Università degli Studi di Modena

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E INFORMATICHE
Corso di Laurea in Informatica

Tesi di laurea

Titolo tesi

Candidato:
Michele Murgolo
Matricola 101851

Relatori:
Prof. Mirco Marchetti
Prof. Giovanni Apruzzese

Anno Accademico 2017-2018

La pagina della dedica

[**?**]

Sommario

Stratosphere Testing Framework (stf) è una framework di ricerca sulla sicurezza della rete per analizzare i modelli comportamentali delle connessioni di rete nel Progetto Stratosphere. Il suo obiettivo è aiutare i ricercatori a trovare nuovi comportamenti malware, etichettare tali comportamenti, creare i loro modelli di traffico e verificare gli algoritmi di rilevamento. Stf funziona utilizzando algoritmi di apprendimento automatico sui modelli comportamentali. L'obiettivo di Stratosphere Project è creare un IPS comportamentale (Intrusion Detection System) in grado di rilevare e bloccare i comportamenti dannosi nella rete. Come parte di questo progetto, stf viene utilizzato per generare modelli altamente attendibili di traffico dannoso consentendo una verifica automatica delle prestazioni di rilevamento. Il framework genera questi modelli da file in formato binetflow, il DIEF salva il traffico internet in file formato flows. Si è scritto un programma in python3 che esegue la conversione batch da flows a binetflow. I file che il programma deve convertire sono numerosi e di grandi dimensioni, ogni giorno di traffico ha una dimensione media pari a 150Mb. Per effettuare una conversione efficiente si è utilizzato un approccio multicore che ha permesso di ottenere uno speed up lineare della conversione.

Indice

Capitolo 1

Introduzione

La continua digitalizzazione nel mondo sta mettendo le aziende a rischio di attacchi informatici più che mai. Negli ultimi anni, grazie alla crescente adozione di servizi cloud e mobili, la sicurezza delle informazioni ha subito un profondo cambio di paradigma dai tradizionali strumenti di protezione verso l'individuazione di attività dannose all'interno delle reti aziendali.

I metodi di attacco sempre più sofisticati utilizzati dai criminali informatici in diverse recenti violazioni della sicurezza su larga scala indicano chiaramente che gli approcci tradizionali alla sicurezza delle informazioni non possono più tenere il passo.

L'analisi dei dati è l'elemento chiave per sfruttare la resilienza informatica. Con attacchi sempre più avanzati e persistenti e il semplice fatto che ogni organizzazione deve proteggersi da tutte le varietà di attacchi mentre un aggressore ha bisogno solo di un tentativo riuscito, le organizzazioni devono ripensare ai propri concetti di sicurezza informatica: Devono andare oltre la pura prevenzione.

big data security analytics è l'approccio alla base di questo miglioramento del rilevamento. Il rilevamento deve essere in grado di identificare i modelli di utilizzo che cambiano ed eseguire analisi complesse su una varietà di fonti di dati che vanno dai registri di server e applicazioni agli eventi di rete e alle attività degli utenti. Ciò richiede di eseguire analisi su grandi quantità di dati correnti e storici.

Negli ultimi anni è emersa una nuova generazione di soluzioni di analisi della sicurezza, in grado di raccogliere, archiviare e analizzare enormi quantità di dati. Questi dati vengono analizzati utilizzando vari algoritmi di correlazione per rilevare le anomalie e quindi identificare possibili attività dannose. L'industria ha finalmente raggiunto il punto in cui gli algoritmi di intelligenza artificiale per l'elaborazione di dati su larga scala sono diventati accessibili utilizzando framework prontamente disponibili.

Ciò consente di combinare analisi storiche e in tempo reale e identificare nuovi incidenti che potrebbero essere correlati ad altri che si sono verificati in passato. Insieme a fonti di intelligence di sicurezza esterne che forniscono informazioni aggiornate sulle ultime vulnerabilità, ciò può facilitare notevolmente l'identificazione di attacchi informatici in corso sulla rete.

È con l'intenzione di utilizzare queste tecnologie che viene presentato in questa tesi un software per la cattura, l'archiviazione e l'analisi di enormi quantità di dati. Come mostrerò nei seguenti capitoli, l'utilizzo di questi software comporta una organizzazione dei dati notevole e verranno evidenziati in special modo le difficoltà nella gestione dei diversi formati che questi tipi di tecnologie comportano. Questa tesi metterà in evidenza l'eterogeità dei software che hanno sempre contraddistinto l'informatica e le soluzioni scelte per risolvere tali problemi.

Nel secondo capitolo verranno in primo luogo presentate le minacce provenienti dalla rete, con particolare enfasi sui tipi di attacchi su larga scala. Successivamente verranno presentato lo stato dell'arte dei sistemi di difesa utilizzati ad oggi per contrastare questi tipi di attacchi. In conclusione del capitolo verrà discussa la scelta delle tecnologie utilizzate in questa tesi.

Nel terzo capitolo verrà introdotto uno speciale tipo di file che sarà il principale punto di enfasi in questa tesi. Dopo di chè verranno presentati i differenti tipi di formati che questo file può assumere e le problematiche dovute ai diversi standard in uso oggi. Dopo una descrizione dettagliate delle varie differenze tra i formati verrà introdotto il software che farà uso di questi file e si presenterà l'utilizzo che questi file hanno in relazione alle tecnologie utilizzate. Infine troverà spazio la presentazione del problema da affrontare.

