Project 4: Feature Extractor con Zeek

Gabriele Tassinari

18 December 2023

Contents

1 Introduction

L'analisi del traffico di rete e la rilevazione di attività sospette sono aspetti critici per garantire la sicurezza delle reti informatiche. In questo contesto, strumenti avanzati e metodologie sofisticate sono essenziali per individuare e mitigare potenziali minacce. All'interno del progetto, ci concentreremo sull'impiego di Zeek, un framework open-source ampiamente utilizzato per l'analisi del traffico di rete, insieme a tecniche di machine learning, per affrontare sfide legate alla sicurezza informatica. Tale approccio consente di ottenere una visione dettagliata e significativa degli elementi chiave del traffico di rete, aprendo la strada a una valutazione più accurata della sicurezza unendo NIDS con Reti Neurali.

Il progetto prende spunto dall'articolo 5"UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)", che si focalizza sul testing e sul confronto di due dataset ampiamente utilizzati per la valutazione dei Network Intrusion Detection Systems (NIDS), ossia KDDCUPP99 e NSLKDD. L'analisi delle features di questi dataset rivela importanti limitazioni, in quanto KDDCUPP99 presenta dati ridondanti che compromettono la precisione della detection, oltre a presentare lacune nella gestione di alcuni dati. Allo stesso modo, NSLKDD, sebbene presenti un miglioramento rispetto a KDDCUPP99, mostra uno sbilanciamento significativo tra i dati relativi al traffico legittimo ed illegittimo. Ciò rende i due Dataset inadatti come rappresentazione realistica di un moderno realistico traffico di rete. L'articolo presenta così un nuovo Dataset, ovvero UNSW-NB15, che supera le limitazioni precedentemente espresse.

Così come è stato fatto per il Dataset UNSW-NB15, questo paper si pone come obiettivo quello di presentare il nuovo Dataset Dataset CIC Modbus 2023 analizzandolo ed estraendo features che permettano di classificare questo dataset come valido o meno per la valutazione di NIDS. In linea con l'approccio delineato nell'articolo 5 precedentemente citato, saranno estratte diverse categorie di feature, tra cui Flow Features, Basic Features, Content Features, Time Features, e Labelled Features. Questo insieme completo di feature fornisce una base solida per l'analisi dettagliata del traffico di rete.

Le regole di estrazione delle feature, implementate nel linguaggio di scripting Bro-code, rappresentano il Core-Business del nostro progetto, che verrà poi integrato e implementato con due estensioni. Verrà infatti implementato realizzando una rete neurale di classificazione che sarà in grado di avere in input un file pcap e di restituire una valutazione immediata sulla natura del traffico, identificandolo come malevolo o legittimo.

Inoltre, verrà utilizzata l'estrazione di features con Zeek per verificare che si possano rilevare attacchi di VLAN-Hopping di tipo Double Tagging. Questo attacco estremamente comune nelle realtà aziendali si basa su una configurazione di Default degli switch Cisco che permette ad un attaccante di "saltare" dalla VLAN di appartenenza, ad una VLAN a cui non dovrebbe avere accesso.

Il paper mira dunque a fornire una macro-visione del funzionamento dei Network Intrusion Destection System, a mostrare come sia possibile personalizzarli e ad integrarli realizzando una rete neurale che aiuti i NIDS per aumentarne l'accuratezza.

