POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Magistrale

Sistemi anti-DDoS distribuiti

Relatore
prof. Guido Marchetto

Studente
Stefano Loscalzo
matricola: s267614

Supervisore aziendale
Centro Ricerche FIAT
dott. ing. Giovanni Giacosa

Anno accademico 2020-2021

Abstract

This short abstract, is typeset with the `abstract` environment (from the `report` document class) just to test if it works. or what concerns working, it works, but in Italian ist title turns out to be “Sommario” in bold face series and normal size; its apperance looks like a bad copy of what one obtains with the `\summary` command. In English, though, its title is “Abstract”, as it should be, since at the beginning of this template a suitable `\ExtendCapions` command was issued.

Please, read the documentation in Italian (file `toptesi-it.pdf` in order to fully understand the difference beteesn “abstract” and “summary” in the context of this bundle.

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. 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 rete neurale allenata su dati provenienti dai router di più sedi aziendali e una successiva mitigazione degli attacchi tramite un agent sugli stessi.

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 applicativi che richiedono molta potenza di calcolo o di banda ai router che compongono la rete. Punteremo a scoprire le anomalie nei dati di rete di singoli flussi e dati aggregati analizzando i dati 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 raccogliere informazioni sull'utilizzo della rete può portare anche alla risoluzioni di problemi non derivanti da attacchi.

1.2 Scenario

Per fare questa tesi 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 (todo: magari spiegare i motivi), per questo motivo il nostro obiettivo è quello di proteggere i servizi aziendali e la rete centrale dai pc connessi alle reti degli uffici. L'organizzazione di Tiesse e di molti suoi clienti è caratterizzata da questo scenario, per questo motivo le prove da me effettuate si basano sull'analisi del traffico proveniente dall'ufficio di Torino e mirano a proteggere i servizi aziendali presenti nella sede centrale di Ivrea a cui l'ufficio di Torino 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.

1.3 Gli attacchi DDoS

Gli attacchi di Denial of Service (DoS) sono degli 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).

1.3.1 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 ana falla a livello applicativo. La seconda categoria invece si basa su tecniche per colpire l'infrastruttura del servizio, per il funzionamento di questa tecnica vengono usati uno o entrambi i seguenti metodi: uno punta 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].

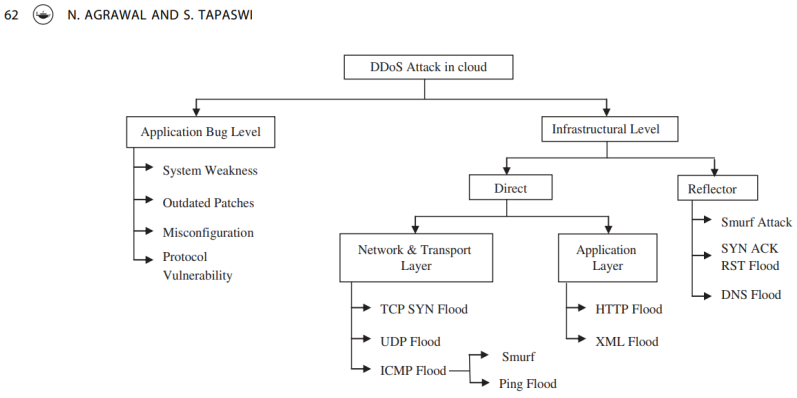

Figura 1.1. Tipologie di attacchi DDoS [4]

