UNIVERSITÀ DI PISA
Dipartimento di Informatica
Corso di Laurea Triennale in Informatica

RILEVAZIONE DI MALWARE TRAMITE L'IDENTIFICAZIONE DI COMPORTAMENTI ANOMALI

RELATORE
Luca **DERI**

CANDIDATO
Alessio
PERUGINI

ANNO ACCADEMICO 2019/20

Indice

Capitolo 1

Introduzione

L’aumento degli attacchi informatici ha reso necessario lo studio e lo sviluppo di nuovi sistemi di difesa. Ci concentreremo sull’analisi del traffico di rete. Osservare il traffico, ci aiuta ad identificare e distinguere comportamenti legittimi, illegittimi e sospetti. Analizzando il traffico generato da applicazioni P2P, mail client, giochi online ed altro, noteremo dei pattern che tendono a mantenere una certa periodicità con i server con cui comunicano. Questa costanza nella loro comunicazione può presentarsi anche da parte di applicazioni malevoli. Quando nella nostra rete compare l'attività di un malware è molto frequente l'attività definita come: beaconing. Il beaconing consiste nella comunicazione tra il client infetto e il server di C2 (Command and Control). Il C2 ha lo scopo di inviare le istruzioni che deve eseguire la macchina infetta. Dalla costante evoluzione degli attacchi, è aumentata anche la prevenzione e rilevazione di questi. Sono stati realizzati sistemi come IDS (Intrusion Detection System) e IPS (Intrusion Prevention System) nel tentativo di limitare e proteggere i propri device.
Una panoramica, sulla tipologia dei malware, è essenziale per comprendere come questi possono variare i loro comportamenti in maniera dinamica. Lo studio, delle attività che svolgono al livello di rete, ci aiuta a capire se è possibile trovare una correlazione tra pattern periodici e presenza di bot/malware. La differenza, tra attacchi burst e slow, mette alla luce eventuali criticità e considerazioni, da affrontare durante la progettazione di sistemi di rilevazione di comportamenti anomali.
L’idea è quella di utilizzare le informazioni proveniente da algoritmi che trovano la periodicità in una comunicazione, nel nostro caso un algoritmo computazionalmente leggero, al fine di analizzare eventuali comportamenti anomali e/o minacce all’interno di pattern periodici. Non andremo ad esaminare sistemi che utilizzano il Machine Learning, in quanto già ben documentati in letteratura. Sarà approfondito lo stato dell’arte, con particolare attenzione alla rilevazione della periodicità.
La discussione e l’analisi di alcuni malware, hanno lo scopo di dimostrare come questi possano lasciare tracce periodiche. Tuttavia la correlazione tra il nostro

algoritmo e l'applicazione per la rilevazione di malware, verrà lasciato come lavoro futuro.
In questo lavoro, presentiamo un algoritmo per rilevare la periodicità del traffico di rete. È stata realizzata una sua implementazione, con un modulo per effettuare una DPI sui flussi sospetti, così da fornire una base per un eventuale sviluppo di questo lavoro.

1.1 Motivazione e obiettivo

Il miglioramento delle tecnologie in ambito AI è stato sposato per realizzare nuovi sistemi di difesa che prevedesse il loro utilizzo, a volte congiunto con altre tecniche quali le Signature Based (basate su una firma del comportamento del malware) per aumentare l'efficacia della rilevazione di minacce. Il lato negativo nell'utilizzo di tecniche AI/ML è quello del loro elevato costo computazionale, richiedendo moltissime risorse che non tutti sono in grado di avere e che per rilevare determinati tipi di attacco potrebbe risultare essere eccessive.
Nell'ambito della sicurezza informatica, sono impiegati algoritmi per rilevare la periodicità del traffico di rete, nel tentativo di realizzare e modellare sistemi che segnalino comportamenti che si discostano da un comportamento definito normale. Il problema di questi algoritmi ricade nella loro complessità e domanda di risorse di calcolo molto onerose. Un esempio è quello nell'utilizzo del Periodogram[1,2] ossia una stima della densità spettrale di un segnale per rilevare comportamenti anomali.
In un ambiente dove gli attacchi si fanno sempre più sofisticati variando in modo molto dinamico il loro comportamento, è importante realizzare sistemi che ci avvertano in realtime di una potenziale minaccia. È proprio nel monitoring a tempo reale, che assume un'importanza fondamentale l'uso di algoritmi leggeri ed efficienti, il più vicino possibile alle soluzioni precedentemente discusse che costituiscono un ottimo strumento per migliorare ed investigare in offline il traffico ricevuto, da utilizzare quindi a supporto dei sistemi real time.
Nella letteratura attuale, viene utilizzato il ML per modellare il comportamento di una determinata famiglia di malware o di bot specifici, nelle comunicazione con i server di Command and Control. Quello che risalta molto spesso da queste analisi è una forte correlazione tra comunicazioni temporizzate che seguono, specialmente nei malware più semplici, pattern ben definiti.
L'obiettivo è quello di mostrare come sia possibile rilevare un traffico periodico senza l'utilizzo di algoritmi computazionalmente onerosi. A seguito di riscontri periodici, andare a verificare se questo traffico sia malevolo o no. Una volta stabilite le comunicazioni che mostrano pattern periodici, effettuare eventuali analisi di DPI dei flussi interessati. In sostanza, utilizzare le informazioni che ci da la periodicità, per verificare se questa sia una potenziale minaccia per la nostra rete.

