Corso di Laurea in Informatica

TESI DI LAUREA

Integrazione di Deep Packet Inspection in un Intrusion Detection System Open Source

Relatore:
Luca Deri

Candidato:
Luca Ferretti

ANNO ACCADEMICO 2023/2024

Indice

1 Introduzione **1**

1.1 Obiettivo 2

2 Stato dell'arte **4**

2.1 Zeek 4

2.1.1 Architettura 6

2.1.2 Workflow di rilevamento 7

2.1.3 Strumentazione e Collezione 8

2.1.4 Creazione dei log 9

2.2 nDPI 10

2.2.1 Deep Packet Inspection 11

2.2.2 Architettura di nDPI 13

2.2.3 nDPI e Cybersecurity 14

3 Architettura del programma **17**

3.1 Struttura e scelte implementative 18

3.2 Vantaggi 19

4 Implementazione **21**

4.1 Architettura e classi di Zeek 21

4.1.1 Compilazione e uso 22

4.2 Analisi dei pacchetti da parte di nDPI 23

4.2.1 log di nDPI 25

5 Validazione **28**

5.1 Casi d'uso 28

5.1.1 Ricerca di una intrusione 29

5.1.2 Visibilità di rete 34

5.2 Performance 36
5.2.1 Uso della memoria 38
5.2.2 Profiling della CPU 40

6 Conclusioni **42**
6.1 Lavori futuri 43

7 Ringraziamenti **44**

1. Introduzione

Il monitoraggio di rete [33] è un'operazione sempre più fondamentale, in quanto permette di comprendere il comportamento di una rete per garantire sicurezza, affidabilità e prestazioni ottimali delle infrastrutture. Attraverso il monitoraggio delle reti, le organizzazioni possono raccogliere e analizzare dati relativi al traffico di rete, consentendo loro di individuare e risolvere problemi, sia per quanto riguarda l'utilizzo delle risorse sia per la rilevazione di minacce.

Una tecnica ormai diffusa per identificare anomalie sul traffico di rete riguarda l'utilizzo di IDS (Intrusion Detection System) [21], un dispositivo software o hardware in grado di generare avvisi in caso di accessi non autorizzati a computer o reti locali. Un IDS consiste in un insieme di tecniche e metodi realizzati per la rilevazione di pacchetti sospetti a livello di rete, di trasporto o applicativo. Possono distinguersi in due categorie di base: sistemi basati su firme (*signature based*) e sistemi basati su anomalie (*anomaly based*). Esiste inoltre una ulteriore suddivisione a seconda di che tipo di infrastruttura analizzano: esistono IDS che analizzano reti locali, quelli che analizzano gli Host e gli IDS ibridi che analizzano entrambi.

I *Signature based detection systems* [18] utilizzano pattern di intrusioni noti e la loro efficienza nel riconoscere attacchi dipende necessariamente dalla frequenza con cui ricevono aggiornamenti dei pattern e sono incapaci di identificare minacce sconosciute.

Gli *Anomaly based detection system* [15] utilizzano un insieme di regole o euristiche che definiscono cosa sia normale e cosa anormale sulla rete, sarà quindi necessaria una prima fase in cui viene definito cosa sia normale per quella specifica rete. Benché riescano ad identificare anche attacchi non presenti nel database hanno come contro un maggior numero di falsi positivi.

Un *Host based IDS* (HIDS) [36] consiste in un agente che analizza localmente l'Host alla ricerca di intrusioni. Le intrusioni vengono rilevate analizzando file di log, system call, modifiche al file system e altre componenti. I *Network Based IDS* (NIDS) [24] vengono posizionati in punti strategici dell'infrastruttura di rete e sono in grado di catturare ed

analizzare dati in cerca di tracce di attacchi. In questo caso parliamo di *packet sniffing*, in quanto vengono catturati i pacchetti che passano attraverso i mezzi di comunicazione. Un *Hybrid Intrusion Detection System* combina le informazioni recuperate dagli agenti in esecuzione negli Host con quelle prelevate dalla rete locale.
Al contrario degli IPS (*Intrusion Prevention System*) [37] che sono in grado di bloccare il traffico in seguito al rilevamento di una violazione, gli IDS si limitano ad osservare e segnalare problematiche. Sarà poi compito di un addetto occuparsi della segnalazione prendendo provvedimenti ritenuti più congrui.

1.1 Obiettivo

Zeek [29] è un analizzatore del traffico di rete passivo open source. Si occupa di registrare le attività di rete che intercetta senza dare alcun tipo di giudizio e senza, nella sua configurazione standard, notificare l'utente di attività particolari in un set di files di log. È possibile generare avvisi al verificarsi di specifici eventi di interesse, attraverso script scritti nel linguaggio del progetto che vengono interpretati a tempo di esecuzione.
Strumenti per il Deep Packet Inspection (DPI), come ad esempio nDPI [8], permettono di estrarre maggiori informazioni da un pacchetto in quanto non si limitano ad ispezionare solamente il suo Header, ma ricercano metadati anche nel payload di esso. Il DPI è in grado di individuare, in aggiunta alle capacità di ispezione di tecnologie di packet sniffing tradizionali, minacce nascoste all'interno del flusso di dati, come possono essere violazione dei diritti, malware, tentativi di esfiltrazione di informazioni ed altro.
Questa tesi nasce dall'esigenza di un software che potesse offrire i servizi di un IDS in modo da monitorare gli eventi di interesse che accadono in rete e allo stesso tempo che potesse permettere di svolgere analisi approfondite sul traffico, attraverso l'uso di tutti quei dati che i pacchetti si portano dietro e che grazie a tecniche di DPI vengono presentati in maniera facilmente consultabili. In particolar modo si era alla ricerca di uno strumento del genere che fosse anche open-source. Proprio per questo, nel corso del seguente elaborato, vedremo come attraverso l'integrazione di nDPI in Zeek siamo in grado di arricchire le capacità di analisi del traffico di rete del framework, consentendo una maggiore precisione

nell'identificazione dei protocolli e delle applicazioni, nonché una migliore comprensione del comportamento del traffico di rete all'interno dell'ambiente monitorato.

Nei capitoli successivi vedremo in maniera introduttiva le architetture e l'utilizzo dei due software 2, verrà data un'introduzione del lavoro effettuato, esponendo motivazioni e benefici di quanto fatto 3. A questo punto esamineremo i passaggi che sono stati necessari per integrare nDPI in Zeek ponendo particolare attenzione sulle parti di interesse dell'architettura 4 ed infine andremo a mostrare dei reali utilizzi mostrando un modo per poter utilizzare Zeek con l'aiuto di nDPI per il riscontro di attività malevola, andremo inoltre ad esporre come il DPI può chiarire il comportamento in rete degli host connessi ad essa 5. Nello stesso capitolo daremo anche una valutazione su quelle che sono le performance con e senza le modifiche apportate.

Per testare le nuove funzionalità integrate all'interno di Zeek verranno usati dataset di attacchi passati che possono essere trovati pubblicamente su Malware Traffic Analysis così come altri dataset contenenti traffico di rete, sempre reperibili pubblicamente[1], per mostrare come nDPI ci aiuti nella visibilità di rete.

[1] https://www.unb.ca/cic/datasets/index.html

2. Stato dell'arte

Il lavoro svolto si basa sull'utilizzo dei due programmi già citati, Zeek ed nDPI. Per poter comprendere al meglio come questi siano stati congiunti e come sono stati utilizzati insieme, è necessario avere una visione di generale del funzionamento di entrambi. Cercheremo in questa sezione di spiegare almeno le parti più rilevanti della loro struttura e quale sono le loro funzionalità principali.

2.1 Zeek

Zeek è un software open source che ha come obiettivo quello di analizzare in maniera passiva il traffico di rete. Durante l'esecuzione vengono creati dei file di log per descrivere le attività della rete. La suddivisione delle informazioni viene effettuata a seconda del protocollo di livello applicativo. Ad esempio troviamo sessioni HTTP, richieste DNS con relative risposte, certificati SSL. Queste informazioni vengono scritte in un formato TSV o JSON ma l'utente può decidere di salvarle in un database esterno.
Inoltre il progetto offre un proprio linguaggio di scripting per poter eseguire specifici task di analisi durante l'esecuzione.
A differenza di altri IDS presenti, come Suricata o Snort [1], che si concentrano maggiormente sul rilevamento di intrusioni e l'invio di allarmi, Zeek consente di eseguire analisi del traffico in tempo reale e analisi forense approfondita. Inoltre Zeek si concentra sull'analisi del contenuto dei pacchetti piuttosto che sul riconoscimento di pattern di attacco basati su firme predefinite. Benché la sicurezza sia un aspetto fondamentale dell'analisi del traffico, Zeek preferisce dare maggiore enfasi ai dati ricavati dal traffico in modo tale che un analista possa scegliere come usarli in base alle esigenze. Ne è la conferma il fatto che non Zeek non generi avvisi nella sua configurazione standard, ma lascia all'utente la possibilità di utilizzare regole per poter inviare delle notifiche.
Quando parliamo di analisi del traffico, i team di sicurezza possono dirsi dipendenti da quattro diverse origini dei dati quando sono alla ricerca di attività sospette e dannose.

Questi includono *fonti di terze parti* come forze dell'ordine, peers e organizzazioni, commerciali o nonprofit, di threat intelligence; *network data*, *infrastructure application data*, che includono logs di un ambiente cloud, e *endpoint data*. Zeek è una piattaforma che si occupa di raccogliere principalmente il secondo tipo di questi dati, i *network data*.
Per quanto riguarda i network data, la pratica che si occupa di classificare ed analizzare questi dati prende il nome di **Network Forensics** [5, 28], il cui obiettivo è appunto quello di analizzare il traffico che si sviluppa lungo una particolare rete. Tale analisi è orientata alla raccolta di dati ed informazioni, le quali vengono esaminate da tecnici specializzati alla ricerca di elementi che possano dimostrare una violazione della rete, un'acquisizione di informazioni riservate o un altro tipo di azione illecita.
Attraverso le indagini di *Network Forensic* è possibile ottenere una classificazione dei dati suddivisi in quattro categorie:

1. **Full content data**, ossia ogni informazione transitata sulla rete senza alcun tipo di filtro. Raramente queste informazioni si rivelano utili, poiché buona parte del materiale acquisito non è pertinente. Per questo motivo vengono preferiti gli **Extracted Content data**, che estraggono il contenuto di una connessione.

2. **Transaction data**, in questo caso viene identificato uno scambio di informazioni tra due host. Tali dati informano su quando e tra chi è avvenuta una connessione, ma non forniscono informazioni sul contenuto.

3. **Statistical data**, questo genere di dati include il numero di byte contenuti in un pacchetto, timestamp di inizio e fine della connessione, protocolli usati, nodi attivi sulla rete. In generale tutti quei metadati che permettano di ottenere informazioni di natura statistica.

4. **Alert data**, si tratta dei dati generati automaticamente dal sistema di identificazione delle intrusioni.

Zeek permette di raccogliere tre di queste tipologie di dati (*transaction data*, *extracted content data*, *alert data*), anche se è conosciuto principalmente per i *transaction data*.
Quando eseguito Zeek generera una collezione di log delle connessioni compatti in cui

vengono descritti i protocolli e le attività viste sulla rete senza alcun pregiudizio, quindi non vi saranno di default avvisi di sicurezza.

2.1.1 Architettura

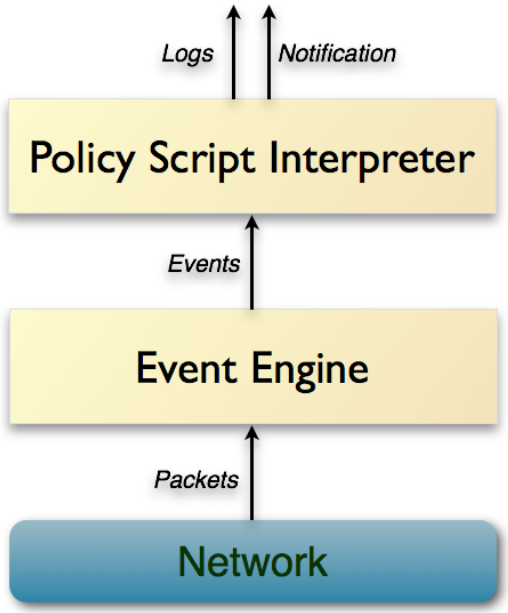


Figura 2.1: Integrazione tra componenti di Zeek

Possiamo vedere Zeek come stratificato su due componenti principali.
La parte più bassa, direttamente a contatto con il flusso di rete, riduce i pacchetti in arrivo in una serie di eventi high-level. Tali eventi riflettono l'attività della rete senza fornire alcun tipo di giudizio, ovvero viene descritto cosa è stato visto ma non perché o se sia significativo.
Ad esempio, ogni richiesta HTTP viene trasformata nel relativo evento *http_request* che trasporta con sé gli indirizzi IP e le porte coinvolte, l'URI richiesto e la versione HTTP. Non viene data alcuna interpretazione, ad esempio se l'URI corrisponda o meno ad un sito malevolo.
La componente che si occupa degli eventi comprende una serie di sottocomponenti, in par-

ticolar modo la pipeline di elaborazione dei pacchetti è composta da: l'input source, riceve il traffico di rete in entrata dalle interfacce di rete, così come tre tipologie di analizzatori.

