POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Magistrale

Anomaly detection per il rilevamento di attacchi DDoS su reti aziendali

Relatore
prof. Guido Marchetto

Studente
Stefano Loscalzo
matricola: s267614

Supervisore aziendale
Tiesse S.p.A.
dott. ing. Francesco Lucrezia

Anno accademico 2020-2021

Abstract

Gli attacchi di denial of Service distribuiti (DDoS) sono uno dei maggiori problemi di sicurezza delle reti. Hanno lo scopo di impedire ad utenti legittimi l'accesso a dei servizi o degradare loro le prestazioni. Contestualmente, gli strumenti a disposizione di chi deve mitigare questo tipo di attacchi evolvono con l'affinamento delle tecniche di riconoscimento del traffico e con la capicita' di monitorare le reti aziendali connesse esposte sulla Internet. La questione di ottenere sempre maggiore padronanza della propria infrastruttura IT si scontra continuamente con l'aumentare e la continua evoluzione degli applicativi, della tecnologia e dell'espansione complessiva della rete Internet stessa. In questa tesi proveremo a identificare anomalie riconducibili ad attacchi DDoS, in un contesto di una rete aziendale con più sedi, usando un riconoscimento delle anomalie effettuato tramite una rete neurale allenata su dati provenienti dai router di più sedi aziendali e una successiva mitigazione degli attacchi tramite un agent sugli stessi. Approfondiremo la tecnica dei cosidetti auto-encorder, un particolare tipo di rete neurale, che si sta diffondendo rapidamente sia in ambito accademico che industriale nell'ambito dell'anomaly-detection. Esporremmo il nostro contributo prendendo come caso d'uso dati raccolti da una sede distaccata della società Tiesse s.p.a., produttrice di router e altri dispositivi di rete, coinvolta nello studio e nella realizzazione di questo progetto.

Indice

Capitolo 1

Introduzione

1.1 Motivazione

Le piccole e medie imprese sono sempre più informatizzate e dipendenti da servizi informatici per poter continuare a lavorare. Per questo motivo scoprire e analizzare il traffico anomalo che passa sulle reti aziendali è sempre più importante, il nostro obiettivo è farlo ad un basso costo, senza aggiungere hardware o applicativi che richiedono molta potenza di calcolo o di banda da parte dei router che compongono la rete. Punteremo a scoprire le anomalie nei dati di rete sui singoli flussi e nei dati aggregati analizzando alcune metriche su un server aggiuntivo che si occuperà anche di mitigare l'attacco informando i router sui flussi da bloccare. Gli attacchi maggiormente presi in considerazione in questa tesi sono gli attacchi di "Distributed Denial of Service", ma i suoi principi possono essere applicati anche ad altre tipologie di anomalie. Inoltre l'analisi dei dati raccolti sull'utilizzo della rete potrà portare anche alla risoluzioni di problemi di rete non derivanti da attacchi.

1.2 Collaborazione con Tiesse s.p.a.

Questa tesi è stata sviluppata in collaborazione con Tiesse s.p.a.: un'azienda che progetta e realizza router e dispositivi M2M, con connettività wired e mobile, interamente in Italia.

L'azienda ha oltre 20 anni di esperienza, ed è stata riconosciuta come uno dei maggiori fornitori di CPE a banda larga e wireless su reti 3G/4G dalla maggior parte delle compagnie telefoniche. Oltre alla produzione e alla progettazione di router e apparecchiature di rete, sviluppa progetti custom, sia hardware che software. I prodotti Tiesse sono particolarmente adatti ad applicazioni Business e Mission Critical per il Corporate Networking.

L'azienda è sempre attiva sull'aspetto di ricerca e sviluppo, anche tramite collaborazioni con il Politecnico di Torino e in particolare negli ultimi anni ha messo il focus su soluzioni di software per la sicurezza e l'analisi dei dati di rete.

La sede centrale dell'azienda si trova ad Ivrea, ma possiede anche sedi distaccate a Torino, Avezzano e Roma.

1.3 Scenario

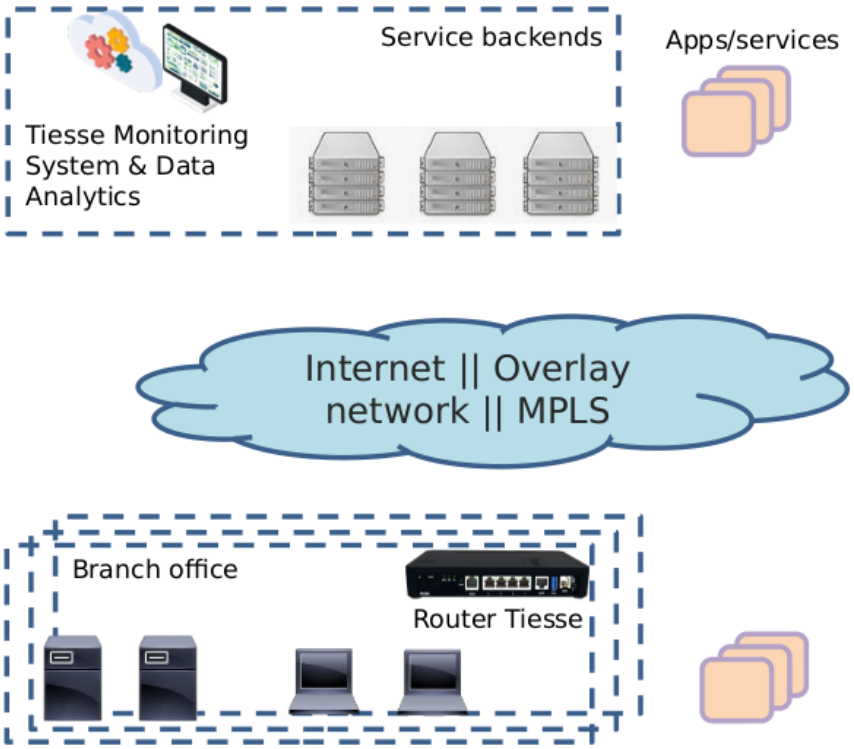

Figura 1.1. Scenario rete Tiesse e clienti

Nello sviluppo della nostra soluzione abbiamo preso in considerazione un tipico scenario aziendale, in cui esiste una sede centrale, ben protetta e su cui sono ospitati i servizi dell'azienda e tante sedi periferiche: uffici, negozi o altro, collegati ai servizi della sede centrale tramite un overlay MPLS o una VPN. Le sedi periferiche sono quelle più esposte sotto l'aspetto della sicurezza, anche solo per il fatto che sono in numero maggiore, spesso non ci sono i responsabili dell IT in sede e solitamente sono meno controllate che la sede centrale. Per questo motivo il nostro obiettivo è quello di proteggere i servizi, gli applicati aziendali e la rete centrale dai dispositivi malevoli connessi alle reti degli uffici. L'organizzazione di Tiesse e di molti suoi

clienti è caratterizzata dallo scenario in figura 1.3. Utilizzando questa struttura di rete per effettuare la raccolta, l'analisi del traffico e le prove, ci siamo basati su esempio di sede periferica, nel nostro caso un ufficio dell'azienda situato a Torino, con l'obiettivo di proteggere i servizi aziendali presenti nella sede centrale di Ivrea a cui l'ufficio è collegato tramite una VPN.

Un altro caso possibile di utilizzo di questa soluzione è la distribuzione dei servizi in cloud, in cui l'azienda non ha il controllo dell'infrastruttura di rete e sfrutta i router thrusted nelle reti degli uffici per analizzare il traffico.

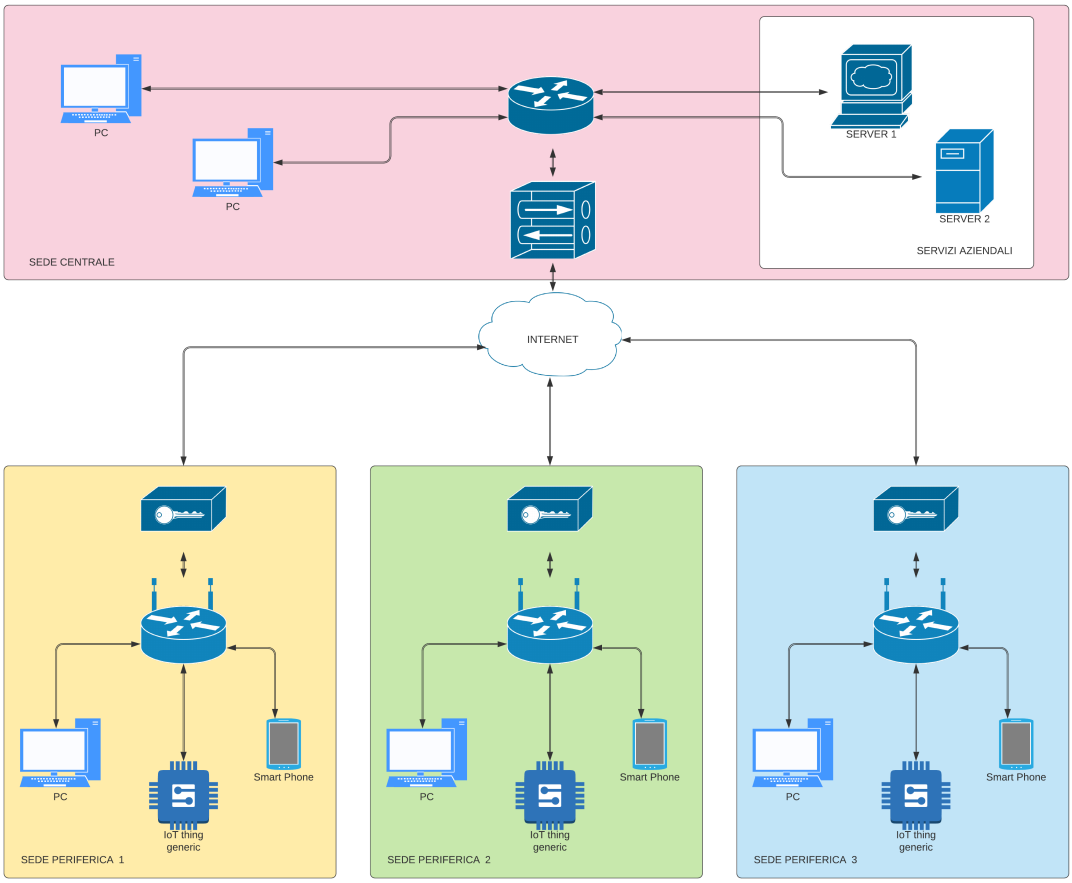

Figura 1.2. Scenario rete Tiesse e clienti

1.4 Gli attacchi DDoS

Gli attacchi di Denial of Service (DoS) sono attacchi nel campo della sicurezza informatica che mirano a interrompere la fruizione di un servizio, fornito da un

host connesso a internet, da parte di utenti legittimi. L'attacco ha l'obiettivo di esaurire le risorse dell'host in modo da non consentirgli di erogare le risposte ai richiedenti. Nel caso in cui la sorgente del traffico che mira a creare disservizi non sia unica, si parla di attacchi di denial of service distribuiti (Distributed Denial of Service).

1.4.1 Tipologia di attacchi DDoS

Gli attacchi DDoS possono essere suddivisi in due categorie principali in base al loro funzionamento. La prima si basa sul mandare alla vittima pacchetti malformati in grado di sfruttare un bug o una falla a livello applicativo. La seconda categoria invece si basa su tecniche utili a colpire l'infrastruttura del servizio, per il funzionamento di questa tecnica vengono usati uno o entrambi i seguenti metodi: il primo si basa sull'interruzione della connessione di rete grazie all'esaurimento della banda o della capacità di processamento dei router o di entrambe, nel secondo caso l'obiettivo dell'attaccante è di esaurire le risorse (es. sockets, CPU, memoria) del server che ospita il servizio [1].

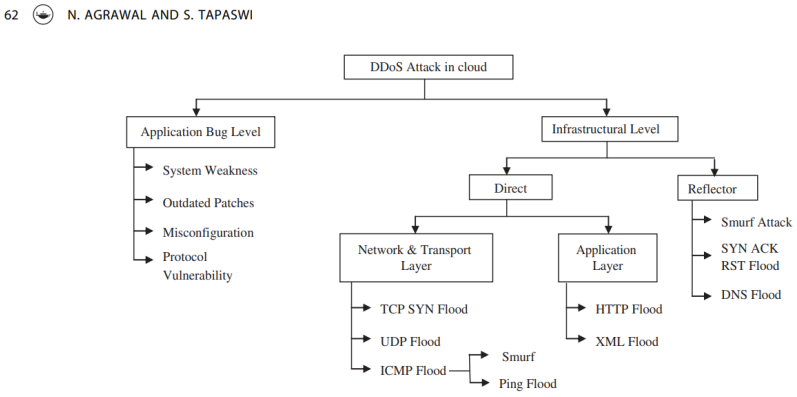

Figure 1. DDoS attack typology of cloud computing. (Adapted with permission from Osanaiye et al., 2016.)

Figura 1.3. Tipologie di attacchi DDoS [4]

L'obiettivo di questa tesi sarà concentrato sul rilevamento e la mitigazione della seconda categoria di attacchi, basata sull'esaurimento delle risorse.

Attacchi basati sul flooding

Network/transport-level DDoS flooding attacks Gli attacchi di denial of service che mirano ad esaurire le risorse di rete si basano sull'invio di molti pacchetti con lo scopo di consumare totalmente la banda o le socket della vittima, queste tipologia di attacco può essere effettuata in maniera diretta: *flooding attacks* e

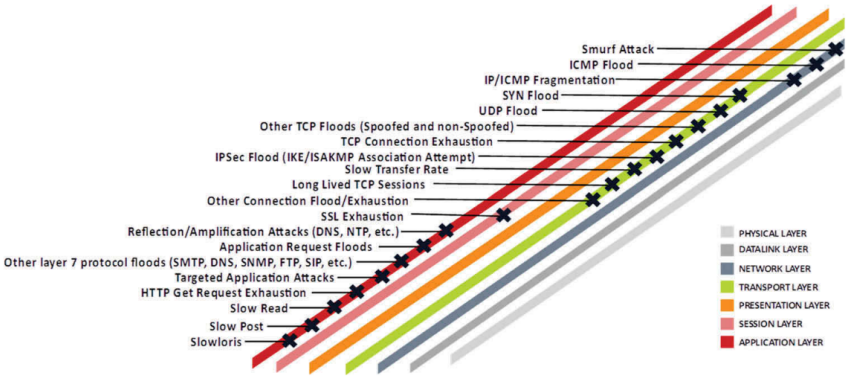

Figure 7. Distributed Denial of Service (DDoS) attack types across network layers.

Figura 1.4. Attacchi per livello [5]

protocol exploitation attacks, nel primo caso la vittima viene inondata di pacchetti (UDP flood, ICMP flood, DNS flood, VoIP flood an etc.), in questo caso la banda aggregata in uscita di tutti gli attaccanti deve essere superiore a quella del servizio che si vuole interrompere, nel secondo caso vengono sfruttane delle caratteristiche dei protocolli della vittima in modo da consumare una grande quantità di risorse (es. TCP SYN flood, TCP SYN-ACK flood, RST/FIN flood e ecc).