Capitolo 2

Stato dell'arte

Questo capitolo si concentrerà sugli approcci che vengono attualmente utilizzati per identificare traffico malevolo generato da bot e malware.
Nell'ambito della sicurezza informatica si fa un ampio utilizzo del ML per realizzare modelli in grado di identificare specifici malware e specifici attacchi.
Andremo ad affrontare lo stato dell'arte delle principali tecniche di detection di vari attacchi e malware.

2.1 Signature based

Utilizzato principalmente in sistemi IDS e IPS. Si basa sulla creazione di regole o firme ed espressioni, utilizzate per identificare determinate sequenze di byte o pacchetti nel traffico di rete. I sistemi open-source più famosi sono Suricata e SNORT, esistono anche delle soluzioni a pagamento che prevedono apparati hardware con del software IDS/IPS preinstallato affiancati anche a dei firewall.
Questo tipo di regole sono scritte appositamente per farle combaciare con il traffico generato da una minaccia, questo prevede la conoscenza a monte del suo comportamento. Una volta studiata la minaccia, si crea una regola specifica che ci consenta di identificarla in modo molto accurato. Lo svantaggio di questo approccio consiste nella necessità di conoscere il comportamento del malware. A fronte di nuovi attacchi, sarà necessario reperire dei campioni per studiarli e creare delle regole sui risultati ottenuti. Un'altro lato negativo è dovuta alla crittografia, sempre in continuo aumento, che mettono in crisi i sistemi DPI che non riescono ad identificare in modo accurato il vero payload dei pacchetti, rendendo necessaria un'analisi di tipo statistica.

2.2 Behavior based

Un sistema behavior-based (Anomaly-based) è un sistema IDS che si basa su una base di dati appresa nel tempo. L'obiettivo è utilizzare questa base di conoscenza per identificare tentativi di intrusione, che andrebbero ad interferire con il normale

comportamento della rete. Alcune implementazioni utilizzano il ML per realizzare tali sistemi, molto gettonati gli algoritmi di classificazione e regressione che utilizzano il "Random forests" [5,6]

2.3 Honeypot based

Basati sulla creazione di macchine vulnerabili, vengono utilizzate dai ricercatori per identificare nuove minacce e procurarsi tutto il materiale necessario per realizzare e migliorare i sistemi di difesa. La creazione di questi sistemi può essere effettuata installando sistemi windows/linux/osx che presentano delle vulnerabilità note al ricercatore. Una volta violate, si avrà accesso ad un'enorme fonte di dati quali: IP, eseguibili, pattern di attacco, tecniche utilizzate e molto altro. Esistono molte tecniche per identificare ambienti honeypot [3], diminuendo così la capacità di identificare nuove minacce.

2.4 Periodic network traffic

Il traffico di rete presenta delle periodicità dovute a programmi che effettuano delle azioni programmate. Nel caso in cui un host delle rete venisse compromesso, si potrebbe assistere a delle comunicazioni periodiche verso il server C&C (Command and Control), che agisce come controllore degli host infetti denominati "bot". I malware meno sofisticati, tendono ad inviare richieste ai server C2 con scadenze che risultano essere facilmente identificabili come periodiche. Sotto questo concetto di periodicità, sono state svolte molte ricerche come quelle dell'utilizzo del *Periodogram* [2,23], mediante una stima della densità spettrale di un segnale riesce a fornire dei dati che consentono di identificare eventuali periodicità.