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

Attacchi basati sul flooding

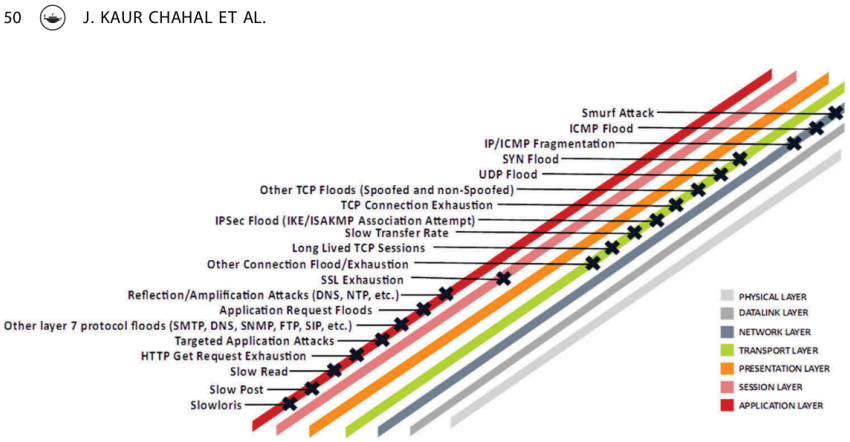

Figure 7. Distributed Denial of Service (DDoS) attack types across network layers.

Figura 1.2. Attacchi per livello [5]

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 che consumino totalmente la banda della vittima, queste tipologia di attacco può essere effettuata in maniera diretta: *flooding attacks* e *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 che non vengono effettuati in maniera diretta invece sfruttano la riflessione o l'amplificazione: nella *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 questo attacco 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 con ip spoofing, specificando come ip sorgente l'ip della vittima, 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 più di 20 volte il pacchetto in arrivo e sfruttando l'ip spoofing la risposta viene manda alla vittima [1].

Application-level DDoS flooding attacks Gli attacchi DDoS al livello applicativo hanno lo scopo di terminare le risorse del server(sockets, cpu, memory, disk/db bandwidth, I/O bandwidth) e di solito usano meno banda, rispetto gli attacchi di alla rete, per questo motivo è anche più difficile identificarli. Le tecniche utilizzate sono simili alle precedenti. Degli esempi sono l'HTTP flooding che grazie con molte richieste, obbliga il server a produrre risposte che possono essere computazionalmente pesanti, oppure 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 che mirano a mantenere la connessione aperta 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 rate variabili [4, 5].

1.3.2 Vittime attacchi DDoS

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 può cambiare anche la portata dell'attacco. Per semplicità possiamo dividere 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.

1.3.3 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 guidate dal *Syn Flood* che copre più del 90% della totalità degli attacchi, seguito da *ICMP flooding* e *UDP flooding* [6] [7].

Attacchi DDoS famosi

Prova prova

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] I bot possono essere controllati dell'artefice dell'attacco tramite tre architetture:

- 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.

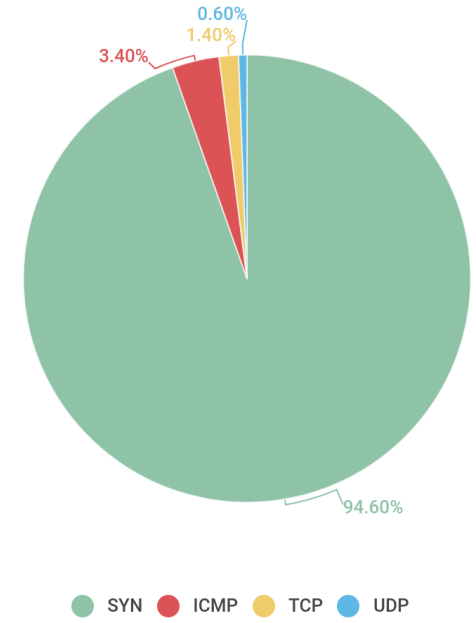

Figura 1.3. Distribuzione di attacchi DDoS per tipologia, Q3 2020 [7]