Gli attacchi che non vengono effettuati in maniera diretta invece sfruttano la riflessione o l'amplificazione: nei *Reflection-based flooding attacks* chi attacca manda un particolare pacchetto, indirizzandolo ad un riflettore e questo riflettore manda le sue risposte alla vittima, in modo da esaurire le risorse della vittima. Un esempio di questa tipologia di attacchi sono lo Smurf e il Fraggle, nel primo vengono mandati ICMP Echo Request ad una sottorete, usando come ip di destinazione l'indirizzo broadcast e specificando come ip sorgente l'ip della vittima, utilizzando l'ip spoofing, causando la risposta di tutti gli host verso l'indirizzo della vittima [2]. Gli *Amplification-based flooding attacks* sfruttano servizi che restituiscono risposte più grandi della richiesta ricevuta, un esempio è il DNS amplification, che riesce a moltiplicare dalle 30 alle 50 volte[20] la banda in uscita dell'attaccante, il suo funzionamento si basa sull'utilizzo dell'ip spoofing mandando un pacchetto con indirizzo ip sorgente della vittima, così il servizio DNS risponderà alla vittima con un flood di pacchetti di dimensioni maggiori [1]. Lo stesso principio è sfruttato anche dal NTP amplification che può amplificare il traffico con un rapporto tra 20:1 e 200:1, arrivando in alcuni casi anche a 556:1 [20].

Application-level DDoS flooding attacks Gli attacchi DDoS al livello applicativo hanno lo scopo di terminare le risorse del server(cpu, memory, disk/db bandwidth, I/O bandwidth) e di solito usano meno banda, rispetto gli attacchi all'infrastruttura di rete rete, per questo motivo è anche più difficile identificarli. Le tecniche utilizzate sono simili alle precedenti. Degli esempi sono l'HTTP flooding che effettuando l'invio di molte richieste HTTP, obbliga il server a produrre risposte che possono essere computazionalmente pesanti, oppure possono sfruttare l'SQL Injection per imporre un lock sul database e bloccare il funzionamento dell'applicazione. Altri attacchi possono essere l'HTTP fragmentation, lo slowpost attack, slowreading attack e lo slowloris attack, tutte tecniche che mirano a mantenere molte connessioni aperte mandando o ricevendo pochi dati per volta [1]. Gli attacchi di tipo applicativo possono essere molto eterogenei e non possono essere mitigati a livello di rete/trasporto, per questo motivo questa tesi prenderà in considerazione solo gli attacchi trattati al paragrafo precedente.

DDoS con obiettivo la riduzione della qualità del servizio L'unico obiettivo possibile degli attacchi DDoS non è la sola interruzione del servizio, ma un altro risultato è la degradazione del servizio, consumando una parte di risorse destinate agli utenti legittimi e creando loro ritardi nelle risposte. Questo risultato può essere raggiunto utilizzando dei packet rate più bassi, e di conseguenza meno rilevabili o dei l'invio d flood di pacchetti con rate variabili [4, 5]. Gli approcci a bassi rate sono utilizzati frequentemente dagli attacchi DDoS a larga scala, poichè l'aggregazione di moltissime fonti a basso rate portano comunque ad un grande risultato finale.

1.4.2 Vittime attacchi DDoS

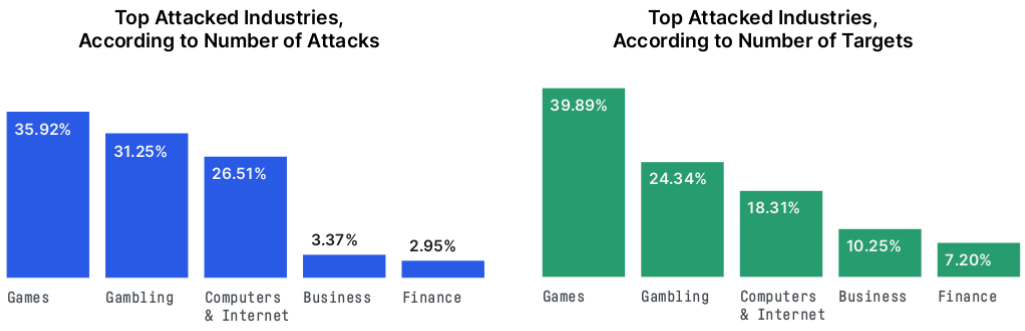

Figura 1.5. Distribuzione vittime DDoS, 2020 [22]

I target degli attacchi DDoS possono variare molto: da un utente domestico ad un governo [3].Per capire maggiormente chi possono essere le vittime di un attacco bisogna analizzare le motivazioni che spingono gli attaccanti e con le diverse motivazioni si può anche capire qual'è il rischio per quanto riguarda la portata dell'attacco. Per semplicità possiamo suddividere gli incentivi di un attacco in cinque principali categorie [1, 3]:

- Beneficio economico o finanziario: sono gli attacchi che riguardano principalmente le aziende, sono considerati i più pericolosi e difficili da fermare, perché mirano ad ottenere benefici finanziari dagli attacchi. I creatori dell'attacco normalmente sono persone con esperienza.
- Vendetta: questa tipologia di attacchi sono mesi in atto da persone, solitamente con uno scarso livello tecnico, a fronte di un'apparente ingiustizia percepita.
- Credo ideologico: alcuni attaccanti si trovano ad effettuare attacchi contro degli obiettivi per motivi ideologici. È una motivazione di attacco meno comune delle altre, ma può portare ad attacchi di grande entità.
- Sfida intellettuale: gli utenti che sviluppano attacchi per questa motivazione che vogliono imparare e sperimentare a lanciare attacchi, spesso sono giovani appassionati di hacking che grazie alla facilità con cui si possono affittare botnets o utilizzare semplici tool riescono ad effettuare con successo DDoS.
- Cyberwarfare: gli attaccanti di questa categoria appartengono ad organizzazioni terroristiche o militari di un paese e sono politicamente motivati ad attaccare risorse critiche di un altro paese. Un grande numero di risorse viene usato per questa tipologia di attacco e può paralizzare le infrastrutture critiche di un paese, portando ad un grave impatto economico.

Le maggiori vittime secondo il report [22] sono le compagnie che operano del campo dei videogiochi e delle scommesse, entrambe comportano un grande livello di rischio e spesso i giocatori si rifiutano di seguire le regole. Le aziende che incentrano il loro business sul settore di Internet e della Computazione si trovano al terzo posto, seguiti dalle attività commerciali e da quelle finanziarie.

1.4.3 Diffusione attacchi DDoS

Nel mondo gli attacchi a fine 2020 la quasi totalità degli attacchi DDoS proveniva da botnets, con target principali in Cina e negli Stati Uniti. Le tipologie di attacco maggiormente utilizzate sono il *Syn Flood* che copre più del 90% della totalità degli attacchi, seguito da *ICMP flooding* e *UDP flooding* [6, 7].

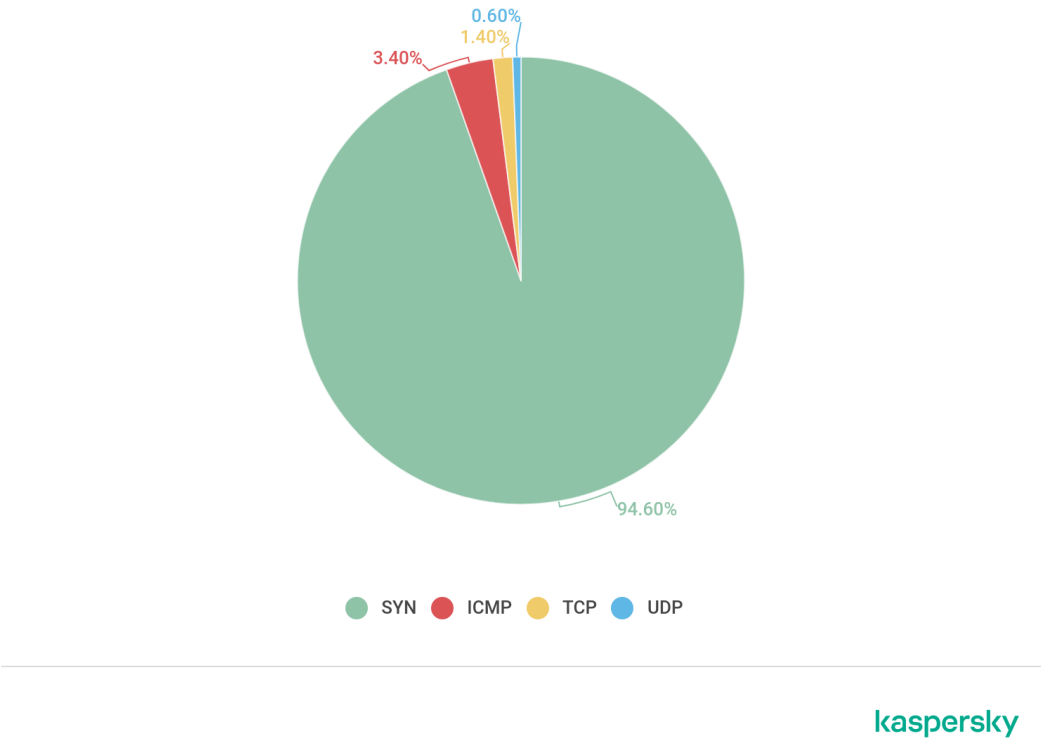

Figura 1.6. Distribuzione di attacchi DDoS per tipologia, Q3 2020 [7]

La maggior parte degli attacchi DDoS hanno una portata inferiore ai 10 Gbps (circa il 65%), ma il 3% di essessi raggiunge cifre incredibili, superiori al 200 Mbps [22].

Attacchi basati su botnets

Gli attacchi basati su botnets sono un grande problema per l'implementazione di sistemi anti-DDoS perché un grande numero di "zombie" rende l'attacco più distruttivo e spesso utilizzano ip spoofing, il che rende più difficile il tracciamento all'indietro per determinare i bot [1].

Il funzionamento delle botnet si basa su tre fasi. La prima consiste nel eclutamento "dell'esercito": fase in cui vengono contagiati dei dispositivi tramite worms (programmi che si autopropagano) che sfruttano falle nella sicurezza. Sono usate tecniche come per esempio la scansione automatica di indirizzi ip casuali per trovare altri soggetti vulnerabili, questa tipologia di scansione però produce molto traffico e può rendere possibile rilevare l'attacco, oppure una hitlist: una tecnica che frutta una lista di possibili macchine potenzialmente vulnerabili, la scansione nella subnet

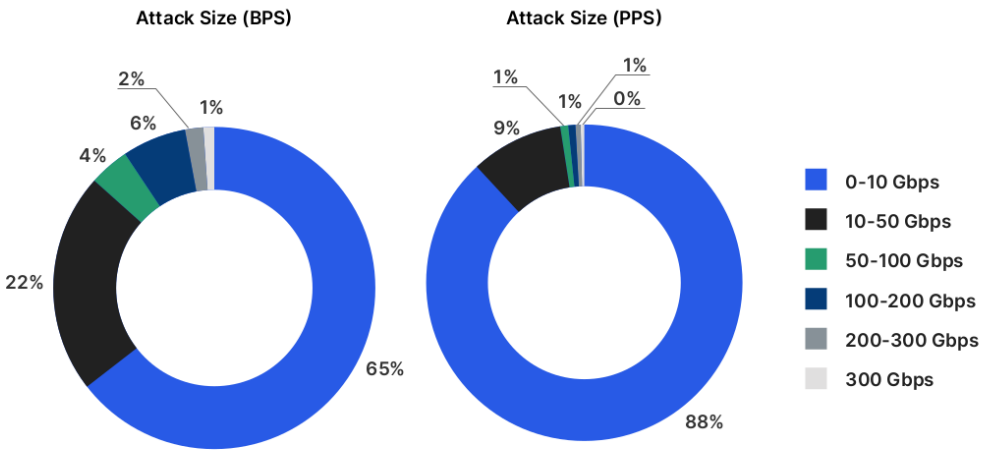

Figura 1.7. Distribuzione di attacchi DDoS per dimensione, 2020 [22]

o l'utilizzo di informazioni dei target basandosi sulle informazioni contenute nelle vittime infettate dai worm.

La fase successiva alla creazione di un "esercito" bisogna avviene la propagazione delle informazioni riguardanti le vittime da attaccare, con la durata e l'ora. I bot possono essere controllati dell'artefice dell'attacco tramite tre architetture [5]:

- IRC-based: architettura client-server in cui ad ogni server si possono collegare centinaia di dispositivi, utilizza un protocollo testuale e utilizzando porte non standard rende molto difficile il riconoscimento del comando per lanciare un DDoS, il quale si può nascondere facilmente nel grande traffico dei server IRC, ma il singolo server a cui si connettono tutti i client può essere considerato un single point of failure.

- Web based: ogni bot scarica periodicamente delle informazioni tramite una richiesta web ad un server, i comandi di questa tipologia di controllo sono i più difficili da tracciare.

- P2P based:le moderne botnet utilizzano una struttura più robusta e flessibile, invece di avere un singolo server centrale chi è al controllo della botnet manda un comando in broadcast a tutti gli agent, questo lo rende più affidabile e l'eliminazione di un agent non rende la rete non funzionante, ma rende la rete più difficile da mantenere e la scoperta di un agent può mostrare tutti gli altri.

L'ultima fase è il tentativo di attacco vero e proprio.

Un esempio di botnet è Mirai, una rete di dispositivi creata attraverso un malware che sfrutta le vulnerabilità dei dispositivi IoT per creare degli host malevoli

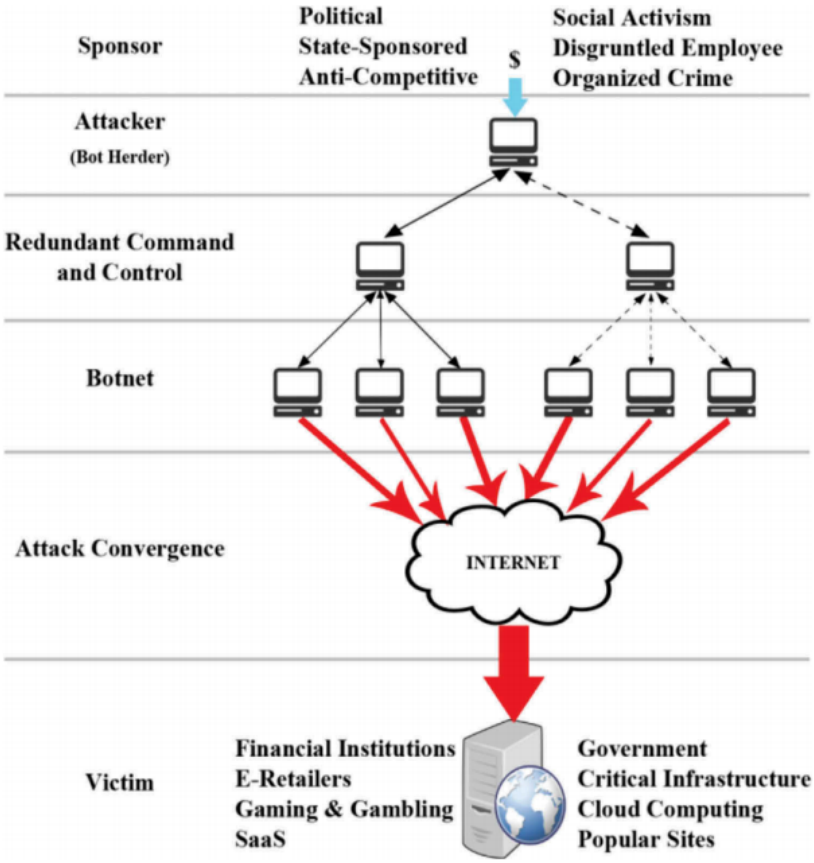

Figura 1.8. Struttura di lancio di attacchi DDoS [5]

da utilizzare in attacchi DDoS di larga scala. Il codice sorgente del bot è stato rilasciato pubblicamente nel 2016, questo ha permesso la creazione di molte sue varianti, lasciando a chiunque la possibilità di crearsi la propria botnet. Il codice per effettuare attacchi DDoS include dieci tipologie diverse di attacco, ognuna configurabile in molti modi e lanciabile attraverso l'utilizzo di un server di comando e controllo [8].