- **analizzatore delle sessioni**, gestisce i protocolli di livello applicativo.
- **analizzatore di pacchetti**, processa i protocolli di livello più basso a partire da quello di collegamento.
- **analizzatore di file**, disseziona il contenuto dei file trasferiti nel corso delle sessioni.

Tale componente permette anche di aggiungere Plug-in appositamente scritti per soddisfare le necessità di casi specifici.
La seconda componente principale di Zeek è quella che si occupa di interpretare gli eventi grazie a dei gestori scritti nell'omonimo linguaggio di scripting. In maniera più specifica questi script possono consegnare qualsiasi proprietà di interesse dal traffico in entrata. Tutti gli output che troviamo di default derivano proprio da questi script. Il linguaggio Zeek permette agli script di mantenere lo stato nel tempo, consentendo di tracciare l'evoluzione di ciò che viene osservato al di là delle connessioni dell'host. Inoltre utilizzando script ad hoc è possibile generare notifiche real-time ed eseguire programmi esterni su richiesta.

2.1.2 Workflow di rilevamento

Come detto precedentemente, Zeek di default è ottimizzato per poter fornire due dei quattro tipi di dati del monitoraggio di rete. Offre i dati delle transazioni ed estrae i dati di contesto. Per Zeek è anche possibile fornire dei dati di allerta sotto forma di avvisi, gli analisti possono anche modificarlo per creare delle allerte per specifici eventi.
La scoperta di incidenti inizia collezionando dati di sicurezza, seguito dalla loro analisi. Durante questa fase, in assenza di un avviso specifico di attività malevola come in questo caso, è possibile distinguere due categorie di ricerca: *matching* (”abbinamento”) e *hunting* (”ricerca”).
Quando si parla di *matching* [23] si fa riferimento alla revisione dei dati alla ricerca di segni di compromissione. *Hunting* [2] significa lavorare senza indicazioni di una eventuale compromissione, facendo affidamento su un'ipotesi iniziale su come attività avverse

potrebbero manifestarsi sui dati raccolti. Il secondo metodo coinvolge un approccio più scientifico. Gli analisti possono utilizzare un archivio di log di Zeek in cerca di compromissioni ed avviare un'investigazione di sicurezza una volta trovato un indirizzo IP, uno user-agent HTTP o uno qualsiasi degli elementi messi a disposizione da Zeek che risulti legato ad attività anomale.
Allo stesso modo possono modellare una ipotesi su come certi comportamenti potrebbero apparire nell'insieme dei dati raccolti, quindi ricercare segni che possano provare o confutare la stessa.
In alternativa può essere affiancato ad un IDS vero e proprio in grado di generare avvisi a seguito di un tentativo di compromissione. Dato che gli IDS spesso non forniscono molti dettagli, gli analisti possono, alla ricezione di un avviso, andare ad investigare sui vari log generati. Basandosi sui dati a disposizione potrebbero essere in grado di comprendere e risolvere l'incidente.
In generale i dati raccolti possono essere affiancati a un qualsiasi strumento che generi degli avvisi al verificarsi di un evento per poter svolgere una analisi più approfondita. Questo può essere fatto anche nel caso in cui l'avviso arrivi da un sistema esterno, ad esempio un server remoto, per non accedervi direttamente in caso quest'ultimo fosse stato realmente compromesso, andando ad ispezionare soltanto i dati generati da Zeek.

2.1.3 Strumentazione e Collezione

Solitamente Zeek viene eseguito facendo analizzare il traffco di una o più interfacce di rete, anche se è comunque possibile processare file PCAP. Zeek scrive i dati in una posizione specificata nei suoi file di configurazione, di default la direcory corrente in cui il programma viene eseguito. Zeek possiede le capacità di riscrivere i log in vari formati e di eseguire specifiche operazioni di gestione di questi, come ad esempio la compressione e l'archiviazione.
La revisione dei log generati può avvenire semplicemente utilizzando strumenti di elaborazione del testo, ad esempio in un ambiente Linux si può pensare di utilizzare *cat*, *grep*, *awk* o qualsiasi altro tool di sorta. A seconda del formato l'utente può scegliere di utilizzare strumenti più specializzati, alcuni dei quali vengono forniti direttamente da Zeek

come ad esempio *zeek-cut*, che permette di isolare i campi di interesse da un log.
Molto spesso gli analisti che utilizzano Zeek inviano i log ad applicazioni specializzate nella conservazione e nell'analisi, indicate come SIEM (Security and Information Event Management) [3].

2.1.4 Creazione dei log

Il sistema di creazione di log di Zeek ruota attorno a tre componenti:

- **Streams**, ogni stream corrisponde ad un singolo log. Definisce i campi, con relativi nomi e tipi, di ogni log.
- **Filters**, ogni stream possiede alcuni filter che determinano quali e come vengono scritte le informazioni di interesse. Alla creazione di una stream ne viene assegnato uno, altrimenti le informazioni non verrebbero scritte su disco. Possono esserne aggiunti di nuovi in modo da scrivere, anche su output diversi, un sottoinsieme delle informazioni.
- **Writers**, ogni filter ha un writer, che definisce il formato di output per le informazioni. Il writer di default è quello ASCII.

Il numero di file che un analista ha a disposizione dipende dal numero di protocolli individuati da Zeek nel traffico ispezionato, per cui non tutti saranno sempre disponibili. Alcuni tuttavia vengono creati indipendentemente dal contesto e dai protocolli riconosciuti. Di questi *conn.log* è tra i più importanti in quanto tiene traccia di tutte le connessioni che sono state stabilite. Inoltre questo file contiene i protocolli estratti da ogni connessione, permettendo di sapere a quale log è opportuno accedere per avere informazioni aggiuntive.
Di seguito un esempio di una entry in formato JSON del file *conn.log*.

```
1 {
2   "ts": 1554410064.698965,
3   "uid": "CMreaf3tGGK2whbqhh",
4   "id.orig_h": "192.168.144.130",
5   "id.orig_p": 64277,
6   "id.resp_h": "192.168.144.2",
```

```
  "id.resp_p": 53,
  "proto": "udp",
  "service": "dns",
  "duration": 0.320463,
  "orig_bytes": 94,
  "resp_bytes": 316,
  "conn_state": "SF",
  "missed_bytes": 0,
  "history": "Dd",
  "orig_pkts": 2,
  "orig_ip_bytes": 150,
  "resp_pkts": 2,
  "resp_ip_bytes": 372,
  "tunnel_parents": []
}
```


Code 2.1: Entry di conn.log in formato Json

2.2 nDPI

nDPI è uno strumento di Deep Packet Inspection, nato come fork dell'ormai deprecato OpenDPI [27], che permette di:

- Rilevare il protocollo applicativo di un flusso.
- Analizzare flussi di traffico criptato.
- Estrarre metadati da protocolli selezionati, così come metriche della rete (ad esempio la latenza) che possano essere utilizzati all'interno di applicazioni di monitoraggio evitando la duplicazione della decodifica dei pacchetti.
- Implementare API per l'analisi del traffico.

Gli strumenti per il DPI sono vari sia open-source che proprietari. Ad esempio Netify offre un'analisi simile a quella di nDPI, anche se oltre a concentrarsi sul riconoscimento di applicazioni di rete fornisce anche informazioni e statistiche dettagliate sull'utilizzo

delle applicazioni e sui modelli del traffico. Tuttavia il numero di protocolli identificati è ben lontano da quello garantito da nDPI. Strumenti come SolarWinds [30] si concentrano maggiormente sulla visibilità ed il controllo dell'infrastruttura IT, tra cui dispositivi di rete, server, applicazioni e servizi cloud, offrendo funzionalità tra cui il monitoraggio delle prestazioni della rete, la gestione degli eventi, la diagnosi dei problemi di rete, la pianificazione delle risorse.
In sintesi, nDPI offre un'alternativa open-source, precisa, efficiente e flessibile per l'analisi del traffico di rete rispetto ad altri DPI commerciali. La sua ampia copertura di protocolli, la precisione nella rilevazione e l'efficienza nell'utilizzo delle risorse lo rendono una scelta popolare per una vasta gamma di applicazioni e scenari di utilizzo.

2.2.1 Deep Packet Inspection

Il DPI [11] è una tecnica di analisi del traffico di rete che ha l'obiettivo di ispezionare in dettaglio i dati scambiati attraverso la rete. Uno strumento per il DPI può prendere provvedimenti come notificare un evento, bloccare o registrare attività. Viene utilizzata per rilevare virus e pacchetti dannosi cercando anomalie dei pacchetti che passano sulla rete di interesse, ma anche per svolgere controlli sulle performance, sull'utilizzo del traffico e sul corretto formato dei pacchetti. Le informazioni vangono ricercate nel payload dei pacchetti piuttosto che nelle intestazioni. Le firme vengono spesso ricercate per rilevare i tipi di flusso e agire di conseguenza. Questo può portare a svantaggi che sono principalmente legati al rallentamento del traffico analizzato, dovuto all'aumento della potenza di calcolo e di RAM necessaria per trattare grandi quantità di pacchetti[32].
Inoltre l'utilizzo di DPI solleva domande sulla privacy [6], in quanto vanno ad analizzare il contenuto del traffico con finalità diverse. Tuttavia non è scopo di questo elaborato approfondire tale tematica.
Per collezionare i pacchetti che transitano sulla rete esistono diversi metodi. Questi includono il port mirroring [38], detto anche SPAN (Switched Port Analyzer), una tecnica che permette di copiare ed inoltrare il traffico di rete da una o più porte ad una porta di monitoraggio. Può essere implementato su vari dispositivi di rete, tra cui switch Ethernet, router e dispositivi di rete virtuale. Tuttavia è comunemente applicato su switch Ethernet,

che consentono agli amministratori di rete di configurare il mirroring attraverso l'interfaccia di gestione del dispositivo. In alternativa possono essere utilizzati dispositivi fisici, i network tap [34], inseriti fisicamente tra dispositivi di rete in grado di duplicare ed inviare il traffico che passa attraverso di essi a strumenti di analisi.

nDPI non è in grado di bloccare un flusso ma soltanto di analizzarlo per estrarre informazioni, come il protocollo applicativo, e dare una stima del rischio. In particolare è in grado di distinguere una grande varietà di protocolli applicativi, pratica che sta diventando sempre più complessa a causa di protocolli criptati [22], peer-to-peer o indipendenti da una specifica porta. In nDPI ad ogni flusso viene associato un protocollo di rete, ad esempio SMTP o DNS, e un protocollo di comunicazione, come può essere Instagram o Skype considerati in questo caso protocolli applicativi anche se non lo sono per l'IETF[1]. Ad esempio possiamo trovare associazioni del tipo *DNS.Facebook* o *QUIC.YouTube*.

