./images/

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

Introduzione

Questa è l'introduzione.

Indice

Elenco delle figure

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

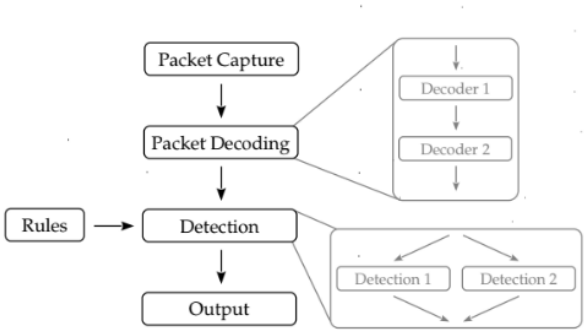

Figura 1.1: Funzionamento del pattern matching

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. [10]

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. Infatti, i pacchetti campionati vengono mandati al collector non appena essi arrivano all'agent. Inoltre l'architettura è estremamente scalabile e permette di posizionare agent in diversi punti della rete, o anche in reti diverse.

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 affidabilitÃ 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.

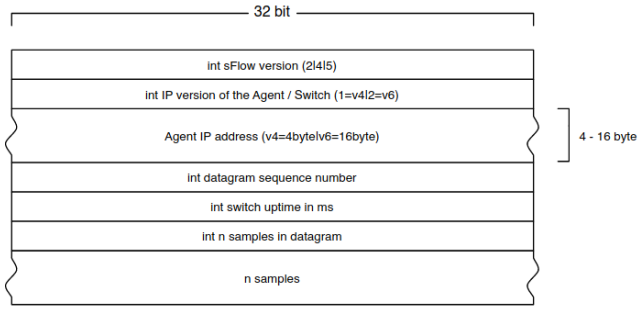

Figura 1.2: Datagram sFlow

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
```

sFlow vs Netflow

Un'altra tecnologia molto simile ad sFlow è NetFlow, sviluppato da Cisco. Essi si differenziano per un motivo importantissimo e che non permette di usare efficacemente NetFlow per monitorare la rete a fianco di un IDS; ovvero che Netflow fornisce solo dati aggregati e non fornisce un troncamento parametrizzabile. Esso infatti esporta solo il Layer 3 del pacchetto, questo fa perdere tutte le informazioni che, nel nostro particolare campo di applicazione, sarebbero state utili, come l'header dei livelli superiori.

Qui è possibile vedere un esempio di un 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 IDS

La sicurezza delle reti, specie di quelle molto estese, è un problema tutt'oggi presente e di un importanza vitale in qualsiasi azienda che operi nel settore ICT (Information and Communications Technologies). Le minacce possono avvenire in qualunque momento [3] ed essere generate 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, quando si sta verificando un aumento dell'installazione di switched point to point network. Quindi 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 loro decisioni di smistamento dei pacchetti.

Vediamo allora come sflow garantisce l'efficacia del sistema di monitoraggio richiesto:

- 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 impatta sulle prestazioni degli apparati poichè viene effettuato in hardware inoltre il consumo di banda è limitato poichè i datagram sFlow sono compressi

1.4 Stack ELK

Lo stack ELK è uno stack formato dai tre programmi (Elasticsearch, Logstash e Kibana) i quali uniti insieme forniscono una metodologia di centralizzazione, esplorazione e visualizzazione dei log. Esso si compone di tre componenti fondamentali:

- **Logstash** : il quale si occupa di raccogliere i log, manipolarli così da renderli consumabili da elasticsearch
- **Elasticsearch** : un database No-SQL particolarmente ottimizzato per la ricerca
- **Kibana** : un visualizzatore dei dati contenuti in Elasticsearch che permette di effettuare ricerche e aggregare i dati in modo da fornirne una visualizzazione utile dei dati di log raccolti

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. Esso 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

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

Ringraziamenti

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