Alcuni attacchi possibili tramite la botnet Mirai sono: flood DNS, STOMP: una attacco che mira ad effettuare il three-way TCP handshake e solo dopo quando la sessione sarà messa in whitelist inizia un ACK flood, GREIP: incapsula nel pacchetto GRE un pacchetto ip con ip sorgente e destinazioni casuali, SYN flood, ACK flood, UDP flood e HTTP flood.

Attacchi DDoS famosi

Prova prova

Parlo dei primi attacchi, fino a quelli moderni menzionando i motivi dell'attacco se scoperti, la tipologia e la portata.

1.5 Organizzazione della tesi

La tesi nei successivi capitoli tratterà lo stato dell'arte dei sistemi anti-ddos esistenti (capitolo 2), mostrando come è possibile riconoscere un attacco e dove è meglio farlo per poterlo mitigare al meglio. Nel capitolo 3 approfondirà maggiormente lo stato dell'arte dei sistemi di anomaly detection, da utilizzare per rilevare un attacco DDoS. Nei capitoli 4 e 5 verrà illustrata la soluzione da noi proposta, focalizzandosi nel quarto sulla tecnica per rilevare un'anomalia e nel quinto su come mitigarla. Nel capitolo 6 vengono menzionati possibili lavori futuri e nel 7 le conclusioni con delle considerazioni sulla soluzione proposta.

Capitolo 2

Sistemi anti-DDoS: stato dell'arte

2.1 Riconoscimento DDoS

La fase di riconoscimento degli attacchi DDoS è un importante passo per la loro mitigazione, questa fase diventa più facile maggiormente ci avviciniamo alla vittima dell'attacco, ma viceversa più ci si allontana dalla sorgente dell'attacco e più diventa difficile identificarla. In letteratura esistono due tecniche per identificare i flussi malevoli: signature-based detection e anomaly-based detection.

2.1.1 Signature-based detection

La signature-based detection è un meccanismo che si basa su attacchi DDoS conosciuti per differenziare la loro firma, dai normali flussi della rete. Queste soluzioni hanno un buon successo con attacchi DDoS conosciuti, ma non sono in grado di rilevare nuove tipologie ti attacco di cui non si conosce ancora la signature. Questi sistemi si possono basare su pattern matching (es. Bro/Zeek), su regole (es. Snort, Suricata), sulla correlazione di informazioni di management sul traffico, o sull'analisi spettrale.

2.1.2 Anomaly-based detection

I sistemi basati sul rilevamento delle anomalie possono riconoscere anche attacchi non conosciuti, basandosi su soglie per differenziare il traffico normale e malevolo, ma la scelta dei limiti oltre i quali considerare il traffico anomalo è una grande sfida per questa tipologia di tecniche. I metodi più diffusi si basano su metodi statistici, di data mining o intelligenze artificiali.

2.2 Contromisure attacchi DDoS

Gli attacchi DDoS si raggruppano ad imbuto dalle sorgenti verso la vittima, per questa ragione più si è vicini alla vittima e più l'attacco sarà facile da riconoscere, ma più difficile da mitigare. Per questa ragione le tecniche di mitigazione vengono suddivise in base al luogo in cui vengono azionate.

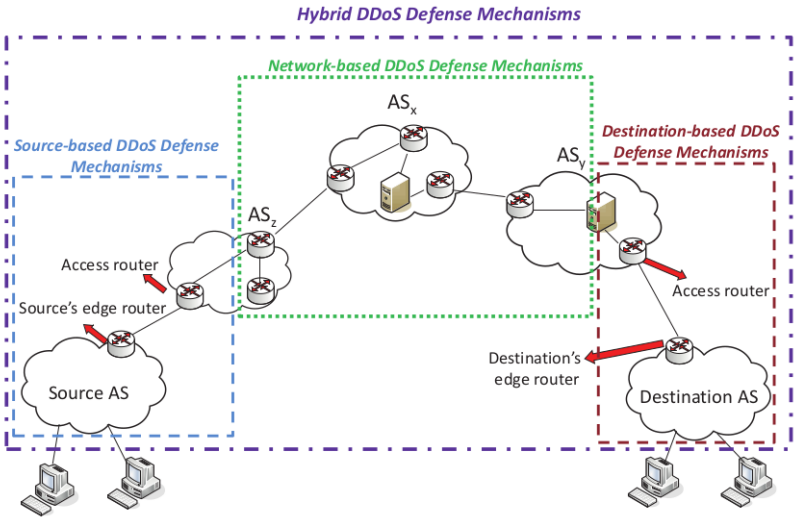

Figura 2.1. Classificazione dei sistemi di difesa in base al luogo di applicazione [1]

2.2.1 Soluzioni alla sorgente

Questa tipologia di soluzioni sono adottate vicino alle sorgenti dell'attacco per impedire agli utenti di una sottorete di generare attacchi DDoS. Queste soluzioni possono essere applicate agli edge router degli Autonomous System (AS) di accesso.

Degli esempi di soluzioni sono [1]:

- Filtri in ingresso e uscita agli edge router delle sorgenti: gli attacchi che si basano sulla riflessione sfruttano la tecnica dell'ip spoofing, lo scopo di questi filtri è di bloccare il traffico che utilizza ip spoofing senza interferire sul traffico legittimo.

- D-WARD: sistema che mira a rilevare attacchi di tipo DDoS flooding monitorando il traffico in ingresso e uscita e comparandolo con dei flussi normali di traffico. I flussi sono filtrati se non rispettano il modello. Per esempio può essere imposto un rapporto massimo tra pacchetti inviati e ricevuti per un flusso TCP che se superato può segnalare il flusso come anomalo.

- MULTOPS (MUlti-Level Tree for Online Packets Statistics): è un'euristica e un struttura dati che permette di rilevare attacchi provenienti da una subnet. Normalmente il traffico in una direzione è proporzionale a quello in quella opposta, una significativa differenza possono indicare che la rete è la sorgente o la vittima di un attacco DDoS.
- MANAnet's Reverse Firewall: è un firewall che a differenza di quelli tradizionali, protegge l'esterno dal traffico in uscita dalla subnet.

I problemi di questa soluzione sono che dovrebbe essere implementata su gli edge router ti tutti gli AS di accesso per permettere una copertura totale, inoltre è difficile differenziare il traffico legittimo, da quello malevolo e non meno importante non è chiaro chi sia il responsabile del mantenimento economico di questo servizio [1].

2.2.2 Soluzioni alla destinazione

Esistono soluzioni che si possono applicare agli edge router della vittima, possono analizzare il comportamento della vittima e il suo traffico usuale e riconoscere le anomalie [1, 2]. Delle soluzioni posizionate in questi luoghi possono essere dei proxy, firewall che gestiscono le connessioni semi aperte in caso di syn flood oppure l'utilizzo sistemi di tracciamento implementati in alcuni router (in caso di ip spoofing), per risalire alla vera sorgente dell'attacco e bloccarla.

Questi sistemi di difesa possono diventare i target degli attacchi, poiché spesso richiedono una grande quantità di memoria e potenza di processamento per effettuare le osservazioni delle misure statistiche [5].

2.2.3 Soluzioni sulla rete

I sistemi anti-DDoS sulla rete si basano sui router o su firewall installati sulla rete dell'operatore. Una prima soluzione adottata è quella del Router based packet filter, la quale si basa sui criteri dell'ingress filtering, ma applicandola ai router nel core della rete. Il traffico per ogni link tendenzialmente viene generato da un ristretto intervalli di indirizzi ip, quando appare un indirizzo ip sospetto viene filtrato, questa soluzione è adatta a rilevare attacchi che utilizzano ip spoofing, ma è inutile nel caso di utilizzo di ip genuini. Altre soluzioni mirano ad identificare i router nel core di internet che sono stati compromessi e si comportano in anomalo, oppure mirano all'installazione di detection systems (DSs) che permettono di rilevare pattern anomali, ma sono computazionalmente molto dispendiose.

2.2.4 Soluzioni distribuite

Le soluzioni distribuite creano una cooperazione tra i luoghi di installazione delle difese, alla sorgente vengono installati i sistemi di difesa per filtrare l'attacco,

mentre sulla rete della vittima viene effettuato il riconoscimento dell'attacco. Questa soluzione porta sia il vantaggio della facilità di riconoscimento degli attacchi possibile alla destinazione, sia l'efficienza dei sistemi per mitigare gli attacchi alla sorgente.

2.3 Momenti in cui mettere in atto le difese

Esistono più momenti in cui mettere in atto le difese contro attacchi DDoS e un buon sistema deve agire su tutti. Il primo momento è prima dell'attacco (attack prevention), in cui vengono installati sistemi di monitoring, sistemi da usare in caso di emergenza, meccanismi per tollerare l'attacco (firewall, IDS, filtri, load balancer e flow control, strategie per identificare gli utenti legittimi e protezioni lato server).Successivamente durante l'attacco devono essere lanciate le adeguate contromisure e dopo l'attacco il sistema di difesa deve identificare la fonte e provvedere con le opportune contromisure, anche collaborando con altri segnalare l'attacco per permettere di difendersi da successivi.

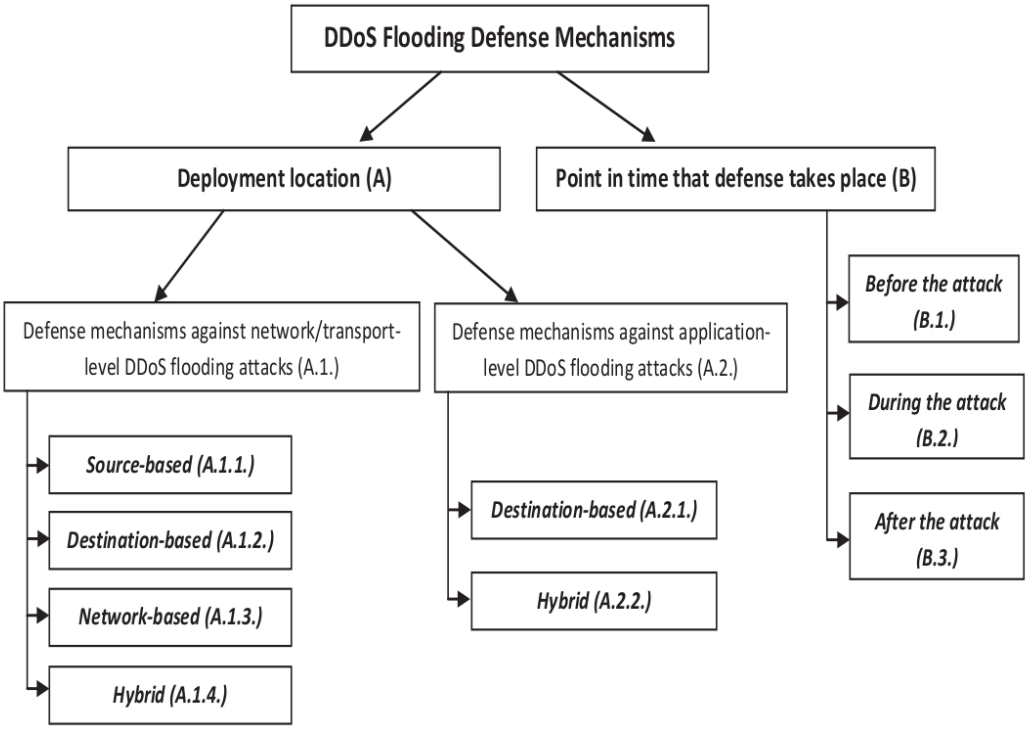

Figura 2.2. Meccanismi di difesa contro DDoS di tipo flooding [1]

2.4 Tolleranza

Capitolo 3

Anomaly Detecion: stato dell'arte

3.1 Cos'è l'anomaly detection?

L'Anomaly detection si riferisce al problema di trovare pattern nei dati che non sono conformi al comportamento aspettato. A queste non conformità ci si riferisce come anomalie [15]. L'importanza del rilevamento delle anomalie è il fatto che un'anomalia nei dati spesso corrisponde ad un'informazione critica nel dominio a cui si riferisce, per esempio nelle reti di computer un traffico anomalo potrebbe significare che un computer è stato hackerato e sta compiendo azioni per il danneggiamento dell'azienda.

3.2 Sfide dell'anomaly detection

Le anomalie sono definite come un pattern che non rispetta il normale comportamento, ma il definire il concetto di normalità è una sfida, i maggiori fattori che influiscono su questa decisione sono [15]:

- La difficoltà nel trovare una regione che comprenda tutti i possibili comportamenti normali è molto difficile e il confine tra azioni normali e anomale spesso non è ben definito.
- Se le azioni anomale sono generate da azioni malevole, il responsabile cercherà di fare in modo che le osservazioni sui dati appaiano normali.
- In alcuni contesti il comportamento si evolve e ciò che è considerato correntemente normale potrebbe essere rappresentativo per il futuro.

- È difficile definire quanto la differenza dalla normalità debba essere considerata anomala, per esempio in medicina piccole variazioni della temperatura corporea possono essere considerate anormali, in finanza la fluttuazione del valore delle azioni potrebbe essere considerato normale.

- La disponibilità di dati già classificati come normali o anomali per verificare il modello è uno dei problemi principali.

- Spesso i dati contengono rumore che tende ad essere simile alle anomalie ed è difficile rimuoverlo o distinguerlo.

3.3 Sistemi di rilevamento delle anomalie

3.3.1 Modalità di apprendimento

I sistemi di anomaly detection necessitano di dataset con esempi di traffico per capire quando devono segnalare un'anomalia. In base alla tipologia di dataset di cui hanno bisogno per effettuare l'apprendimento, i sistemi di anomaly detection possono essere classificati in tre modi.

Apprendimento superivisionato I sistemi di anomaly detection basati su apprendimento supervisionato hanno prestazioni superiori ai metodi non supervisionati, poichè usano esempi etichettati di dati. I metodi supervisionati imparano il limite di separazione da un set di dati annotato durante una fase di allenamento e successivamente classifica le istanze da testare basandosi sul modello imparato. Avendo bisogno di etichette di difficile reperibilità, questo metodo è poco usato rispetto a quello semi-supervisionato e a quello non supervisionato, ma ha come vantaggi oltre al fatto di essere più preciso anche la velocità della fase di test, basandosi su un modello pre calcolato.

Apprendimento semi-superivisionato L'apprendimento semi-supervisionato è una tecnica che che per funzionare assume che tutti i dati usati per l'allenamento siano etichettati in un'unica classe, quella della normalità. Tutte le istanze di dati di dati di test che superano un certo limite intorno alla normalità verranno etichettate come anomale. I vantaggi di questo metodo è che l'uso di dati etichettati può portare ad un grosso miglioramento di performance rispetto alle tecniche non supervisionate, ma rischiano di non essere rappresentative di alcuni casi e sono inclini all'overfitting.

Apprendimento non superivisionato Questa tecnica si basa su un set di dati normali che contengono che contengono una piccola quantità di dati anomali, l'algoritmo i occuperà di creare un punteggio di separazione basandosi su proprietà intrinseche dei dati.

3.3.2 Metodo di detection

Una volta classificati i sistemi di anomaly detection in base al metodo di apprendimento, possono essere ulteriormente suddivisi in sei sottocategorie basate sulla tecnica di rilevamento [11, 10].