La maggior parte dei protocolli oggigiorno sono HTTP/TLS-based. nDPI include il supporto per il rilevamento di protocolli string-based: query DNS, campi di intestazione HTTP di Host/Server, TLS/QUIC Server Name Indication. Di seguito un esempio del rilevamento dell'applicazione Netflix.

```
{ "netflix.com", NULL, "netflix" TLD, \NetFlix", NDPI_PROTOCOL_NETFLIX,
    NDPI_PROTOCOL_CATEGORY_STREAMING, NDPI_PROTOCOL_FUN },
{ "nflxext.com", NULL, "nflxext" TLD, "NetFlix", NDPI_PROTOCOL_NETFLIX,
    NDPI_PROTOCOL_CATEGORY_STREAMING, NDPI_PROTOCOL_FUN },
{ "nflximg.com", NULL, "nflximg" TLD, "NetFlix", NDPI_PROTOCOL_NETFLIX,
    NDPI_PROTOCOL_CATEGORY_STREAMING, NDPI_PROTOCOL_FUN },
{ "nflximg.net", NULL, "nflximg" TLD, "NetFlix", NDPI_PROTOCOL_NETFLIX,
    NDPI_PROTOCOL_CATEGORY_STREAMING, NDPI_PROTOCOL_FUN },
{ "nflxvideo.net", NULL, "nflxvideo" TLD, "NetFlix", NDPI_PROTOCOL_NETFLIX,
    NDPI_PROTOCOL_CATEGORY_STREAMING, NDPI_PROTOCOL_FUN },
{ "nflxso.net", NULL, "nflxso" TLD, "NetFlix", NDPI_PROTOCOL_NETFLIX,
    NDPI_PROTOCOL_CATEGORY_STREAMING, NDPI_PROTOCOL_FUN },
```

[1]Internet Engineering Task Force, https://www.ietf.org/

2.2.2 Architettura di nDPI

La libreria di nDPI è composta da due componenti principali, una parte centrale responsabile della gestione dei pacchetti grezzi e un dissector per rilevare i protocolli supportati, questa seconda parte viene gestita tramite plugin.
nDPi si occupa di decodificare dal livello 3 in poi ed ha bisogno di ricevere pacchetti già divisi in flussi (suddivisi secondo IP/porta di sorgente/destinazione, protocollo) e quindi devono essere già stati gestiti gli incapsulamenti fino al livello 2 (ad esempio VLAN). I dissector per i vari protocolli sono registrati in base ad attributi quali protocollo e porta standard. Ciò significa che, ad esempio, il dissector per HTTP specifica come default TCP/80. In questo modo i pacchetti che appartengono ad un flusso non ancora classificato possono essere passati a tutti i dissector a partire dal più probabile, diminuendo il numero di dissector da testare e di conseguenza il tempo impiegato a trovare quello corretto. Se un flusso rimane non classificato (quindi tutti i dissecotr sono stati provati senza alcun riscontro), nDPI può supporre il protocollo controllando se ce ne fosse uno registrato con tale protocollo/porta. Questo accade non tanto per la limitazione del numero dei dissector, ma potrebbe essere dovuto al fatto che non tutti i pacchetti del flusso sono stati passati ad nDPI (ad esempio il programma è stato avviato successivamente all'inizio del flusso).
Allo stesso modo ogni flusso mantiene uno stato dei dissector che non hanno ottenuto un riscontro positivo in modo da poterli ignorare alla successiva iterazione. L'analisi si interrompe nel momento in cui un protocollo è stato trovato. Questo richiede mediamente non più di otto pacchetti.
Come già accennato, il traffico di rete si sta spostando verso la comunicazione criptata. Per motivi di privacy e sicurezza, HTTPS sta rimpiazzando HTTP non solo in comunicazioni sicure ma anche in attività della quotidianità come scambio di messaggi o navigazione in Internet. Identificare tale traffico come SSL [25] non è sufficiente e c'è la necessità di una maggiore caratterizzazione. Quando viene usato traffico criptato, l'unica parte dello scambio di dati che può essere decodificata è lo scambio di chiave iniziale. nDPI contiene un decoder per SSL che riesce ad estrarre l'host name del server contattato, tale informazione viene poi inserita tra i metadati del flusso. Con questo approccio è possibile

identificare servizi noti e assegnare ad essi un tag a seconda del nome del server. Ad esempio una comunicazione criptata con un server con nome 'api.twitter.com' verrà marcato come Twitter. Inoltre è possibile individuare certificati SSL auto-firmati, che potrebbero significare che la connessione in questione non è sicura.
nDPI contiene internamente una configurazione per il riscontro di molti protocolli noti, rilevati utilizzando la tecnica descritta sopra. Oltre a ciò è possibile aggiungere a runtime un file di configurazione che estenda l'insieme di protocolli riconosciuti in modo tale da poter aggiungerne di nuovi senza dover modificare il dissector. Con l'avvento dei *Content Delivery Netwroks* (CDN) [35], questo risulta essere l'unica via per poter identificare il protocollo applicativo in quanto un singolo server (identificato con un unico indirizzo IP) può fornire due servizi differenti da due clienti. Come ripiego, nDPI è in grado di riconoscere applicazioni specifiche attraverso l'indirizzo IP. Ad esempio Apple fornisce una varietà di servizi distinti (iCloud, iMessage, ...) che vengono identificati da nDPI, ma potrebbe marcare come Apple tutte le connessioni non meglio identificate con un indirizzo IP di provenienza registrato da Apple (ovvero 17.0.0.0/8).

2.2.3 nDPI e Cybersecurity

Tecniche per il DPI possono essere impiegate, tra le altre cose, per stabilire se un determinato traffico è o meno malevolo [7]. nDPI associa ad ogni flusso un rischio per identificare le comunicazione che potrebbero presentare problemi di sicurezza, oltre a fornire una descrizione dei problemi riscontrati (es. "HTTP suspicious user-agent", "TLS connections not carrying HTTPS (e.g. a VPN over TLS)", "Known protocol on non standard port").
Questo viene fatto attraverso:

1. Firme digitali di un handshake, negoziazione tra client e server, di una comunicazione TLS/SSL attraverso il metodo JA3 [16]. I parametri scambiati vengono catturati ed analizzati per creare un hash univoco da poter confrontare con un database di firme note per identificare l'applicazione o software che ha generato la connessione.

2. Firme dei certificati TLS, calcolata utilizzando algoritmi crittografici come SHA-1, SHA-256 o MD5, e serve a identificare in modo univoco il certificato all'interno

di un sistema o di una rete. Viene usato per garantire l'autenticità e l'integrità dei certificati TLS/SSL utilizzati nelle connessioni sicure.

3. Controllando l'entropia dei bytes [31], una metrica utilizzata per misurare la distribuzione dei bytes. Questa è fortemente correlata all'applicazione che si sta utilizzando, è quindi possibile creare dei confini all'interno dei quali tale valore deve risiedere.

Lo score calcolato è dato dalla somma di tutti i rischi identificati su un certo flusso, i quali a loro volta vengono ottenuti sommando lo score dato dal client e lo score dato dal server.

nDPI supported risks:

Id	Risk	Severity	Score	CliScore	SrvScore
1	XSS attack	Severe	250	225	25
2	SQL injection	Severe	250	225	25
3	RCE injection	Severe	250	225	25
4	Binary application transfer	Severe	250	125	125
5	Known protocol on non standard port	Medium	50	25	25
6	Self-signed Certificate	High	100	90	10
7	Obsolete TLS version (older than 1.2)	High	100	90	10
8	Weak TLS cipher	High	100	90	10
9	TLS Expired Certificate	High	100	50	50
10	TLS Certificate Mismatch	High	100	50	50
11	HTTP Suspicious User-Agent	High	100	90	10
12	HTTP Numeric IP Address	Low	10	5	5
13	HTTP Suspicious URL	High	100	90	10
14	HTTP Suspicious Header	High	100	90	10
15	TLS (probably) not carrying HTTPS	Low	10	5	5
16	Suspicious DGA domain name	High	100	90	10
17	Malformed packet	Low	10	5	5
18	SSH Obsolete Client Version/Cipher	High	100	90	10
19	SSH Obsolete Server Version/Cipher	Medium	50	5	45
20	SMB Insecure Version	High	100	90	10
21	TLS Suspicious ESNI Usage	Medium	50	25	25
22	Unsafe Protocol	Low	10	5	5
23	Suspicious DNS traffic	High	100	90	10
24	SNI TLS extension was missing	Medium	50	25	25
25	HTTP suspicious content	High	100	90	10
26	Risky ASN	Medium	50	25	25
27	Risky domain name	Medium	50	25	25
28	Possibly Malicious JA3 Fingerprint	Medium	50	25	25
29	Possibly Malicious SSL Cert. SHA1 Fingerprint	Medium	50	25	25
30	Desktop/File Sharing Session	Low	10	5	5
31	Uncommon TLS ALPN	Medium	50	25	25
32	TLS certificate validity longer than 13 months	Medium	50	25	25
33	TLS suspicious extension	High	100	90	10
34	TLS fatal alert	Low	10	5	5
35	Suspicious entropy	Medium	50	25	25
36	Clear-text credentials	High	100	90	10
37	DNS packet larger than 512 bytes	Medium	50	25	25
38	Fragmented DNS message	Medium	50	25	25
39	Text contains non-printable characters	High	100	90	10

Figura 2.2: Risk Score di nDPI

Molti prodotti per la Cybersecurity tendono ad applicare policy molto stringenti, dividendo il mondo in maniera netta. La realtà è più complessa di così e il moderno deve coesistere con l'antico. Il principio dell'utilizzo di uno score può essere considerato del tutto affidabile solamente nel caso in cui non vi siano falsi positivi, che altrimenti porterebbero a generare falsi allarmi.
Alcuni esempi tipici di eccezioni:

- IP privati con certificati TLS auto-firmati.
- Protocolli/host non sicuri che non possono essere aggiornati ma che forniscono un servizio specifico ad alcuni client.
- Applicazione eseguita su porta non standard (ad esempio SSH su porta 2222).

In modo da identificare cambiamenti significativi di un host, dato che è difficile stabilire a priori quali comportamenti e valori ogni host dovrebbe tenere nel tempo, viene utilizzato il Double Exponential Smoothing [9] che implementa la previsione dei dati dando anche dei confini inferiori/superiori.
Inoltre nDPI riesce ad identificare comunicazioni a basso rumore che possono nascondersi facilmente nel traffico generale, i Beacons. Queste vengono spesso usate da malware per comunicare con chi li ha lanciati, possono indicare un fallimento nella connessione, sono usati per identificare attività di monitoraggio o controlli periodici.
Per identificarli viene tenuta traccia delle quadruple $\langle$*IP sorgente/destinazione, porta di destinazione, protocollo di trasporto*$\rangle$. Ogni volta che nDPI individua un nuovo flusso una quadrupla viene creata o aggiornata. Le quadruple che non comunicano da molto tempo o la cui comunicazione è incostante vengono scartate.

3. Architettura del programma

Con questa tesi si è cercato di sviluppare uno strumento completamente open-source che potesse svolgere le funzioni di un IDS e allo stesso tempo fornisse una visibilità di rete maggiore attraverso l'utilizzo del DPI con il vantaggio di mantenere tutti i flussi con relativi metadati all'interno di specifici file di log per poter comprendere al meglio cosa sta succedendo in rete in qualsiasi momento.

Guardando lo scenario attuale per quanto riguarda soluzioni analoghe ci accorgiamo che i prodotti disponibili sono ben pochi. Una scelta diffusa è rappresentata dall'integrazione di un qualsivoglia strumento per il DPI all'interno di Suricata[1]. Ne è un esempio il DPI di Enea Qosmos[2]. Questo tuttavia non è un programma open-source e parlando di efficienza, come possiamo leggere nelle slides introduttive, il traffico viene duplicato e quindi analizzato due volte, sia da Suricata che dal tool di DPI aumentando l'overhead sia della CPU sia della memoria. Al contrario con l'integrazione presentata con questo elaborato, nDPI riceve il traffico direttamente da Zeek. Non è quindi necessario l'aggiunta di una seconda sonda.

Parlando di integrazione, in primo luogo è stato necessario definire la struttura per poter integrare il codice di nDPI in quello di Zeek. Come verrà spiegato di seguito, questo è stato fatto attraverso la modifica del codice sorgente.

Successivamente, per testare il programma esteso, sono stati usati sia dataset di attacchi registrati in passato sia dataset di traffico che potremmo definire come generato da quotidiane attività umane. Con questi verrà mostrato come grazie al DPI le ricerche di minacce sul traffico di rete possano essere semplificate e velocizzate.

In generale grazie ad nDPI è possibile vedere aspetti che di default Zeek non andrebbe a mostrare, come ad esempio il rischio associato che può essere un buon monito per analizzare nel dettaglio ogni connessione che ne riporti almeno uno. Infatti Zeek, come già

[1] https://www.youtube.com/watch?v=g9cICnssAcE

[2] https://suricon.net/wp-content/uploads/2023/08/SURICON2022-Synold-Boosting-Suricata-with-DPI.pdf

detto, non fa assunzioni sul traffico ma si limita a salvare le informazioni estratte da esso. Questa integrazione permette, inoltre, una suddivisione delle connessioni più granulare, grazie al maggior numero di protocolli che nDPI riesce ad identificare.