Nel quarto capitolo sarà descritto in dettaglio l'installazione del software di cui si farà uso. In seguito si descriveranno le scelte effettuate per risolvere il problema nell'uso di diversi formati di file e l'automatizzazione di tale soluzione. Infine saranno presentati i diversi metodi atti a perfezionare l'automatizzazione per renderla efficiente e la scelta che è stata effettuata.

Nel quinto capitolo verranno mostrati i risultati dei test e i benchmark effettuati con lo scopo di confermare al livello pratico quanto mostrato sotto forma teorica nel capitolo precedente. Saranno descritte in modo dettagliato le condizioni sotto le quali sono stati effettuati i test e limiti della scalabilità della soluzione. Il tutto verrà seguito da grafici esplicativi.

Nel sesto capitolo infine, verrà fatto un riassunto della tesi ribadendo l'obiet-

tivo, cosa si è svolto in questa tesi e i risultati ottenuti.

Capitolo 2

Stato dell'arte

Network Intrusion Detection System (IDS) è un sistema di rilevamento delle intrusioni: una tecnologia di sicurezza efficace, che può rilevare, prevenire e possibilmente reagire agli attacchi informatici, uno dei componenti standard nelle infrastrutture di sicurezza. Monitora le fonti di attività del traffico della rete e distribuisce varie tecniche per fornire servizi di sicurezza. L'obiettivo principale degli *IDS* è quello di rilevare tutte le intrusioni in modo efficiente, l'implementazione consente agli amministratori di rete di rilevare violazioni degli obiettivi di sicurezza.

Esistono diversi tipi di tecniche per rilevare le intrusioni. In questa tesi si è fatto uso di un *IDS* che utilizza algoritmi di *machine learning*.

Machine learning può essere definito come la capacità di un programma per computer di apprendere e migliorare le prestazioni su una serie di attività nel tempo. Le tecniche di *machine learning* si concentrano sulla costruzione di un modello, *behavioral model*, di sistema che migliora le sue prestazioni in base ai risultati precedenti.

Una *botnet* è una rete di computer compromessi sotto il controllo di un attore malintenzionato. Un bot si forma quando un computer viene infettato da un malware che ne consente il controllo da terze parti. I computer infetti sono noti anche come *zombie* per la loro capacità di operare in direzione remota senza la conoscenza dei loro proprietari. Negli ultimi anni il *botnet detection* è stato un tema caldo a causa dell'aumento dell'attività malevola.

L'utilizzo di *IDS* è un approccio utile per fare *botnet detection* sul traffico di rete, si osserva il traffico di dati nella rete e si cercano comunicazioni sospette che possono essere fornite da *bot*.

Capitolo 3

Analisi del problema

3.1 Analisi traffico di rete

Nell'era digitale il traffico di rete è aumentato notevolmente e con esso gli attacchi di tipo informatico, per questo motivo è necessario trovare soluzioni per analizzare questo grande quantitativo di pacchetti che attraversano i dispositivi di rete delle aziende di grosse dimensioni. Nel mondo informatico di oggi è molto importante poter determinare rapidamente e con precisione l'origine e la portata di un potenziale attacco su una rete al fine di poterlo contrastare in modo efficace. Per fare ciò viene costruito un *audit trail* di informazioni collezionate dal traffico di rete usando una combinazione di network flow e PCAP. Un audit trail è un file che contiene una registrazione cronologica di attività relative alla sicurezza per consentire la ricostruzione e l'esame di eventi.

3.1.1 Network Flow

Un flow è una sequenza di pacchetti inviati da una sorgente ad una destinazione che hanno degli attributi in comune:

- indirizzo IP sorgente
- indirizzo IP destinazione
- porta sorgente
- porta destinazione
- protocollo

Se i pacchetti che attraversano un dispositivo di rete hanno questi attributi in comune possono essere raggruppati in un flow.

3.1.2 Packet Capture

Per packet capture (PCAP) si intende la cattura di traffico internet che attraversa un dispositivo di rete. Un packet capture intercetta i singoli pacchetti e li archivia. Nei sistemi operativi Unix è utilizzata la libreria **libpcap** mentre nei sistemi Windows si fa utilizzo di **WinPcap**

3.1.3 NetFlow

NetFlow è un protocollo di analisi di rete che offre la possibilità di raccogliere informazioni dettagliate sul traffico mentre attraversa un'interfaccia. NetFlow è una tecnologia proprietaria di Cisco. I dispositivi di rete conformi a NetFlow possono raccogliere statistiche sul traffico ed esportarle come record verso un NetFlow collector, un server che esegue l'analisi del traffico.

Cisco definisce un flow come una sequenza unidirezionale di pacchetti che condividono tutti i seguenti 7 valori:

- interfaccia di ingresso
- indirizzo IP sorgente
- indirizzo IP destinazione
- protocollo IP
- porta sorgente TCP o UDP, 0 per altri protocolli
- IP Type of Service