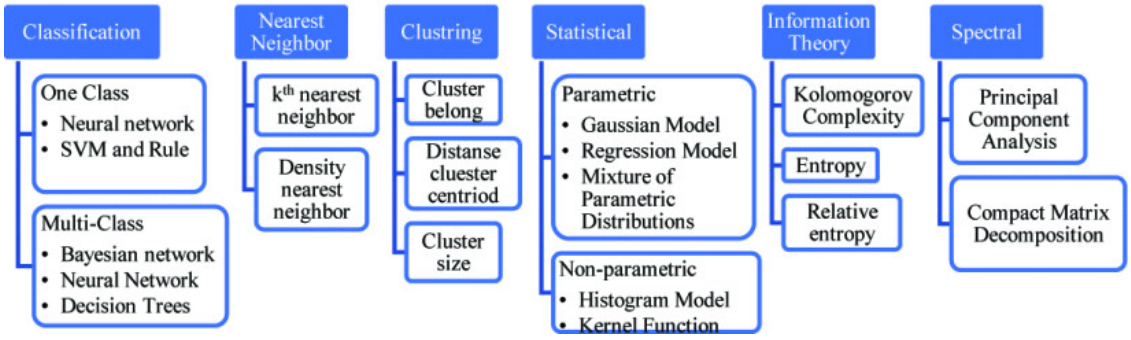

Figura 3.1. Classificazione metodi di anomaly detection

- Classification based: si basano su un set di dati etichettato che distingue una o più categorie e si basa su un modello (classificatore) che impara a distinguere le classi durante la fase di allenamento.
- Statistical based: assumono che i dati normali si presentino con una maggiore probabili rispetto a quelli anomali e segnalano come anomali regioni di dati a bassa probabilità.
- Information theory: analizzano il contenuto dei dati per verificare se sono presenti irregolarità.
- Nearest neighbor: si basa sull'assunzione che i dati normali si concentrino in zone dense e le anomalie si presentino lontano da queste zone.
- Clustering-based: raggruppano le istanze di dati simili in gruppi, i dati che non appartengono a nessun gruppo sono considerati anomali.
- Spectral: trasferisce i dati in un sottospazio con minore dimensionalità in modo che le istanze anomale siano significativamente differenti.

3.3.3 Output of Anomaly Detection

Un'importante caratteristica è come sono rappresentate le anomalie in uscita dal modello, esistono due metodi: le etichette, tecnica con la quale i dati vengono etichettati con una categoria, tendenzialmente binaria per esempio possiamo etichettare i dati come normali o anomali. L'altro metodo è l'uso di un "anomaly score": un punteggio per ogni istanza di dati, che indica quando si discosta dalla normalità, imponendo delle soglie possiamo verificare quali dati sono anomali [13, 14, 15].

3.4 Classificazione delle anomalie

3.4.1 Tipologia di anomalie

Un aspetto importante dell'anomaly detection è l'analisi delle anomalie che possono presentarsi, di conseguenza le anomalie possono essere classificate nel seguente modo:

- Anomalie puntuali: un singolo dato che si discosta dalla normalità. Questo è il caso più semplice e su cui si concentra la maggior parte delle ricerche sui dati anomali. Un esempio è un utente che tutti i giorni scarica 1GB di dati quando arriva in ufficio, ma un giorno ne scarica 10.

- Anomalie contestuali: quando un insieme di dati si comporta in modo anomalo in un determinato contesto, per esempio il numero di acquisti su un sito durante il periodo di Natale è più alto che durante il resto dell'anno.

- Anomalie collettive: quando un'istanza di dati è anormale rispetto all'intero dataset, in questo caso i dati in sè non sono anomali, ma lo diventano quando presi insieme, un esempio è l'elettrocardiogramma, in cui se ci sono bassi valori per un lungo periodo possono identificare un problema.

3.4.2 Applicazione anomaly detection

Le applicazioni dell'anomaly detection possono essere molteplici, qui un breve elenco delle possibili applicazioni [15]

- Fraud detection: sono sistemi con lo scopo di riconoscere frodi, i occupano di rilevare attività illegali in diversi campi come quella assicurativo, finanziario e telefonico.

- Medical and Public Health Anomaly Detection: lavorano sui dati dei pazienti, che possono avere anomalie per diverse ragioni: errori di registrazione, errori

degli strumenti di misura o una condizione anormale delle condizioni del paziente. Questi sistemi mirano a riconoscere anomalie puntuali usando sistemi semi supervisionati.

- Industrial Damage Detection: i dispositivi industriali sono soggetti ad usura e danneggiamenti, sistemi di anomaly detection applicati in queste situazioni possono permettere di prevenire o rilevare in anticipo questi problemi.
- Image Processing: permette di riconoscere anomalie significative in immagini durante il tempo (motion detection) o regioni anomale in immagini statiche.
- Anomaly Detection in Text Data: questi sistemi riconoscono principalmente gli argomenti o gli eventi in una collezione di documenti, le anomalie sono causate da nuovi eventi interessanti o argomenti anomali.
- Sensor Networks: anomalie nei rilevamenti dei sensori possono significare un difetto o un'intrusione.
- Intrusion detection systems(IDS): sono dei sistemi che puntano ad identificare attività malevole nei sistemi informativi, gli IDS possono essere installati su un computer, in quel caso si parla di "Host Intrusion Detection (HIDS)" oppure su una rete, in quel caso si parla di "Network Intrusion Detection (NIDS)", gli IDS possono essere signature-based oppure anomaly based, nel primo caso non si è in grado di riconoscere nuove tipologie di attacchi. In questo contesto vengono analizzate le anomalie collettive e sono preferiti sistemi di apprendimento semi-supervionato o non supervisionato, perchè le etichette per i dati anomali non sono quasi mai disponibili. In questa tesi mi concentrerò sull'analisi di anomalie di rete basandomi su sistemi di anomaly detection.

3.4.3 Tipologia degli attacchi di rete

Lo scopo della sicurezza di rete è di proteggere la confidenzialità, l'integrità e di assicurare la disponibilità delle risorse, quindi una minaccia viene definita come qualunque cosa cosa che ha caratteristiche finalizzate a compromettere la rete [13]. Un elenco di possibili minacce è:

- Denial of service (DoS)
- Probe: viene sondata la rete per ottenere informazione sulla struttura della rete target o degli host, principalmente per scopi di ricognizione.
- User to root (U2R): è una tipologia di attacco, quando l'attaccante ha l'obiettivo di ottenere illegalmente accesso ad un account amministrativo per manipolare o fare uso illecito di risorse.
- Remote to user (R2U): è un attacco in cui l'attaccante ottenere l'accesso di una macchina target, in modo di avere i privilegi di utilizzare la sua rete.

3.5 Le reti neurali

3.5.1 Funzionamento delle reti neurali

Le reti neurali sono composte da diversi strati (layer) di nodi, ciascuno collegato al successivo.

L'unità base di ogni rete neurale è il nodo, che rappresenta il neurone artificiale, il tipo base è il "perceptron": un nodo con più input e un output rappresentato come una funzione a gradino caratterizzata dalla funzione 3.5.1, in cui si può notare che solo se la sommatoria dei valori in ingresso supera una certa soglia, il neurone si attiva.

$$output = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases}$$

Questo permette di compiere decisioni in base all'input.

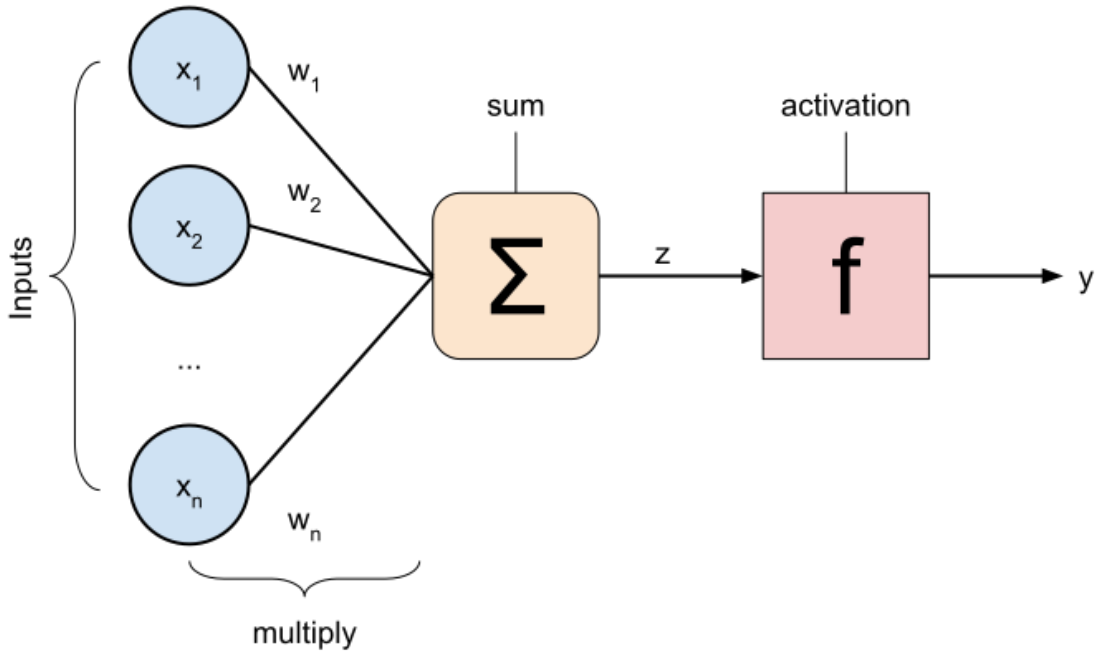

Figura 3.2. Neurone artificiale

Come si può notare dalla figura con l'evoluzione delle reti neurali non viene più usata solo la funzione a gradino, ma anche delle generiche funzioni rappresentate come "f". Un esempio comunemente usato di evoluzione del perceptron sono i neuroni di Sigmoid, i quali invece di ritornare una funzione a gradino in uscita, ritornano il risultato della funzione di sigmoid 3.1 [9].

$$\sigma(z) = \frac{1}{1 + \exp(-\sum_j w_j x_j - b)} \tag{3.1}$$

Le reti neurali sono composte da gruppi di neuroni artificiali organizzati in livelli e tipicamente composte da almeno tre strati: uno di input, uno o più livelli intermedi definiti "livelli nascosti" e un livello di uscita. Ogni livello può contenere uno o più neuroni che possono essere collegati tra loro solo con dei collegamenti in direzione dai nodi di input a quelli di output, in questo caso si parla di "feedforward" oppure "recurrent" in cui sono previste delle connessioni di feedback a neuroni dello stesso livello o a livelli precedenti.

ALLENAMENTO parlare di come vengono allenate

3.5.2 Autoencoders

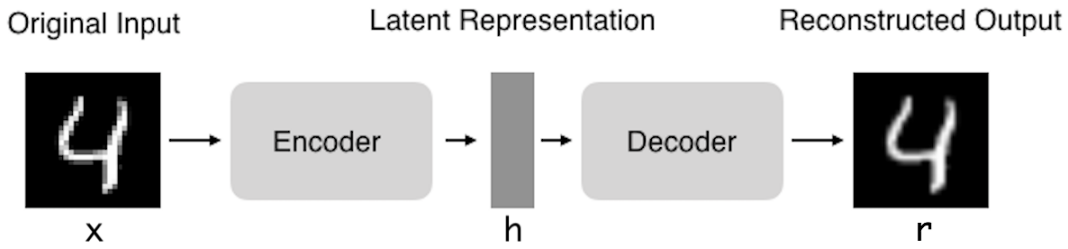

Figura 3.3. Struttura di un autoencoder

Gli autoencoder sono una tipologia di reti neurali non supervisionata, anche se quasi sempre vengono usati in maniera semi-supervisionata, che permettono di generare nuovi dati, effettuando in uscita la ricostruzione dei dati forniti in ingresso. Gli autoencoder sono composti da tre parti, l'encoder, lo spazio latente e il decoder, negli autoencoder la dimensione dell'input è uguale a quella dell'output, e nel mezzo ci sono dei livelli che si occupano prima di comprimere (encoder) le dimensioni, fino a raggiungere lo spazio latente, una forma "compressa" dell'input, seguita da una serie di livelli (decoder) che si occupano ti ritornare alle dimensioni originarie 3.3. La capacità di comprimere i dati e decomprimerli si basa sull'allenamento della rete, di conseguenza ogni rete è specifica per una tipologia di dati da comprimere e decomprimere, questo la caratterizza dagli altri sistemi di compressione, per esempio gzip, inoltre è una tipologia di compressione "lossy", con perdita, perchè l'uscita sarà simile all'ingresso, ma degradata. Una popolare applicazione degli autoencoder è l'anomaly detection. Gli autoencoder si occupano di ricostruire i dati iniziali, imparando effettivamente a riprodurre una funzione identità, per questo

motivo allenandoli solo con dati di istanze normali quando si troveranno davanti ad un dato anomalo falliranno la ricostruzione dell'input e di conseguenza produrranno un grande errore di ricostruzione [14].

3.6 Soluzioni esistenti di Anomaly Detection

Esistono in letteratura sistemi di anomaly detection basati su autoencoders, che possono essere allenati per il riconoscimento di anomalie di rete basandosi su correlazioni non lineari tra le features, degli esempi sono:

- Kitsune [24]: un NIDS, basato sistema di anomaly detection basato su autoencoders, che permette il riconoscimento di traffico normale e anomalo senza una supervisione per l'allenamento. Il sistema ha l'obiettivo di rilevare attacchi di tipo DDoS, Man in the Middle, scansioni di rete e botnet in una rete con dispositivi tradizionali e IoT.
- La soluzione proposta nell'articolo (come lo cito?) [25], la quale sfrutta una rete convoluzionari di autoencoders (per una riduzione dei tempi di training) per superare le prestazioni dei sistemi di rilevamento tradizionali.
- L'obiettivo della soluzione [26] è di rilevare attacchi in di rete utilizzando i dati provenienti dai router di rete, generati tramite NetFlow, inoltre si pone l'obiettivo di spiegare in cosa consiste l'anomalia nei dati. Per il rilevamento delle anomalie si basa sui Variational Autoencoder, uno sviluppo della versione tradizionale.

3.7 Motivazione scelte

Capitolo 4

Soluzione proposta

4.1 Introduzione

Il nostro obiettivo è di creare un sistema di anomaly detection con struttura distribuita: nei router delle sedi periferiche dell'azienda vengono raccolte informazioni sul traffico e in un server posizionato fisicamente nella sede centrale vengono analizzati e in base al risultato vengono prese decisioni sulle future azioni da intraprendere. Questa tipologia di architettura permette di sfruttare dispositivi già esistenti e non aggiunge molta complessità all'infrastruttura informatica aziendale, quindi per svolgere questo lavoro ci siamo basati su alcuni software e sull'hardware aziendale con i relativi vantaggi e svantaggi rispetto ad una soluzione completamente ex-novo. Inoltre essendo una soluzione distribuita permette di bloccare gli attacchi e il traffico malevolo lontano dalla sede centrale, in modo da essere più efficace e non influenzare il comportamento di altre sedi in caso di un attacco proveniente da una o più sedi periferiche. La raccolta e l'analisi dei dati viene effettuata utilizzando un sistema centralizzato, questo permette di avere un unico dispositivo con la capacità necessaria a svolgere l'elaborazione, che al tempo stesso possiede una visione completa sulle sorgenti e quindi sarà in grado di identificare facilmente da quale sede proviene il traffico anomalo.

4.2 Gestione dei dati

I dati vengono collezionati tramite un demone sui router (collectd), inviati ad un server (go-graphite) e visualizzati tramite una dashboard (Grafana). Il tool di anomaly detection da noi sviluppato si interfaccia direttamente con il database per la lettura dei dati e la scrittura dei risultati. Qui in seguito approfondirò maggiormente il flusso della gestione dei dati.