- P2P based: [5] pagina 46

1.4 Organizzazione della tesi

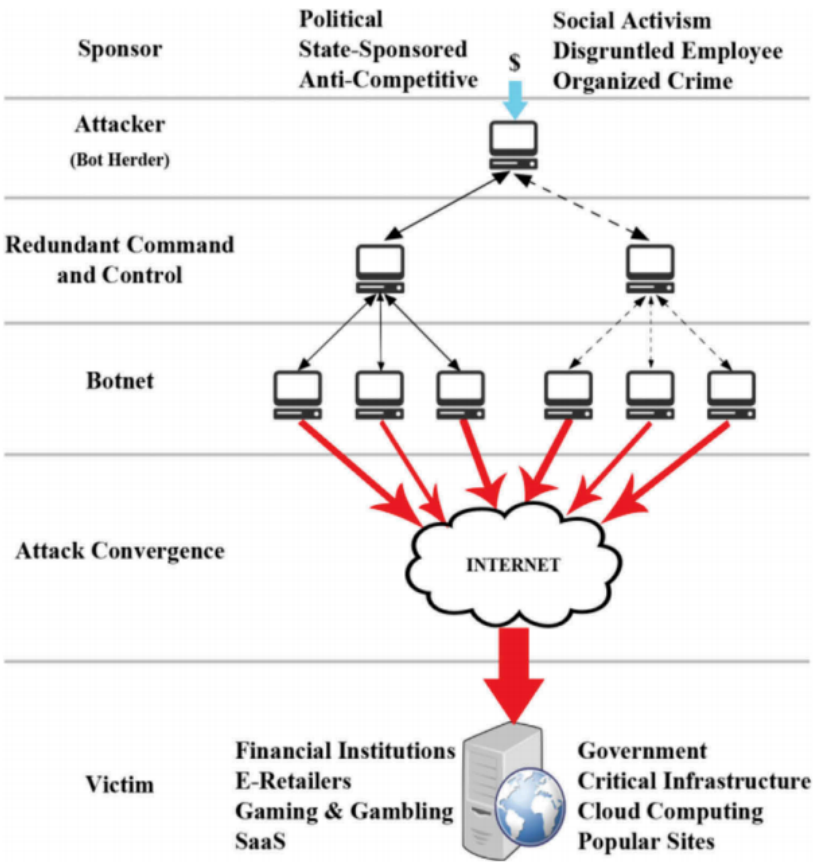

Figura 1.4. Struttura di lancio di attacchi DDoS [5]

Capitolo 2

Stato dell'arte dei sistemi anti-DDoS

2.1 Riconoscimento DDoS

La fase di riconoscimento degli attacchi DDoS è un importante passo per la mitigazione degli attacchi, questa fase diventa più facile maggiormente ci avviciniamo alla vittima dell'attacco, ma 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), sulla correlazione di informazioni di management sul traffico, o sull'analisi spettrale.

2.1.2 Anomaly-based detection

I meccanismi di rilevamento delle anomalie possono riconoscere attacchi anche su attacchi non conosciuti, basandosi su soglie per differenziare il traffico normale e malevolo, ma la scelta di esse 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 ramificano 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.

2.2.1 Soluzioni alla sorgente

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

Degli esempi di soluzioni sono:

- Filtri in ingresso e uscita agli edge router delle sorgenti:
- D-WARD:
- MULTOPS:
- MANAnet's Reverse Firewall:

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, l'utilizzo sistemi di tracciamento implementati in alcuni router (in caso di ip spoofing), 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 Tolleranza

Capitolo 3

Stato dell'arte dell'Anomaly Detecion

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 [12]. 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 [12]:

- 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 per esempio 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 Metodo di detection

- Classification based
- Statistical anomaly detection
- Information theory
- Clustering-based

3.3.2 Dati in input

pagina 6 [12]

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 [11, 12, 10].

3.3.4 Modalità di apprendimento

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. Le tipologie di reti c

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'over fitting.

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.

3.4 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

- 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
- Image Processing
- Anomaly Detection in Text Data
- Sensor Networks
- 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

Analizzando

3.5 Le reti neurali

Spiego cosa sono le reti neurali, come funzionano e come vengono allenate.

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.

