Alla mia famiglia,
che mi ha sempre sostenuto
in ogni mia scelta.

Introduzione

Questa è l'introduzione.

Indice

Introduzione **i**

1 Lo stato dell'Arte **1**

1.1 Intrusion Detection System 1

1.1.1 Suricata, una breve introduzione 4

1.2 sFlow 8

1.2.1 Il campionamento 9

1.2.2 Tools per l'utilizzo di sFlow 11

1.3 Packet sampling e Intrusion Detection 13

1.3.1 Metodi di rilevazione flow-based 14

1.4 Elasticsearch e lo stack ELK 17

Conclusioni **19**

A Prima Appendice **21**

B Seconda Appendice **23**

Bibliografia **25**

Elenco delle figure

1.1 Funzionamento del pattern matching 6
1.2 Datagram sFlow 10
1.3 13
1.4 Lo stack ELK 18

Elenco delle tabelle

Capitolo 1

Lo stato dell'Arte

In questo capitolo si va ad illustrare lo stato dell'Arte delle tecnologie utilizzate. Si illustreranno le principali qualità degli Intrusion Detection Systems e le caratteristiche principali che hanno portato durante i test alla scelta di un software rispetto che un altro. Si passerà poi a presentare sFlow, illustrandone i benefici e le principali differenze con NetFlow e di come esso viene attualmente utilizzato per affiancare un IDS in reti molto estese e complesse. Infine si darà una breve presentazione dello stack ELK, (Elasticsearch-Logstash-Kibana) e di come esso sia utilizzato nell'ambito della Network Security.

1.1 Intrusion Detection System

Un Intrusion Detection System (IDS) [4] è un dispositivo o un' applicazione software che monitora una rete o un sistema per rilevare eventuali attività dannose o violazioni delle policy. Qualsiasi attività o violazione rilevata viene in genere segnalata ad un amministratore o raccolta a livello centrale utilizzando un Security Information and Event Management (SIEM). Un SIEM combina output provenienti da più sorgenti e utilizza tecniche di filtraggio degli allarmi per distinguere le attività dannose dai falsi allarmi.

Esiste un' ampia gamma di IDS, che varia dal software antivirus fino ai sistemi gerarchici che controllano il traffico di un' intera backbone. La classificazione più comune è tra:

- **Network-based Intrusion Detection Firmware (NIDS)**: ad esempio un sistema che monitora il traffico di rete passante attraverso alcuni punti strategici di una rete. Esempi famosi sono: Suricata [6] , Snort [7] e BRO [8] .

- **Host-based Intrusion Detection (HIDS)** : ad esempio un software che monitora alcuni file importanti del sistema operativo su cui è installato. Un esempio famoso di HIDS è OSSec [5]

Il panorama degli Intrusion Detection System (IDS) è al giorno d'oggi in continua evoluzione. Tuttavia è possibile operare una seconda e importante classificazione in base a due criteri principali che ne determinano il funzionamento:

- Sistemi *signature-based detection*

- Sistemi *anomaly-based detection*

Un IDS *signature-based* analizza i pacchetti passanti su una rete utilizzando il concetto di *signature*: Una signature è un pattern che corrisponde ad un tipo di attacco noto. [7] Esempi di signature possono essere:

- un tentativo di connessione a TELNET con username "root", che corrisponde ad una violazione delle policy di sicurezza

- un email con oggetto "Immagini gratis!" e un allegato con nome "freepics.exe", che sono caratteristici di un attacco nostro

- tentativi ripetuti nel tempo di connessione SSH ad intervalli sospetti, che identificano un possibile attacco bruteforce su SSH.

Il rilevamento signature-based è molto efficace nel rilevare minacce note, ma in gran parte inefficace nel rilevare minacce precedentemente sconosciute, minacce mascherate dall'uso di tecniche di evasione e molte varianti di minacce note. Per esempio, se un aggressore ha modificato il malware nell' esempio precedente per usare un nome file di "freepics2.exe", una firma che cercava "freepics.exe" non lo corrisponderebbe. Il rilevamento basato sulla firma è il metodo di rilevamento più semplice in quanto confronta il campione corrente, come un pacchetto o una voce di registro, con un elenco di firme utilizzando operazioni di confronto tra stringhe.