3.1.3.1 Componenti di NetFlow

Una architettura NetFlow ha i seguenti componenti

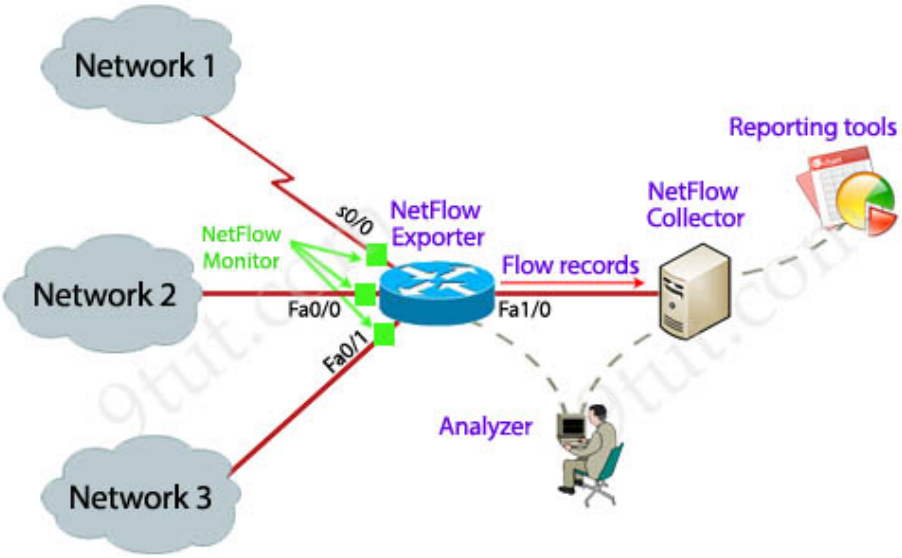


Figura 3.1: Architettura NetFlow

NetFlow monitor un componente applicato a un'interfaccia che raccoglie informazioni sui flow. I NetFlow monitor sono costituiti da un record e una cache.

NetFlow exporter Aggrega i pacchetti in flows e ne esporta i record verso uno o più *flow collectors*. Quando dei pacchetti arrivano al NetFlow exporter, vengono ispezionati singolarmente per uno più attributi che vengono utilizzti per determinare se il pachetto è univoco o è simile agli altri pacchetti. Se il pacchetto presenta attributi simili viene classificato nello stesso flow.

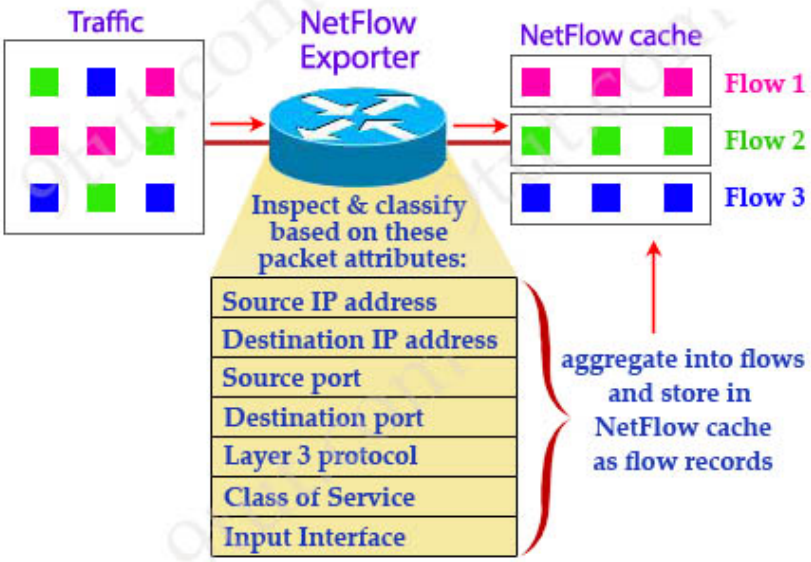


Figura 3.2: NetFlow exporter

Dopo aver esaminato questi attributi, il NetFlow exporter li aggrega in record di flow e li salva in un database che può essere una cache NetFlow o un NetFlow collector.

NetFlow collector Responsabile della ricezione, conservazione e pre-elaborazione dei dati di un flow ricevuti da un *flow exporter*. Solitamente è un software separato in esecuzione su un server di rete. I record NetFlow vengono esportati in un NetFlow collector tramite protocollo UDP.

Analysis application Analizza i dati dei flows ricevuti nel contesto del rilevamento delle intrusioni o del profilo di traffico.

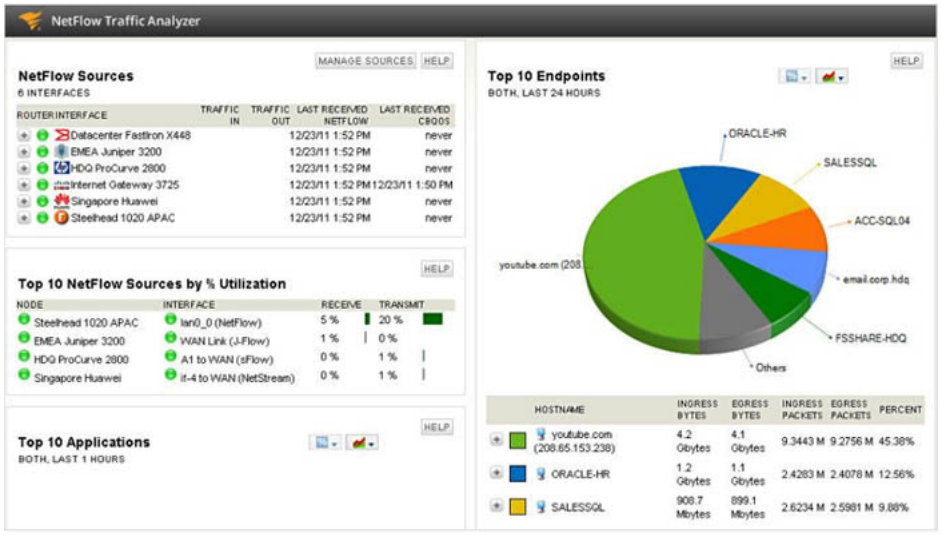


Figura 3.3: SolarWinds NetFlow Traffic Analyzer

L'analisi dei flow può aiutare a determinare le statistiche del traffico in generale, ma non è sufficiente quando è necessario analizzare una conversazione specifica nel dettaglio. È quindi necessario un utilizzo di entrambe le tecnologie per la monitorazione del traffico.