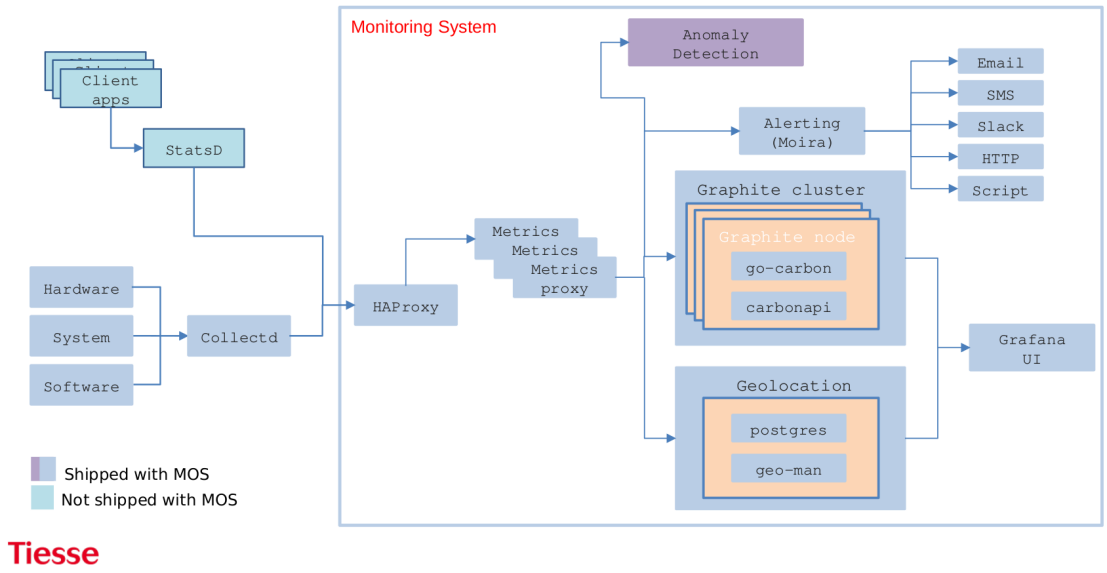

Figura 4.1. Architettura di MOS (Monitor System di Tiesse)

4.2.1 collectd

Collectd è un demone che raccoglie metriche di sistema e di applicazioni, trasferisce e salva dati di computer e dispositivi di rete. Collect ha una struttura modulare, vedi figurra 4.2.1 in cui è possibile abilitare centinaia di plugin per la raccolta di metriche di sistema dai casi più generali a quelli più specifici ed inoltre è possibile scrivere i propri plugin per integrarlo ulteriormente. I plugin usati sono "write_graphite": plugin che permette di scrivere le metriche raccolte su un database graphite, "conntrack": plugin che permette di contare il numero di voci nella connection tracking table di Linux, "interface" : plugin che colleziona informazioni sul traffico su un'interfaccia, quindi pacchetti al secondo, bytes al secondo ed errori sull'interfaccia.

Inoltre per avere ulteriori dati a disposizione ho scritto un plugin che si occupa di aggiungere delle metriche sul conteggio dei pacchetti non possibile con i plugin standard.

4.2.2 NDPI

NDPI è un software open-source per il deep-packet inspection basato su OpenDPI, il suo scopo è di riconoscere protocolli conosciuti anche su porte non standard, inoltre può essere personalizzato per il riconoscimento di applicazioni aziendali (è vera questa cosa?). nDPI è implementato nei router Tiesse e permette l'analisi dei pacchetti senza un degrado prestazionale.

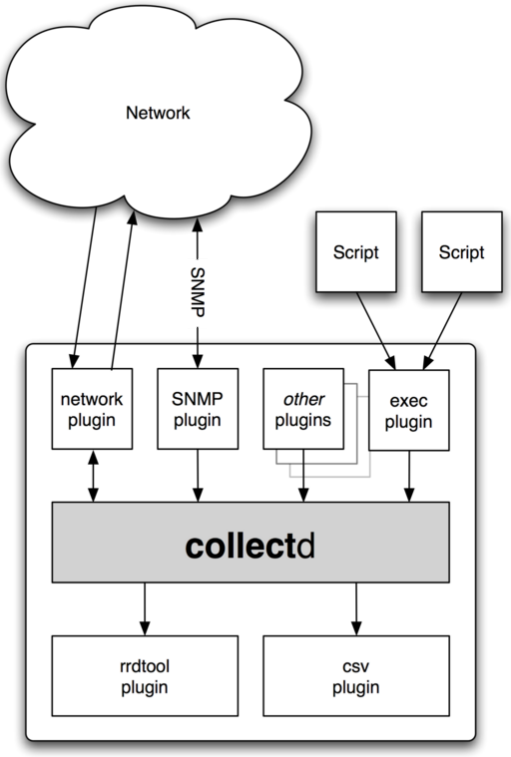

Figura 4.2. Architettura di collectd

4.2.3 graphite

è un software open source per il monitoraggio che può funzionare sia su hardware economico, che su un'infrastruttura cloud. Può essere usato per monitorare le performance di siti, applicazioni, server e nel caso di Tiesse è usato per monitorare informazioni sull'uso dei router, come per esempio il numero totale di router connessi, quelli raggiungibili, il throughput, l'uptime e la velocità della connessione xDSL. L'obiettivo di graphite è il salvataggio di serie temporali di dati numerici e la successiva condivisione e visualizzazione. Graphite è composto da tre parti (come si può vedere dalla figura 4.2.3):

- carbon: è un service ad altre prestazioni che si occupa di ricevere le metriche con formato "(timestamp, value)" da salvare.

- whisper: un semplice database che salva sul filesystem le sequenze temporali di dati.

- graphite-web: è un'interfaccia utente e delle API le quali restituiscono i dati per renderizzare i grafici da visualizzare.

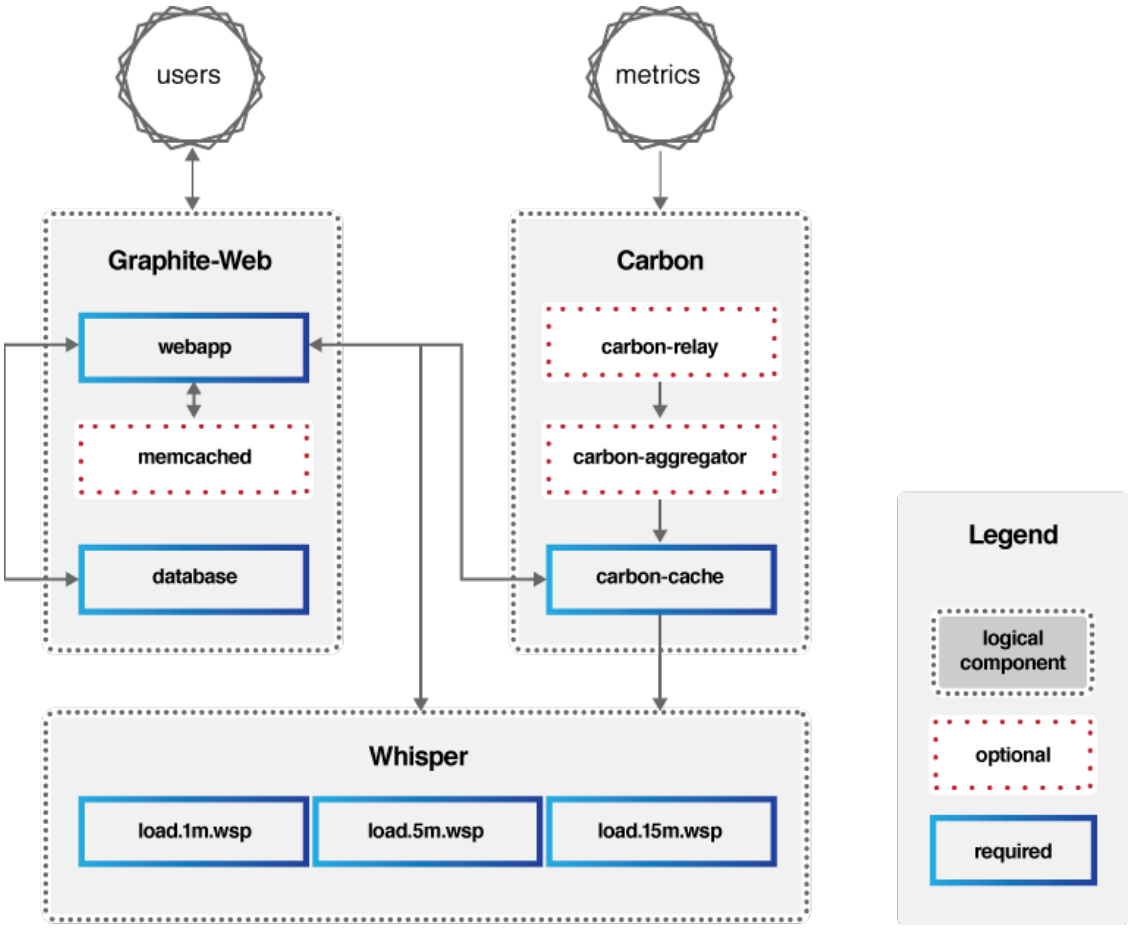

Figura 4.3. Architettura di Graphite

go-graphite è un'implementazione in Golang di Graphite, rispetto alla versione originale di carbon il backend go-carbon è più veloce in tutte le condizioni (la differenza di prestazioni varia in base alla macchina su cui è installato) 4.2.3. Le carbonapi invece sono un subset delle api di graphite-web e le vanno a sostituire essendo dalle 5 alle 10 volte più veloci.

4.2.4 Grafana

è un software open source che permette la visualizzazione e la generazione delle metriche tramite una web application. Permette di creare dashboard dinamiche

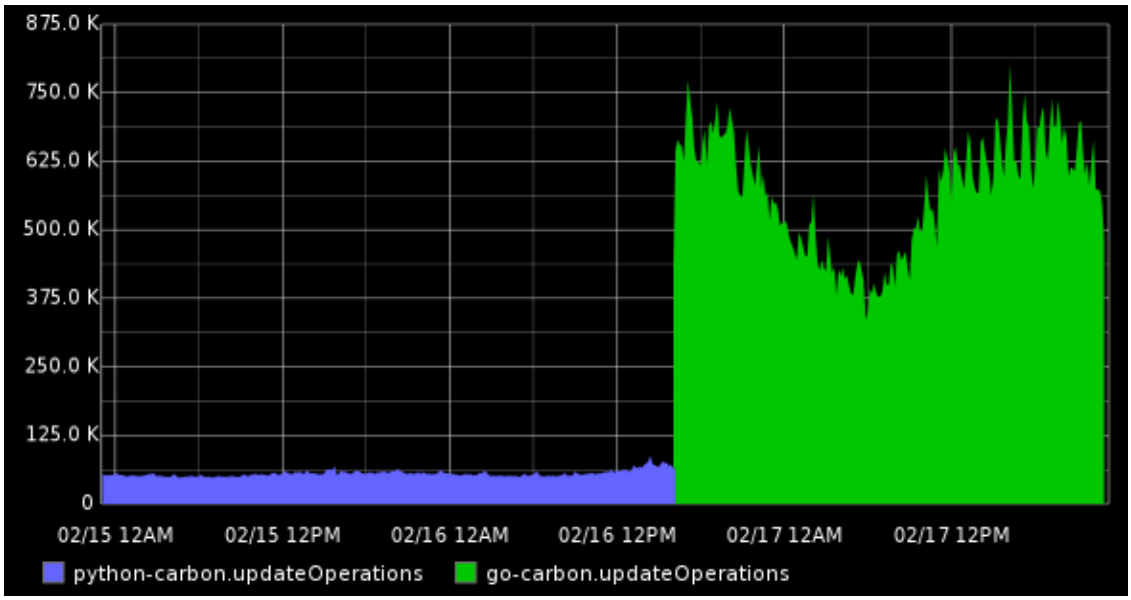

Figura 4.4. Differenza di prestazioni tra carbon e go-carbon

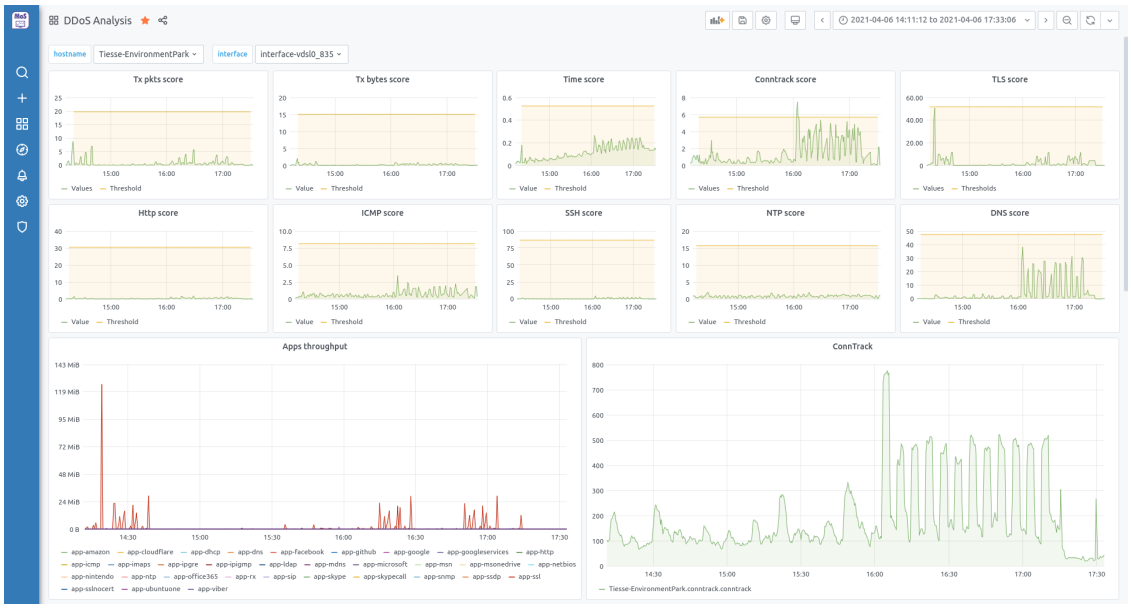

Figura 4.5. La dashboard per l'anomaly detection su Grafana

interrogando le api di graphite-web. Nel mio caso ho organizzato una dashboard in modo da visualizzare sia i dati provenienti dai router da analizzate, sia gli anomaly score calcolati dal software di anomaly-detection 4.2.3.

Riassumendo i dati provenienti dai plugin di collect, dal mio plugin e da NDPI vengono collezionati da collectd e inviati tramite il plugin write_graphite al backend go-carbon, che si occupa di ricevere i dati e salvarli sul file system. Per la visualizzazione dei dati viene usato grafana, che permette di visualizzare i dati richiedendo i dati alle carbonapi e contemporaneamente il mio tool di anomaly detection richiede i dati per analizzarli, ogni volta che ottiene dei risultati li manda a go-carbon per salvarli nel database 4.2.