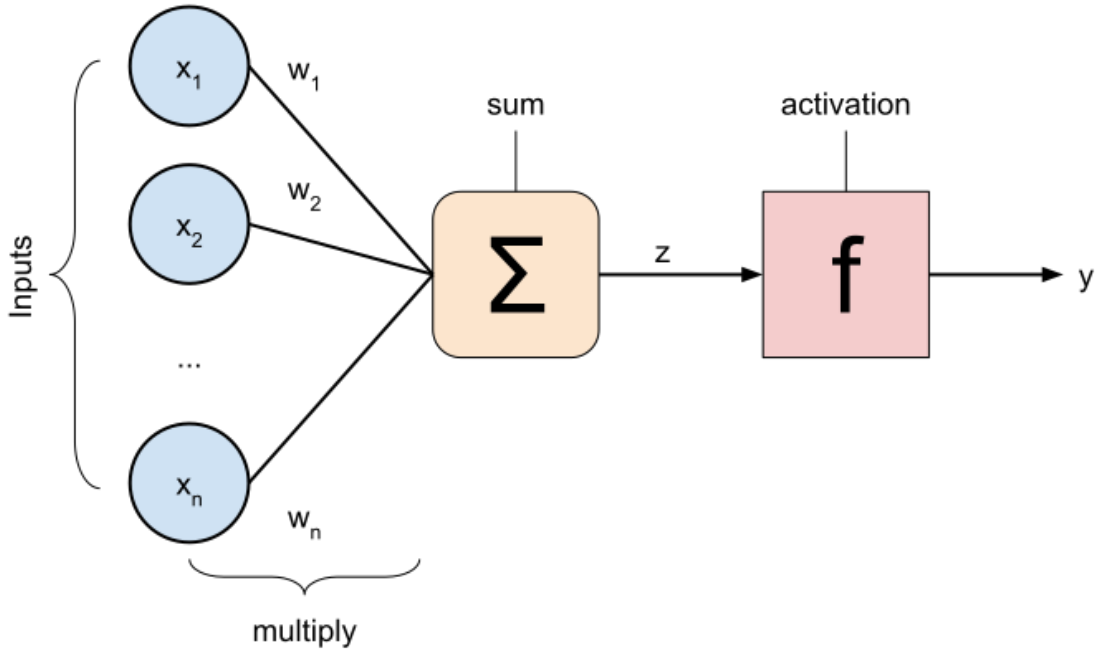

Figura 3.1. Neurone artificiale

Come si può notare dalla figura con l'evoluzione delle reti neurali non viene più solo usata la funzione a gradino, ma anche delle generiche rappresentate genericamente 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 [8].

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

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.

3.5.2 Autoencoders

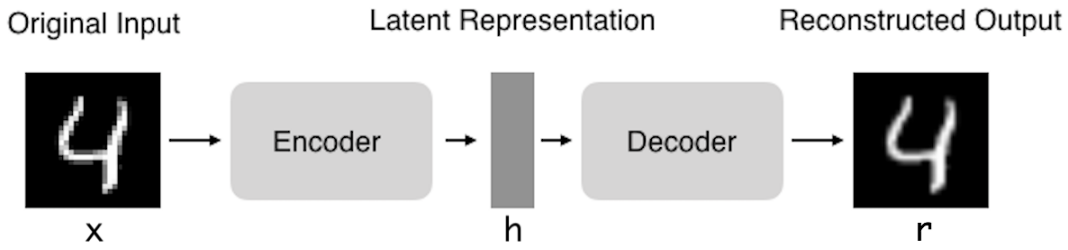

Figura 3.2. Struttura di un autoencoder

Cosa sono gli autoencoder e perchè sono utili nell'anomaly detection.

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 3.2. La capacità di comprire 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 [11].

Gli autoencoders sono stati usati da questo, quello e quell'altro ancora per sviluppare questi sistemi di anomaly detection:

- articolo 1
- articolo 2
- articolo 3

3.6 Soluzioni esistenti di Anomaly Detection

Qua illustro delle soluzioni esistenti prese dagli articoli.

3.7 Motivazione

Prova

Capitolo 4

Soluzione proposta - Titolo provvisorio

4.1 Introduzione

Il mio obiettivo è di creare un sistema di anomaly detection con struttura distribuita: nei router delle sedi periferiche raccolgo informazioni sul traffico e in un server posizionato fisicamente nella sede centrale lo analizzo e prendo decisioni sulle azioni da compiere. Per svolgere questo lavoro mi sono dovuto basare su alcuni software e sull'hardware aziendale con i relativi vantaggi e svantaggi rispetto ad una soluzione completamente ex-novo.