È importante rendersi conto che una risposta efficae agli incidenti è tutta una questione di dimensioni. Raccogliendo solo PCAP si potrebbero avere troppi dati in un tempo troppo breve, usare PCAP per scoprire con chi uncomputer era connesso su un segmento occupata della rete è, nel migliore dei casi, una lunga query e, nel peggiore dei casi, impossibile (****QUESTO PARAGRAFO VA RISCRITTO****).

Pertanto l'approccio migliore per le organizzazioni è quello di utilizzare prima NetFlow per poi integrarsi con PCAP in un secondo momento.

3.2 Strumenti software utilizzati

Verranno ora descritti i software utilizzati in questa tesi. I software utilizzati sono multipiattaforma, in questa tesi sono stati adoperati su sistema operativo basato su una distribuzione GNU/Linux.

3.2.1 Audit Record Generation and Utilization System

Argus (Audit Record Generation and Utilization System) è la prima implementazione del monitoraggio dei flow, è un progetto open source e multipiattaforma.

Quest'ultima particolarità lo rende molto interessante poichè supportando molti sistemi operativi, trai quali Windows, MaxOSX, Linux, Solaris, FreeBSD, OpenBSD, IRIX e OpenWrt, può essere adoperato in quasi tutte le reti comprendo la maggior parte degli host. La sua architettura è di tipo server/client. Il server recupera i pacchetti ricevuti da una o più interfacce di rete disponibili su una una macchina, argus assembla poi questi pacchetti in dati binari che rappresentano dei flow. Lo scopo dei client è di quello di leggere i dati dei flow.

Argus viene utilizzato da molte università e aziende per registrare dei flow che vengono utilizzati sia nell'analisi immediata dell'utilizzo della rete, sia nell'analisi storica. I record Netflow di Argus offrono un rapporto fino a 10.000:1 dalla dimensione del pacchetto al record scritto sul disco, che consente alle installazioni di salvare i record per molto più tempo rispetto alle acquisizioni di pacchetti completi.

campo	**descrizione**
StartTime	record start time
Dur	record total duration
Proto	transaction protocol
SrcAddr	source IP address
Sport	source port number
Dir	direction of transaction
DstAddr	destination IP address
Dport	destination port number
State	transaction state
sTos	source TOS byte value
dTos	destination TOS byte value
TotPkts	total transaction packet count
TotBytes	total transaction bytes
SrcBytes	src ->dst transaction bytes
srcUdata	source user data buffer
dstUdata	destination user data buffer
Label	metadata label

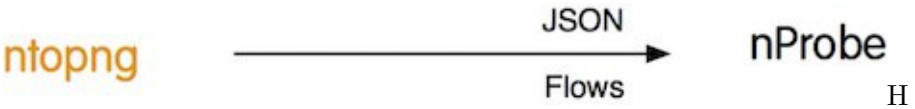
3.3 nProbe

Negli ambienti commerciali, NetFlow è probabilmente lo standard de facto per la contabilità e la fatturazione del traffico di rete. nProbe è un software in grado di reccogliere, analizzare ed esportare report sul traffico di rete utilizzando il formato standard Cisco NetFlow. È disponibile per la maggior parte dei sistemi operativi sul mercato.

Nel monitoraggio basato sui flow ci sono due componenti principali: il flow exporter e il flow collector. Solitamente NetFlow è un pradigma in modalità push poichè i dispositivi di rete non hanno quasi nessun tipo di memoria di archiviazione e quindi inviano i dai il più presto possibile verso un collector. Questa architettura non è ottimale poichè la sonda sta inviando gli stessi dati a tutti i collector e anche perchè nel caso in cui debba essere aggiunto un nuovo collector, la sonda deve essere riconfigurata. Un altro problema è che i dati scambiati sono in chiaro, il che significa che chiunque intercetta i flussi inviati dalla sonda può scoprire cosa succede nella rete monitorata. Ntopng ha ripristinato questo paradigma utilizzando un'architettura in modalità polling.

Con l'utilizzo di ZMQ ntopng si iscrive in modo dinamico alla sonda, comunica alla sonda il tipo di dati a cui è interessata nel flow e la sonda invia solo questa informazione, senza inviare tutti i flow a ntopng. Questa pratica ottimizza il traffico di rete e limita i cicli della CPU a quelli realmente necessari per continuare a raccogliere flows.

3.3.0.0.1 ZeroMQ ZeroMQ è una libreria di messagistica asincrona ad alte prestazione destinata all'utilizzo in applicazioni distribuite o concorrenti. L'API ZeroMQ fornisce socket che possono rappresentare una connessione molti-a-molti tra endpoint. Operando con una granularità del messaggio, richiedono l'uso di un pattern di messagistica e sono particolarmente ottimizzati per quel tipo di pattern.