N. Hubballi e D. Goyal [13], propongono un approccio basato su un riepilogo dei dati per dataset molto grandi. Il loro metodo consiste nel rappresentare un grande insieme di punti di dati in un formato compresso di tipo lossless. La tecnica si basa sul fatto che le comunicazioni periodiche mostrano bassa varianza ed alta deviazione standard del loro tempo di interarrivo. Mentre i flussi di natura randomica mostrano un'alta varianza. Quindi utilizzano la SD delle comunicazioni stabilite tra 2 host, come misura per decidere se hanno una comunicazione periodica o no. Il loro algoritmo è governato dalla soglia della SD che va scelta dal system administrator, tenendo conto del tipo di traffico che riceve.

M. Haffey ed altri [14] utilizzano un approccio basato su SQL da utilizzare come modello computazionale per la rilevazione di traffico periodico. Si concentrano sull'analisi di traffico proveniente da un edge network, contenente solo dati di

utilizzo reale senza realizzare dataset artificiali. Molti ricercatori a volte valutano le nuove tecniche utilizzando dati artificiali che escludono traffico periodico legittimo [15,16]. L'analisi e la caratterizzazione del traffico utilizzando l'approccio SQL fa delle assunzioni per stabilire se il traffico è periodico. Utilizzando 5 parametri quali: L'evento del network, il numero minimo di eventi, il periodo minimo e massimo e la soglia della varianza. Concludono che il traffico periodico è molto presenta in ambito videoludico, CDN, P2P, servizi basati sul cloud e traffico malevole (nel caso specifico ZeroAccess e Sality Botnet). Le applicazioni P2P compongo circa le metà della totalità del traffico periodico.

G. Bartlett ed altri [17] si concentrano nel trovare flussi con bassa periodicità attraverso la tecnica di signal-processing utilizzando wavelets discrete. L'estrazione dei timeseries è fondamentale per poter applicare il metodo. Si procede nel generare una serie temporale di eventi che rappresentano il traffico di rete. Viene scartato ciò che non ci interessa e si mappano gli eventi d'interesse dentro una serie temporale fissa di eventi per periodo di tempo. L'evento d'interesse, viene considerato quando arrivano i flag SYN-ACK. Si procede alla creazione di bin di durata prefissata: n=durata/timebinsize. Una volta ottenuto la serie temporale di eventi si passa all'analisi multi-resolution utilizzando le Haar wavelets. I coefficienti dei wavelet rappresentato l'energia per una certa serie X in un percorso P. Sfruttando le proprietà delle wavelet di Haar è possibile aumentare o diminuire una frequenza nell'albero del filtro. Diventa così possibile utilizzare la Frequenza e l'energia per rilevare la periodicità. Concludono che è possibile utilizzare questo approccio per controllare il proprio traffico per monitorare cambiamenti inaspettati.

Quelli sopra citate, sono tra le più note tecniche utilizzate per rilevare periodicità, senza l'utilizzo del ML. La letteratura ha approfondito molto l'applicazione di tecniche di ML, per studiare e realizzare modelli, in grado di identificare comportamenti più o meno specifici di famiglie di malware. Ma non c'è stato ancora uno studio approfondito, nel trovare una comunicazione malevola a partire da un'informazione generale quale la periodicità.