4.2 Obiettivo

4.2.1 Gestione dei dati

I dati vengono collezionati tramite un demone sui router (collectd), inviati al server (go-graphite) e visualizzati tramite una dashboard(graphana). Il tool di anomaly detection da me pensato 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.

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 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 da me 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

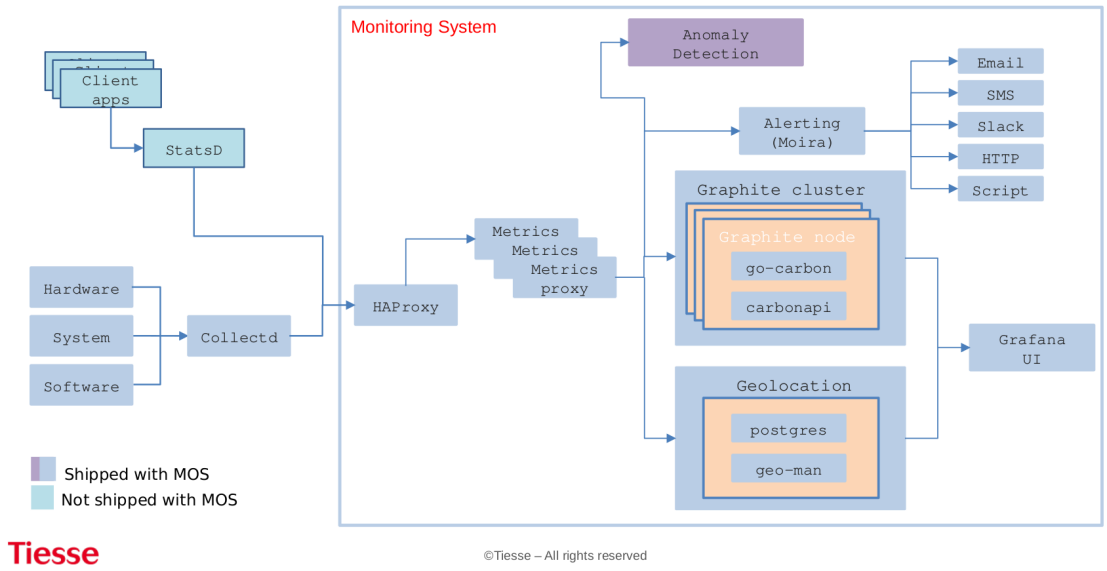

Figura 4.1. Architettura di MOS (Monitor System di Tiesse)

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.

Il mio plugin di collect

NDPI è un software per il deep-packet inspection basato su OpenDPI

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 4.2.1):

- 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.