H

3.4 Stratosphere IPS

In questa tesi si è utilizzato Stratosphere Testing Framework, un *Network Intrusion Detection System* che genera modelli comportamentali delle connessioni di reti. Il suo obiettivo è aiutare i ricercatori a trovare nuovi comportamenti malware, etichettare tali comportamenti, creare i loro modelli di traffico e verificare gli algoritmi di rilevamento. Stf funziona utilizzando algoritmi di apprendimento automatico sui modelli comportamentali.

L'obiettivo di Stratosphere Project è quello di creare un *IDS* comportamentale in grado di rilevare e bloccare i comportamenti dannosi nella rete.

Come parte di questo progetto, stf viene utilizzato per generare modelli altamente attendibili di traffico dannoso consentendo una verifica automatica delle

campo	descrizione
IPV4_SRC_ADDR	IPv4 source address
IPV4_DST_ADDR	IPv4 destination address
IPV4_NEXT_HOP	IPv4 next hop address
INPUT_SNMP	input interface SNMP idx
OUTPUT_SNMP	output interface SNMP idx
IN_PKTS	incoming flow packets (src ->dst)
IN_BYTES	incoming flow bytes (src ->dst)
FIRST_SWITCHED	SysUptime (msec) of the first flow pkt
LAST_SWITCHED	SysUptime (msec) of the last flow pkt
L4_SRC_PORT	IPv4 source port
L4_DST_PORT	IPv4 destination port
TCP_FLAGS	cumulative of all flow TCP flags
PROTOCOL	IP protocol byte
SRC_TOS	Type of service byte
SRC_AS	source BGP AS
DST_AS	destination BGP AS
IPV4_SRC_MASK	IPv4 source subnet mask
IPV4_DST_MASK	IPv4 dest subnet mask
L7_PROTO	layer 7 protocol (numeric)
BIFLOW_DIRECTION	1=initiator, 2=reverseInitiator
FLOW_START_SEC	seconds (epoch) of the first flow packet
FLOW_END_SEC	seconds (epoch) of the last flow packet
OUT_PKTS	outgoing flow packets (dst ->src)
OUT_BYTES	outgoing flow bytes (dst ->src)
FLOW_ID	serial flow identifier
FLOW_ACTIVE_TIMEOUT	activity timeout of flow cache entries
FLOW_INACTIVE_TIMEOUT	inactivity timeout of flow cache entries
IN_SRC_MAC	source MAC address
OUT_DST_MAC	destination MAC address

prestazioni di rilevamento.

Stratosphere IPS non è strettamente un *IPS* nel senso che può impedire l'intrusione. Usa l'acronimo *IPS* perchè l'*IPS* di Stratosphere può bloccare connessioni malevoli usando il firewall del computer. Tuttavia, a causa della natura delle connessioni di traffico, l'*IPS* di Stratosphere necessita di un po' di tempo per rilevare il comportamento dannoso e quindi non può bloccare i primi pacchetti nella connessione.

L'*IPS* di Stratosphere è in grado di rilevare e bloccare connessioni di rete molto fini e pericolose, e quindi dovrebbe essere visto come un complemento delle attuali misure di sicurezza della rete.

3.4.1 Il significato dei modelli comportamentali

Il nucleo di *Stratosphere IPS* è composto dai modelli comportamentali di reti e algoritmi di rilevamento. I modelli comportamentali rappresentano ciò che una connessione specifica fa nella rete durante la sua vita. Il comportamento è costuito analizzando la sua periodicità, le dimensioni e la durata di ciascun flusso. Sulla base di queste caratteristiche a ciascun flusso viene assegnata una lettera e il gruppo di lettere caratterizza il comportamento della connessione.

Prendiamo come esempio una connessione generata da una botnet che ha il seguente modello comportamentale

88∗y∗y∗ i ∗H∗H∗H∗y∗0yy∗H∗H∗H∗y∗y∗y∗y∗H∗h∗y∗h∗h∗H∗H∗h∗H∗y∗y∗y∗H∗

Questa catena di stati che chiamiamo modello comportamentale evidenzia alcune delle caratteristiche del canale C&C. In questo caso ci dice che i flussi sono altamente periodici (lettere *h*, *i*), con qualche periodicità persa vicino all'inizio (lettere *y*). I flussi hanno anche una grande dimensione con una durata media. I simboli tra le lettere sono correlati al tempo trascorso tra i flussi. In questo caso il simbolo '*' significa che il flusso è separato da meno di un'ora. Guardando le lettere si può vedere che questa è una connessione piuttosto periodica, e controllando efficacemente i suoi flussi confermiamo tale ipotesi. Con l'utilizzo di questo tipo di modelli siamo in grado di generare le caratteristiche comportamentali di un gran numero di azioni dannose. L'immagine seguente mostra i criteri di assegnazione delle lettere per i modelli comportamentali

	Size Small			**Size Medium**			**Size Large**		
	Dur. Short	**Dur. Med.**	**Dur. Long**	**Dur. Short**	**Dur. Med.**	**Dur. Long**	**Dur. Short**	**Dur. Med.**	**Dur. Long**
Strong Periodicity	a	b	c	d	e	f	g	h	i
Weak Periodicity	A	B	C	D	E	F	G	H	I
Weak Non-Periodicity	r	s	t	u	v	w	x	y	z
Strong Non-Periodicity	R	S	T	U	V	W	X	Y	Z
No Data	1	2	3	4	5	6	7	8	9

Symbols for time difference:

Between 0 and 5 seconds:	.
Between 5 and 60 seconds:	'
Between 60 secs and 5 mins:	+
Between 5 mins and 1 hour:	*
Timeout of 1 hour	0

Gli algoritmi di rilevamento utilizzano modelli comportamentali proprietari dannosi per rilevare nuove connessioni sospette nella rete. Il rilevamento viene attualmente eseguito utilizzando algoritmi basati su catene di *Markov*. La prima parte dell'algoritmo consiste nell'apprendimento e nell'etichettatura del traffico di verità di base. Questo traffico viene utilizzato per creare modelli verificati di

comportamenti di rete noti e stabili.