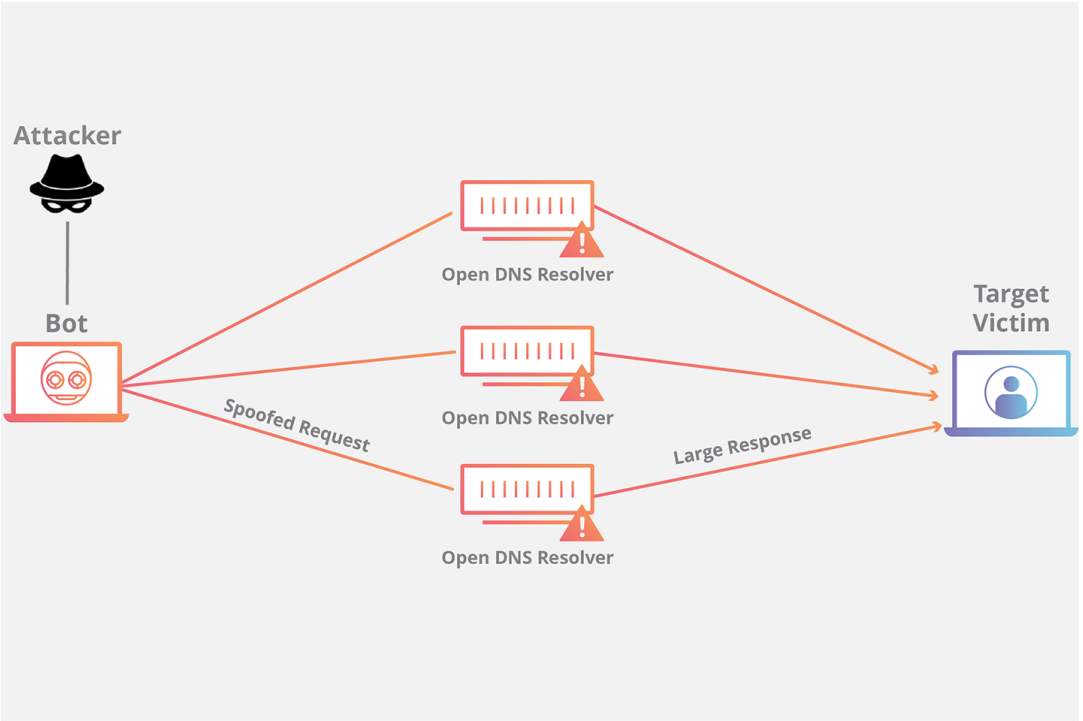
2.5 DDOS Detection

Il termine DDOS (Distributed Denial of Service) indica un tipo d'attacco che ha lo scopo di rendere irraggiungibile un server, un servizio o un'infrastruttura.
Gli attacchi DDOS possono variare per intensità e tecnica d'utilizzo. Quelli più comuni sono attacchi volumetrici, basati sull'intensità e sul numero di pacchetti che vengono inviati in un breve lasso di tempo.
Questo attacco può essere effettuato in L3, L4 ed anche utilizzando un mix di questi.

I SYN Flood attacks, sfruttano il design dell'algoritmo three-way del TCP. Una volta ricevuto il SYN flag la sessione viene aperta fino a quando non viene chiusa dal client. Questo fa sì che un attacco massiccio porta all'esaurimento della connection table memory, scartando così eventuali richieste legittime.

Con la diffusione del protocollo http non sono mancati anche attacchi costruiti su misura. Uno di questi è slowloris. L'attaccante apre numerose connessioni http simultanee cercando di mantenere la connessione sempre aperta. Questo tipo d'attacco è considerato basso e lento, in quanto ideato per mandare in down un server mediante una singola macchina.

DNS Amplification sfrutta la grande disponibilità di banda dei server DNS per attaccare a loro insaputa i propri target. Questo attacco prevede di forgiare delle richieste dns utilizzando l'ip della vittima, che genera risposte molte grandi. In questo modo il server DNS considera legittima la richiesta ed invia la risposta alla vittima. La disparità di banda che c'è tra l'attaccante e la vittima consente di inviare richieste dns molto piccole a fronte di una risposta molto grande. Questo consente al bot di inviare molte più richieste, che si tradurranno in risposte spropositatamente più grandi, portando la vittima ad un eventuale downtime. Questo tipo di amplificazione consente di mandare query di 500 bytes e ricevere risposte di 3-4 kB [12].



Le mitigazioni sono molteplici. La più efficace è quella di controllare che l'IP sorgente non sia un IP spoofed. Questo può essere effettuato da parte degli ISPs che possono rifiutare tutti i pacchetti del proprio traffico interno che contengono un ip spoofed.

Un’altro modo è quello di ridurre il numero dei resolver DNS e di garantire il servizio solo ai dispositivi provenienti da una zona sicura.