Un IDS che usa *anomaly-based* detection, utilizza il concetto di anomalia: ovvero una deviazione del comportamento della rete osservato al momento da quello che è considerato normale in base a quanto osservato in precedenza. Un IDS che utilizza un rilevamento anomaly-based ha profili che rappresentano il comportamento normale di utenti, host, connessioni di rete o applicazioni. I profili sono sviluppati monitorando le caratteristiche dell'attività tipica per un periodo di tempo. Ad esempio, un profilo di una rete potrebbe indicare che l' attività Web comprende in media il 13% della larghezza di banda della rete al confine Internet durante le normali ore di lavoro giornaliere. L' IDS utilizza quindi metodi statistici per confrontare le caratteristiche dell'attività corrente con le soglie relative al profilo, ad esempio rilevando quando l'attività Web comprende una larghezza di banda significativamente maggiore del previsto e avvisando un amministratore dell'anomalia. L' IDS inoltre potrebbe utilizzare tecniche di intelligenza artificiale per determinare se un comportamento della rete sia da ritenersi normale o anomalo.

Sebbene nell'ultimo periodo l'intelligenza artificiale stia facendo la sua comparsa in ogni ambito dell'informatica gli IDS signature based rappresentano tuttora un'importante fetta (se non la maggioranza) degli IDS in uso nei più importanti data center del mondo ed è per questo che vale la pena studiarli.

In questo elaborato ci si focalizzerà sugli IDS signature based e se ne analizzeranno le loro prestazioni combinate ad altre tecnologie che verranno

introdotte in seguito.

Come anticipato sopra, tra i maggiori esponenti degli IDS attualmente utilizzati abbiamo:

- Snort: Un IDS sviluppato a partire dagli anni '90, acquisito da Cisco nel 2013 e che è tuttora il più utilizzato in ambito enterprise.
- Suricata: Un IDS del nuovo millennio, sviluppato a partire dal 2009 da Open Information Security Foundation (OISF) e che vanta molteplici vantaggi sopra gli altri IDS.

In questo elaborato si è preferito utilizzare per motivi di performance e di implementazione, Suricata. I dettagli di questa scelta saranno chiari più avanti quando saranno state introdotte le principali caratteristiche di Suricata.

1.1.1 Suricata, una breve introduzione

Suricata è un IDS Open Source sviluppato da OISF che fa uso di pattern matching per il riconoscimento di *threat*, violazioni della policy e comportamenti malevoli. Esso è inoltre capace di rilevare numerose anomalie nel protocollo all'interno dei pacchetti ispezionati. Le anomalie rilevabili, tuttavia, sono diverse da quelle degli IDS anomaly-based citati sopra. Le prime infatti sono scostamenti dall'utilizzo lecito di protocolli ben definiti e standardizzati : *Ad esempio: richieste DNS che non sono destinate alla porta 53, oppure richieste HTTP con un header malformato.* Il secondo tipo di anomalia invece è relativo ad un deviazione dal comportamento standard della specifica rete su cui l'IDS è stato tarato, ed è quindi un concetto di anomalia molto più lasco.

Deep Packet Inspection

La rilevazione di queste anomalie in Suricata va sotto il nome di *Deep Packet Inspection* o *Stateful Protocol Analysis*, ovvero il processo di confronto

di determinati comportamenti accettati dal signolo protocollo (HTTP, FTP, SSH, ecc...) con il comportamento osservato al momento del campionamento. Se un IDS usa questa tecnica vuol dire che esso è in grado di comprendere e monitorare lo stato della rete, del trasporto e dei protocolli applicativi che possiedono una nozione di stato. Ad esempio, quando un utente avvia una sessione del File Transfer Protocol (FTP), la sessione si trova inizialmente nello stato non autenticato. Gli utenti non autenticati devono eseguire solo alcuni comandi in questo stato, come la visualizzazione delle informazioni della guida in linea o la fornitura di nomi utente e password. Inoltre una parte importante dello stato di comprensione è l' accoppiamento delle richieste con le risposte, quindi quando si verifica un tentativo di autenticazione FTP, l' IDS può determinare se il tentativo è riuscito trovando il codice di stato nella risposta corrispondente. Una volta che l' utente si è autenticato con successo, la sessione si trova nello stato autenticato e agli utenti è permesso eseguire una qualsiasi delle decine di comandi disponibili. Mentre eseguire la maggior parte di questi comandi mentre sono in stato non autenticato sarebbe considerato sospettoso.