La seconda parte dell'algoritmo consiste nell'utilizzare questi modelli di verità di base noti e verificati per rilevare comportamenti simili in reti sconosciute. *Stratosphere IPS* catturerà il traffico in un computer client e confronterà ogni connessione sconosciuta con i modelli conosciuti di comportamento del traffico. Poichè il modo in cui viene effettuato il rilevamento e come vengono creati i modelli, ciascun modello comportamentale può corrispondere a un'ampia gamma di comportamenti simili senza essere troppo generico. I modelli sono quindi utili per trovare comportamenti simili senza il rischio di generare troppi falsi positivi.

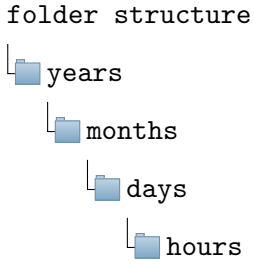
3.5 Problematiche dovute all'utilizzo di due diversi formati

Come si è potuto notare nelle sezioni *2.2* e *2.3*, gli header di *nProbe* ed *Argus* presentano delle differenze che non permettono di essere utilizzati in modo intercambiabile. I file che produce in output *Argus* presentano 17 cambi, ben 12 in meno rispetto ai file di output prodotti da *nProbe*. L'utilizzazione di due diversi formati presenta errori quando si cerca di utilizzare file prodotti da *nProbe* in Stratosphere IPS.

Questa incompatibilità ha portato alla necessità di una conversione: i file prodotti dal *DIEF* devono essere convertiti in file di formato usato da *Argus*. Questa conversione deve essere efficiente e precisa.

3.6 Presentazione del problema

Il *DIEF* salva i file in una struttura gerarchica fissa ben definita: ci sono 4 livelli di subdir, in cui il primo livello indica l'anno, il secondo il mese, il terzo il giorno e il quarto l'ora. All'interno dell'ultima subdir, quella delle ore, ci sono 60 file uno per ogni minuto della giornata.



I file dei minuti sono compressi usando il programma *gzip*, pertanto c'è da tenerne conto nella soluzione per la conversione.

Capitolo 4

Soluzione proposta

4.0.0.0.1 Preparazione environment Per lo sviluppo del programma e lo studio del framework è stata creata una piattaforma dedicata alla *Security Analytics*. L'installazione verrà effettuata su *virtual machine* dedicata con Ubuntu 16.04 LTS.

4.1 Installazione Stratosphere IPS

Sulla macchina virtuale è stato installato il framework di Stratospehere IPS. Per l'installazione si sono seguiti i seguenti passaggi:

- Installazione del programma git *2.7.4*

  ```
  $ sudo apt install git
  ```

- Clonazione repository github del framework

  ```
  $ git clone https://github.com/stratosphereips/StratosphereTestingFrame
  ```

- Installazione del programma *python-pip*

  ```
  $ sudo apt install python-pip
  ```

- prettytable *0.7.2-3*

  ```
  $ sudo apt install python-prettytable
  ```

- transaction *1.4.3-3*

  ```
  $ sudo apt install python-transaction
  ```

- persistent *4.1.1-1build2*

```
$ sudo apt install python-persistent
```

- zodb *5.4.0*

```
$ sudo pip install zodb
```

- sparse *1.1-1.3build1*

```
$ sudo apt install python-sparse
```

- dateutil *2.4.2-1*

```
$ sudo apt install python-dateutil
```

4.2 Installazione di Argus

- libpcap *1.7.4-2*

```
$ sudo apt install libpcab-dev
```

- bison *3.0.4*

```
$ sudo apt install bison
```

- flex *2.6.0-11*

```
$ sudo apt install flex
```

- Installazione dell'ultima versione di argus *3.0.8.2* dal sito http://qosient.com/argus/dev/argus-latest.tar.gz

- Installazione dell'ultima versione di argus-client *3.0.8.2* dal sito http://qosient.com/argus/dev/arg clients-latest.tar.gz

4.3 Utilizzo del programma stf

Per eseguire il programma lo si esegue con

```
./stf.py
```

```
    Stratosphere Testing Framework
            _    __
           | |  / _|
       ___| |_| |_
      / __| __|  _|
      \__ \ |_| |
    ... |___/\__|_|  ...
    0.1.2alpha


[*] Amount of experiments in the DB so far: 0
[*] Amount of datasets in the DB so far: 0
[*] Amount of groups of connections in the DB so far: 0
[*] Amount of groups of models in the DB so far: 0
[*] Amount of notes in the DB so far: 0
stf >
```

Per caricare un dataset si utilizza il comando

```
datasets -c /absolute/path/file.binetflow
```

Per generare la connessione si utilizza il comando

```
connections -g
```

Infine, per generare i modelli, il comando

```
models -g
```

Per visualizzare il behavioral model si utilizza il comando

```
models -L [id]
```

```
test: stf > models -l
[] Groups of Models
+-------------------+------------------+------------+--------------+
| Group of Model Id | Amount of Models | Dataset Id | Dataset Name |
+-------------------+------------------+------------+--------------+
| 0-1               | 446              | 0          | test         |
+-------------------+------------------+------------+--------------+
test: stf >
```

4.4 Conversione

I due formati che vengono utilizzati hanno formati diversi, si è scelto di tenere soltanto i campi utilizzati dal programma *Argus* poichè è il formato che viene utilizzato da Stratosphere. I campi di *nProbe* che quindi non compaiono nei file di tipo **.binetflow* vengono scartati.