3.1 Struttura e scelte implementative

L'implementazione è stata eseguita andando a modificare il codice già presente. Le linee guida per l'estensione di Zeek presenti sulla documentazione ufficiale consigliano di creare un Plug-in esterno ogni qualvolta si vogliano aggiungere nuove funzionalità per non dover modificare il codice sorgente. Come già detto nella sezione 2.1.1 troviamo tre tipi di analizzatori, ognuno dei quali riceve un insieme di parametri dal proprio gestore. Per il nostro caso d'uso il più congruo sarebbe l'analizzatore delle sessioni, poiché nDPI ha bisogno di mantenere uno stato dei pacchetti visti precedentemente in un flusso. Tuttavia, il manager di questi analizzatori invia una quantità di informazioni non sufficienti per le analisi che nDPI deve svolgere. Ad esempio non viene ricevuto l'header del pacchetto ma solamente l'header IP, andando a perdere il timestamp o altre informazioni necessarie ad nDPI per svolgere le proprie ricerche nel modo più accurato possibile.
Gli analizzatori di pacchetti ricevono invece il pacchetto completo. Di contro, utilizzando questa tipologia, avremmo dei pacchetti scollegati dalla connessione di provenienza. Questo implicherebbe la necessità di una struttura apposita per tenere traccia delle connessioni, comportando un notevole aumento della memoria utilizzata.
Andando ad operare direttamente sulle strutture già definite di Zeek i problemi di sorta non sussistono, poiché sono già presenti classi che contengono tutte le informazioni di cui nDPI necessita. Per facilitare la generazione di eventi è stata comunque aggiunta una classe che risiede sotto gli analizzatori delle sessioni, che mantiene tra le altre cose le informazioni che verranno poi scritte sul log. Inoltre si è aggiunto uno script contenente l'handler dell'evento generato alla conclusione dell'analisi di una connessione da parte di nDPI che si occupa della creazione dei record contenenti le informazioni estratte. Tale record è necessario in quanto dice a Zeek quali sono le informazioni che devono essere

scritte sul log.

Per quanto riguarda la scrittura, si è scelto di salvare tutti i flussi e non soltanto quelli per cui nDPI riesce a trovare un'applicazione, in modo tale da avere una visione di insieme di ciò che sta succedendo sulla rete. Allo stesso modo troviamo connessioni per cui abbiamo un protocollo applicativo ma a cui mancano altri metadati che non sono stati individuati. Le connessioni con le rispettive informazioni ricavate sono salvate su un log a se stante rispetto agli altri per una questione di maggior leggibilità. Questo è stato possibile in quanto l'overhead generale, come vedremo successivamente, non è risultato essere sostanziale. Inoltre la maggior parte delle informazioni che Zeek riesce ad estrarre sono le stesse di nDPI. Per non rischiare di mantenere informazioni ridondanti si è scelto di omettere tali dati e tenere soltanto quelli più interessanti che non vengono ottenuti attraverso le ricerche svolte da Zeek. Questi comprendono principalmente l'applicazione riscontrata, con relativa categoria e il tipo di rischio.

3.2 Vantaggi

A seguito del completamento del lavoro, ci ritroviamo con uno strumento in grado di classificare il traffico di rete , sia attraverso le applicazioni di una connessione sia in base al tipo di rischio associato ad essa. Inoltre offre tutti i vantaggi di un qualsiasi progetto open-source rispetto ad altre soluzioni commerciali come ad esempio DPI R&s PACE2 [4]. Parliamo della possibilità di ispezionare il codice sorgente, quindi completa trasparenza tra sviluppatori e clienti finali, così come la libera modifica e ridistribuzione.

Analizzare il traffico attraverso i semplici log generati da Zeek potrebbe risultare dispersivo e impiegare più risorse di quelle a disposizione. Grazie all'aggiunta di un DPI è possibile fare una prima selezione di quei flussi che potrebbero presentare un problema, per poter successivamente svolgere un'analisi approfondita attraverso l'uso dei dati estratti da Zeek.

Un altro aspetto molto importante è quello della visibilità di rete [13]. Come già detto anche il traffico comune è protetto da crittografia, questo rende più complesso estrarre dati significativi dal traffico al fine di comprendere cosa stia succedendo.

L'utilizzo di un DPI come nDPI, che riesce a distinguere più di 300 protocolli applicativi, permette in qualsiasi ambiente di ottenere più informazioni su ciò che viene fatto. Le informazioni riguardanti i servizi maggiormente contattati possono aiutare a svolgere una profilazione sulle abitudini di individui in contesti diversi in modo tale da poter fornire maggiori risorse ai servizi più utilizzati, offrire servizi mirati da parte di ISP (Internet Service Provider). Allo stesso modo, sfruttando la classificazione per protocolli applicativi, si può pensare di implementare ad esempio un parental control se ci troviamo in una rete domestica, bloccare l'accesso a servizi di gaming in ufficio o a risorse didattiche in una scuola durante un test.

Le applicazioni da questo punto di vista sono molteplici e spaziano da profilazioni con finalità di marketing [17], ridistribuzione delle risorse a servizi di interesse a restrizioni sul traffico in determinate sedi. Uno strumento quale Zeek che salva su files di log tutte le informazioni ricavate delle connessioni individuate può permettere di svolgere queste analisi a distanza di tempo.

4. Implementazione

Per l'integrazione di nDPI all'interno di Zeek si è scelto di modificare la struttura esistente del progetto. Anche se altre implementazioni, precedentemente esposte nella sezione 3.1, più in linea con l'idea di Zeek sarebbero stati possibili, questo è risultato essere il modo più diretto di realizzazione.
Per la concretizzazione delle nuove parti del programma i linguaggi utilizzati sono stati il C++ e Zeek, linguaggio di scripting dell'omonimo software.

4.1 Architettura e classi di Zeek

Per poter comprendere come è stato svolto il lavoro di implementazione è necessario prima capire come opera Zeek e introdurre brevemente l'architettura prestando una maggiore attenzione agli elementi interessanti per la realizzazione del progetto, rispetto a quanto già fatto precedentemente.
Zeek può intercettare i pacchetti di rete da varie fonti, compresi file pcap, interfacce di rete e dispositivi di mirroring delle porte. Questo viene fatto attraverso la libreria libpcap[10]. Tutte le informazioni dei pacchetti estratti sono contenute nella classe Packet.
Ogni pacchetto catturato è legato ad una connessione, identificata con la classe Conn. Per ogni nuovo pacchetto della connessione il sistema riconosce, in base al protocollo, quali analizzatori devono essere utilizzati, che verranno quindi applicati alla connessione.
Una volta che un analizzatore è stato aggiunto ad una connessione, Zeek lo usa per decodificare i pacchetti associati e interpretarne il contenuto. Come visto nella sezione 2.1.1 troviamo tre tipologie di analizzatori, che vengono selezionati in base ai protocolli riscontrati. È possibile per questo che il pacchetto venga inviato a più analizzatori, ad esempio un analizzatore per un protocollo di livello più basso (arp, ethernet, etc) e un analizzatore per il livello applicativo (http, netbios, etc). L'utente può crearne di nuovi da aggiungere dinamicamente a Zeek tramite la definizione di nuove istanze di analizzatori di protocollo e specificando le regole e le azioni che devono essere eseguite per ogni protocollo rilevato.

Gli analizzatori sono in grado di estrarre una vasta gamma di informazioni dai pacchetti di rete, ad esempio indirizzi IP e porte, richieste HTTP, risposte DNS, sessioni SSL/TLS. Queste informazioni vengono poi utilizzate per generare eventi, registrare log e generare report sull'attività di rete.
Durante l'analisi Zeek genera una serie di eventi per ogni attività di rete rilevata. Questi eventi forniscono informazioni dettagliate sul traffico di rete, inclusi dettagli sulle connessioni stabilite e le connessioni terminate, richieste HTTP, handshake TLS e altro ancora.

4.1.1 Compilazione e uso

Per la compilazione è stato necessario modificare il makefile in modo tale da collegare la libreria di nDPI al progetto. L'utilizzo di nDPI è lasciato a discrezione dell'utente, che potrà esprimere la volontà di integrarlo o meno durante la compilazione con una apposita flag, come mostrato nel codice 4.1. Questa ha bisogno del path in cui è stato clonato il repository di nDPI.

```
1    $ ./configure --with-ndpi=/path/to/nDPI
```

Code 4.1: Compilazione per l'integrazione di nDPI

La compatibilità tra i due programmi non è risultata essere un problema benché Zeek sia scritto prevalentemente in C++ mentre nDPI in C. Infatti il C++ permette di utilizzare librerie scritte per l'appunto in C semplicemente specificandolo nel seguente modo.

```
1 extern "C" {
2     #include <ndpi_api.h>
3     #include <ndpi_typedefs.h>
4  }
```

Code 4.2: Inserimento degli header di nDPI in Zeek

Il codice sorgente è attualmente reperibile alla pagina `https://github.com/lucaferret/zeek`

4.2 Analisi dei pacchetti da parte di nDPI

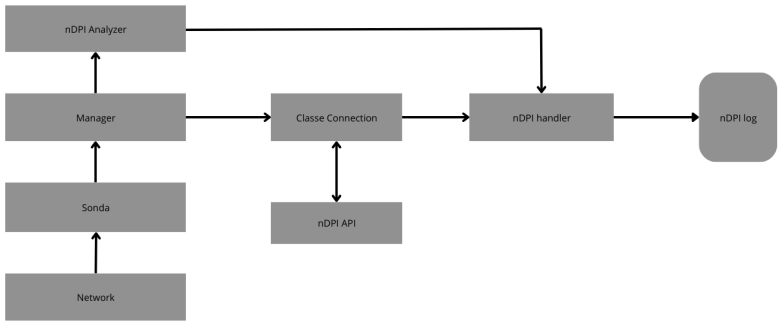


Figura 4.1: Implementazione di nDPI

La figura 4.1 illustra brevemente come è stata effettuata l'estensione di Zeek per poter analizzare i pacchetti attraverso nDPI.

La classe Connection ha visibilità di tutti i pacchetti inerenti alla connessione, oltre a mantenere metadati del tipo numero di pacchetti visti, pacchetto in entrata o in uscita, e altro ancora. Ciò ci permette di poter istanziare la struttura dati di nDPI che mantiene le informazioni estratte dai pacchetti del flusso, tra cui il protocollo applicativo, direttamente all'interno di tale classe. Per poter inviare i pacchetti all'API di nDPI è stato creato un apposito metodo per il loro inoltro. Le informazioni ricavate vengono inserite nella struttura citata. All'avvio di Zeek viene inoltre creata una struttura dati generale, il cui tipo è ancora una volta definito da nDPI, in cui sono contenute informazioni generali riguardanti i flussi.

In aggiunta è stato necessario dichiarare un apposito analyzer per poter generare eventi ogni qualvolta nDPI ottenga il protocollo di una determinata connessione. Questo contiene anche un record in cui vengono salvate le informazioni da dover scrivere nel log, oltre

ad un metodo per il loro inserimento che viene invocato al termine di una connessione. In particolare, a seguito della chiusura di una connessione, vengono inseriti i valori ricavati nel record e successivamente viene generato un evento che avvisa l'handler di salvare i valori da scrivere sul file. La scrittura viene fatta immediatamente dopo. Questa è gestita dall'handler di fine connessione ed ha priorità minore rispetto all'altra, in modo tale da permettere l'inserimento nel record di tutti i valori ricavati prima della loro scrittura su file.

La ricerca di un protocollo da parte di nDPI viene portata avanti fino a che non vi è un riscontro o fino al raggiungimento di un prefissato numero di iterazioni. Se al termine della connessione o al termine dei tentativi a disposizione nessun protocollo è stato individuato viene fatto un ultimo tentativo di ricerca, come già spiegato nella sezione 2.2.2.

Ad ogni nuovo pacchetto viene calcolato se i tentativi fatti fino ad allora siano abbastanza o meno attraverso il codice 4.3

```
1 #define MAX_PACKET_UDP 24
2 #define MAX_PACKET_TCP 80
3
4 u_int enough_packets = ( ((proto == TRANSPORT_UDP) &&
     nDPI_packet_processed > MAX_PACKET_UDP) || ((proto ==
     TRANSPORT_TCP) && nDPI_packet_processed > MAX_PACKET_TCP)) ? 1 :
      0;
5 enough_packets |= nDPI_flow->fail_with_unknown;
```

Code 4.3: Calcolo del numero massimo di pacchetti

Sia il massimo numero di pacchetti UDP che TCP è dato dai casi limite dei due protocolli. Completata l'identificazione del protocollo, i metadati estratti non vengono immediatamente scritti nel log. Questo infatti viene ultimato al termine della connessione (o all'interruzione del programma). Tale approccio è stato scelto in quanto permette di svolgere un ultimo controllo nel caso in cui non sia stato identificato precedentemente alcun protocollo e, in caso di successo, verrà scritto sull'apposito log.