Un altro attacco il Ping Of Death che consiste nell’invio di un pacchetto IP malformato, al fine di causare un buffer overflow con relativo disservizio [11,12].
S. Pande ed altri hanno realizzato una tecnica che utilizza il ML per prevenire questi tipi di attacchi. Nel caso specifico hanno utilizzato Random Forest (RF) utilizzato come classificatore. Questo restituisce diversi alberi di decisioni. Ogni albero viene costruito da avvii iniziali alternativi a partire dalla prima informazione utilizzando un classificatore ad albero.

Per altri tipi di DDOS e rilevamenti vi riporto alla lettura di Z. Al-Jarrah [12] che affronta ed approfondisce una dettagliata collezione di tecniche.

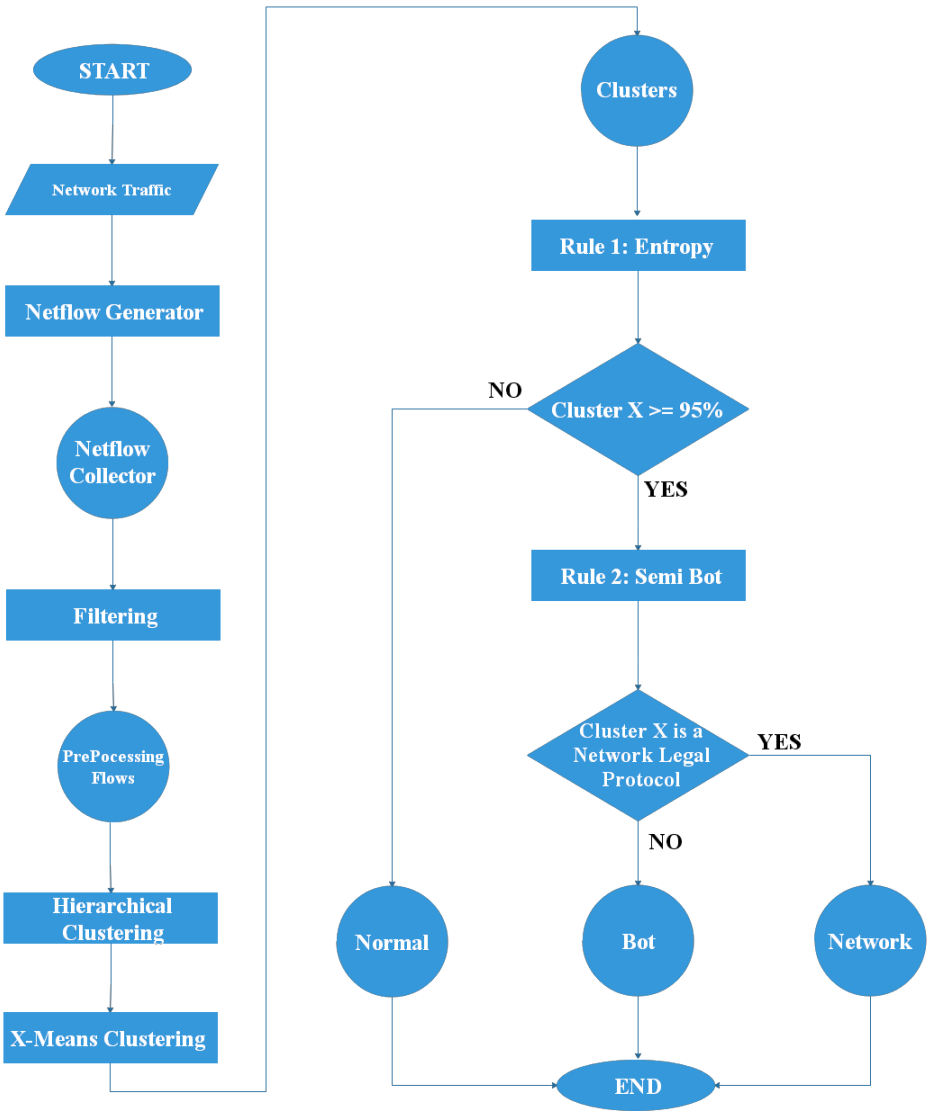
2.6 SHAPING MALWARE CON ML

In letteratura è stato approfondito l’utilizzo del ML per identificare i comportamenti anomali di determinati malware e botnet. Le tecniche utilizzate mirano ad effettuare uno shaping, ossia studiare e rilevare comportamenti propri a distinguere uno specifico malware dal resto del traffico legittimo. Citeremo alcuni dei studi più popolari che essenzialmente si distinguono per l’utilizzo di modelli e classificatore diversi.