4.3 Selezione features

La scelta delle feature da utilizzare dipende da quale obiettivo su vuole ottenere, il mio obiettivo primario in questa tesi è di rilevare gli attacchi DDoS in uscita verso la sede centrale, quindi basandomi sugli attacchi più famosi e frequenti ho delineato una lista di parametri da osservare. Questi parametri sono:

- *bytes trasmessi al secondo*:questa metrica è utile, abbinata ad altre, per la rilevazione di attacchi che mirano alla saturazione della banda.
- *pacchetti trasmessi al secondo*: questa metrica ha uno scopo simile alla precedente oppure aiuta ad avere informazioni sugli attacchi di tipo flooding.
- *numero di connessioni aperte*: il numero di connessioni aperte è un indicatore importante per identificare tutti gli attacchi che mirano a saturare le connessioni di un server.
- *numero di pacchetti con flag syn*: questa metrica è molto utile per la rilevazione di syn flood o port scanning.
- *tls throughput*: tutto il traffico inviato su un canale sicuro tls non può essere analizzato più nello specifico, quindi viene raggruppato in questa categoria.
- *dns throughput*: conoscere il traffico relativo al traffico dns può essere utile come indice di un attacco contro il server DNS aziendale oppure di un attacco di DNS amplification.
- *ssh throughput*: potrebbe segnalare anomalie sull'utilizzo improprio di macchine nella sede centrale tramite delle connessioni ssh.
- *ntp throughput*: potrebbe segnalare anomalie riguardanti attacchi DDoS di NTP amplification.
- *icmp throughput*: è un indicatore di attacchi, per esempio durante un syn flood usando ip spoofing, con ip della sottorete dell'attaccante, al ritorno dei syn ack si vede un aumento degli icmp che indicano che l'host non è esistente o che la porta destinazione a cui è destinato il pacchetto è chiusa. Inoltre gli icmp possono anche essere usati direttamente per degli attacchi.

- *ora del giorno*: l'ora del giorno viene aggiunta per caratterizzare al meglio il traffico lungo la giornata.

Tutte le feature vengono poi derivate su un tempo di 10/30s, per avere dei "rate" confrontabili tra lavoro.

La scelta della features è nata da un compromesso tra i dati necessari per rilevare al meglio le anomalie, la riduzione dei dati da salvare sul server e di conseguenza l'uso di banda usata per il trasferimento. Inoltre un problema che ho dovuto tenere in considerazione è l'utilizzo di un acceleratore hardware nei router Tiesse, il quale permette un incremento della velocità di routing, ma non permette di analizzare nel kernel i pacchetti.

4.4 Il nostro tool

Per una migliore raccolta e monitoraggio di features con diversa granularità, abbiamo sviluppato un ulteriore software in grado di estrarre metriche relative a server/servizi aziendali specifici o metriche non disponibili tramite i plugin ufficiali di collectd, come per esempio il syn rate. Per svilupparlo abbiamo programmato un modulo kernel, che comunicava con un plugin per collectd da noi creato tramite un process file system (procfs), per una discussione più approfondita rimando al capitolo 5.

Questa soluzione è funzionante ed è stata testata in laboratorio, però non viene utilizzata per la raccolta di dati necessaria a causa dell'impossibilità di introdurre software aggiuntivo in un router in produzione, da cui dipendeva la produttività dell'ufficio. Inoltre i router Tiesse sfruttano un acceleratore hardware, in cui solo il primo pacchetto di ogni flusso passa tramite il kernel e i successivi vengono inoltrati automaticamente in hardware.

4.4.1 Struttura

Il programma è scritto in Python 3, con l'utilizzo di TensorFlow e Keras librerie per il "machine learning", effettua le richieste http tramite la libreria requests e gestisce i dati grazie a NumPy e Pandas: librerie per il calcolo matriciale e la gestione di tabelle e serie. Il tutto, per assicurare una maggiore portabilità, ed assicurare la possibilità di effettuare il deploy su qualsiasi macchina, viene eseguito all'interno di un container Docker, la cui immagine viene creata e avviata tramite "docker-compose".

I dati all'interno del repository sono organizzati nel seguente modo:

```
/
├── data ..... Cartella dove verrà salvato il modello e le immagini generate
├── src ..... Cartella con i sorgenti del programma
│   └── requirements.txt ..... Librerie di python necessarie
```

- `utility.py` ..... File in cui sono presenti le funzioni comuni
- `update_db.py` ..... Funzioni per caricare i risultati su graphite
- `evaluate.py` File con le funzioni per valutare se sono presenti anomalie
- `train.py` ..... File con funzioni per eseguire l'allenamento della rete
- `model.py` ..... File con funzioni per generare il modello
- `test.py` ..... Script che esegue la creazione di un modello, il train e la successiva valutazione di anomalie, basandosi su un file di impostazioni
- `main.py` .... Script che esegue la creazione e il train del modello se non presente e a intervalli regolari verifica le anomalie

- `test_configs`
  - `test_docs_v3.json`..... Un esempio di file di configurazione usato per effettuare i test della validità del modello
- `Dockerfile`
- `docker-compose.yml`
- `README.md`

4.4.2 Keras e TensorFlow

Tensorflow 2 è una libreria open-source sviluppata da Google per il machine learning che fornisce moduli ottimizzati per la realizzazione di algoritmi di machine learning e la loro esecuzione in maniera efficiente su CPU, GPU e TPU. Inoltre permette di scalare agevolmente su un'architettura con molti device.

Keras è un insieme di API ad alto livello di TensorFlow 2 che forniscono astrazioni essenziali per incentrarsi sulla facilità d'uso, la modularità e l'estendibilità per la risoluzioni di problemi relativi al machine learning, con una particolare concentrazione sui moderni problemi di deep learning.

4.4.3 Elaborazione dei dati in input

Le metriche relative alle applicazioni vengono aggiornate ogni 30 secondi, a differenza dei 10 secondi con cui aggiorno i dati collezionati da collectd, per questo motivo ogni volta che devo richiedere dei dati al server devo effettuare due richieste, tramite la libreria requests, alle carbonapi.

Un esempio di dati restituiti dalla prima richiesta è mostrata nel codice 4.1, in cui per ridurre la quantità di dati da mostare nell'esempio, ipotizzo che l'intervallo di tempo richiesto sia di venti secondi.

Listing 4.1. Esempio di risposta delle carbonapi

```
1 [
2     {
3         "target": "Tiesse-EnvironmentPark.conntrack.conntrack",
4         "datapoints": [
```

```
            [
                135,
                1619681610
            ],
            [
                134,
                1619681620
            ]
        ],
        "tags": {
            "name": "Tiesse-EnvironmentPark.conntrack.conntrack"
        }
    },
    {
        "target": "Tiesse-EnvironmentPark.interface-vdsl0_835.
           if_packets.tx",
        "datapoints": [
            [
                21.199091,
                1619681610
            ],
            [
                26.401902,
                1619681620
            ]
        ],
        "tags": {
            "name": "Tiesse-EnvironmentPark.interface-vdsl0_835.
               if_packets.tx"
        }
    },
    {
        "target": "Tiesse-EnvironmentPark.interface-vdsl0_835.
           if_octets.tx",
        "datapoints": [
            [
                4202.222148,
                1619681610
            ],
            [
                5862.413248,
                1619681620
            ]
        ],
        "tags": {
            "name": "Tiesse-EnvironmentPark.interface-vdsl0_835.
               if_octets.tx"
        }
    }
]
```

I dati ricevuti per essere elaborati devono essere memorizzati in una tabella con il corretto formato, vedi l'esempio di tabella 4.1, per questo motivo i dati del json appena ricevuto vengono elaborati, sostituendo per prima cosa il timestamp, il quale viene trasformandolo da un unix timestamp ad uno che indica i secondi passati dalla mezzanotte e successivamente tutti gli elementi verranno aggiunti ad un dizionario con il corretto formato, come mostrato nell'estratto di codice 4.3.

index	timestamp	.conntrack.conntrack	..if_packets.tx	..if_octets.tx
1	45810	71.0	7.500424	2416.445886
2	45820	74.0	4.89923	51069.831629415
3	45830	72.0	5.000281	1224.055123

Tabella 4.1. Tabella di esempio dei dati ricevuti relativi alle features di collect.

Listing 4.2. Funzione usata per scaricare i dati dal server

```
def get_data(s_time, e_time, hostname: str, interface: str,
    username: str = None, password: str = None):
    r = requests.get(
    f"http://{host}:{port}/api/datasources/proxy/4/render?"
    f"target={'&target='.join(target_list(hostname,interface))}"
    f"&from={s_time}&until={e_time}&format=json",
    auth=HTTPBasicAuth(username, password)
    )
    # se il valore ritornato non e' 200 OK viene
    # lanciata un'eccezione, gestita a livelli superiori
    print(r.status_code, '-', r.url)
    if r.status_code != 200:
        raise Exception('Get features data error')

    # converto la risposta in un json
    # e poi in un dizionario con chiave il target
    # e come valore i datapoints
    json_resp = r.json()

    # todo: probabilmente ci sono modi piu' efficienti
    data = data_to_dict(json_resp)
    data_list = list()
    for i in range(len(json_resp[0]['datapoints'])):
        line: dict = dict()
        timestamp =
        datetime.fromtimestamp(json_resp[0]['datapoints'][i][1])
        # trasformo il tempo in secondi dalla mezzanotte
        line['timestamp'] = timestamp.hour * 3600 \
                            + timestamp.minute * 60 \
                            + timestamp.second
        tl = target_list(hostname, interface)
```

```
32          for e in tl:
33              line[e] = data[e][i][0]
34          data_list.append(line)
35
36      # creo la matrice con pandas ed elimino le righe con valori
37      # mancanti in rari casi le carbon api mi restituiscono dei
38      # valori nulli, per questo motivo cancello le righe in cui
39      # c'e' almeno un valore nullo
40      matrix = pd.DataFrame(data_list).dropna()
41
42      # successivamente richiedo anche i dati delle app, e quando
43      # li ricevo, dopo averli manipolati, li unisco tutti
44      # in un'unica tabella
45      app_data_matrix =
46      get_app_data(s_time, e_time, hostname, username, password)
47
48      matrix = pd.merge(app_data_matrix, matrix)
49
50      # questa funzione mi permette di ritornare l'elenco
51      # delle feature attualmente utilizzate, senza impostare
52      # host e interfaccia da utilizzare
53      matrix.columns = total_target_list('', '')
54      return matrix
```

Ricevuti i dati e organizzati nel formato corretto, effettuo una nuova richiesta alle carbon api, per ricevere le statistiche relative alle applicazioni, il procedimento è simile a quello precedentemente descritto, l'unica differenza è che i dati sul database vengono salvati da ndpi ogni 30 secondi, per questo motivo devo estenderli mostrando lo stesso dato più volte, in modo da avere lo stesso numero degli elementi ottenuti precedentemente.

```
1  for element in app_list:
2      # se l'app non esiste nella risposta metto uno 0
3      line[element] =
4       data[element][i][0] if element in data else 0.0
5  # estendo i dati, replicando lo stesso dato ogni 10s per un totale
       di 30s
6  for j in range(3):
7      line['timestamp'] += 10
8      data_list.append(line.copy())
```

index	tabella	da	rifare	SBAGLIATA!
1	45810	71.0	7.500424	2416.445886
2	45820	74.0	4.89923	51069.831629415
3	45830	72.0	5.000281	1224.055123

Tabella 4.2. Tabella di esempio dei dati ricevuti relativi alle app di nDPI

La standardizzazione dei dati è la fase subito successiva, Durante il train calcolo la media e la deviazione standard sui dati per ogn, successivamente standardizzo e salvo la media e deviazione standard di ogni feature su un file. Durante la fase di evaluate per standardizzare i dati uso i dati salvati precedentemente sul file e standardizzo i dati.

Scelta della lunghezza delle sequenze dei dati. Utilizzando i dati standardizzati genero "N-K" sequenze di lunghezza "K" elementi, dove "N" è il numero di elementi ricevuti precedentemente. Queste sequenze mi servono per uniformare i dati di allenamento e valutazione usando una lunghezza fissa per la sequenza di dati da dare in input alla rete neurale. La scelta della lunghezza della sequenza è molto importante, perchè da essa dipende la capacità di rilevamento delle anomalie: più il valore sarà grande e più sarà preciso il riconoscimento delle anomalie e meno il programma sarà soggetto a falsi positivi, ma al tempo stesso renderà di difficile identificazione i picchi anomali e il tempo necessario per rilevare un'anomalia sarà maggiore. Dopo un attenta analisi, spiegata successivamente, abbiamo scelto una lunghezza della sequenza pari a 12 elementi, tenendo anche in considerazione che il fatto che il 25% degli attacchi ha una durata inferiore ai 10 minuti [22], quindi è importante rilevare attacchi di breve durata.

4.4.4 Modello della rete

Il modello di una rete neurale serve a descrivere le interconnessioni, le diverse tipologie e la composizione dei livelli di una rete neurale, è un oggetto della libreria Keras, che può essere facilmente creato grazie alla funzione "tf.keras.Model()". Nel nostro caso ogni livello prevedeva un solo livello in ingresso e uno in uscita, per questo motivo abbiamo usato il "sequential model" di Keras. La scelta della rete è molto importante per ottenere un buon risultato nel rilevamento delle anomalie, per questo motivo abbiamo effettuato molte prove con diverse tipologie di reti: sia Dense feedforward, sia LSTM recurrent, i cui risultati verranno analizzati alla conclusione del capitolo.

Nelle reti dense abbiamo usato un modello a tre o cinque livelli: uno di input e uno di output con dimensioni (lunghezza_sequenza, numero_features) e livelli con prima una riduzione e poi un incremento dei nodi, come nei classici autoencoders. Avendo in ingresso dei dati con più dimensioni ho dovuto linearizzare i dati in ingresso (flatten) e ricostruire il vettore multidimensionale in uscita (reshape), nei nodi intermedi invece abbiamo fatto delle prove con diverse configurazioni di numero di nodi. Per creare la rete sono stati usati "Dense layers", in cui cui i nodi di due livelli sono interamente connessi tra loro e sono state usate come funzioni di attivazioni "relu" (Equazione 4.1) per i nodi intermedi e "linear", una funzione lineare, per i nodi dell'ultimo livello, questo permette di rappresentare anche i

numeri negativi. Inoltre abbiamo aggiunto dei "Dropout layers", un livello che imposta casualmente l'input del livello a zero, con una frequenza determinata, durante la fase di train, questo permette di ridurre il problema dell'overfitting.

$$f(x) = x^+ = max(0, x) \tag{4.1}$$

Nelle reti di tipo LSTM

Il modello viene infine compilato e salvato su file, in modo da potere essere usato successivamente.

Listing 4.3. Funzione usata per generale il modello della rete neurale

```
sequential_model = keras.Sequential(
    [
        layers.Flatten(input_shape=shape),
        layers.Dense(25, activation='relu'),
        layers.Dropout(rate=0.2),
        layers.Dense(8, activation='relu'),
        layers.Dropout(rate=0.2),
        layers.Dense(25, activation='relu'),
        layers.Dropout(rate=0.2),
        layers.Dense(shape[1]*shape[0], activation='linear'),
        layers.Reshape((shape[0], shape[1]))

    ]
)
```

4.4.5 Train

Il train deve essere effettuato su degli intervalli di tempo in cui non si sono verificate anomalie, per questo motivo viene fatta l'assunzione che tutto il traffico generato non presenti anomalie a meno di piccoli intervalli in cui è stato volutamente generato traffico malevolo per scopo di test. Volendo se l'amministratore di sistema notasse altri periodi di traffico anomalo potrà escluderli dai dati di allenamento. Inoltre per avere un maggiore numero di dati, e velocizzare l'apprendimento dell'allenamento della rete, è possibile utilizzare i dati provenienti da più router delle sedi periferiche, se ipotizziamo che il traffico generato sia simile tra loro. Dalla selezione degli intervalli di tempo su cui si vuole effettuare il train per semplicità in questa prima versione di algoritmo vengono esclusi i weekend: giorni della settimana in cui, nel nostro caso, il traffico è molto diverso dagli altri, volendo risolvere il problema si potrebbe usare una seconda rete da usare solo nei giorni festivi o Nella fase di train per prima cosa verrà effettuato la raccolta e la trasformazione dei dati dal server, come spiegato precedentemente e avere caricato da file il modello generato. Successivamente prima di effettuare il train vero e proprio, grazie alla funzione "train_test_split" della libreria sklearn i dati vengono divisi in due insiemi, quello di train e quello di test, con il 15% dei dati usati per il test e il restante per il train.