Prestazioni

Una singola istanza di Suricata è capace di ispezionare traffico proveniente da una rete multi-gigabit, questo grazie al forte paradigma multi thread utilizzato nel core del programma. Inoltre esso dispone di un supporto nativo per l'accelerazione hardware, l'analisi di pacchetti con GPUs e supporta nativamente PF_RING [9] e PF_PACKET, due tipologie di socket altamente performanti.

Pattern Matching

Il Pattern Matching è il processo di confronto del pacchetto osservato sulla rete è una signature salvata all'interno di quelle che vengono definite *rules*. Il suo funzionamento può essere riassunto dalla figura 1.1. [2] Ogni pacchetto passante sull'interfaccia di rete monitorata dall'IDS viene quindi

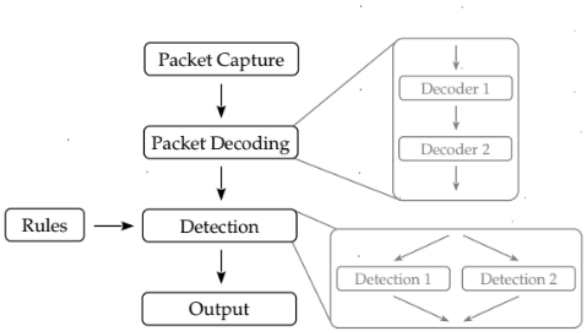

Figura 1.1: Funzionamento del pattern matching

decodificato e poi analizzato parallelamente per riscontrare similitudini con più pattern (rules). Una tipica regola per il suddetto patter matching è nella forma seguente:

```
alert tcp 1.1.1.1 8909 -> 192.168.1.0/24 80
```

Viene quindi indicata l'azione da intraprendere (in questo caso 'alert'), il protocollo , indirizzo/i di sorgente, porta/e sorgente, indirizzo/i di destinazione e porta/e di destinazione del pacchetto con cui fare match. Queste regole possono essere personalizzate oppure è possibile scaricarne di già confezionate da molteplici siti che offrono questo servizio. La praticità dell'utilizzare questa specifica sintassi sta nel fatto che essa è quasi del tutto identica a quella di Snort. Per cui le regole per l'uno o per l'altro software sono intercambiabili.

Altre caratteristiche

Infine una delle caratteristiche fondamentali di Suricata sta nel fatto che esso può funzionare in due modi distinti:

- In modalità online: viene monitorata una interfaccia specifica in modalità *promisqua*, ossia tutti i pacchetti passanti per quella determinata interfaccia vengono decodificati e analizzati.

- In modalita offline: viene monitorato un file pcap contenente del traffico "registrato" in precedenza e che costituisce un punto di riferimento per l'analisi prestazionale delle regole o dell'istanza di Suricata da analizzare.

(FORSE QUESTA PARTE PUOI METTERLA NEL CONTESTO DI SPERIMENTAZIONE) Concludiamo quindi elencando i motivi che hanno portato durante le fasi di sperimentazione alla scelta di Suricata rispetto ad altri software:

- Velocità (Multi-threading)
- Capacità di analizzare pcap in modalità offline
- Open Source
- Possibilià di analizzare facilmente i log grazie al formato in json
- Il fatto che si tratti di un software "giovane", pensato fin da subito per rispondere alle attuali esigenze di monitoraggio

1.2 sFlow

L' esplosione del traffico internet sta portando a larghezze di banda superiori e una maggiore necessità di reti ad alta velocità. Per analizzare e ottimizzare le reti è necessario un sistema di monitoraggio efficiente. [12]

sFlow [1] è una tecnologia sviluppata dalla InMon Corporation per monitorare il traffico all'interno di grandi reti contenenti switches e routers che utilizza il campionamento di pacchetti. In particolare, esso definisce i meccanismi di campionamento implementati in un *sFlow Agent* e il formato dei dati campionati mandati da tale Agent.