2 Background Tecnologico

2.1 Zeek

Zeek, precedentemente noto come Bro, è un potente framework open-source progettato per l'analisi del traffico di rete. Utilizzato ampiamente nelle comunità di sicurezza informatica e nella ricerca, Zeek si distingue per la sua flessibilità, la capacità di adattarsi a una vasta gamma di scenari e la fornitura di dettagliate informazioni sull'attività di rete. Operando così come un sensore di rete che ispeziona e analizza il traffico in transito, fornendo una panoramica approfondita delle attività di rete, inclusi dettagli relativi a connessioni, protocolli applicativi, anomalie e potenziali minacce. Il suo successo è attribuibile ala sua architettura modulare ed al suo linguaggio di scripting, che consente agli utenti di personalizzare e ampliare le funzionalità in base alle proprie esigenze specifiche. Zeek è particolarmente efficace nel rilevare attività sospette o potenzialmente dannose, fornendo agli amministratori di rete e agli esperti di sicurezza uno strumento avanzato per monitorare, analizzare e rispondere agli eventi di rete.
Zeek offre una serie di caratteristiche fondamentali che lo rendono un potente strumento per l'analisi di rete. Una delle sue peculiarità è la capacità di decomporre il traffico di rete in unità semantiche chiamate "eventi". Gli eventi di Zeek rappresentano diverse attività osservate nella rete, dalla creazione di connessioni

alla trasmissione di dati. Questo approccio event-driven consente agli utenti di concentrarsi su aspetti specifici del traffico di rete che sono rilevanti per le loro esigenze di analisi. Inoltre, Zeek supporta numerosi protocolli di rete, offrendo un'ampia copertura di analisi che spazia dai protocolli di trasporto comuni a quelli più specifici delle applicazioni. La modularità di Zeek consente agli utenti di integrare nuovi script per analizzare e interpretare nuovi protocolli, adattando facilmente il framework alle mutevoli esigenze del panorama di rete.

2.2 Reti Neurali

Il Machine Learning (ML) rappresenta un approccio computazionale all'apprendimento automatico, permettendo alle intelligenze artificiali di imparare complessi procedimenti senza richiedere una specifica codifica umana. Nei contesti di ML, il Supervised Learning coinvolge l'uso di dati etichettati per addestrare algoritmi, mentre l'Unsupervised Learning si basa su dati non etichettati per individuare pattern senza una risposta di riferimento. Il Semi-Supervised Learning è una combinazione di entrambi, mentre il Reinforcement Learning permette all'algoritmo di apprendere tramite interazioni dinamiche con un ambiente, ricevendo "premi" o "punizioni" in base alle prestazioni.

Nel campo del Deep Neural Network Learning, le intelligenze artificiali utilizzano dataset suddivisi in training, validation e testing set per addestrare reti neurali. Concetti chiave includono Epoche, rappresentanti il numero di presentazioni del dataset alla rete, Batch Size, che indica quanti esempi di addestramento vengono utilizzati in un'iterazione, e l'Overfitting, un problema in cui il modello memorizza troppo i dati di addestramento. Le Reti Neurali Convoluzionali (CNN) sono una categoria specifica di reti neurali adatte per l'analisi di dati grid-based, come immagini, e operano attraverso due fasi principali: Forward propagation e Backward propagation.

Inoltre, le Generative Adversarial Networks (GAN) sono un tipo speciale di rete neurale utilizzate per generare nuovi dati sintetici, mentre le reti neurali di classificazione svolgono un ruolo cruciale nell'assegnare etichette o categorie a input specifici, contribuendo a risolvere problemi di classificazione in vari domini applicativi. La diversità di approcci e concetti nel vasto campo delle reti neurali offre strumenti potenti per l'apprendimento automatico e l'analisi avanzata dei dati.

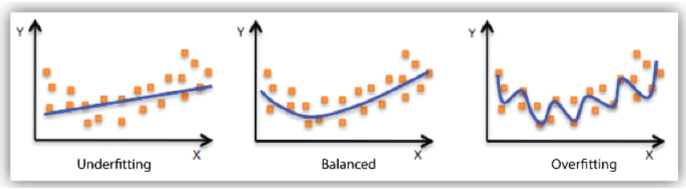

Figure 1: Addestramento AI.