4.2.1 log di nDPI

Per poter salvare i dati sul file corrispondente è stato necessario creare una nuova Stream 2.1.4. Questo è stato possibile a seguito della definizione di un *record* contenente tutti i campi di interesse da dover scrivere, a cui viene associato un identificatore unico che identifichi la nuova stream. La creazione avviene tramite l'invocazione della funzione *Log::create_stream*. Per la scrittura su file è necessario richiamare la funzione *Log::write* che si occupa dell'invocazione del *Writer*.

```
1  export {
2      redef enum Log::ID += { LOG };
3
4      type Info: record {
5      ## Timestamp for when the event happened.
6          ts: time              &log;
7          ## An unique identifier of the connection.
8          uid: string           &log;
9      ## The connection 4-tuple of endpoint addresses/ports.
10         id: conn_id           &log;
11     ## The transport layer protocol of the connection.
12         l4_proto: transport_proto &log;
13         ## The application layer protocol.
14         proto: string         &log &optional;
15         ## The protocol of the connection.
16         app_proto: string     &log &optional;
17         ## Category of application protocol
18         category: string      &log &optional;
19         ## Number of packet that nDPI has analyzed to find the
       protocol
20         num_packet_analyzed: count      &log &optional;
21         ## Indicates if the connection is encripted.
22         encripted: string     &log &optional;
23         ## Name of the host server
24         host_server: string   &log &optional;
25         ## Identifier of the risk type of the connection
```

```
26          risk_type: string    &log &optional;
27          ## Risck score of the connection calculated by nDPI
28          risk_score: count    &log &optional;
29      };
30
31      redef record connection += {
32      ndpi: Info &optional;
33      };
34
35      event zeek_init() &priority=5
36      {
37      Log::create_stream(NDPI::LOG, [$columns=Info, $path="ndpi"]);
38      }
39
40      event connection_state_remove(c: connection)
41      {
42      if ( c?$ndpi )
43          Log::write(NDPI::LOG, c$ndpi);
44      }
45  }
```

Code 4.4: Creazione della Stream

Il codice 4.4 mostra i campi che vengono definiti per il log. Nello specifico i dati ricavati grazie ad nDPI sono i seguenti:

Protocollo applicativo, protocollo della connessione (e.g. whatsapp, youtube), categoria del protocollo (e.g. download, network), numero di pacchetti analizzati prima di ottenere un riscontro, connessione criptata o meno, nome dell'host server contattato, il tipo di rischio, il risk score.

Il resto dei campi vengono ricavati da Zeek e sono necessari al fine di ricollegare i dati di nDPI con i dati presenti nel resto dei log.

Ognuno di essi presenta l'attributo *optional* in quanto non è garantito che nDPI riesca ad individuare tutte le informazioni per ogni connessione. Il record creato viene aggiunto a quello già esistente *connection*. Come viene specificato nella documentazione di Zeek, si

sceglie di farlo in quanto permette a qualunque evento che abbia come argomento il record *connection* di accedere ai nuovi valori.
La nuova stream viene creata all'avvio di Zeek, così da non perdere alcuna informazione.
L'utilità di questi dati verrà approfondita nella sezione 5.

5. Validazione

L'obiettivo di questa sezione è quello di mostrare come poter utilizzare al meglio i dati estratti attraverso nDPI per analizzare il traffico e velocizzare la ricerca di minacce. Per farlo mettiamo a confronto un'ipotetica ricerca di attività malevole con e senza le informazioni aggiuntive. Vogliamo anche denotare quale sia l'utilizzo addizionale delle risorse introdotto da nDPI.
Il software è stato testato in un ambiente linux, in particolare su una macchina Ubuntu 23.10 con 2 core e 6GB di RAM.
Per quanto riguarda i dati utilizzati, si tratta di dataset facilmente reperibili in rete. Si è scelto di seguire questo approccio piuttosto che utilizzare traffico live in modo tale da avere già un riscontro su quale fosse il problema di un determinato traffico e, sapendo in anticipo quali indirizzi fossero quelli di interesse, mostrare al meglio l'utilità dei maggiori dettagli offerti attraverso l'utilizzo di nDPI. Nella sezione che segue cerchiamo di mostrare come un'analisi dei log prodotti da Zeek potrebbe verificarsi al fine di ricercare prove di un'avvenuta compromissione del sistema nel traffico raccolto o di svolgere uno studio per quanto riguarda le attività svolte in una rete.

5.1 Casi d'uso

Zeek, mano a mano che le connessioni del traffico monitorato terminano, scrive le informazioni ottenute nei log. Ispezionandoli è quindi possibile scoprire quali host si sono connessi alla/alle macchine monitorate, quale servizio stiano utilizzando. Una analisi generica, utilizzando Zeek con le proprie impostazioni predefinite, parte solitamente dal file *conn.log* dato che, come già detto, tiene traccia di tutte le connessioni stabilite con le macchine di interesse. In generale un'analista utilizza i log come supporto per rispondere ad una domanda prefissata e convalidare la propria ipotesi. La limitazione di Zeek per l'analisi del traffico rimane quella di non fornire supposizioni su potenziali minacce. In un

ambiente in cui le connessioni registrate sono migliaia, controllarne una ad una non è un opzione vagliabile. È quindi necessario l'appoggio di uno strumento che possa guidare nell'analisi.
Con l'aggiunta di nDPI, un'analisi generale che tenta di identificare se ci sono azioni sospette nel traffico, parte non più da conn.log (o da un qualsiasi altro file in caso di ricerche specifiche), ma dal nuovo file introdotto, ndpi.log. Questo permette di avere un riscontro immediato su quali connessioni presentano problemi in modo tale da analizzare nel dettaglio l'attività di tali host. In particolare una possibilità è quella di seguire le connessioni a cui è stato associato un rischio di qualsiasi tipo per vedere se sono collegate ad attività malevole ad esempio andando a controllare se l'indirizzo IP di un certo host sia presente in una qualche blacklist. L'esempio 5.1.1 mostra un utilizzo in questo specifico caso.
Successivamente, con l'esempio 5.1.2, ci concentreremo maggiormente sull'utilizzo delle informazioni ricavate da nDPI per poter tracciare un profilo per quanto riguarda le attività che vengono svolte in rete in uno scenario reale, escludendo in questo caso la presenza di intrusioni così da concentrarci maggiormente sulle applicazioni contattate dai vari host di una rete.

5.1.1 Ricerca di una intrusione

Per questo esempio useremo un dataset[1] contenente del traffico generato da un malware IcedID (anche noto come Bokbot). Si tratta di un trojan con l'obiettivo di rubare le credenziali di accesso bancarie salvate nei moduli di auto compilazione dei browser di navigazione Web o altre informazioni finanziarie. Inoltre può essere usato per trasportare altri malware. In questo caso l'infezione è avvenuta a causa del download di una applicazione, AnyDesk, attraverso un link infetto.
Questo tipo di attacco segue solitamente tre distinte fasi:

- Consegna iniziale del payload malevolo. Solitamente contenuto in un file .zip. Una volta estratto ed eseguito il file, solitamente un documento word, viene scaricato ed eseguito il primo file DDL.

[1] Google ads lead to fake software pages pushing IcedID, https://www.malware-traffic-analysis.net/2022/12/14/index.html

- il primo DDL si occupa di mandare informazioni al server riguardo all'ambiente vittima e scaricare il DDL successivo.

- nella terza fase viene ricevuto un secondo DDL ed il trojan effettivo che verrà caricato in memoria.

Successivamente all'esecuzione, che in questo caso è stata eseguita utilizzando in aggiunta uno script di Zeek con l'obiettivo di estrarre file dalle connessioni[2] se presenti, ci ritroviamo con i file mostrati in figura 5.1.

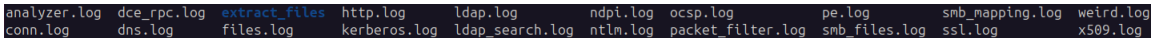


Figura 5.1: Log generati da Zeek

Partendo da ndpi.log, andiamo ad isolare i flussi che contengono un risk score. In questo esempio si è scelto di utilizzare il formato TSV per la scrittura su log piuttosto che il JSON, andremo quindi ad impiegare i vari tool per l'elaborazione del testo presenti in un ambiente Unix. In questo caso possiamo utilizzare lo strumento *awk*; sapendo che lo score minimo è 10 e che la colonna che identifica tale valore è l'ultima il comando 5.1 svolge il compito. In questo caso l'output prodotto viene inserito in un file in modo tale da poterlo riutilizzare in seguito senza alterare *ndpi.log*.

```
1 $ cat ndpi.log | awk '{ if ($(NF) >= 10) print }' &> file.txt
```

Code 5.1: Isolamento dei flussi con un risk score

Vediamo che tra i flussi rimanenti si ripetono per molte volte gli stessi indirizzi IP di destinazione, così come i server name.
Per aumentare la leggibilità possiamo andare ad eliminare quelle connessioni che presentano lo stesso indirizzo IP così come il protocollo dell'applicazione e il nome del server, gli ultimi due parametri vengono usati per evitare di eliminare servizi diversi ma con lo stesso indirizzo IP. Anche in questo caso possiamo utilizzare *awk*.

```
1 $ awk '!seen[$5,$9,$13]++' file.txt &> file.txt
```

[2]https://docs.zeek.org/en/master/frameworks/file-analysis.html#file-analysis-framework

Code 5.2: Eliminazione di connessioni con stesso IP destinazione, applicazione, name server

Osservando le connessioni così isolate, in particolar modo il protocollo estratto da nDPI, possiamo vedere che molte di queste vengono contrassegnati come Microsoft/Microsoft365. Inoltre notiamo la presenza di connessioni con protocollo SMB (nel caso specifico SMBv1, versione non più in uso di default in quanto presenta lacune di sicurezza benché possa ancora essere attivata su sistemi moderni), un protocollo per lo scambio di file e la comunicazione tra dispositivi di una rete principalmente usato da Windows. Questi elementi ci fanno supporre che il traffico sia stato registrato su una macchina Windows. Supponiamo di essere interessati a capire quali host esterni si siano connessi alla macchina che ha catturato il traffico e perché, escludendo le connessioni contrassegnate con il protocollo Microsoft (controllando ad esempio con VirusTotal [19] possiamo vedere che tali connessioni sono state effettivamente stabilite con server di Microsoft).
Eliminare le ultime connessioni citate può essere fatto sempre utilizzando un tool unix, *grep*, utile per andare ad evidenziare le linee, ad esempio di un file, che contengono o non contengono (come vediamo nell'esempio 5.3 in cui viene usata la flag -v) determinate parole.

```
1 $ grep -v "Microsoft" file.txt &> file.txt
```

Code 5.3: Eliminazione connessioni con Microsoft

Per mantenere soltanto gli IP pubblici andiamo ad utilizzare uno script Python per comodità.

```
1     import ipaddress
2     with open(file_path) as file:
3         counter = 0 # Necessario in quanto le prime 3 righe del file
4                     #  indicano il contenuto con relativi tipi
5         for line in file:
6             counter = counter + 1
7             arg = line.replace('\n','').split('\t')
```

```
8                 if counter > 3 and not ipaddress.ip_address(arg[4]).
    is_private:
9                     print(arg)
```