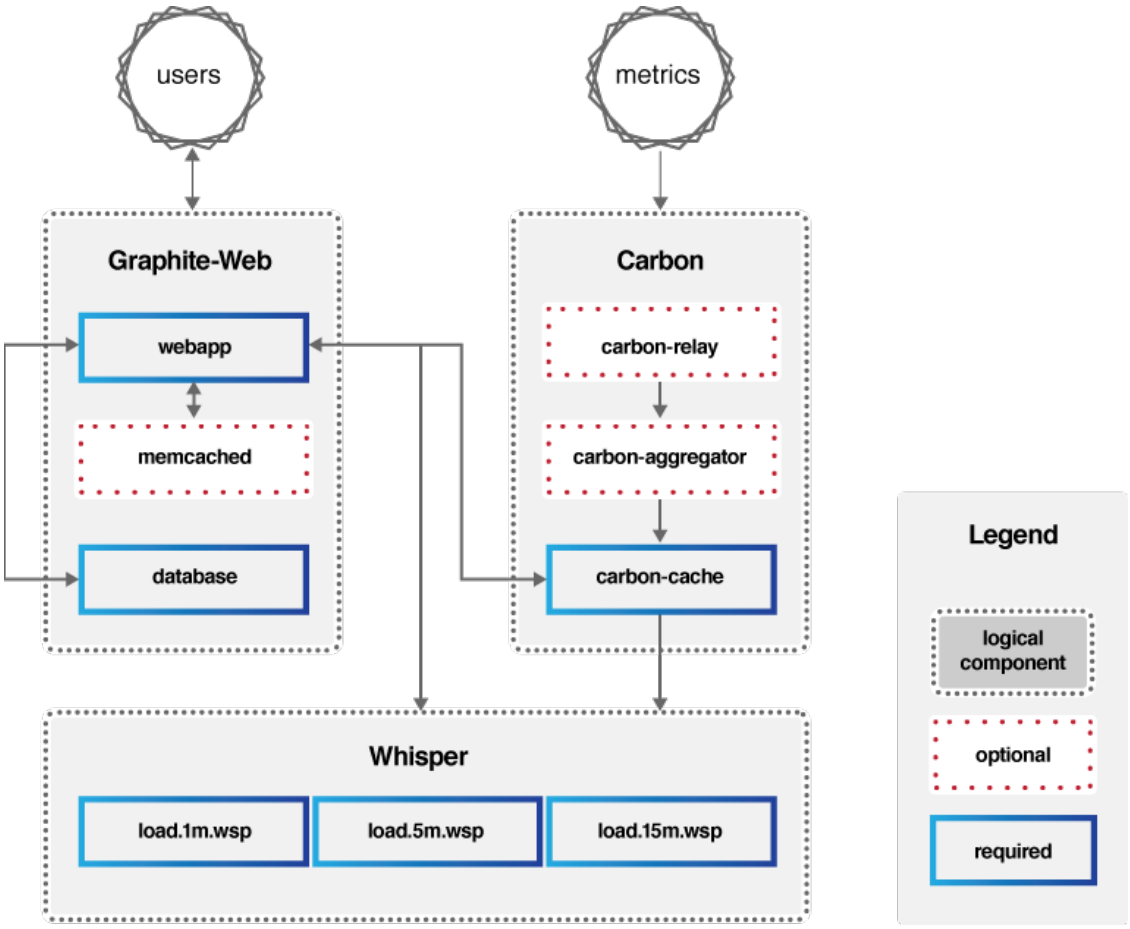

Figura 4.2. Architettura di Graphite

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) 4.2.1. Le carbonapi invece sono un subset delle api di graphite-web e le vanno a sostituire essendo dalle 5 alle 10 volte più veloci.

Grafana è un software open source che permette la visualizzazione e la generazione delle metriche tramite una web application. Permette di creare dashboard dinamiche interrogando le api di graphite-web. Nel mio caso ho organizzato una