2.3 Dataset CIC Modbus 2023

Il Dataset che si è scelto di utilizzare per l'estrazione delle feature e per allenare la rete neurale è il dataset "CIC Modbus 2023". Questo dataset risulta estremamente variegato e contiene ogni tipologia di traffico di rete.Questo dataset è stato generato attraverso la cattura di file pcap che simulano il traffico di rete, sia legittimo che malevolo, all'interno di una rete fittizia. La struttura del dataset riflette la suddivisione tra il "Dataset Attack" e il "Dataset Benign". La creazione dei file pcap del dataset è stato realizzato con l'ausilio di Wireshark e Docker e prevede la seguente struttura 2. All'interno dell'architettura simulata, gli Secure IED, identificati come IED1A (185.175.0.4) e IED4C (185.175.0.8), sono considerati sicuri. Analogamente, il Secure SCADA HMI (185.175.0.3) è incluso tra i dispositivi considerati sicuri. Tuttavia, sono presenti anche un IED non sicuro (185.175.0.5) e un SCADA HMI non sicuro (185.175.0.3), rendendoli potenzialmente vulnerabili.

Un elemento chiave dell'architettura è rappresentato dal Central Agent (185.175.0.6), il quale riceve i cosiddetti "Detection score" dagli agent presenti in ogni dispositivo sicuro. Contestualmente, è presente un Attacker (185.175.0.7), responsabile degli attacchi visibili nel "Dataset Attack".

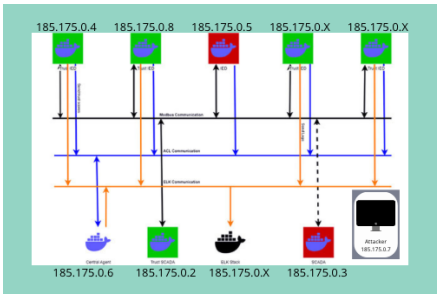

Figure 2: Architettura del Dataset.

La comprensione della struttura del dataset permette ora di comprendere meglio le statische e i log ottenuti.

3 Progetto

3.1 Estrazione Feature

Il progetto mira ad insegnare come viene effettuata l'estrazione di Features con Zeek utilizzando dunque il linguaggio "Bro-Code" per analizzare il traffico di rete per estrarre features ed in seguito processarle per qualunque esigenza. Per fare ciò è stato utilizzzato il Dataset "CIC Modbus 2023" il quale è stato dato in input alle regole Zeek scritte per l'estrazione delle stesse features presenti nell'articolo di riferimento 5. Le features estratte, complessivamente 37, sono estremamente variegate, permettendo di offrire una panoramica dettagliata della connessione sotto analisi. Esse sono suddivise in diverse categorie, ciascuna focalizzata su aspetti specifici della comunicazione:

- **Flow Feature:** Queste features sono progettate per identificare in modo univoco una connessione. La Figura 3 mostra un esempio grafico di una di queste features.

#	Name	T.	Description
1	*srcip*	N	Source IP address
2	*sport*	I	Source port number
3	*dstip*	N	Destination IP address
4	*dsport*	I	Destination port number
5	*proto*	N	Transaction protocol

Figure 3: Flow Feature.

- **Basic Feature:** Queste features forniscono informazioni di carattere generale sulla connessione, contribuendo a costruire una comprensione di base. La Figura 4 offre una visualizzazione rappresentativa di una Basic Feature.

#	Name	T	Description
6	*state*	N	The state and its dependent protocol, e.g. ACC, CLO, else (-)
7	*dur*	F	Record total duration
8	*sbytes*	I	Source to destination bytes
9	*dbytes*	I	Destination to source bytes
10	*sttl*	I	Source to destination time to live
11	*dttl*	I	Destination to source time to live
12	*sloss*	I	Source packets retransmitted or dropped
13	*dloss*	I	Destination packets retransmitted or dropped
14	*service*	N	http, ftp, ssh, dns ...,else (-)
15	*sload*	F	Source bits per second
16	*dload*	F	Destination bits per second
17	*spkts*	I	Source to destination packet count
18	*dpkts*	I	Destination to source packet count

Figure 4: Basic Feature.