Il monitoraggio si compone di due elementi fondamentali:

- **sFlow Agent**: ovvero un qualsiasi apparato in grado di campionare i pacchetti secondo le specifiche di sFlow e di inviarli ad un *Collector*. l'Agent è un componente molto versatile ed estremamente performante dell'architettura sFlow che può essere impersonato anche da uno switch o da un router, senza degradarne le prestazioni. Il campionamento e la raccolta dei dati del nodo viene fatta in hardware e non presenta overhead nemmeno su reti Gigabit.

- **sFlow Collector**: ovvero una macchina in qualsiasi parte del mondo in grado di raccogliere i dati sFlow e di elaborarli.

L'architettura e le modalità di campionamento usati in sFlow offrono numerosi vantaggi tra cui quello di avere una visione di tutta la rete (*network-wide*) in tempo reale. La visione *network-wide* è possibile poichè sempre più produttori stanno equipaggiando i loro apparati di rete con un modulo sFlow; inoltre recentemente è stato aggiunto il supporto a sFlow anche all'interno di *OpenvSwitch*, un famoso software usato nel campo della *Software Designed Networking*. La visione in tempo reale invece è dovuta al fatto che i pacchetti campionati vengono mandati al collector non appena essi passano per l'agent. Un ulteriore punto a favore sta nel fatto che si tratta di un' architettura estremamente scalabile che permette di posizionare gli agent in diversi punti

della rete, o anche in reti diverse, garantendo una visione totale della rete anche in contesti *Multi-homed*.

1.2.1 Il campionamento

Il campionamento è la parte fondamentale del protocollo sFlow, ed è anche il motivo per cui esso si distacca da altre tecnologie simili come NetFlow.

sFlow usa due modalità operative:

- **Counter Sampling** : un campione dei contatori delle interfacce dell'apparato, che viene schedulato internamente dall' Agent su base temporale (*polling*).
- **Packet Based Sampling**: il campionamento di uno ogni N pacchetti sulla base di un opportuno parametro N (*sampling rate*). Questo tipo di campionamento non permette risultati accurati al 100% ma quantomeno permette di avere risultati di un'accuratezza accettabile e comunque parametrizzabile

Il pacchetto campionato viene esportato verso l'sFlow Collector tramite UDP [11]. La mancanza di affidabilità nel meccanismo di trasporto UDP non influisce in modo significativo sulla precisione delle misurazioni ottenute dall' Agent sFlow. Se i Counter Sampling vengono persi, al superamento dell' intervallo di polling successivo verranno inviati nuovi valori. La perdita dei Packet Flow Sample si riflette in una leggera riduzione della frequenza di campionamento effettiva. L' uso di UDP inoltre riduce la quantità di memoria necessaria per il buffer dei dati. UDP ù più robusto di un meccanismo di trasporto affidabile (es. TCP) perchè sotto sovraccarico l' unico effetto sulle prestazioni complessive del sistema è un leggero aumento del ritardo di trasmissione e un maggior numero di pacchetti persi, nessuno dei quali ha un effetto significativo su un sistema di monitoraggio.

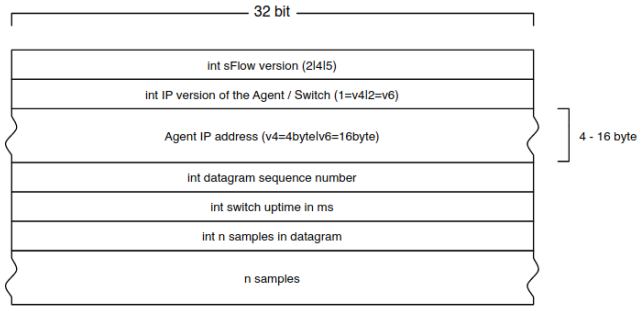

Figura 1.2: Datagram sFlow

Sflow Datagram

Il payload del pacchetto UDP inviato al Collector contiene l' sFlow Datagram, il quale è composto da: versione di sFlow, ip dell'Agent, sequence-number, il numero di campioni contenuti e fino a 10 campioni tra flow samples e counter samples (figura 1.2).