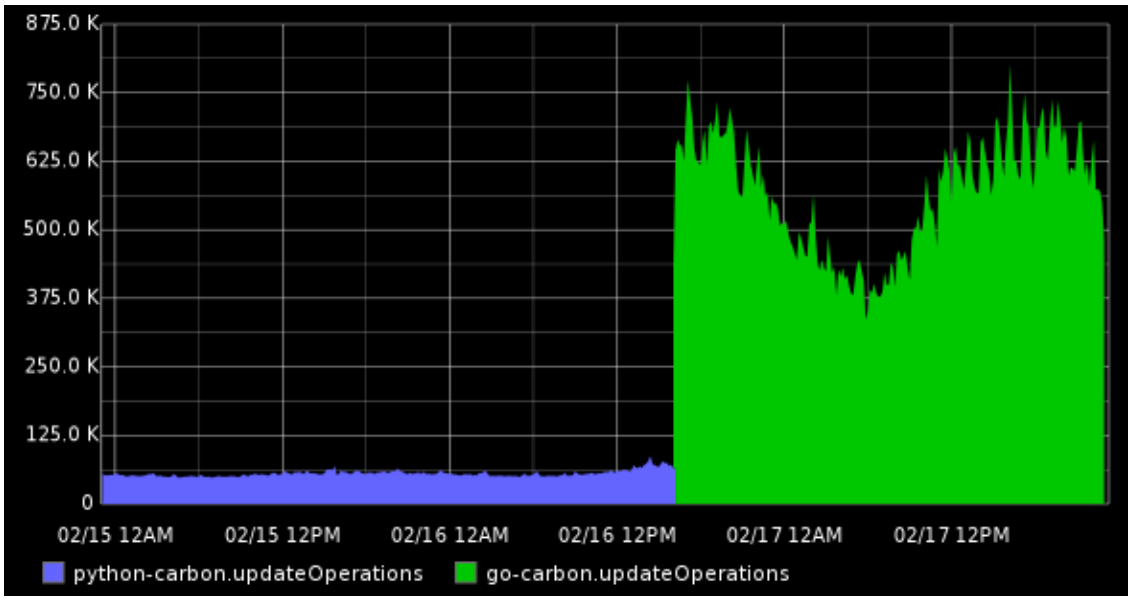

Figura 4.3. Differenza di prestazioni tra carbon e go-carbon

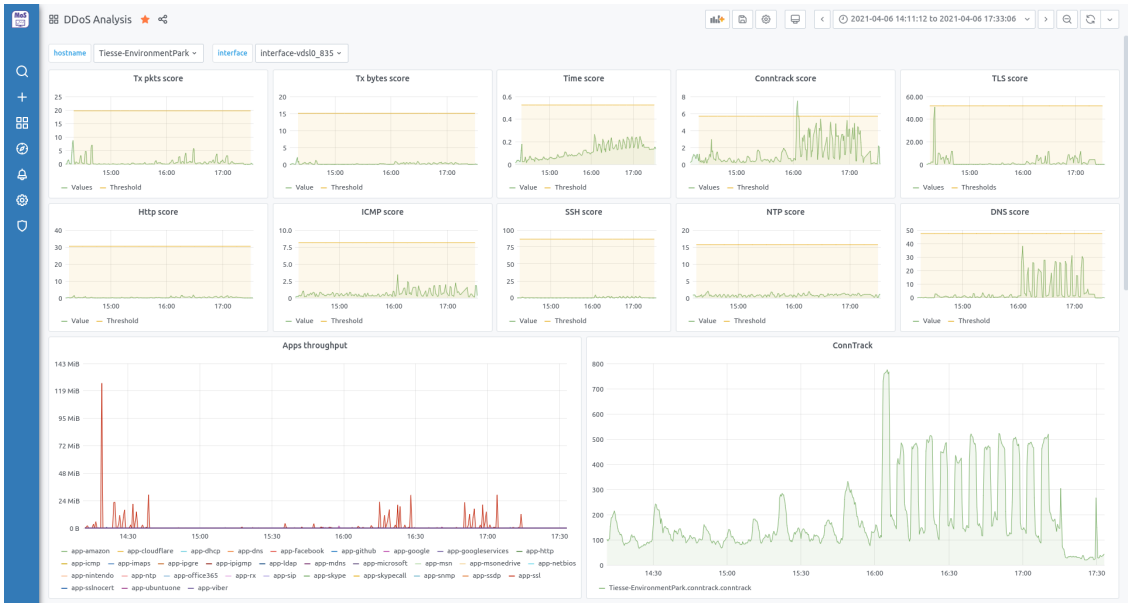

Figura 4.4. La mia dashboard su Grafana

dashboard in modo da visualizzare sia i dati provenienti dai router da analizzate, sia gli anomaly score calcolati dal software di anomaly-detection 4.2.1.

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 4.2.1.

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.
- *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 mio tool

Cosa fa? Perchè l'ho fatto?

Per rilevare le anomalie, una volta decise le features da analizzare, ho dovuto decidere quale sistema dei precedentemente elencati adottare. La scelta è ricaduta sull'utilizzo di una rete neurale di autoencoder, la quale permette di allenare facilmente il modello in modo semi-supervisionato, grazie all'utilizzo della sola classe normale di traffico.