- **Content Feature:** Queste features analizzano il contenuto dei pacchetti, tipicamente di tipo TCP, contribuendo a identificare caratteristiche specifiche della comunicazione. La Figura 5 presenta un esempio di Content Feature.

#	Name	T	Description
19	*swin*	I	Source TCP window advertisement
20	*dwin*	I	Destination TCP window advertisement
21	*stcpb*	I	Source TCP sequence number
22	*dtcpb*	I	Destination TCP sequence number
23	*smeansz*	I	Mean of the flow packet size transmitted by the src
24	*dmeansz*	I	Mean of the flow packet size transmitted by the dst
25	*trans_depth*	I	the depth into the connection of http request/response transaction
26	*res_bdy_len*	I	The content size of the data transferred from the server's http service.

Figure 5: Content Feature.

- **Time Features:** Queste features si concentrano sulle tempistiche relative ai pacchetti all'interno di una connessione. Un esempio grafico è mostrato nella Figura 6.

#	Name	T	Description
27	*sjit*	F	Source jitter (mSec)
28	*djit*	F	Destination jitter (mSec)
29	*stime*	T	record start time
30	*ltime*	T	record last time
31	*sintpkt*	F	Source inter-packet arrival time (mSec)
32	*dintpkt*	F	Destination inter-packet arrival time (mSec)
33	*tcprtt*	F	The sum of 'synack' and 'ackdat' of the TCP.
34	*synack*	F	The time between the SYN and the SYN_ACK packets of the TCP.
35	*ackdat*	F	The time between the SYN_ACK and the ACK packets of the TCP.

Figure 6: Time Feature.

- **Attack Feature:** Queste features consentono di identificare diverse tipologie di attacchi, contribuendo a migliorare la robustezza del sistema. Un esempio di Attack Feature è illustrato nella Figura 7.

#	Name	T	Description
48	*attack_cat*	N	The name of each attack category. In this data set, nine categories (e.g., Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms)
49	*Label*	B	0 for normal and 1 for attack records
***Type (T.) N:** nominal, **I:** integer, **F:** float, **T:** timestamp* and ***B:** binary*			

Figure 7: Attack Feature.

Il codice completo per l'estrazione di queste feature è disponibile QUI. La repository Github presenta script per l'estrazione di tutte le features sopra citate, istruzioni per l'utilizzo, automazioni BASH per l'estrazione automatica delle feature partendo dal dataset fornito e codice python per la creazione della rete neurale che analizzeremo nella prossima sezione.

3.1.1 Statistiche Dataset

Sottoponendo dunque l'intero dataset "CIC Modbus 2023" alle regole Zeek create, è stato possibile dunque estrarre le suddette features che hanno permesso di mettere a confronto il dataset Modbus con il dataset di partenza, "Unsw-nb15", ovvero quello dell'articolo 5. Il processo è stato da me automatizzato tramite script Bash che mi hanno permesso di estrarre tutte le feature in maniera semplice e veloce. Gli output ottenuti sono stati dunque messi a confronto ottenendo il seguente risultato:

Statistical Features	Unsw-nb15	Bening	Attack
N° of flows	987,627	4,559,770	9,055,995
Src_bytes	4,860,168,866	1,478,356,764	3,179,361,798
Dst_bytes	44,743,560,943	1,128,175,330	2,368,174,501
Src_pkts	41,168,425	26,006,278	49,821,269
Dst_pkts	53,402,915	20,094,581	38,436,078
TCP	771,448	4,550,004	9,042,823
UDP	301,528	8,589	11,964
ICMP	150	1,177	1,208
Other	150	60	80

3.2 Attacchi strategie

Per quanto riguarda le Attack Feature sono stati realizzati script che permettono di rilevare alcuni tipi di attacchi. Le stategie utilizzate sono le seguenti:

Tipologia di attacco	Teoria Attacco	Strategia Detection
Shellcode	Inserimento di codice eseguibile nei dati di input per sfruttare vulnerabilità del software.	Monitoraggio dei comportamenti anomali nel traffico di rete e analisi del comportamento del sistema.
Baseline replay attack	Ripetizione di comunicazioni precedenti per ottenere accesso non autorizzato.	Rilevamento di pattern di comunicazione ripetuti e implementazione di meccanismi di autenticazione avanzati.
Brute force write	Tentativi massivi e automatizzati di scrittura per violare le autorizzazioni.	Limitazione dei tentativi, controllo degli accessi e monitoraggio delle attività di scrittura.
DDoS	Sovraccarico delle risorse di un sistema per renderlo inutilizzabile.	Filtraggio del traffico per rilevare un massivo numero di molteplici tutte verso un unico target.
Fuzzing	Inserimento di dati di input casuali per identificare vulnerabilità.	Analisi dell'applicazione in modo intensivo, validazione rigorosa degli input e monitoraggio degli errori.
Loading payloads	Caricamento di dati dannosi per sfruttare vulnerabilità e ottenere accesso.	Analisi approfondita dei dati in ingresso, scansione dei file alla ricerca di contenuti dannosi.
Modify length parameter	Modifica dei parametri di lunghezza per causare errori e sfruttare le debolezze del sistema.	Validazione accurata delle lunghezze dei dati, controllo degli errori e implementazione di controlli di sicurezza robusti.
Reconnaisance	Raccolta di informazioni sul sistema o sulla rete per pianificare attacchi successivi.	Monitoraggio delle attività di scansione, analisi dei log e implementazione di firewall avanzati.

Per ogni attacco è stato dunque necessario comprendere come funzionasse e quali fossero le features comuni in modo da riuscire ad essere scoperto tramite Zeek. Effettuato ciò sono stati dunque creati gli script necessari per la detection. L'intero codice è disponibile su Github@4utotune, dove è possibile vedere le stegie di rilevazione per ogni singolo attacco che spaziano dal riconoscimento di pattern come '/x41' per il fuzzing (sequenza di A) o '/x90' per il shellcode (NOP sled) al salvataggio delle connessione come ad esempio per il Baseline Replay Attack (oneroso computazionalmente) ed al rilevamento di input non della dimensione desiderata come nel Modify Length Parameter o al rilevamento di connessioni su porte non convezionali come per il Shellcode o Backdoor.

3.3 Estensione Rete Neurale

Compreso il processo di estrazione feature con Zeek, è stato dunque deciso di ampliare il nostro NIDS, inserendo una rete neurale in modo da aiutare il NIDS aumentandone l'accuratezza. Si è infatti pensato di utilizzare le reti neurali per individuare nuovi pattern per il rilevamento degli attacchi, pattern che magari a volte sfuggono nella creazione degli script Zeek. Svolgendo infatti l'attività progettuale con la professoressa Montanari inerentemente all'esame di Sicurezza dell'Informazione dove è stato possibile produrre il documento "Analisi della sicurezza dei modelli per il riconoscimento facciale", ho compreso che le reti neurali sono uno strumento potentissimo in grado di individuare uno schema anche quando apparentemente questo non sembra esistere; infatti studio effettuato, consisteva nel creare una Generative Adversarial Network per generare il volto di una persona partendo da un vettore di embedding, ovvero un array di 512 features che identifica i punti chiave del volto e che viene salvato sui nostri dispositivi mobili per permettere il riconoscimento facciale.
Questo progetto mi ha permesso di comprendere la potenza delle reti neurali. Ho dunque pensato di applicarla anche a questo progetto. Ciò è dovuto al fatto che per rilevare degli attacchi con zeek è necessario effettuare delle "personalizzazioni" nel quale si definisce ad esempio una soglia limite che se superata attiva il trigger che fa scattare l'alert per un attacco di DDoS o di Brute Force Write, oppure ad esempio occorre a volte inserire parole chiave per il rilevamento di Reconnaisance. Mi sono dunque chiesto se fornendo in input un file pcap ad una rete neurale e fornendo le opportune label che identificano il traffico benigno (0) o malevolo (1), riuscissi a generare una rete neurale che riuscisse in autonomia ad identificare nuovi pattern per la rilevazioni di attacchi, in modo da aiutare il lavoro dei NIDS.