Code 5.4: Eliminazione IP privati

Il risultato ottenuto contiene 15 flussi. Essendo relativamente pochi possiamo pensare di riportare gli indirizzi su VirusTotal e vedere se qualcuno di questi rappresenta una potenziale minaccio. Di questi dieci sono stati riconosciuti come malevoli da almeno una compagnia di sicurezza. La traccia dell'attacco ci conferma che questi host sono esattamente quelli che hanno preso parte nell'attacco.

```
#fields ts      uid     id.orig_h       id.orig_p       id.resp_h       id.resp_p       l4_proto        proto   app_proto       category        num_packet_analyzed     encripted     h
ost_server      risk_type risk_score
1671043143.837796       CH7tAi2zf3NZ8doIik      172.17.5.135    49781   143.198.92.88   80      tcp     -       HTTP    Web     7       NO      klepdrafooip.com        * HTTP Susp Us
er-Agent *      100

1671043210.879701       CYYQ193IX6kBXzfGF6      172.17.5.135    49786   94.140.114.40   443     tcp     -       TLS     Web     7       YES     primsenetwolk.com       * Self-signed
Cert * TLS (probably) Not Carrying HTTPS *      110

1671043222.349917       CL5dxP17LWoaHf9a28      172.17.5.135    49794   94.140.114.40   443     tcp     -       TLS     Web     7       YES     onyxinnov.lol   * Self-signed Cert * T
LS (probably) Not Carrying HTTPS *      110

1671045063.008761       C0GN391dBuILoThgf8      172.17.5.135    49835   158.255.211.126 443     tcp     -       TLS     Web     7       YES     trashast.wiki   * Self-signed Cert * T
LS (probably) Not Carrying HTTPS *      110

1671049795.845585       CobNDB3eUImeCvETtc      172.17.5.135    49931   176.105.202.212 80      tcp     -       HTTP    Download        7       NO      176.105.202.212  * Binary App T
ransfer * HTTP Susp User-Agent * HTTP/TLS/QUIC Numeric Hostname/SNI *   260

1671049817.866298       Cqi8Hp24UbXNA4fnX3      172.17.5.135    49932   172.67.130.194  443     tcp     -       TLS     Web     11      YES     kingoflake.com  * TLS (probably) Not C
arrying HTTPS *         10

1671050199.836265       CfWePj111hALZRPir2      172.17.5.135    49950   199.127.62.132  80      tcp     -       HTTP    Download        7       NO      199.127.62.132   * Binary App T
ransfer * HTTP Susp User-Agent * HTTP/TLS/QUIC Numeric Hostname/SNI *   260

1671050209.320525       CifLsa17ONj4tSeFTl      172.17.5.135    49951   108.177.235.187 443     tcp     -       TLS     Web     3       YES     -       * TCP Connection Issues *    5
0

1671050209.901189       C3E4Ev26q085qGW9Bl      172.17.5.135    49951   108.177.235.187 443     tcp     -       TLS     Web     9       YES     bukifide.com    * TLS (probably) Not C
arrying HTTPS *         10

1671054189.628456       C1FuAR1dczj3bOlXul      172.17.5.135    53245   46.4.182.102    80      tcp     -       TLS     Web     7       YES     (empty)  * Known Proto on Non Std Port
* TLS (probably) Not Carrying HTTPS * Missing SNI TLS Extn *    110

1671054163.301643       C8cRtpxddKOUTjao1       172.17.5.135    53211   190.61.121.35   443     tcp     -       HTTP    Download        7       NO      190.61.121.35    * Binary App T
ransfer * Known Proto on Non Std Port * HTTP Susp User-Agent * HTTP/TLS/QUIC Numeric Hostname/SNI *     310
```

Figura 5.2: Connessioni potenzialmente legate ad un attacco

Partendo da queste possiamo utilizzare i vari log prodotti da Zeek per caratterizzare al meglio le connessioni ricavate.

Possiamo notare che il primo di questi flussi usa il protocollo HTTP, usiamo l'apposito log *http.log* per controllare l'attività di esso. Per cercarlo all'interno del file utilizziamo l'UID, l'identificatore unico del flusso.

```
#fields ts      uid     id.orig_h       id.orig_p       id.resp_h       id.resp_p       trans_depth     method  host    uri     referrer        version user_agent      origin  request
_body_len       response_body_len       status_code     status_msg      info_code       info_msg        tags    username        password        proxied orig_fuids      orig_filenameso
rig_mime_types  resp_fuids      resp_filenames  resp_mime_types
1671043144.243193       CH7tAi2zf3NZ8doIik      172.17.5.135    49781   143.198.92.88   80      1       GET     klepdrafooip.com        /       -       1.1     -       -       0     1
864269  200     OK      -       -       (empty) -       -       -       -       -       FoEq4H3atH4lNe6MT5      -       application/x-gzip
```

Figura 5.3: Dettagli HTTP prima connessione

Così facendo possiamo vedere che l'operazione effettuata è una **GET**, quindi sappiamo che è stato richiesto un file, il tipo MIME ci fa capire che si tratta di un file .tar.gz. Possiamo esaminare *file.log* e vedere se Zeek è riuscito ad estrarre l'elemento.

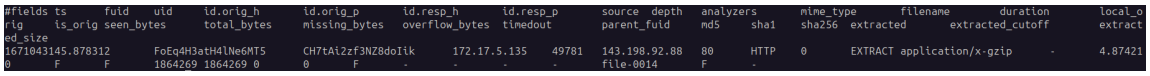


Figura 5.4: File estratto dalla prima connessione

Sempre utilizzando l'UID andiamo a cercare la linea corrispondente per scoprire che è stato estratto e che può essere trovarlo nella directory textitextracted_files/file-0014.
Il secondo flusso è una connessione TCP. nDPI ha riscontrato il rischio *Self signed Certificate*, andiamo quindi a controllare questo parametro. I certificati TLS risiedono in *x509.log*. Per accedere alla giusta linea abbiamo bisogno dell'identificatore del certificato che possiamo trovare in *ssl.log*. Tale file identifica le connessioni ancora una volta attraverso l'UID.

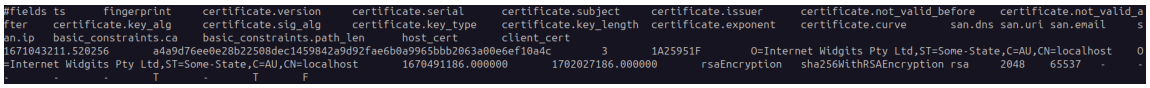


Figura 5.5: Certificato TLS della seconda connessione

L'entry di *x509.log* ci fornisce varie informazioni. In particolare possiamo osservare il parametro *Subject* per ottenere informazioni inerenti al server che è stato contattato per questa connessione. Notiamo che il certificato è stato emesso con il default-name *Internet Widgits PTy Ltd* da *Some-state*. Tale nome viene dato da OpenSSL quando crea un certificato ed è usato da varie famiglie di Malware come ad esempio *IcedID, AsyncRAT, DcRAT, Vawtrak, PhantomNet*. Un altro campanello d'allarme è rappresentato dal campo Common-Name, in questo caso *localhost* e non un FQDN (Fully Qualified Domain Name) come solitamente hanno i server pubblici.
Le due connessioni successive, sempre TLS, presentano lo stesso certificato. È comune per i Malware IcedID contattare il server C2 [20] a seguito dell'infezione in attesa di istruzioni da eseguire.

Vediamo che ci sono altre tre connessioni HTTP, ognuna delle quali ha scaricato un file. Ritroviamo quindi il pattern tipico degli attacchi IcedID.
Questo breve esempio mostra come le informazioni salvate da Zeek siano un valido alleato nella ricerca di tracce di intrusioni. Come abbiamo visto in questo caso possiamo trovare certificati, file estratti, server di provenienza e altri metadati non utilizzati in questo esempio. Come si è mostrato, tuttavia, senza una direzione prestabilita rimane difficile districarsi tra tutte le connessioni visualizzate che in caso di reti più grandi possono diventare diversi GB al giorno. I metadati aggiunti da nDPI possono essere un aiuto fondamentale per poter attuare una ricerca di tracce di compromissione.

5.1.2 Visibilità di rete

Questo secondo esempio pone maggiore attenzione sui protocolli che nDPI riesce a riconoscere, mettendoli a confronto con quelli che Zeek analizza nella sua configurazione standard. Il dataset utilizzato[3] [26] contiene del traffico che è stato appositamente generato basandosi su attività reali quotidiane in un ambiente domestico. Abbiamo quindi un insieme di informazioni che potremmo ritrovare in contesti reali di tutti i giorni.
Parlando del numero di diversi protocolli individuati sia da Zeek che da nDPI, abbiamo 16 protocolli del primo contro gli 89 riscontrati dal secondo. Inoltre nDPI offre una classificazione dei protocolli (Download, Email, Streaming, Chat) e per questo esempio troviamo 23 distinti sottoinsiemi, riportati nella tabella 5.1.
La maggior parte delle connessioni stabilite fa parte della categoria Network. Qui troviamo i protocolli DNS e OCSP. In successione appare la categoria Web, che comprende oltre a vari Motori di ricerca, Google e Yahoo in questo dataset, connessioni per il quale nDPI non è riuscito a rilevare una applicazione. Non sorprende che Advertisement rappresenti buona parte delle connessioni stabilite in quanto hanno assunto un ruolo preponderante quando parliamo di navigazione in rete. Grazie alla traccia che descrive questo dataset conosciamo dei dettagli sulla topologia della rete da cui è stato catturato questo traffico, sappiamo infatti che sono presenti due server. Il traffico dei gruppi Cloud, RemoteAccess appartengono principalmente all'accesso di questi, oltre che a terminali che hanno con-

[3] https://www.unb.ca/cic/datasets/ids-2017.html

Categoria	Num. connessioni	Durata media in sec
Network	234355	2.399
Web	99497	30.584
Advertisement	20057	47.828
System	10578	10.7
SocialNetwork	3129	74.478
Cloud	2755	30.059
Media	1356	79.038
RemoteAccess	1060	1.339
Download	572	15.96
Email	367	1.66
Shopping	310	68.148
ConnCheck	193	133.663
RPC	165	24.9
SoftwareUpdate	71	18.549
Music	69	33.029
AdultContent	59	36
Game	45	59.422
Streaming	34	86.735
VoIP	32	109.125
Video	19	53,632
VPN	18	10.722
Chat	12	23.75

Tabella 5.1: Categorie delle connessioni riscontrate

tattato il servizio di AmazonAWS o server Microsoft. Tra i download effettuati troviamo principalmente file trasferiti da uno dei server e download di aggiornamenti di windows. Troviamo anche file scaricati attraverso Torrent o da Pastebin, oltre a varie connessioni

con il protocollo ADS_Analytic_Track, quindi inerenti sempre ad ADS in questo caso prevalentemente di Google.
Esclusi quei processi per cui un utente non ha diretto controllo, quindi aggiornamenti di sistema, richieste DNS o altri protocolli raggruppati come ConnCheck (si tratta in questo caso di controlli svolti dal browser o dal sistema operativo per quanto riguarda la possibilità di connettersi ad un certo servizio), vediamo che le connessioni che hanno durata maggiore sono quelle inerenti all'intrattenimento, come social networks, servizi di streaming o giochi. Grazie ad nDPI riusciamo a sapere facilmente quali sono ad esempio i social network utilizzati su questa rete, mostrati nell'immagine 5.6.

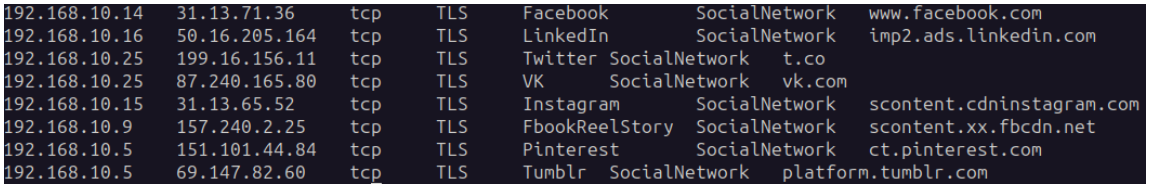


Figura 5.6: Social Network utilizzati

L'utilizzo di un DPI fornisce una maggior chiarezza quando parliamo di azioni che vengono svolte sulla rete, non solo attinenti ad intrusioni come visto in questo esempio. Risulta infatti semplice comprendere quali applicazioni vengono contattate con più frequenza. Una possibile applicazione può essere quella di distribuire più banda a quei servizi che vengono contattati maggiormente, in questo caso si potrebbe pensare di avvantaggiare servizi di streaming, oppure fornire maggiori risorse ad aggiornamenti di sistema nel caso in cui non fossero presenti connessioni di una determinata categoria.

5.2 Performance

Con l'aggiunta di nDPI il carico sulla CPU e la memoria necessari all'esecuzione aumenta. Decidere se questo aumento sia significativo o meno rispetto ai vantaggi portati spetta al singolo. Nel corso di questa sezione andremo soltanto a mostrare quanto sia questo aumento in modo che ognuno possa trarre le proprie conclusioni in maniera autonoma.
Zeek offre vari strumenti per poter misurare le risorse che utilizza. Offre una serie di

script nel linguaggio Zeek che possono essere usati per estrarre alcune metriche inerenti alle performance.