4.4.1 Struttura

Posso parlare di come è strutturato il programma, che librerie ho usato 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

Come ricevo i dati dal server? Come li elaboro? Che api uso? Genero tante piccole sequenze di 8 elementi (spiegare il motivo della scelta della sequence length => trade-off tra tempo di rilevamento dell'anomalia e mitigazione dei picchi e dei falsi positivi) e perchè lo faccio, come normalizzo i dati e perchè. Come fornisco i dati in input alla rete.

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 che mi restituisce la prima richiesta è (qui per ridurre la quantità di dati ipotizzo che la richiesta sia di 20 secondi):

```
[
    {
        "target": "Tiesse-EnvironmentPark.conntrack.conntrack",
        "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"
        }
    }
]
```


Per elaborare i dati ricevuti devo avere una tabella con il corretto formato, vedi l'esempio di tabella 4.1, per questo motivo elaboro i dati del json ricevuto precedentemente, sostituendo per prima cosa il timestamp, trasformandolo da un unix

timestamp ad uno che indica i secondi passati dalla mezzanotte e successivamente aggiungo tutti gli elementi ad un json con il corretto formato, come mostrato nell'estratto di codice 4.2.

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 4.1. 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' un'eccezione,
    # che verra' 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)
            for e in tl:
                line[e] = data[e][i][0]
            data_list.append(line)

    # creo la matrice con pandas ed elimino le righe con valori
    # mancanti in rari casi le carbon api mi restituiscono dei
    # valori nulli, per questo motivo cancello le righe in cui
    # c'e' almeno un valore nullo
    matrix = pd.DataFrame(data_list).dropna()

    # successivamente richiedo anche i dati delle app, e quando
    # li ricevo, dopo averli manipolati, li unisco tutti
    # in un'unica tabella
    app_data_matrix =
    get_app_data(s_time, e_time, hostname, username, password)

    matrix = pd.merge(app_data_matrix, matrix)

    # questa funzione mi permette di ritornare l'elenco
    # delle feature attualmente utilizzate, senza impostare
    # host e interfaccia da utilizzare
    matrix.columns = total_target_list('', '')
    return matrix
```

Una volta ricevuti i dati e organizzati nel formato corretto, effettuo una nuova richiesta alle carbon api, per ricevere i dati relativi alle applicazioni, il procedimento è simile a quello precedentemente descritto, l'unica differenza è che estendo i dati, mostrando lo stesso dato più volte, in modo da avere lo stesso numero di elementi ottenuti precedentemente.

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

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. Abbiamo scelto una lunghezza della sequenza di otto elementi

4.4.4 Modello della rete

Spiego come ho usato Keras per generarlo. Spiego come sono arrivato a scegliere il modello e comparo i risultati tra 2/3 modelli totalmente diversi come numero di nodi, sarebbe interessante provare a mettere anche un modello con retroazione visto che stiamo usando sequenze temporali.

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 nostra rete è composta da un semplice autoencoder a 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 ho usato progressivamente un layer con 25 nodi, uno spazio latente con 8 e un livello successivo di nuovo con 25 nodi. La scelta delle dimensioni dei livelli centrali è stata presa analizzando più configurazioni e considerando che usando una lunghezza della sequenza di otto elementi, con dieci features significa avere 80 nodi in input e output, di conseguenza i nodi nei livelli centrali devono essere meno. 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.

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

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

Listing 4.2. 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.

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

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

Quando eseguire il train?

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. 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 Parlare come funzionano i tool che ho fatto

4.5.2 Risultati

Capitolo 5

Mitigazione degli attacchi

5.1 Introduzione

Prova

5.1.1 Bloccare l'ip spoofing

L'ip spoofing permettere di usare la tecnica dell'amplification

5.2 Funzionamento

Prova prova

5.2.1 eBPF

5.2.2 BCC

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?

5.3 Test sulle anomalie

5.3.1 Tool utilizzati

5.3.2 Risultati

Capitolo 6

Lavoro futuro

Capitolo 7

Conclusioni

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

[9] Keras documentation

[10] 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)

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

[12] Varun Chandola, Arindam Banerjee and Vipin Kumar, *Anomaly detection: A survey*, 2009. (https://dl.acm.org/doi/abs/10.1145/1541880.1541882)