto notare. A Giacomo, il fratello mancato, un legame che poche persone posso dire avere. Quella persona che non serve che dica "io ci sono" perché sai che nel momento del bisogno c'è.

Un ringraziamento a Giulia, un amica, una ragazza, un sostengo. La ragazza con più pazienza che abbia mai incontrato. Tu hai sopportato i momenti di panico e di gioia senza pretendere niente in cambio.

Ringrazio i miei compagni di università, tutti i ragazzi del culto. Sebbene qualcuno non sia riuscito a finire l'università rimane comunque un compagno. In particolare ringrazio Zampetta, Samu, Gheb,Bando Bandolero, Tommi vic, Costi, Dile, Erne, Fefè, Matte, Matte, Paola, France, Cate, Loren (zo), Totò, Nicola... A voi dedico un grandissimo grazie perché con ognuno di voi ho passato bellissimi momenti, dai caffè alle macchinette, le infinite camminate nell'università all'esplorazione di luoghi misteriosi, le partite di ping pong, i pranzi con Rick e Morty, le feste di compleanno sotto la pioggia, nelle vigne e mille altre avventure che non si concluderanno adesso. Ma soprattutto a voi che non mi avete mai lasciato solo in questo posto e spero di aver fatto anch'io uguale per voi.

Ringrazio Ucci, Ale, Paolo e Antu, coloro che con una birra hanno sempre fatto dimenticare tutto. Voi che avete sempre cambiato i vostri piani per includere me con tutti i problemi, dalla schiena, alla pancia, ai piedi.

Ad Alle, l'amica dalle infinite passeggiate, che mi ha insegnato che nessuno deve mai

poterti mettere i piedi in testa e ogni tanto a chi se lo merito non è sbagliato rispondere a tono.

A Nicco, Rocco, Matte, Gaia e Marta che mi hanno insegnato che effettivamente ci si può divertire anche studiando, lanciandosi oggetti, nascondendo sedie e banchi... ma anche studiando.

Un grazie a Blu celeste, con cui ho passato serate e viaggi bellissimi e mai più riusciremo a vivere così felici e spensierati.

Ed infine un enorme grazie a tutti voi che non ho nominato, ma che avete dato il vostro piccolo contributo in questo lungo percorso.

Concludo con una perla: "la vita è come una spiaggia , calpestandola si lascia l'impronta", vivete la vita lasciando un segno la dove passate, non rimanete anonimi o sufficienti, fate la differenza.