Nell’articolo *“Using Machine Learning Techniques to Identify Botnet Traffic”[22],* viene presentato un approccio all'identificazione del traffico di una botnet IRC. Il loro lavoro è basato su una prima parte di creazione del dataset, in questo caso ottenuto dal traffico del loro campus, e da un’ultima fase che prevede la scelte del classificatore. I classificatori presi in considerazione sono J48, naive Bayes, e Bayesian networks. Fatta una comparazione delle performance, hanno verificato che il naive Bayes identificata in modo più equo il traffico problematico.

P. Amini e altri in *“Analysis of Network Traffic Flows for Centralized Botnet Detection”*[24] propongono un approccio per identificare le botnet centralizzate che utilizzano HTTP. Nel caso specifico si parla di Zeus1, Zeus2 e Citadel. Utilizzano un clustering gerarchico, un clustering con X-Means e una classificazione basata su regole (rule-based). Ognuno di questo metodo è stato utilizzato per un scopo preciso. Il clustering gerarchico per migliorare la velocità e l'accuratezza, nella separazione dei flussi netflow. X-Means, per aumentare la coesione dentro il cluster, ed aumentare la distanza dagli elementi esterni. La classificazione rule-based per identificare cluster di tipo bot, semi-bot e non-bot. I risultati ottenuti

ottengono un accuracy del 95%. Il metodo che propongono è sintetizzato nel flow chart seguente:



I dettagli dell'implementazione sono disponibili nel paper sopra citato [24].

"Malware Detection Based on Mining API Calls"[26] è una pubblicazione che utilizza un metodo per la detection attraverso delle API. La loro tecnica consiste nel leggere le chiamate API utilizzate in una collezione di file. Successivamente genera un set di possibili feature. Queste caratteristiche verranno poi utilizzate per il training del classificatore, che dovrà scoprire malware non visti. Utilizzano Random Forest come classificatore hanno verificato che mostra la miglior perfomance. Il risultato dello studio conclude con il raggiungimento di una detection pari al 99,7% con un accuracy del 98.3%.

Capitolo 3

Tipi di malware

Nel mondo dei malware si tende a fare una distinzione tra quelle che sono le classi e le famiglie. Una classe identifica un certo comportamento che può essere comune alle famiglie che vi appartengono. Una famiglia è un malware che appartiene ad una classe e che avrà dei comportamenti più o meno riconoscibili. Vi sono molti studi su come classificare i malware, ognuno utilizza approcci leggermente diversi ma che vengono accomunati dall'utilizzo del ML [27, 28, 29]. In questo capitolo affronteremo una descrizione delle famiglie e delle classi dei malware in cui si è concentrata la ricerca e lo sviluppo del software di questo tirocinio. Va inoltre ricordato che non tutti i malware hanno la stessa gravità. Esistono malware creati solo per il gusto di infastidire l'utente, dove non vi è alcuno scopo criminale, ed altri con l'intento di rubare informazioni utili per realizzare un profitto. I malware su cui ci soffermeremo saranno quelli forgiati appositamente per azioni criminali e di APT.

3.1 Classi di malware

Una classe di malware identifica un comportamento generico. Mediante queste classi è possibile realizzare una panoramica generale della gravità della minaccia. È possibile creare classi di malware più o meno specifiche, le più comuni sono: Adware, Spyware, Worm, Trojan, Rootkit, Phishing, Bot, Ransomware e Backdoor.

3.1.1 Bot

Un *bot* è un host infetto che entrerà a far parte di una rete di altri infetti, formando così una *botnet*. Una botnet è una rete di bot controllata dal *botmaster*, colui che decide le azioni che devono eseguire le loro vittime. Un bot comunica principalmente con un server chiamato *C2* o *C&C* (Command and Control) che si occupa di impartire le azioni da eseguire. In base a quanto è sofistacata una botnet questa può essere equipaggiata con molti tool che possono essere di qualsiasi tipo quali: ransomware, DDoS, Keylogging e molto altro. La pericolosità di questi

malware risiede nel far parte di una rete in cui noi stessi possiamo partecipare attivamente ad un attacco verso un target innocente. Il tutto senza la nostra consapevolezza. Un esempio molto comune è quello di ritrovarsi l'ip della propria macchina segnalato come spam. Com'è possibile? Grazie ad altri bot che, attraverso bruteforce o altre tecniche, riescono ad entrare su un nostro server, solitamente mal configurato, effettuano azioni illegittime verso altri server. Gli ip considerati come spam sono, solitamente dovuti ad attacchi volumetrici quali DDoS od email spam.