È bene notare che, come già precedentemente detto, nDPI viene chiamato per ogni pacchetto del flusso, ma solamente fino a quando non avrà ottenuto un riscontro del protocollo dell'applicazione o fino al raggiungimento del numero massimo di tentativi. Al contrario, gli analyzer di Zeek, vengono invocati ad ogni pacchetto del flusso qualsiasi siano le informazioni rilevate.

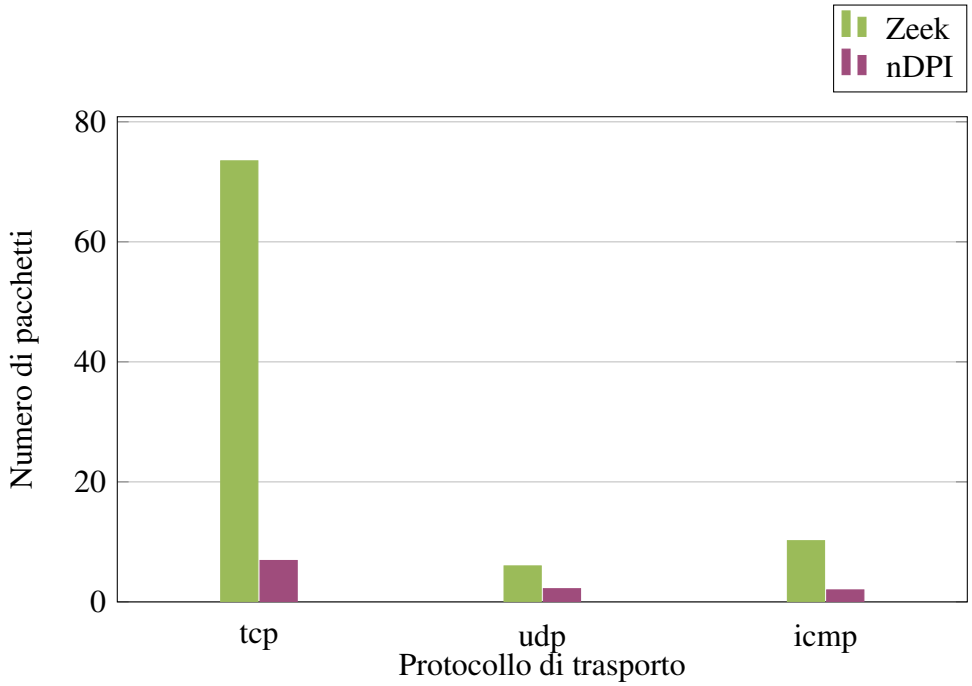


Figura 5.7: Pacchetti analizzati in media da nDPI

Il grafico 5.7 mostra il numero medio dei pacchetti per ogni connessione riconosciuta da Zeek messi a confronto con il numero di pacchetti che nDPI impiega in media per riconoscere un'applicazione. Da notare che Zeek considera ICMP come un protocollo di livello 4. Vediamo chiaramente come nDPI riesca a riconoscere l'applicazione di una connessione TCP con circa 7 pacchetti in media, mentre ne impiega circa 2 per connessioni sia UDP che ICMP.

5.2.1 Uso della memoria

Per quanto concerne l'uso della memoria, possiamo utilizzare appositi tool esterni che si occupano della profilazione della stessa. Uno strumento terzo per la gestione della memoria supportato da Zeek è jemalloc [12], un'implementazione di malloc che enfatizza il supporto scalabile della concorrenza e tenta di evitare la frammentazione. Destinato all'uso come allocatore di memoria fornito dal sistema, fornisce molte funzionalità di introspezione, gestione e ottimizzazione della memoria oltre alla funzionalità di allocazione standard. In questo caso è stato usato per confrontare le prestazioni e l'utilizzo della memoria di Zeek nella sua configurazione standard e con l'implementazione di nDPI, consentendoci di identificare eventuali differenze significative nell'allocazione e nell'uso della memoria durante l'analisi del traffico di rete.

Per il supporto all'interno di Zeek è sufficiente, oltre a scaricare il Software, utilizzare la flag *–enable-jemalloc* durante la configurazione.

Fatto ciò sarà possibile eseguire Zeek con l'opzione *MALLOC_CONF* che scaricherà la profilazioni della memoria ogni 256MB allocati (la scrittura viene controllata dall'impostazione *lg_prof_interval*: $2^{28} = 256MB$)

```
1 $ MALLOC_CONF="prof:true,prof_prefix:jeprof.out,prof_final:true,
    lg_prof_interval:28" zeek -C -i eth0
```

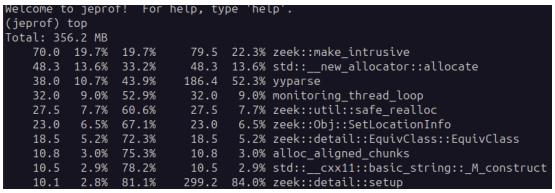
Code 5.5: Avvio di Zeek con il supporto di Jemalloc

I file generati da Jemalloc avranno una denominazione del tipo $jeprof.out.\langle pid \rangle ...$ e possono essere processati con l'utility *jeprof*. Possiamo utilizzare il comando 5.6 per accedere alla shell di controllo del tool. In questo modo andremo a processare tutti i file prodotti durante l'esecuzione.

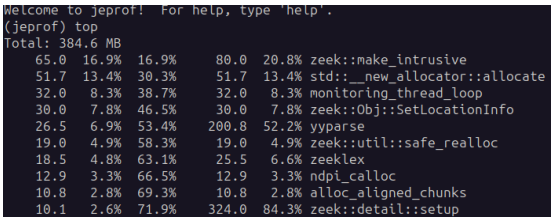
```
1 $ jeprof $(which zeek) jeprof.out.*
```

Code 5.6: Uso dell'utility jeprof

Eseguendo i comandi sopra con il dataset mostrato nell'esempio 5.1.1 il risultato è mostrato di seguito.



(a) Uso della memoria di Zeek



(b) Uso della memoria di Zeek con nDPI

Figura 5.8: Confronto riguardo l'utilizzo della memoria

In questo specifico caso, dopo una singola esecuzione, otteniamo un incremento di circa l'8% passando dai 356.2 MB ai 384.6 MB allocati. L'immagine 5.8 mostra come le chiamate all'API di nDPI costituiscano buona parte della memoria allocata in più a seguito delle modifiche, in questo caso parliamo di 12.9 MB.

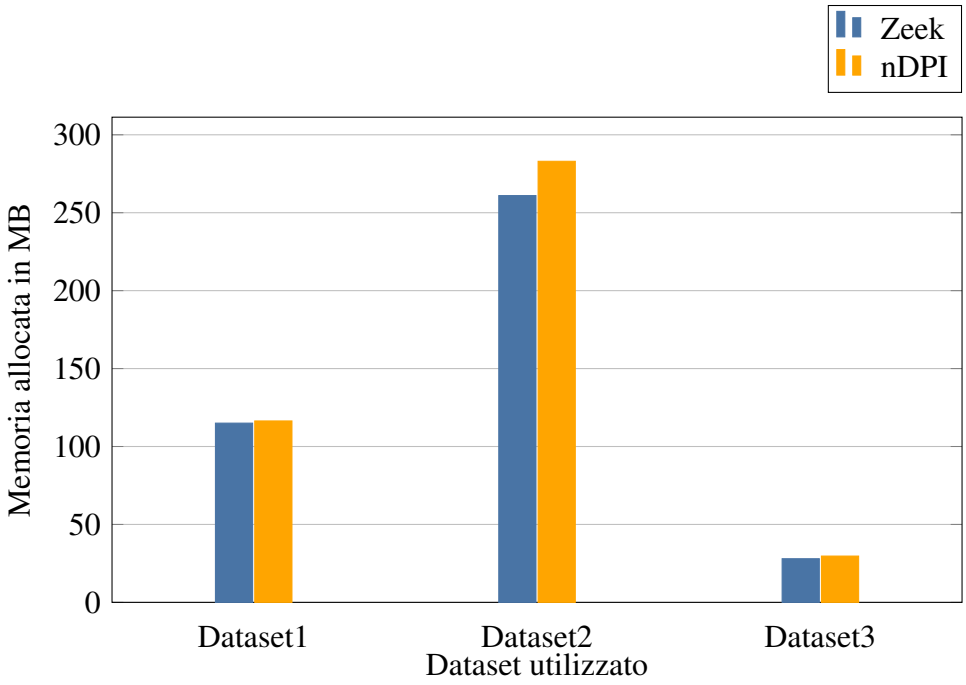


Figura 5.9: Memoria allocata da Zeek con e senza nDPI

Per avere una stima più veritiera i test sopra sono stati eseguiti con più dataset, sempre reperibili pubblicamente on-line, per più volte per ogni dataset, ottenendo un aumento

dell'allocazione media della memoria di circa il 6% maggiore quando viene utilizzato nDPI. La figura 5.9 mostra l'allocazione necessaria per tre diversi dataset che sono stati utilizzati.

5.2.2 Profiling della CPU

Oltre alla memoria un altro fattore determinante è rappresentato dall'utilizzo della CPU, nonché dal tempo di esecuzione del programma. Per raccogliere tali informazioni si è scelto di utilizzare uno script python che tracciasse l'uso della CPU ogni mezzo secondo.

```
1    import psutil
2    import subprocess
3
4    zeek_process = subprocess.Popen(['zeek', '-r', 'file.pcap'])
5
6    def calculate_cpu_usage():
7        while zeek_process.poll() is None:
8            cpu_percent = psutil.cpu_percent(interval=0.5, percpu=
   False)
9            cpu_data.append(cpu_percent)
10   calculate_cpu_usage()
```

Code 5.7: Monitoraggio uso della CPU

Per questa analisi è stato utilizzato il dataset visto nell'esempio 5.1.2. Si tratta di un file di oltre 10GB che può quindi darci un'idea delle risorse richieste per analizzare il traffico di un generico giorno.

Osservando la figura 5.10 si può notare che il programma esteso con nDPI ha un picco di utilizzo a circa metà dell'esecuzione e che il tempo di esecuzione è maggiore. La media del carico della CPU in questo caso passa da circa il 28.9% a circa il 30.7%. Riscontriamo quindi un aumento dell'utilizzo medio di quasi il 10%. Per quanto riguarda il tempo di esecuzione passiamo da 6 minuti e 10 secondi nella configurazione standard a 6 minuti e 30 secondi nella versione estesa con nDPI, un aumento del 3.6%.

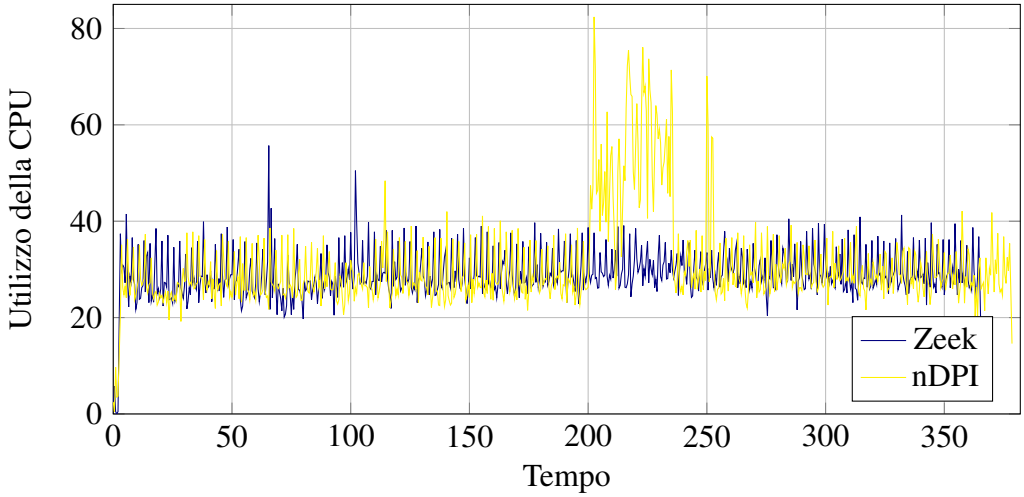


Figura 5.10: Utilizzo della CPU di Zeek nel tempo

6. Conclusioni

Il lavoro svolto è servito a proporre un prodotto presente nello scenario della sicurezza di rete, o quanto meno non abbiamo trovato programmi mainstream che utilizzassero tecniche di DPI all'interno di un IDS escludendo tecniche rudimentali come quella citata nel capitolo 3.