Per i restanti campi si rimanda alla seguente tabella di conversione

Tabella 4.1: Tabella di conversione

binetflow	flow
start time	first switched
duration	last switched - first switched
protocol	protocol
source address	ipv4 source address
source port	source port
direction	biflow direction
destination address	ipv4 destination address
destination port	destination port
state	-
source tos	source tos
destination tos	-
tot packets	input packets + output packets
tot bytes	input bytes + output bytes
source bytes	input bytes
source data	-
destination data	-
label	-

4.5 Automatizzazione conversione

La conversione è stata automatizzata con la scrittura di uno script in *python*. Si è scelto di scrivere un programma usando questo linguaggio per la facilità di utilizzo nel lavorare con i file e per le performance.

Il problema richiede lo sviluppo di un programma che converte file in modalità batch.

Pseudocodice del programma

Algorithm 1 Single core version

```
1: procedure HYDRA
2:     for all file in path do
3:         read data from file
4:         convert data into new format
5:         append data into new file
```

Come descritto nel capitolo 3, i file hanno una struttra gerarchica per data. Si esegue quindi un ciclo *for* che prende tutti i file in modo ricorsivo.

Si è scelto di leggere il file una riga alla volta perchè le grandi dimensioni dei file non permettono un approccio diverso. Un approccio più veloce sarebbe stato quello di leggere i file per intero nella memoria principale ma non è possibile per le grandi dimensioni dei file.

Per ogni riga che viene letta si elimina il carattere speciale che divide i valori e li si inserisce in un vettore principale. Per ogni riga letta si ha quindi un vettore con posizioni dei campi fissi (es. l'IP sorgente sarà sempre in posizione vettore[0]).

Si crea un vettore per ogni campo che dovrà essere convertito e dal vettore principale vengono appesi i valori. Successivamente si apre un file nel formato *binetflow* e vengono convertiti e appesi i valori da ogni vettore.

4.6 Rendere efficiente la conversione

Nel programma descritto in precendenza viene generato un solo file di output in cui vengono convertiti tutti i file dati in input. Questa soluzione è comoda perchè da migliaia di file si ha un solo file con i dati convertiti, ma presenta il problema di creare un file con dimensioni enormi e di difficile gestione (il file può raggiungere dimensioni tali da rendere difficile anche solo aprirlo in lettura) su cui crearci i modelli comportamentali.

Il programma è inoltre inefficiente poichè è single core e ha come collo di bottiglia la scrittura su un unico file.

La soluzione proposta seppure sia teoricamente corretta non può avere un'applicazione nel mondo reale. Bisogna cambiare quindi strategia per rendere la conversione più veloce sfruttando le macchine multi core e per avere in output file di dimensioni accettabili.

4.6.1 Possibili soluzioni

Si possono pensare diverse soluzioni che migliorerebbero in modo significativo il programma visto in precedenza.

- Lavorare su chunk di file
- Meccanismi di lock
- Scrittura su file 1:1

4.6.1.0.1 Lavorare su chunk di file Per velocizzare il programma e sfruttare i processori disponibili si potrebbe assegnare ad ogni processore un file da leggere e convertire. Quando il processore termina la conversione dei dati scrive sul file in output. In questo modo si dividrebbe il tempo di esecuzione sul numero di processori disponibili. Questa soluzione rende il più veloce possibile la conversione dei file ma i processori finiscono per scrivere sullo stesso file senza avere nessuna regola di precedenza, questo crea problemi perchè le scritture in output non sono ordinate e non è possibile creare modelli comportamentali affidabili. Le scritture su file devono essere ordinate e sequenziali. Inoltre questa soluzione ha il problema di scrivere sempre su unico file e come detto in precedenza questo tipo di soluzione non è possibile.

4.6.1.0.2 Meccanismi di lock Un modo per risolvere i problemi precedenti è quello di utilizzare il concetto di semaforo.

In informatica un semaforo è un tipo di dato astratto gestito da un sistema operativo multitasking per sincronizzare l'accesso a risolrse condivise tra processi. È composto da una variabile intera e da una coda di processi. Quando un processo apre il file per scriverci viene impostato un semaforo che segnala che la risorsa è occupata, se un altro processore prova ad aprire lo stesso file per scriverci gli sarà negato l'accesso dal semaforo fino a quando l'altro processo non rilascerà la risorsa.

In questo modo si risolve il problema delle scritture ordinate, ma c'è da tener conto che i semafori riducono la velocità di esecuzione dell'algoritmo poichè mentre un processore occupa la risorsa tutti gli altri processori devono mettersi in

coda per aspettarne il rilascio. Questa soluzione sebbene rallenti l'esecuzione del programma rispetto alla soluzione precedente è comunque molto più veloce della versione single core perchè si guadagna comunque molto tempo nella lettura e conversione dei dati che viene effettuata alla massima velocità possibile. Questa soluzione risolve anche il problema dell'ordinamento delle scritture.