3.1.2 Trojan

Un *trojan* è un programma che nasconde il suo reale comportamento dietro false apparenze. Viene utilizzato principalmente per installare una *backdoor* che consente di ottenere il controllo della macchina. Una volta ottenuto il controllo si procede con l'esecuzione e/o download di altri malware che possono variare di ogni tipo. In genere gli attacchi vengono per phishing o per social engineering, in entrambi i casi si cerca di ottenere l'accesso ad una parte della rete e procedere poi con mosse laterali per assumerne il controllo più completo possibile.

3.2 Esempi di malware

Per analizzare i comportamenti di classi e famiglie di malware, è necessario studiarli procurandosi una copia e avviandola in dei laboratori, su cui è possibile indagare in modo più approfondito. In questo capitolo andremo ad analizzare 2 campioni molto diffusi. Il primo riguarda Zeus, uno dei malware bancari più diffusi nel decennio scorso, fino alle nuovo varianti che continuano a vivere ancora oggi. Il secondo riguarda lo studio di una nuova minaccia denominata "Sunburst", apparsa il 20 Dicembre del 2020. Questi esempi ci consentono di individuare alcune caratteristiche che andremo poi a generalizzare e ad approfondire nei capitoli successivi.

3.2.1 Zeus

Zeus (Conosciuto anche come Zbot) è un malware bancario apparso per la prima volta nel 2006 a seguito di una pubblicazione del reverse engineering del trojan PRG. È stati sviluppato per trarre profitto anche dal suo sviluppo attraverso un design modulare, consente di personalizzarlo e venderlo ad una fetta di mercato più ampia. L'obiettivo è quello di rubare informazioni sensibili per effettuare frodi fiscali. Il suo ecosistema è caratterizzato da 3 entità: bot, C&C e il server di configurazione. I dati che ruba sono inviati ad una *dropzone* attraverso canali

differenti. Può utilizzare sistemi con frequenza d'invio variabile da 2 a 10 minuti. Le prime varianti sono basate su un server C&C centralizzato che vengono bloccati da le community di sicurezza. L'ultima variante denominata *GameOver Zeus* si basa su una comunicazione *peer-to-peer*. Evoluto anche nelle capacità d'attacco che vanno oltre quelle dei comuni trojan bancari per spaziare ad attacchi DDoS, malware dropping e bitcoin mining. La prima variante, risulta avere un periodo di connessione pari a 5 minuti ed un keep-alive di 20 minuti. GameOver presenta un keep-alive di 13 minuti [27].

3.2.2 Sunburst

È un malware backdoor molto recente, scoperto nel dicembre del 2020.Siamo di fronte ad una minaccia molto sofisticata, che utilizza le più alte tecniche di intrusione esistenti [33, 34, 35].
La sua diffusione, è stata resa possibile dalla compromissione dell'ambiente di sviluppo della fase di build del software Orion di SolarWinds.
A livello di rete, ciò che facilita la sua rilevazione è l'utilizzo dei domini usati per comunicare con i C2. Il loro algoritmo, di generazione di domini, utilizza pattern che mimano la nomenclatura dei cloud host, come quelli di amazon. Nella generazione includono anche dettagli riguardante la vittima. Questo ha favorito ai criminali, di decidere quale bersaglio seguire e ai ricercatori di realizzare tool per decodificare le richieste DNS.
La problematica maggiore è il periodo di latenza del malware. La backdoor rimane in uno stato dormiente per un periodo dai 12 ai 14 giorni. Questo rende difficile individuare il momento esatto in cui si è stati compromessi. Ulteriori controlli e randomizzazioni, vengono effettuate prima di intraprendere nuove azioni. Questi prevedono la ricerca di determinati tool e processi, utilizzati per la ricerca di malware. Se i test rilevano la presenza di software di sicurezza, si procede con il tentativo di disabilitarli e riavviare la backdoor. Una volta superati i sistemi di controllo, si passava all'invio delle informazioni ad un sottodominio di *avsvmcloud.com*. Sunburst controlla anche che l'ip, che viene restituito da una certa richiesta, sia in un certo range, altrimenti termina il programma.
Anche l'utilizzo dei server di C2 viene cambiato in modo dinamico dall'attaccante. Se questo considera la vittima particolarmente interessante, può decidere di cambiare il C2 con cui comunica. In questo modo è possibile scegliere il server equipaggiato con più tool e comandi.
Il motivo per cui è stato particolarmente difficile rilevare questa backdoor, è dovuta anche al mascheramento del traffico, reso molto simile al software Orion Improvement Program (OIP). Ogni attacco è effettuato utilizzando l'ip della stessa nazione della vittima.