Nel corso dei capitoli è stata approfondita l'integrazione di nDPI all'interno di Zeek, un software di monitoraggio di rete utilizzato per il monitoraggio in tempo reale. Ciò ha successivamente permesso di analizzare le differenze che sussistono quando viene deciso di utilizzare un DPI per l'analisi del traffico. Attraverso l'esecuzione di Zeek sia nella sua forma standard che estesa con nDPI, sono emerse differenze significative nella capacità di rilevamento e nell'analisi del traffico di rete.

Zeek, nella sua forma standard, si è dimostrato utile in quanto permette di salvare tutte le connessioni in modo da poter svolgere analisi in qualsiasi momento o mandando i dati raccolti a software specifici, così come nella possibilità di utilizzare script a tempo di esecuzione per tenere sotto controllo eventi di interesse. Benché riesca a rilevare e classificare i protocolli più comuni, la sua capacità di identificare protocolli applicativi specifici e analizzare il contenuto dei pacchetti è limitata.

L'integrazione di nDPI in Zeek ha apportato miglioramenti significativi in termini di rilevamento e classificazione dei protocolli di rete. Grazie alla libreria nDPI, Zeek è ora in grado di identificare un'ampia varietà di protocolli applicativi, compresi quelli meno comuni. Questo consente agli analisti di rete di ottenere una visione più approfondita del traffico di rete e di individuare potenziali minacce in modo più preciso.

Tuttavia, l'integrazione ha introdotto alcuni compromessi sulle prestazioni del sistema. L'analisi approfondita del traffico richiede più risorse computazionali e può comportare un aumento dell'utilizzo della CPU e della memoria. Questo, come visto nella sezione 5.2, è minimizzato dal fatto che l'analisi non viene svolta per ogni pacchetto della connessione ma solamente fino ad ottenere un riscontro o fino al raggiungimento del limite superiore di iterazioni. Come visto, il protocollo viene acquisito dopo pochi pacchetti,

indipendentemente dalla durata della connessione.
In sintesi, l'integrazione di nDPI in Zeek offre un vantaggio significativo nell'analisi approfondita del traffico di rete, consentendo agli analisti di ottenere una comprensione più dettagliata delle attività di rete.

6.1 Lavori futuri

Parlando di possibili sviluppi che possono partire a seguito di questo lavoro, è giusto sottolineare la possibilità di integrare le modifiche apportate nel progetto originale in modo tale che questa versione possa essere di facile accesso a più persone possibili. Come già detto questo richiederebbe la riscrittura del codice così da rispettare le linee guida, il che consiste nell'utilizzo di una classe apposita anziché le modifiche del codice sorgente della classe Conn. Tali modifiche, come descritto nella sezione 3.1, avrebbero richiesto un maggiore studio per l'integrazione non portando ad un reale vantaggio in termini di efficienza.
Altri possibili sviluppi riguardano l'inserimento del prodotto perfezionato in tool esistenti. Un esempio è rappresentato da Security Onion [14], uno strumento per il monitoraggio delle reti che già utilizza Zeek, in particolar modo i log prodotti vengono analizzati per comprendere le attività della rete, e che potrebbe trarre vantaggio dalla classificazione minuziosa che offre nDPI.
Allo stesso modo sarebbe interessante svolgere un confronto in termini di performance tra lo strumento sviluppato e altri IDS presenti sul mercato, non necessariamente Open-Source come Suricata o Snort, in cui venisse integrato nDPI come è stato fatto in questo elaborato.

7. Ringraziamenti

Alle persone che ci sono state durante questi anni, a quelle che si sono aggiunte durante questo percorso, a quelle che non lo hanno visto concludere, a me che l'ho portato a termine. Grazie.

Non ci saranno nomi, non ci saranno riferimenti. Chi sta leggendo questo frammento sa di essere stato fondamentale, sia che lo stia leggendo il giorno della mia laurea, sia a distanza di anni. Una risata tra una linea di codice e l'altra, una chiamata di incoraggiamento prima di un esame o postuma per le congratulazioni, un supporto in un momento in cui magari pensavo di mollare, di non riuscire a finire ciò che avevo iniziato, un dungeon dopo cena ("qualcuno fermi quest'uomo"), un aperitivo per distrarsi, una foto di Tigro ricevuta, una chiacchierata dopo cena, un massimale di stacco in compagnia, un panino a Pontedera a seguito dell'esame più stressante di questi anni. Ogni momento di questi anni è stato intenso e sono grato di averli vissuti dal primo all'ultimo.

Ho iniziato pieno di paure, di dubbi e non credevo di arrivare un giorno a scrivere questi ringraziamenti carico di emozioni. Ho avuto modo di perdermi, di ritrovarmi, di piangere, di sorridere. Per la prima volta in molto tempo sono soddisfatto di dove sono arrivato. È giunto il momento di porre un punto su questa avventura che tanto mi ha fatto penare e altrettanto mi ha insegnato, che sia un punto di partenza per continuare a costruire.

Sei forte.

Bibliografia

[1] Eugene Albin and Neil C. Rowe. A realistic experimental comparison of the suricata and snort intrusion-detection systems. In *2012 26th International Conference on Advanced Information Networking and Applications Workshops*, pages 122–127, 2012.

[2] Fatimah Aldauiji, Omar Batarfi, and Manal Bayousef. Utilizing cyber threat hunting techniques to find ransomware attacks: A survey of the state of the art. *IEEE Access*, 10:61695–61706, 2022.

[3] Sandeep Bhatt, Pratyusa K. Manadhata, and Loai Zomlot. The operational role of security information and event management systems. *IEEE Security & Privacy*, 12(5):35–41, 2014.

[4] Tomasz Bujlow, Valentín Carela-Español, and Pere Barlet-Ros. Independent comparison of popular dpi tools for traffic classification. *Computer Networks*, 76:75–89, 2015.

[5] V. Corey, C. Peterman, S. Shearin, M.S. Greenberg, and J. Van Bokkelen. Network forensics analysis. *IEEE Internet Computing*, 6(6):60–66, 2002.

[6] Eric H. Corwin. Deep packet inspection: Shaping the internet and the implications on privacy and security. *Information Security Journal: A Global Perspective*, 20(6):311–316, 2011.

[7] Luca Deri and Francesco Fusco. Using deep packet inspection in cybertraffic analysis. In *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 89–94, 2021.

[8] Luca Deri, Maurizio Martinelli, Tomasz Bujlow, and Alfredo Cardigliano. ndpi: Open-source high-speed deep packet inspection. In *2014 International Wire-*

less Communications and Mobile Computing Conference (IWCMC), pages 617–622, 2014.

[9] Henrique Dornel, EDS Christo, Kelly Alonso Costa, and DPM Souza. Traffic forecasting for monitoring in computer networks using time series. *Int. J. Adv. Eng. Res. Sci*, 6(7), 2019.

[10] Vitor Duarte and Nuno Farruca. Using libpcap for monitoring distributed applications. In *2010 International Conference on High Performance Computing & Simulation*, pages 92–97, 2010.

[11] Reham Taher El-Maghraby, Nada Mostafa Abd Elazim, and Ayman M. Bahaa-Eldin. A survey on deep packet inspection. In *2017 12th International Conference on Computer Engineering and Systems (ICCES)*, pages 188–197, 2017.

[12] Jason Evans. Tick tock, malloc needs a clock. In *Applicative 2015*, Applicative 2015, New York, NY, USA, 2015. Association for Computing Machinery.

[13] Graham Finnie. The role of dpi in an sdn world. *White paper, Heavy Reading*, 2012.

[14] Ross Heenan and Naghmeh Moradpoor. Introduction to security onion. 2016.

[15] VVRPV Jyothsna, Rama Prasad, and K Munivara Prasad. A review of anomaly based intrusion detection systems. *International Journal of Computer Applications*, 28(7):26–35, 2011.

[16] Hyundo Kim, Minsu Kim, Joonseo Ha, and Heejun Roh. Revisiting tls-encrypted traffic fingerprinting methods for malware family classification. In *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*, pages 1273–1278, 2022.

[17] Andreas Kuehn and Milton Mueller. Profiling the profilers: deep packet inspection and behavioral advertising in europe and the united states. *Available at SSRN 2014181*, 2012.

[18] Vinod Kumar and Om Prakash Sangwan. Signature based intrusion detection system using snort. *International Journal of Computer Applications & Information Technology*, 1(3):35–41, 2012.

[19] Christian Leka, Christoforos Ntantogian, Stylianos Karagiannis, Emmanouil Magkos, and Vassilios S. Verykios. A comparative analysis of virustotal and desktop antivirus detection capabilities. In *2022 13th International Conference on Information, Intelligence, Systems & Applications (IISA)*, pages 1–6, 2022.

[20] Zeyu Li, Meiqi Wang, Xuebin Wang, Jinqiao Shi, Kexin Zou, and Majing Su. Identification domain fronting traffic for revealing obfuscated c2 communications. In *2021 IEEE Sixth International Conference on Data Science in Cyberspace (DSC)*, pages 91–98, 2021.

[21] Hung-Jen Liao, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16–24, 2013.

[22] Eva Papadogiannaki and Sotiris Ioannidis. A survey on encrypted network traffic analysis applications, techniques, and countermeasures. *ACM Comput. Surv.*, 54(6), jul 2021.

[23] Sara Qamar, Zahid Anwar, Mohammad Ashiqur Rahman, Ehab Al-Shaer, and Bei-Tseng Chu. Data-driven analytics for cyber-threat intelligence and information sharing. *Computers & Security*, 67:35–58, 2017.

[24] Qais Qassim, Abdullah Mohd Zin, and Mohd Juzaiddin Ab Aziz. Anomalies classification approach for network-based intrusion detection system. *Int. J. Netw. Secur.*, 18(6):1159–1172, 2016.

[25] Khamar Ali Shaikh, A Karthik Bhat, and Minal Moharir. A survey on ssl packet structure. In *2017 2nd International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS)*, pages 1–5, 2017.

[26] Iman Sharafaldin, Arash Habibi Lashkari, Ali A Ghorbani, et al. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, 1:108–116, 2018.

[27] Chaofan Shen and Leijun Huang. On detection accuracy of l7-filter and opendpi. In *2012 Third International Conference on Networking and Distributed Computing*, pages 119–123, 2012.

[28] Gulshan Shrivastava. Network forensics: Methodical literature review. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 2203–2208, 2016.

[29] Robin Sommer. Bro: An open source network intrusion detection system. In *Security, E-learning, E-Services, 17. DFN-Arbeitstagung über Kommunikationsnetze*, pages 273–288. Gesellschaft für Informatik e.V., Bonn, 2003.

[30] Wenguang Song, Mykola Beshley, Krzysztof Przystupa, Halyna Beshley, Orest Kochan, Andrii Pryslupskyi, Daniel Pieniak, and Jun Su. A software deep packet inspection system for network traffic analysis and anomaly detection. *Sensors*, 20(6):1637, 2020.

[31] Zhengzhi Tang, Xuewen Zeng, and Yiqiang Sheng. Entropy-based feature extraction algorithm for encrypted and non-encrypted compressed traffic classification. *Int. J. Innov. Comput. Inf. Control*, 15:845–860, 2019.

[32] Pere Barlet-Ros Tomasz Bujlow, Valentín Carela-Español. Independent comparison of popular dpi tools for traffic classification. 76(0):75–89, 2015.

[33] Pang-Wei Tsai, Chun-Wei Tsai, Chia-Wei Hsu, and Chu-Sing Yang. Network monitoring in software-defined networking: A review. *IEEE Systems Journal*, 12(4):3958–3969, 2018.

[34] Toshihide Tsuzuki. High-availability and low-latency tap for network monitoring. *FUJITSU Sci. Tech. J*, 49(3):370–373, 2013.

[35] A. Vakali and G. Pallis. Content delivery networks: status and trends. *IEEE Internet Computing*, 7(6):68–74, 2003.

[36] L. Vokorokos and A. Baláž. Host-based intrusion detection system. In *2010 IEEE 14th International Conference on Intelligent Engineering Systems*, pages 43–47, 2010.

[37] Zongjian Wang and Xiaobo Li. Intrusion prevention system design. In Zhicai Zhong, editor, *Proceedings of the International Conference on Information Engineering and Applications (IEA) 2012*, pages 375–382, London, 2013. Springer London.

[38] Jian Zhang and Andrew Moore. Traffic trace artifacts due to monitoring via port mirroring. In *2007 Workshop on End-to-End Monitoring Techniques and Services*, pages 1–8, 2007.