3.3.1 Strategia e codice

Ho dunque creato una rete neurale che potesse prendere in input un file pcap e restituisse un numbero binario. Tuttavia ciò ha creato non poche problematiche. La prima problematica l'ho riscontrata nell'individuazione del corretto formato di input, infatti non è possibile fornire un generico file pcap ad una rete per allenarla a causa della disomonogenità dei dati. Ho dovuto dunque effettuare del preprocessing del dataset per uniformarlo e quindi riuscire ad effettuare il training della rete neurale. Durante questa fase di preprocessing è stato necessario leggere i file pcap fino ad un numero N di pacchetti ed estrarne le feature in modo da avere un input omogeneo. Questo poichè ad esempio i pacchetti arp o tcp o udp hanno una struttura differente ed una eventuale disomogeneità degli input potrebbe poi generare problemi con la normalizzazione dei dati in fase di training. Per estrarre le features non è stato utilizzato Zeek bensì, le librerie Python, Scapy e DPKT. Inizialmente ho utilizzato Scapy che però si è rivelato poco efficiente per l'estrazione di features poichè non supporta il multithreading e dunque ad esaminare il vasto dataset a disposizione impiegava circa 12/13 ore. Ho dunque deciso di utilizzare dpkt, più efficiente e veloce; grazie a questa libreria ho estratte le seguenti features, che ho ritenuto più rilevanti per l'analisi del traffico di rete:

```
file_features.append([
            protocol, length,
            src_port, dst_port, flags,
            ip_version, ttl, identification, fragment_offset,
            seq_number, ack_number
        ])
```

Ho voluto opportunamente non estrarre il src_ip e il dst_ip poichè sono estramemtne conosciuti casi di overfitting, ovvero casi in cui una rete neurale ha imparato in maniera errata individuando quindi pattern su aspetti non rilevanti per il problema. In particolare provenendo il mio dataset dall'architettura fornita sopra 2, non si voleva correre il rischio che ad esempio siccome abbiamo un solo IED non sicuro (185.175.0.7), dal quale provengono tutti gli attacchi, non volevo che la rete imparasse che se vede src_ip = 185.175.0.7 automaticamente classificasse il traffico come malevolo.

Una volta uniformato il dataset, mi sono occupato di creare la mia rete neurale, nello specifico ho creato la rete neurale come segue:

```
model = Sequential()
model.add(LSTM(128, input_shape=(max_sequence_length, features.shape[2])))
```

```
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))
```

La struttura della rete neurale è un modello sequenziale composto da strati LSTM (Long Short-Term Memory) e strati densi, una architettura comunemente utilizzata per la classificazione binaria (1 o 0). Nel dettaglio sono stati utilizzati i seguenti layer:

- **LSTM**: Questo è uno strato LSTM con 128 unità. LSTM è un tipo di strato ricorrente utile per catturare relazioni a lungo termine nei dati sequenziali.
- **Dense(64, activation='relu')**: Si tratta di uno strato completamente connesso (densamente connesso) con 64 unità e funzione di attivazione ReLU (Rectified Linear Unit). Questo strato è progettato per apprendere pattern dai dati di output dello strato LSTM.
- **Dropout(0.5)**: Dropout è una tecnica di regolarizzazione che impedisce al modello di diventare troppo dipendente da specifici percorsi di input durante l'addestramento. Impedisce il sovradattamento e migliora la generalizzazione del modello. Qui, 0.5 indica la frazione delle unità di input da lasciare cadere durante l'addestramento.
- **Dense(1, activation='sigmoid')**: L'ultimo strato è uno strato densamente connesso con una sola unità di output e funzione di attivazione sigmoide. La funzione sigmoide è comunemente utilizzata per problemi di classificazione binaria perché restituisce un valore compreso tra 0 e 1, che può essere interpretato come la probabilità di appartenenza alla classe positiva.