L'allenamento viene effettuato usando il metodo "fit" della classe model e usando in input sia per le x, che per le y, l'insieme di train appena generato e l'insieme di test per validare il modello. Inoltre è abilitato l'EarlyStopping, in cui se per più di cinque iterazioni non si hanno miglioramenti, viene stoppato il processo di allenamento, questo serve per ridurre i tempi di train e ridurre la possibilità di overfitting.

Come spiegato precedentemente, l'utilizzo degli autoencoders permette di ottenere degli "anomaly score" per ogni sequenza di dati, quindi terminata la fase di allenamento della rete devo decidere sopra quale valore devo considerare i dati anomali. Le soglie sopra i quali considero i dati anomali vengono prese dal massimi valore degli anomaly score per ogni feature: avendo ipotizzato che tutti i dati forniti per il train non siano anomali, devo assicurarmi che effettuando la valutazione delle anomalie su quei dati nulla risulti anomalo. Per calcolare l'anomaly score calcolo il valore medio dei valori assoluti delle differenze tra il valore originale e quello ricostruita.

$$anomaly\_score = \frac{\sum_{elementi\_sequenza} |valore\_originale - valore\_ricostruito|}{lunghezza\_sequenza} \tag{4.2}$$

La scelta di questo sistema permette di evidenziare sia anomalie sul traffico anomalo dovuto a variazioni verso l'alto dei valori normali, sia verso il basso, ma anche i valori ondulatori possono essere notificati, perchè la somma delle distanze delle ricostruzioni del valore assoluto di ogni istante da quello reale si discosterà molto, nonostante il valore medio sia in linea con quello aspettato.

Le soglie calcolate vengono salvate su un file, da potere usare successivamente per la valutazione delle anomalie.

Quando eseguire il train?

Un importante aspetto da tenere in considerazione è la stazionarietà dei dati

4.4.6 Evaluate

Allenato il modello a ricostruire l'input e calcolate sopra le quali considerare gli anomaly score anomali è possibile verificare se il traffico generato in un intervallo di tempo da un determinato router è anomalo. Per mettere in atto questa fase, dopo avere trattato i dati in ingresso nello stesso modo dei dati per l'allenamento, leggo dal file generato precedentemente le soglie e verifico se ogni sequenza di k elementi è anomala, calcolando gli anomaly score come effettuato nell'ultima fase del train. Se gli anomaly score superano le soglie si è verificata un'anomalia, a questo punto viene segnalata all'amministratore di rete e viene attivata la fase di mitigazione sul router.

Come detto precedentemente l'utilizzo della rete potrebbe variare nel tempo, anche nel breve periodo o tra un allenamento e il successivo, per questo motivo è importante non segnalare come anomalo le piccole variazioni che hanno superato i

precedenti massimi, per questo motivo è stato introdotto un ulteriore margine sulla soglie. Nel nostro caso la sede di Torino presenta un traffico molto variabile nel breve periodo per questo motivo abbiamo deciso di segnalare le anomalie solo nel caso in cui le soglie vengano superate del 120%.

Questa fase viene ripetuta automaticamente dal software ogni periodo di tempo definito (nell'ordine dei trenta secondi) per i dati di ogni router di cui si vogliono monitorare le anomalie.

4.5 Test sulle anomalie

L'analisi delle anomalie in questa tesi ha l'obiettivo principale di rilevare attacchi DDoS, quindi basandoci sullo studio degli attacchi più popolari abbiamo selezionato un ristretto elenco di attacchi possibili da riprodurre:

- SYN flood
- ICMP flood
- UDP flood
- DNS flood
- DNS amplification

4.5.1 Tool utilizzati

Per effettuare le varie tipologie di attacchi sono stati usati software open-source reperibili sul web e sono stati scritti dei tool per adeguarsi al meglio alle nostre esigente.

hping3 è un tool in grado di generare pacchetti di rete TCP/IP personalizzati. Il nostro utilizzo è stato per generare attacchi di tipo SYN flood, ICMP flood e UDP flood. Il tool, oltre a permettere di generare i pacchetti personalizzati da mandare alla vittima, permette di regolare la portata dell'attacco, tramite le possibilità di sceltà del rate a cui inviare i pacchetti.

Per cosa è stato usato, quali sono i limiti di hping3 e perchè dobbiamo

Il nostro toool Hping3 non consente di riprodurre tutte le tipologie di attacco a noi necessarie, per questo motivo è stato sviluppata un'applicazione per costruire pacchetti di tipo DNS da utilizzare per creare attacchi di tipo DNS flood e DNS amplification. Il tool è sviluppato in C++, per una maggiore efficienza e per raggiungere un maggiore throughput durante gli attacchi, utilizzando la libreria

"libtins"[1]. Il programma genera delle DNS query indirizzate verso i DNS server configurati in un file di configurazione. A scopo di test, per non sovraccaricare il server DNS aziendale, abbiamo utilizzato dei Raspberry Pi, con in esecuzione dei container di CoreDNS: semplici istanze DNS configurate in modo da mantenere in cache le risposte alle richieste effettuate e abbiamo effettuato solo richieste di risoluzione di un particolare dominio da parte del software di attacco. Inoltre per semplicità d'uso il software permette anche di effettuare attacchi di tipo syn flood con ip spoofing.

4.6 Risultati

Le reti neurali sono sistemi non lineari decisamente complessi, con risultati abbastanza imprevedibili, difficili da padroneggiare nella loro interezza. Per questo motivo abbiamo svolto alcuni test sperimentali sui dati reali generati come spiegato precedentemente.

Le reti neurali usata per i test sono state allenate usando i dati di circa due mesi e i test sono stati effettuati per verificare sia l'esistenza di falsi positivi, che di falsi negativi.

4.6.1 Test effettuati

Per verificare qual'è la migliore lunghezza della finestra da utilizzare o qual'è il migliore modello di rete neurale, sono stati effettuati dei test su 20 intervalli di dati (Tabella 4.3) che presentano delle peculiarità e su cui la rete non era stata precedentemente allenata. Inoltre oltre ad analizzare i risultati dei test controllo che il modello approssimi veramente i dati in ingresso e che i buoni risultati non siano dovuti solo a valori di soglie calcolate durante l'allenamento molto alte.

Per mostrare un risultato visivo delle ricostruzioni sono stati disegnati dei grafici approssimati, mostrare un grafico con i dati ricostruiti dell'intervallo iniziale sarebbe impossibile, perchè fornisco alla rete in input delle sequenze di N elementi e non l'intero intervallo, quindi per avere un'idea di come potrebbe essere il grafico ricostruito sono stati presi i primi valori di ogni sequenza, in modo da avere tutti i dati per ricomporre l'intero intervallo.

4.6.2 Scelta della finestra

Per la scelta della lunghezza della finestra abbiamo effettuato alcuni test

[1] http://libtins.github.io/

#test	data	inizio	fine	Tipologia	Da considerare anomalo?
1	10/03/21	12.13	12.19	Syn Flood 100pps 40s	SI
2	10/03/21	11.24	11.30	Upload 500MB su Google Drive	SI (è un evento mai successo)
3	10/03/21	11.16	11.21	Download 600MB da Google Drive	NO
4	22/03/21	08.01	08.24	Traffico di una normale mattinata	NO
5	09/03/21	17.12	17.19	Picco di traffico NTP	NO
6	23/03/21	08.57	17.46	Traffico di una giornata normale	NO
7	31/03/21	10.28	10.42	Picchi di traffico NTP	NO
8	31/03/21	10.50	11.02	Syn Flood di 2min 50pps	SI
9	31/03/21	11.01	11.09	Syn Flood di 30s 50pps	SI
10	31/03/21	11.05	11.15	Port scanning (nmap)	SI
11	31/03/21	11.24	11.33	ICMP Flood 100 pps	SI
12	31/03/21	11.33	11.43	UDP Flood 100pps	SI
13	31/03/21	11.57	12.06	DNS Amplification (4Mbps)	SI
14	31/03/21	12.11	12.20	DNS Flood (4Mbps)	SI
15	18/05/21	13.49	14.00	Picco di connessioni (2500)	Dipende
16	18/05/21	12.57	13.19	Picco di connessioni	NO
17	18/05/21	12.47	13.05	Picco di trasmissione	NO
18	18/05/21	08.50	10.30	Connessioni alte	NO
19	19/05/21	7.47	11.05	Traffico di una normale mattinata	NO
20	19/05/21	11.24	17.28	Traffico di un pomeriggio con dei picchi di traffico NTP	NO

Tabella 4.3. Tabella dei test

4.6.3 Scelta del modello e risultati

Capitolo 5

Mitigazione degli attacchi

5.1 Introduzione

Mitigare gli attacchi DDoS è un problema di più difficile risoluzione rispetto alla sola rilevazione, perchè bisogna conoscere maggiori informazioni sulla provenienza del flusso malevolo e il problema dei "false positive" è maggiormente sentito: se durante la fase di detection rileviamo un falso positivo e lo notifichiamo all'amministratore di sistema non è un grande problema perchè sarà lui ad effettuare una successiva analisi prima di intraprendere azioni correttive. Se ci prefiggiamo l'obiettivo di bloccare automaticamente i flussi malevoli dei falsi positivi significherà degradare la connessione ad utenti legittimi.

5.1.1 Ip spoofing

L'ip spoofing è una tecnica che permette di costruire pacchetti IP con indirizzo IP sorgente modificato con lo scopo di fingersi un altro dispositivo o nascondere la propria identità.È un grande problema di sicurezza delle reti, principalmente perchè permette effettuare attacchi DDoS, permettendo di effettuare attacchi come l'amplificazione DNS oppure rendendo più difficile l'identificazione della sorgente dell'attacco nelle altre tipologie.

Bloccare l'ip spoofing Per mitigare questo problema abbiamo introdotto una regola iptables nel router 5.1, la quale impedisce l'inoltro di pacchetti provenienti da sottoreti diverse da quella in cui è situato il router. Iptables è un firewall integrato nel kernel linux.

Listing 5.1. Esempio di regola iptables

```
mettere la regola qui
```

Questa soluzione impedisce solo parzialmente l'utilizzo dell'ip spoofing, perchè sarà sempre possibile generare pacchetti con ip spoofing provenienti da qualsiasi indirizzo ip della sottorete in cui si trova l'attaccante.

5.2 Tool utilizzati

Prova prova

Per mitigare un attacco devo conoscere un maggior numero di informazioni riguardo al traffico in forma non aggregata, per questa motivazione utilizzo degli ulteriori tool per raccogliere dati sui flussi transitanti per un'interfaccia in maniera non aggregata.

5.2.1 Netflow

Netflow è un software introdotto inizialmente nei router Cisco nel 1996, successivamente è stato creata un'estensione standardizzata dall'IETF, chiamata IPFIX. Netflow è uno dei tool di monitoring più famosi e permette di monitorare e registare informazioni riguardo i flussi che attraversano una determinata interfaccia.

Poichè non è possibile estendere le features di netflow e non rispecchiano completamente i nostri interessi abbiamo deciso di progettare il nostro agent utilizzando altre soluzioni.

5.2.2 Modulo Kernel

Nella sezione [**?**] è stata accennata la possibiltà di raccogliere dati con maggiore granularità tramite l'utilizzo di un modulo kernel scritto appositamente.

Nella caso in cui si vogliano aggiungere dei software per collezionare dati maggiormente granulari bisogna intervenire lato kernel, perchè in userspace non esistono meccanismi per analizzare tutti i pacchetti in transito su un'interfaccia. Di conseguenza per aggiungere delle funzioni al kernel Linux esisto solitamente tre soluzioni: la prima è quella di fare aggiungere il codice ufficialmente al suo interno, questa soluzione è la più complicata: sono necessarie le giuste motivazioni per convincere i manteiner ad adottare quella soluzione e inoltre passeranno anni prima della distribuzione in distribuzioni stabili. La seconda è quella di creare un modulo kernel personalizzato, i moduli kernel sono una porzione di codice compilato che possono essere inseriti nel kernel a run-time. Questo metodo è sicuramente più rapido, ma potrebbe portare a vulnerabilità, problemi di compatibilità all'aggiornamento del kernel e rischi di deadlock per l'intero sistema.La terza possibilità è l'uso di eBPF, che esploreremo successivamente.

La prima versione del nostro software è stata sviluppata scrivendo un modulo kernel che utilizzava netfilter. Questa soluzione è la strada tradizionalmente usata nel mondo Linux e in Tiesse per l'implementazione delle proprie personalizzazioni

nei prodotti. Netfilter è un componente di tutte le moderne distribuzioni Linux, che permette di intercettare e manipolare pacchetti in transito dal router, usato nel kernel permette di registrare delle callback, eseguite alla ricezione di ogni pacchetto su un determinato punto d'aggancio. Per la scrittura del modulo siamo partiti dalla compilazione del codice sorgente del kernel del router. successivamente è stato scritto un programma in linguaggio C, che sfruttando netfilter aggancia ad un hook, più in particolare "NF_INET_POST_ROUTING", una funzione da eseguire alla ricezione di ogni pacchetto, la quale incrementa dei contatori di un vettore. Terminata la scrittura, il modulo deve essere compilato, nel nostro caso cross compilato per architecture "arm64". La compilazione generare un file con estensione .ko e dopo averlo copiato sul router, dovrà essere inserito dinamicamente nel kernel del router tramite il comando "insmod [nome_modulo].ko".

Inoltre in userspace è stato scritto un plugin per collectd, il quale ogni volta che necessita dei dati effettua una lettura su un "process file system" (procfs), uno pseudo-filesystem usato per accedere alle informazioni fornite dai processi kernel, tramite il quale il nostro modulo kernel ritorna un JSON con il nome della metrica presa in considerazione e il valore.

Questa soluzione, nonostante fosse la soluzione usata da sempre in azienda, presentava alcuni problemi come la maggior difficoltà di comunicazione tra user-space e kernel-space, la mancanza di strutture dati già esistenti e soprattutto un problema in questo modulo kernel rischiava di portare ad un completo malfunzionamento dell'intero router, per questa ragione abbiamo deciso di adottare una soluzione più moderna.

5.2.3 Berkley Packet Filter (BPF)

Il Berkley Packet Filter (BPF) è una tecnologia introdotta nel kernel di alcuni sistemi operativi, tra cui il kernel linux, su cui è stata introdotta a partire dalla versione 2.1.75 del kernel, nel 1997. BPF consiste in una macchina virtuale, con una virtual cpu special purpose, integrata nel kernel, che permette il filtraggio e l'analisi dei pacchetti in transito su un'interfaccia. È una soluzione che permette di eseguire un bytecode in maniera sicura nello spazio kernel utilizzando una sandbox ed eliminando l'overhead delle system call e del context switching tra kernel e user. Per effettuare il filtering tramite BPF deve essere generato del codice in un particolare assembly in grado di funzionare sulla CPU virtuale, che verrà richiamato alla ricezione di ogni pacchetto sull'interfaccia determinata (event driven). Un esempio del suo utilizzo è tcpdump che sfrutta BPF per effettuare il filtraggio in maniera efficiente. L'esempio di bytecode 5.2 permette di filtrare i pacchetti IPV4, per farlo verifica se l'ethertype è uguale a 0x800 [18].