Un *flow sample* consiste di due parti:

- un *packet data*: che contiene l'header del pacchetto campionato (il quale consiste nel pacchetto originale troncato fino ad una lunghezza parametrizzabile)

- un *extended data*: che contiene informazioni aggiuntive, ad esempio in caso di uno switch con VLANs abilitate fornisce la VLAN sorgente e di destinazione.

Come possiamo notare quindi avvengono due tipi di campionamento diversi: un campionamento di un pacchetto ogni N e un troncamento del pacchetto designato fino ad una dimensione massima T (solitamente fissata ad un massimo di 256 Bytes). Occorre notare che il troncamento del pacchetto effettuato ci permette di avere una visibilià dal Layer 2 al Layer 7 dello stack OSI [11]

E‘ necessario puntualizzare che il datagram contenente il *Packet Sample* a differenza del *Counter Sample* viene inviato al collector appena il pacchetto viene campionato o comunque non oltre un secondo più tardi dal campionamento, anche se non si è riempito il buffer di 10 campioni raccolti.

1.2.2 Tools per l'utilizzo di sFlow

La InMon Corporation mette a disposizione due tool fondamentali per l'utilizzo dello standard sFlow, che impersonificano l'Agent e il Collector.

hostsflowd

Hostsflowd è un tool per sistemi UNIX [1] che svolge la funzione di Agent, ovvero osserva i pacchetti passanti per una determinata interfaccia e applica ad essi i meccanismi di campionamento discussi sopra. Esso permette di specificare tramite il suo file di configurazione: la frequenza di campionamento (*Sampling Rate*), l'indirizzo del collector e la sua porta, l'intervallo per il polling del *Counter Sample* e la grandezza dell'header del pacchetto campionato.

Si tratta di un tool molto versatile e dai requisiti molto bassi, tuttavia durante i test è emersa una limitazione importante, ossia che non è possibile troncare pacchetti ad una dimensione più grande di 256 Byte, inoltre non è possibile impostare un sampling rate di 1/1 (ovvero campionare tutto). Questo potrebbe non essere una limitazione determinante in produzione ma in fase di testing si è stati costretti a prendere strade alternative, che hanno portato allo sviluppo del simulatore *pcap2sflow*. Ulteriori motivazioni di questa scelta saranno fornite nel capitolo 2.

sflowtool

Sflowtool è un toolkit per decodificare dati sFlow. Esso svolge quindi la funzione di Collector, ed è in grado di stampare in STDOUT i dati decodifi-

[1] Ne esiste una versione anche per Windows

cati o, cosa molto più importante nel contesto di applicazione di questa tesi, spacchettare i campioni e riscriverli in formato pcap.

Ad esempio con il comando seguente è possibile esportare i pacchetti campionati in un file pcap che conterrà i campioni raccolti dall'Agent, troncati secondo i parametri specificati nell'Agent:

```
sflowtool -t > capture.pcap
```

Alternative a sFlow

Ci sono molte tecnologie che a prima vista potrebbero sembrare simili ad sFlow, probabilmente per la presenza della parola "flow", tipo NetFlow, OpenFlow o IPFIX, tuttavia essi si differenziano da sFlow per alcuni motivi fondamentali: NetFlow e IPFIX sono protocolli di esportazione del flusso che mirano ad aggregare i pacchetti in flussi. Successivamente, i record di flusso vengono inviati a un punto di raccolta per la conservazione e l' analisi. [12] sFlow, tuttavia, non ha alcuna nozione di flussi o aggregazione di pacchetti. Inoltre, mentre l' esportazione del flusso può essere eseguita con campionamento 1:1 (cioè considerando ogni pacchetto), questo non è tipicamente possibile con sFlow, in quanto non è stato progettato per farlo. Il campionamento è parte integrante di sFlow, con l' obiettivo di fornire scalabilità per il monitoraggio in tutta la rete. [13]

La caratteristica fondamentale necessaria per la riuscita del progetto portato avanti in questa tesi stà proprio nel fatto che i dati non siano dati aggregati, come avviene in NetFlow o IPFIX. Un aggregazione dei dati non permette infatti un'analisi degli header dei pacchetti invalidando quindi la funzionalità di monitoraggio contro gli attacchi informatici.