Questo modello, ottenuto in maniera empirica, risulta il più accurato tra quelli creati e mi ha permesso di ottenere una accuracy di 0.8421 ovvero dell'84%.

3.4 Estensione VLAN Hopping

3.4.1 Vlan Hopping

Il termine VLAN (Virtual Local Area Network) indica un insieme di tecnologie che permettono di segmentare il dominio di broadcast che si crea in una rete locale, basata tipicamente su switch, in più reti locali logicamente non comunicanti tra loro, ma che condividono globalmente la stessa infrastruttura fisica della rete. Semplificando molto il concetto è possibile dire che, una VLAN è una rete fittizia di dispositivi che appartengono a una o più reti LAN.
Le applicazioni di questa tecnologia sono tipicamente legate ad esigenze di separare il traffico di gruppi di lavoro o dipartimenti di un'azienda, per applicare diverse politiche di sicurezza. Ad esempio queste vengono spesso utilizzate nelle grandi aziende per separare logicamente i computer dei dipendenti con ad esempio i PLC (ovvero Controllori Logici Programmabili).
Il funzionamento delle VLAN può essere riassunto come segue:

(a) Scenario senza VLAN

(b) Scenario con VLAN

Figure 8: Funzionamento delle VLAN

In un attacco di tipo Double Tagging, l'attaccante una volta ottenuto accesso alla rete e sfruttando le configurazioni di default e non attente degli switch, è in grado di comunicare con altre VLAN all'interno

della rete. Si noti che questo attacco è solamente One-way, infatti funziona in solo in andata, l'attaccante non ha dunque modo di forzare la vittima a generare una risposta in modo da riceverla. Questo attacco viene quindi spesso usato in modalità DoS, ovvero Denail of Service, una tipologia di attacco che prende di mira la disponibilità di un servizio, creando disagi dovuti all'impossibilità della macchina di erogare il servizio richiesto.

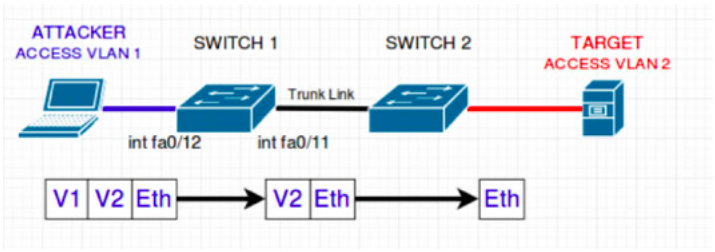

Figure 9: Double Tagging Funzionamento

Il primo switch elimina il primo tag, identificando il pacchetto come un frame uscente dalla VLAN di appartenenza. Il frame raggiunge così la VLAN, che in questo caso collega i due switch. Essendo la VLAN di appartenenza dell'attaccante e la VLAN che collega gli switch, entrambe identificate con VLAN 1 ovvero la VLAN Nativa, il frame viene identificato come appartenente alla VLAN nativa e dunque non è modificato quando passa in un trunk. Avendo inoltre un secondo tag, il frame viene instradato verso il secondo switch. Questo switch vedendo che il secondo tag identifica la "sua" VLAN, provvede ad eliminare il tag ed ad instradare il frame all'host destinatario appartenente alla seconda VLAN.