Listing 5.2. Esempio delle istruzioni per il filtering dei pacchetti IP in BPF assembly

```
user@linux$ tcpdump -d ip
(000) ldh [12]
```

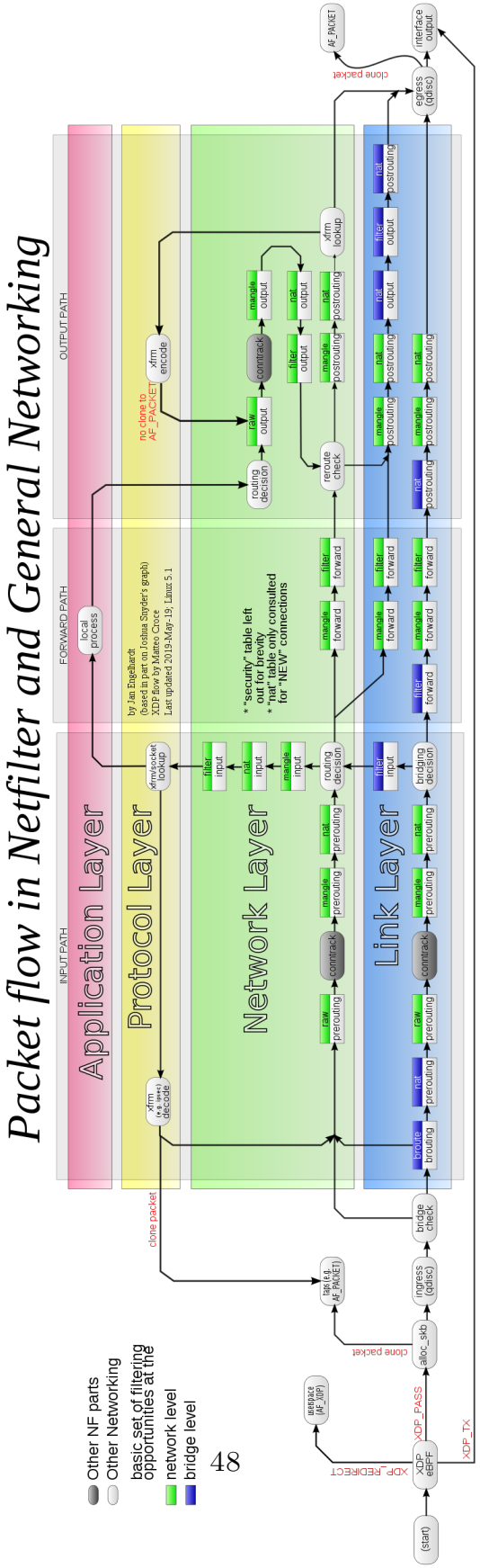

Figura 5.1. Flusso dei pacchetti in Netfilter

```
(001) jeq #0 x800 jt 2 jf 3
(002) ret #96
(003) ret #0
```

5.2.4 extended Berkeley Packet Filter (eBPF)

La extended Berkeley Packet Filter è una versione che estende la versione originale ed è stata introdotta nella versione 3.18 del kernel Linux, ai giorni nostri la versione originale è completamente obsoleta e anche il codice scritto per la vecchia versione, per esempio tcpdump, viene tradotto trasparentemente per la nuova [16].

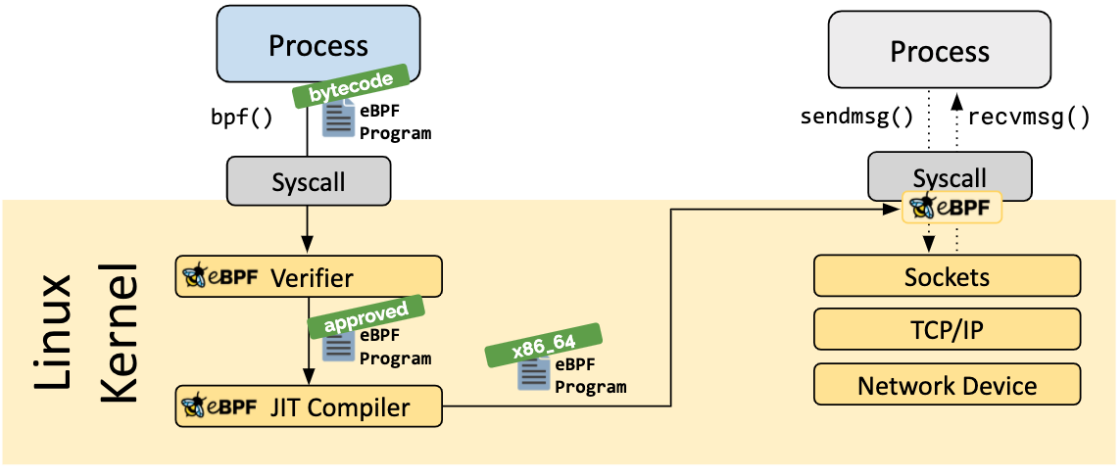

Figura 5.2. Architettura eBPF [19].

Prima che il bytecode sia caricato nel kernel Linux vengono effettuati due passaggi, come si può notare dall'immagine 5.2.4. La prima è la "Verication", la quale si occupa di verificare che il programma sia sicuro da eseguire e verifica che: il programma abbia lunghezza limitata, che non ci siano accessi a indirizzi di memoria non validi e che abbia una fine, di conseguenza controlla che non siano presenti cicli infiniti. Effettuata la verifica viene eseguita la "Just-in-Time (JIT) compilation", la quale traduce il bytecode generico in istruzioni specifiche per la macchina che lo sta eseguendo, questo permette di incrementare le prestazioni e rendere i programmi eBPF efficienti come i moduli kernel compilati nativamente [19]. eBPF inoltre fornisce strumenti utili per lo sviluppo, come le mappe e gli helpers.

Le mappe sono un importante aspetto dei programmi eBPF, esse permettono di memorizzare dati in delle strutture chiave/valore in kernel space, che possono essere condivise con altri programmi eBPF oppure con applicazioni in user space [16].

La chiamata di generiche funzioni a livello kernel non è consentita, sia per mantenere la compatibilità del programma non solo con una determinata versione del kernel, per questo motivo sono stati introdotti gli "helpers", inoltre permettono l'esecuzione di istruzioni o permettono di eseguire alcuni task non permessi dall'assembly eBPF per motivi di sicurezza.

Un ulteriore caratteristica di eBPF è la possibilità di potere concatenare l'esecuzione dei programmi, chiamandoli a cascata,in maniera simile alla "execve()", durante l'esecuzione.

I vantaggi di un programma eBPF rispetto alla scrittura di un modulo kernel sono molteplici: permettono l'esecuzione sicura del codice senza la possibilità che vada a corrompere il kernel, non c'è rischio che le nuovi versioni del kernel vadano a interrompere il suo funzionamento e garantisce le stesse prestazioni.

5.2.5 eXpress Data Path (XDP)

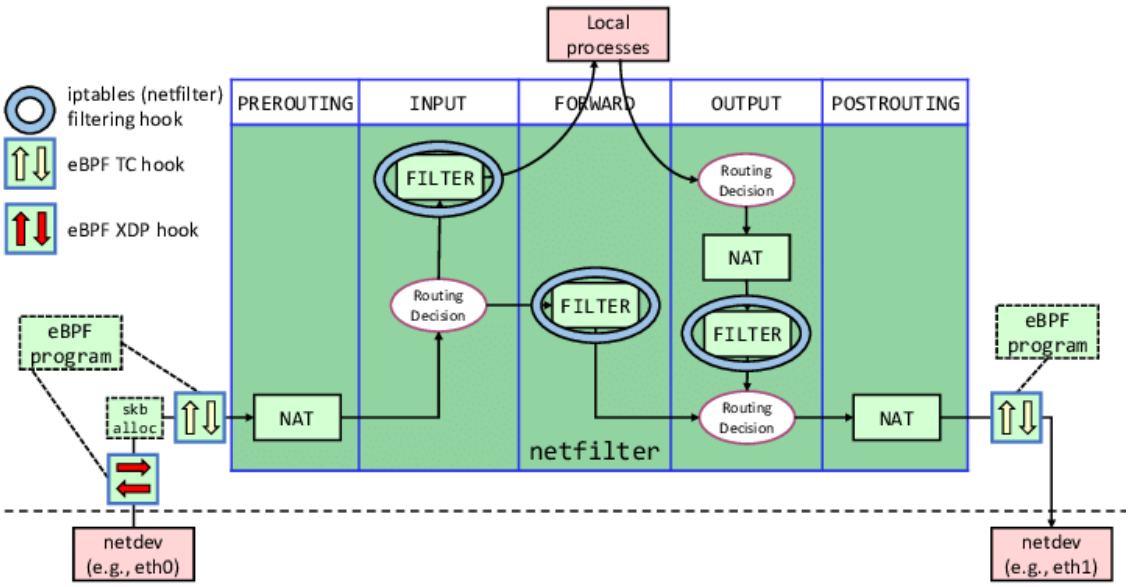

Figura 5.3. Hooks eBPF

Il kernel Linux supporta più punti dello stack di rete in cui è possibile agganciare l'esecuzione dei programmi eBPF. XDP (eXpress Data Path) è un hook ad alte performance che permette di eseguire un programma eBPF al primo punto possibile, questo permette grandi performance, perchè il processamento avviene prima che qualsiasi altro processamento possa avvenire. XDP è un punto d'aggancio ideale per il filtering dei pacchetti malevoli o inaspettati e in qualsiasi comune protezione anti DDoS [17]. Il programma inserito su un hook XDP può compiere cinque azioni possibili:

- XDP_PASS: lascia passare il pacchetto nel normale stack di rete
- XDP_DROP: ignora semplicemente il pacchetto, senza compiere ulteriori azioni
- XDP_TX, restituisce il pacchetto alla stessa interfaccia, normalmente modificato
- XDP_ABORTED: è un valore di ritorno che il programmatore non dovrebbe utilizzare, segnala un che si è verificato un errore nel programma eBPF, per esempio una divisione per zero e contemporaneamente ignora il pacchetto come l'XDP_DROP
- XDP_REDIRECT: reindirizza il pacchetto verso un'altra scheda di rete, o in user space

5.2.6 BPF Compiler Collection (BCC)

È un framework che permette la scrittura di programmi python con programmi eBPF incorporati al loro interno. Il framework ha come obiettivo primario i casi in cui eBPF è utilizzata per raccogliere statistiche o compiere azioni in user space al verificarsi di eventi. L'esecuzione del programma eBPF genererà il bytecode che sarà caricato nel kernel [19]. Il programma python potrà comunicare con quello eBPF grazie all'utilizzo delle mappe precedentemente menzionate.

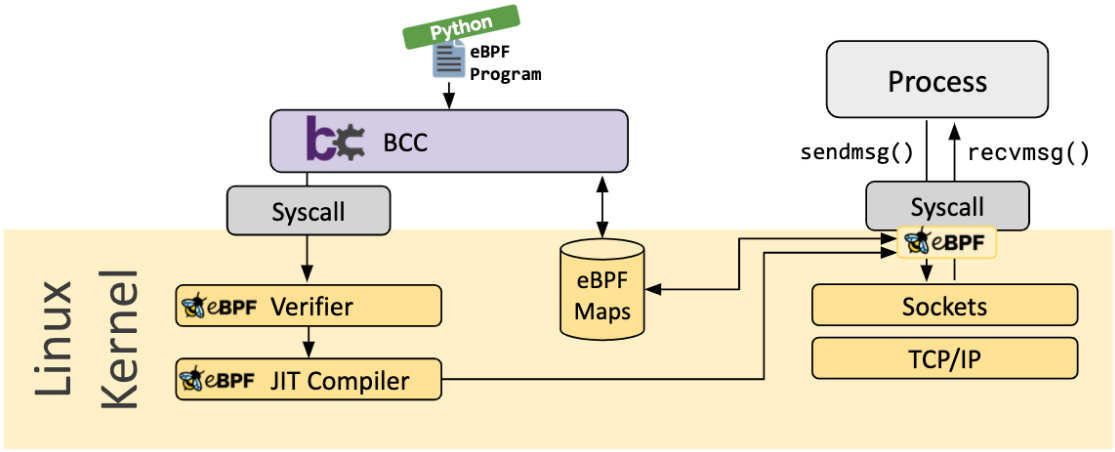

Figura 5.4. Funzionamento di BCC [19]

5.3 Funzionamento

Qua posso parlare di due alternative, la prima è riutilizzare il sistema di anomaly detection simile a quello presentato precedentemente elencando tutti i problemi e i vantaggi.

Il secondo consiste nel raccogliere i dati come prima, e creare un ranking per ogni feature risultata anomala precedentemente e a quel punto blocco i flussi sopra una certa soglia, ma quale?

Mentre per l'ip spoofing come la gestisco?

5.4 Test sulle anomalie

5.4.1 Tool utilizzati

5.4.2 Risultati

Capitolo 6

Lavoro futuro

Capitolo 7

Conclusioni

Sicuramente una cosa da migliorare è la sicurezza per assicurarsi che neanche le sedi periferiche non siano infettate

Elenco delle figure

Bibliografia

[1] Saman Taghavi Zargar, James Joshi and David Tipper *A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks*, 2013. ()

[2] s

[3] Tasnuva Mahjabin, Yang Xiao, Guang Sun and Wangdong Jiang *A survey of distributed denial-of-service attack, prevention, and mitigation techniques*, 2017. (https://journals.sagepub.com/doi/10.1177/1550147717741463)

[4] Neha Agrawal and Shashikala Tapaswi *Defense schemes for variants of distributed denial-of-service (DDoS) attacks in cloud computing: A survey*, 2017. (https://doi.org/10.1080/19393555.2017.1282995)

[5] Jasmeen Kaur Chahal, Abhinav Bhandari and Sunny Behal *Distributed Denial of Service Attacks: A Threat or Challenge*, 2019. (https://doi.org/10.1080/13614576.2019.1611468)

[6] kaspersky securitylist.com *DDoS Report - DDoS attacks in Q4 2020* , 2020. (https://securelist.com/ddos-attacks-in-q4-2020/100650/)

[7] kaspersky securitylist.com *DDoS Report - DDoS attacks in Q3 2020* , 2020. (https://securelist.com/ddos-attacks-in-q4-2020/100650/)

[8] slide mirai

[9] Michael A. Nielsen *Neural Networks and Deep Learning* , Determination Press, 2015. (http://neuralnetworksanddeeplearning.com/)

[10] https://www.researchgate.net/publication/332078049_Data_Analytics_Methods_for_Anomaly_Detection_Evolution_and_Recommendations

[11] https://www-sciencedirect-com.ezproxy.biblio.polito.it/science/article/pii/S1084804515002891#f0015

[12] Keras documentation

[13] Mohiuddin Ahmed, Abdun Naser and Mahmood Jiankun-Hu, *A survey of network anomaly detection techniques* , 2016. (https://doi.org/10.1016/j.jnca.2015.11.016)

[14] Raghavendra Chalapathy and Sanjay Chawla, *Deep Learning for Anomaly Detection: A Survey* , 2019. (https://arxiv.org/abs/1901.03407v2)

[15] Varun Chandola, Arindam Banerjee and Vipin Kumar, *Anomaly detection: A survey*, 2009. (https://dl.acm.org/doi/abs/10.1145/1541880.1541882)
[16] https://docs.cilium.io/en/stable/bpf/
[17] https://docs.cilium.io/en/stable/concepts/ebpf/intro/
[18] slide risso
[19] https://ebpf.io/what-is-ebpf
[20] https://www.imperva.com/blog/ntp-flood-explained/
[21] https://www.imperva.com/docs/DS_Incapsula_The_Top_10_DDoS_Attack_Trends_ebook.pdf
[22] https://www.imperva.com/resources/resource-library/reports/global-ddos-threat-landscape/
[23] https://keras.io/examples/timeseries/timeseries_anomaly_detection/
[24] https://arxiv.org/abs/1802.09089
[25] https://ieeexplore.ieee.org/abstract/document/8363930
[26] https://ieeexplore.ieee.org/abstract/document/8802833