Come ulteriore prova si fornisce un esempio di dati provenienti da un sistema di monitoraggio che utilizza NetFlow (con un ragionamento analogo è possibile fornire la stessa prova per IPFIX).

Output del comando *nfdump*:

Date flow start	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Packets	Bytes	Flows
2010-09-01 00:00:00.459	0.000	UDP	127.0.0.1:24920	->	192.168.0.1:22126	1	46	1
2010-09-01 00:00:00.363	0.000	UDP	192.168.0.1:22126	->	127.0.0.1:24920	1	80	1

Figura 1.3:

1.3 Packet sampling e Intrusion Detection

La sicurezza delle reti, specie di quelle molto estese, è un problema tutt'oggi presente e di un importanza vitale per qualsiasi azienda che operi nel settore ICT (Information and Communications Technologies). Le minacce possono presentarsi in qualunque momento [3] e provenire sia dall'interno che dall'esterno. Riuscire ad identificare queste minacce in tempo è il primo passo verso la soluzione di questo difficile problema. Per fare ciò è necessario avere un'ampia e continua sorveglianza della rete. Storicamente la sorveglianza di una grande rete era (ed è tuttora) affidata a sonde (*probes*), posizionate in punti strategici della rete. Questo è stato sufficientemente accurato fino ad ora, ma visto l' aumento delle switched point to point network, il monitoraggio *probe-based* non è più sufficiente. Implementare il monitoraggio già dall'interno di uno switch o di router sta diventando sempre più una necessità. Tuttavia le esigenze di mercato tendono a preferire l'ampiezza di banda sulla sicurezza, per cui la funzione di monitoraggio deve essere relegata come funzione secondaria all'interno di questi apparati di rete. E' necessario quindi che questa funzione operi con il minimo overhead possibile, al fine di non degradare le prestazioni dell'apparato. E' qui che entra in gioco la tecnologia sFlow, essa permette di delegare la parte di analisi del traffico ad un altro componente della rete (*il collector*), lasciando allo switch o al router le risorse per effettuare le normali decisioni di smistamento dei pacchetti.

Il rilevamento delle intrusioni basato sui flow è un modo innovativo di rilevare le intrusioni nelle reti ad alta velocità. Il rilevamento delle intrusioni basato sui flow controlla solo l' header del pacchetto e non ne analizza il *payload*. Analizzare il *payload* del pacchetto infatti sarebbe computazionalmente inaccettabile su reti di grandi dimensioni e con un elevato flusso di dati.

Questa limitazione però nel caso di sFlow si presuppone che non invalidi la qualità del sistema di monitoraggio poichè la stragrande maggioranza degli attacchi è riconoscibile tramite un'analisi degli header al di sopra del livello di Livello 3.

Vediamo ora più nel dettaglio come è possibile costruire con sFlow un sistema di monitoraggio efficace e che rispetta le richieste di mercato:

- Il sistema deve avere una sorveglianza continua network-wide: sFlow può essere configurato su ogni apparato di rete
- I dati devono essere sempre disponibili per rispondere efficacemente: sFlow manda immediatamente i pacchetti campionati al Collector con UDP
- I dati devono essere sufficientemente dettagliati per caratterizzare l'attacco: sFlow permette di esportare più di 128 Byte (comprendendo quindi ad esempio anche header di livello 7)
- Il sistema di monitoraggio non deve esporre gli apparato ad attacchi: sFlow non impatta sulle prestazioni degli apparati poichè viene effettuato in hardware inoltre il consumo di banda è limitato poichè i datagram sFlow sono ottimizzati

1.3.1 Metodi di rilevazione flow-based

L'analisi dei dati provenienti dai *flow generators* al fine di garantire una rilevazione efficace della maggior parte degli attacchi ha portato nel corso degli anni all'utilizzo di molteplici approcci, che utilizzano tre macroaree dell'intrusion detection moderna:

- *Statistical Analysis*
- *Machine Learning Analysis*
- *Signature-based Analysis*

I metodi statistici *Statistical Analysis* costruiscono un profilo del traffico di rete normale utilizzando una funzione statistica dei parametri del traffico di rete; questo profilo del traffico normale viene utilizzato per controllare il traffico in arrivo al momento attuale; la somiglianza tra il traffico di rete e il profilo del traffico di rete normale viene calcolata utilizzando metodi statistici. Se la misura di somiglianza è al di sopra della soglia predefinita, il flusso è marcato come dannoso o normale. [15] I metodi statistici non necessitano di conoscenza pregressa di un determinato attacco per determinare se esso è effettivamente un attacco, e questo si è dimostrato particolarmente efficace nel rilevare attacchi di tipo DoS (*Denaial of Service*). Essi tuttavia possono essere aggirati mantenendo l'impatto dell'attacco al di sotto della soglia predefinita. Inoltre l'efficacia di questi metodi in reti reali è minata dalla difficoltà intrinseca che sta nel calcolare le statistiche sul traffico.

I metodi che utilizzano tecniche di *Machine Learning* si basano principalmente, su reti neurali, *Support Vector Matrix* (SVM), *Decision Tree* e *Clustering*. Essi presentano numerosi vantaggi rispetto ai metodi statistici quali l'adattamento del comportamento in base ad una finestra temporale. Inoltre i risultati degli esperimenti condotti con Machine Learning hanno evidenziato un'ottimo tasso di riconoscimento degli attacchi in un processo di "apprendimento" relativamente breve. Tuttavia anche questo metodo ha degli svantaggi importanti quali quelli la difficoltà di generare dati campioni per l'apprendimento, inoltre quest'ultimo si è rivelato computazionalmente molto costoso. Infine essi hanno un alto tasso di falsi positivi (ovvero allarmi generati da un pacchetto benevolo).

In ultimo abbiamo la *Signature-based Analysis*, tema principale di questo elaborato, che utilizza l'header dei pacchetti campionati per fare pattern matching con un dataset di regole statiche. Questo approccio porta con se tutti i vantaggi e gli svantaggi degli IDS Signature-based discussi nei paragrafi precedenti, con il solo svantaggio aggiunto di non poter ricorrere all'utilizzo della Stateful Analysis poichè il campionamento non garantisce che tutti i pacchetti di una determinata sessione siano catturati. Inoltre occorre ribadire

che tale metodo è utilizzabile solo nel caso in cui si abbiano a disposizione almeno gli header di Livello 3, per quanto detto nella sezione "Alternative a sFlow", da cui ci si accorge che sFlow rappresenta l'unica alternativa di applicazione.

1.4 Elasticsearch e lo stack ELK

L'analisi dei dati, specie se di grosse dimensioni, come i dati generati dai test o dai log, è un problema che va affrontato con sistematicità al fine di poter trarre delle conclusioni efficaci dal lavoro svolto. È stato necessario quindi utilizzare tool appositamente studiati per la *Data Analysys* come lo stack ELK, ed in particolare Kibana.

Elasticsearch [16] è un motore di ricerca basato su Lucene. Esso fornisce un motore capace di ricerca full-text, distribuito e con capacità multitenant. Possiede un' interfaccia web HTTP e una strutura schema-free di documenti con la sitassi JSON. Elasticsearch è sviluppato in Java ed è rilasciato con licenza Open Source Apache. I client ufficiali sono disponibili in Java,. NET (C), PHP, Python, Apache Groovy e molti altri linguaggi. A differenza di un *Database Management System* relazionale, esso è *schema-free* anche se non *schema-less* (come MongoDB per esempio) il che significa che non è necessario definire i tipi (string, integer, ecc.) dei dati prima di inserirli, ma comunque possibile. [18] Secondo DB-Engines, [17] Elasticsearch è il motore di ricerca più popolare in ambito enterprise e rappresenta un tool fondamentale nell'analisi e nella centralizzazione dei log.