3.4.2 Strategia e codice

Per riuscire a testare ed a rilevare questo attacco con Zeek, è stato quindi necessario quindi necessario scaricare da internet un file pcap contente una legittima comunicazione di una VLAN, ed in seguito crearne uno che contenesse una comunicazione malevola.
Per la creazione del file pcap conentente una comunicazione malevola, ho utilizzato la libreria Scapy che mi ha permesso di creare pacchetti personalizzati:

```
packet = Ether(dst="ff:ff:ff:ff:ff:ff")/\
         Dot1Q(vlan=1)/\
         Dot1Q(vlan=2)/\
         Dot1Q(vlan=2)/\
         IP(src="10.1.2.3",dst="10.1.2.254")/\
         ICMP()
```

Avendo ottenuto così i file pcap necessari, ho provveduto per analizzarli con Zeek. In primo luogo ho testato lo script Zeek sul file pcap legittimo, ottenendo i risultati previsti. È stato infatti possibile ottenere le vlan rispettive ad ogni comunicazione:

Listing 1: Output comunicazione legittima

```
Connection: CNrqu32dIONRfWOdHk
131.151.107.254:520/udp -> 255.255.255.255:520/udp, vlan: 104
Connection: C0WQv23L7Q82KaxnXd
131.151.32.254:520/udp -> 255.255.255.255:520/udp, vlan: 32
Connection: Cmz90322de5PXZWXc3
131.151.111.254:520/udp -> 255.255.255.255:520/udp, vlan: 108
```

Utilizzando successivamente lo script per analizzare il file pcap malevolo è stato possibile notare e dunque rilevare l'anomalia inerentemente le VLAN. L'output ottenuto è il seguente:

Listing 2: Output comunicazione legittima

```
Connection: [orig_h=10.1.2.3, orig_p=8/icmp, resp_h=10.1.2.254, resp_p=0/icmp]
10.1.2.3:8/icmp -> 10.1.2.254:0/icmp, vlan=1, inner_vlan=2
```

Si può dunque vedere la differenza tra le i due valori delle VLAN che confermano che l'attacco è andato a buon fine; è stato dunque possibile passare da una vlan ad un'altra. Si è dunque mostrato come sia possibile rilevare con Zeek l'attacco di Vlan Hopping di tipo Double Tagging, da cui è poi possibile attuare dinamiche reazione all'attacco.

4 Conclusioni

Il progetto ha dunque esplorato in dettaglio l'analisi avanzata del traffico di rete attraverso l'uso di Zeek, l'estrazione di features e l'implementazione di una rete neurale. I risultati ottenuti delineano un avanzamento significativo nel campo dei Network Intrusion Detection Systems (NIDS) e della sicurezza delle reti.

L'utilizzo di Zeek sul Dataset CIC Modbus 2023 ha consentito un'analisi dettagliata delle connessioni, che ha permesso di analizzare positivamente il Dataset fornito. Inoltre l'implementazione di una rete neurale di classificazione ha ampliato le capacità di valutazione in tempo reale del traffico, migliorando l'accuratezza della rilevazione di attività malevole.

Infine è stata effettuata un'analisi approfondita degli attacchi di VLAN-Hopping, che ha contribuito ad una comprensione più approfondita delle minacce in questo contesto. L'utilizzo di Zeek ha quindi permesso di identificare scenari di attacco e dunque ipotizzare strategie di mitigazione.

In conclusione, il progetto non solo contribuisce alla comprensione avanzata dei NIDS, ma offre anche un punto di partenza per futuri sviluppi nel campo della sicurezza delle reti e nell'implementazione di NIDS unendo i features extractor con reti neurali. L'evoluzione continua di Zeek e delle tecnologie correlate promette progressi continui nella sicurezza e nella resilienza delle infrastrutture informatiche.

5 Fonti fornite

@INPROCEEDINGS7348942, author=Moustafa, Nour and Slay, Jill, booktitle=2015 Military Communications and Information Systems Conference (MilCIS), title=UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set), year=2015, volume=, number=, pages=1-6, doi=10.1109/MilCIS.2015.7348942

@artice author=University of Brunswick, site=https://www.unb.ca/cic/datasets/modbus-2023.html, title=CIC Modbus Dataset 2023