Rimane il problema della scrittura su un unico file che però può essere risolto facilmente decidendo di scrivere su un nuovo file quando raggiunge una dimensione specificata.

Se si implementa la divisione del file di output questa soluzione può considerarsi efficiente anche se rimane il collo di bottiglia introdotto dai semafori che costringe i processi ad aspettare in coda quando una risorsa è impegnata.

4.6.1.0.3 Scrittura su file 1:1 In questa soluzione ogni processore apre un file, lo legge. lo converte e scrive la conversione su un proprio file. Questa soluzione è decisamente la più semplice tra quelle proposte ed è anche la più efficiente poichè i processori non entrano mai in conflitto tra di loro cosicchè da effettuare letture, conversioni e scritture alla massima velocità possibile dalla macchina. Un problema che può creare questa soluzione è la produzione di una grande quantità di file in output. Si pensi di dover una sola settimana di traffico di rete, sono circa 10 mila file.

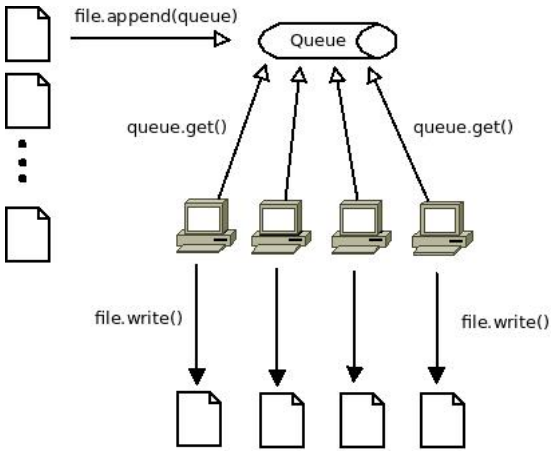
4.6.2 Scelta effettuata

Si è scelto di procedere con l'ultima soluzione presentata perchè data la grande quantità di informazioni da convertire si preferisce l'approccio più veloce a discapito dell'ordinamento finale.

Pseudocodice del programma multi core

Algorithm 2 Multi core version

```
1: procedure HYDRA
2:     for all file in path do
3:         Queue[] ← file
4:         read data from file
5:         spawn 4 process
6:         while Queue[] not empty do
7:             filename ←Queue.get()
8:             convert data into new format
9:             append data into new file
```



Il programma, dato in input la root directory dei file, cerca ricorsivamente tutti i file e li inserisce in una coda, chiama poi una funzione a cui assegna 4 processori. La funzione ha un ciclo while che si ripete fino a quando la coda popolata da tutti i file non è vuota. Ogni processore che entra in questa funzione toglie un file dalla coda e lo elabora.

Capitolo 5

Esperimenti e risultati

5.1 benchmark single core

Sono stati effettuati 10 test in entrambe le modalità. I risultati vengono riportati a seguire. tutti i benchmark sono stati effettuati con */usr/bin/time*

1. 2:28:03 98%CPU
2. 2:27:98 98%CPU
3. 2:28:86 98%CPU
4. 2:28:97 95%CPU
5. 2:24:52 99%CPU
6. 2:34:21 94%CPU
7. 2:28:36 98%CPU
8. 2:35:02 93%CPU
9. 2:25:59 99%CPU
10. 2:28:75 96%CPU

La media calcolata è quindi di **2:28:38 97%CPU**

5.2 Benchmark multi core

Sono stati effettuati 10 test in entrambe le modalità. I risultati vengono riportati a seguire. Tutti i benchmark sono stati effettuati con */usr/bin/time*

1. 0:54:82 265%CPU
2. 0:36:13 375%CPU
3. 0:38:24 367%CPU
4. 0:40:25 377%CPU
5. 0:40:14 368%CPU
6. 0:44:28 351%CPU
7. 0:41:67 368%CPU
8. 0:40:72 383%CPU
9. 0:40:83 381%CPU
10. 0:41:19 382%CPU

La media calcolata è quindi di **0:41:83 362%CPU**

Capitolo 6

Conclusioni

6.1 Prestazioni

Sia *T(p)* il tempo di esecuzione in secondi di un certo algoritmo su *p processori*. Di conseguenza sia *T(1)* il tempo di esecuzione del codice parallelo su 1 processore. La *misura di scalabilità* o *speedup* relativo di un algoritmo parallelo eseguito su p processori si calcola come:

$$S(p) = \frac{T(1)}{T(p)}$$

Con i risultati ottenuti si ha uno *speedup relativo* di

$$S(p) = \frac{T(148)}{T(40)} = \mathbf{3{,}7}$$

In un sistema ideale, in cui il carico di lavoro potrebbe essere perfettamente partizionato su p processori, lo speedup relativo dovrebbe essere uguale a p. In questo caso si parla di **speedup lineare**.

Si definisce *efficienza* il rapporto

$$E(p) = \frac{S(p)}{p}$$

Idealmente, se l'algoritmo avesse uno speedup lineare, si avrebbe $E(p) = 1$

Più l'efficienza si allontana da 1, peggio stiamo sfruttando le risorse di calcolo disponibili nel sistema parallelo.

$$E(p) = \frac{S(3{,}7)}{4} = \mathbf{0{,}925}$$