Elasticsearch infatti è sviluppato dallo stesso team di pari passo con Logstash e Kibana e i tre programmi insieme formano quello che viene chiamato lo stack ELK.

Esso si compone di tre componenti fondamentali:

- **Logstash** : è una "pipeline di elaborazione dati" che può raccogliere dati da varie fonti, trasformarli e inviarli a vari consumatori, tra cui Elasticsearch
- **Elasticsearch** : come descritto prima, un database No-SQL particolarmente ottimizzato per la ricerca
- **Kibana** : è uno strumento di visualizzazione web-based che si integra con Elasticsearch atto a fornire un modo efficace per navigare e visualizzare i dati, utilizzando una varietÃ di grafici, grafici e tabelle.

Figura 1.4: Lo stack ELK

Lo stack ELK è ad oggi lo standard *de facto* per l'esplorazione dei log, esso si va ad affiancare a tutti i programmi critici all'interno di una infrastruttura e risulta particolarmente utile nell'esplorazione dei log degli IDS. Collegato ai log di un IDS permette infatti acquisire in pochissimi secondi conoscenza su cosa accade nella rete e su che tipo di attacchi sono in corso. I dati vengono fruiti all'operatore in maniera veloce e comprensibile tramite dashboard, fornendo una fotografia estremamente informativa dello stato attuale.

In figura è illustrato il funzionamento della pipeline di analisi usando lo stack ELK.

Conclusioni

Queste sono le conclusioni.
In queste conclusioni voglio fare un riferimento alla bibliografia: questo è il mio riferimento [3, 4].

Appendice A

Prima Appendice

In questa Appendice non si è utilizzato il comando: `\clearpage{\pagestyle{empty}\cleardoublepage}`, ed infatti l'ultima pagina 8 ha l'intestazione con il numero di pagina in alto.

Appendice B

Seconda Appendice

Bibliografia

[1] RFC 3176 InMon Corporation's sFlow for monitoring traffic in Switched and Routed Networks

[2] A survery on Network Security Monitoring System, Ibrahim Ghafir, Vaclav Prenosil, Jakub Svoboda, Mohammad Hammoudeh, 2016 4th International Conference on Future Internet of Things and Cloud Workshops

[3] Traffic Monitoring with Packet-Based Sampling for Defense against Security Threats, Joseph Reves ans Sonia Panchen

[4] "NIST â Guide to Intrusion Detection and Prevention Systems (IDPS)" (PDF). February 2007.

[5] https://ossec.github.io/

[6] https://github.com/OISF/suricata

[7] https://www.snort.org/

[8] https://www.bro.org

[9] https://www.ntop.org/products/packet-capture/pf_ring/

[10] sFlow, I Can Feel Your Traffic, Elisa Jasinska, Amsterdam Internet Exchange

[11] Reti di Calcolatori, Larry Peterson, Bruce Davie, Apogeo, 2012

[12] Hofstede, Rick; Celeda, Pavel; Trammell, Brian; Drago, Idilio; Sadre, Ramin; Sperotto, Anna; Pras, Aiko. "Flow Monitoring Explained: From Packet Capture to Data Analysis with NetFlow and IPFIX". IEEE Communications Surveys Tutorials. IEEE Communications Society

[13] http://blog.sflow.com/2009/05/scalability-and-accuracy-of-packet.html

[14] Flow-based intrusion detection: Techniquesand challenges, Muhammad Fahad Umer, Muhammad Sher, Yaxin Bi. Computers & Security 70 (2017) 238-254

[15] Intrusion detection system: A comprehensive review, Hung-JenLiao, Chun-HungRichard Lin,Ying-ChihLin, Kuang-YuanTung

[16] https://www.elastic.co/

[17] https://db-engines.com/en/ranking/search+engine

[18] SCADA STATISTICS MONITORING USING THE Elastic Stack(Elasticsearch, Logstash, Kibana), James Hamilton, Brad Schofield, Manuel Gonzalez Berges, Jean-Charles TournierCERN, Geneva, Switzerland

Ringraziamenti

Qui possiamo ringraziare il mondo intero!!!!!!!!!!
Ovviamente solo se uno vuole, non è obbligatorio.