Tutti questi accorgimenti hanno reso molto complicata la detection di Sunburst ed ancora non è stato possibile realizzare un “kill switch” completo.
Questo è l’esempio di come un attacco di tipo “Slow” (Capitolo 4.2) sia molto difficile da individuare. Anche a fronte di sistemi avanzati non è semplice capire queste minacce subdole, ed quindi fondamentale una costante analisi, ricerca e sviluppo di sistemi in grado di segnalare quanto prima possibile eventuali breach.

Capitolo 4

Tipi di comportamento dei malware

I comportamenti che può assumere un malware dipende da quanto questo sia sofisticato e dall'obiettivo del target. I malware più semplici non passeranno inosservati anche in assenza di sistemi di monitoraggio specifici, in quanto effettueranno grossi volumi di azioni anomale. Quelli più ingegnerizzati, sono un vero problema, in quanto tenderanno a nascondersi. Utilizzando tecniche sofisticate ed eventuali zero-day.

4.1 Burst

Azioni, che comportano un elevato carico nel sistema di rete, in un lasso di tempo molto breve, sono considerati di tipo burst. Solitamente sono attacchi DDoS, nel tentativo di saturare la rete e mettere offline vari servizi. Bruteforce su porte standard, quali le RDP e le SSH. Nel tentativo di indovinare le credenziali, vengono effettuati grandi quantità di login in poco tempo. Questo tipo di attacchi sono piuttosto evidenti e molto semplici da identificare. Vengono spesso utilizzati anche per esfiltrare dati importanti in poco tempo, dove l'obiettivo non è rimanere nascosti, ma riuscire a prendere quante più cose nel breve tempo possibile, richiedendo un eventuale riscatto.

4.2 Slow

I malware che ricadono nella categoria Slow sono solitamente più sofisticati e con molta probabilità appartenenti a delle APT (Advanced Persistent Threat). Una volta che riescono a violare un sistema questi rimangono in una stato di riposo per periodi non definiti. Quando è necessario comunicare con i server C2, vengono effettuate richieste con tempo randomizzato rendendo complicata la loro rilevazione. Nascondono le loro comunicazioni con il botmaster, cercando di simulare traffico legittimo, tentando di eludere così eventuali sistemi di score attribuiti a vari sistemi di IDS. In questa categoria è comune vedere una rotazione di indirizzi ip, riducendo ulteriormente sospetti su indirizzi ip richiesti in modo

ricorrente. Sono molto subdoli e pericolosi, tuttavia combinando tecniche miste tra Behaviour-based e Signature-based è possibile individuarli, il problema più grande è individuarli in tempo piuttosto brevi, operazione non banale che richiede un continuo aggiornamento delle regole e del training dei nostri modelli di ML.

Capitolo 5

PERIODICITÀ

La periodicità è un segnale oggettivo che va saputo interpretare. Infatti questa non può essere sempre considerata come un aspetto negativo. In molti sistemi informatici sono presenti software che effettuano polling. Un esempio è quello di verificare se una macchina o un servizio è andato offline. Se andassimo ad ispezionare il loro traffico, noteremo che sarà presente un tracciato periodico corrispondente con i timer di polling che abbiamo impostato.
Il malware beaconing rientra in comportamenti periodici. Il beaconing è una comunicazione ricorrente, che effettua il software malevolo con uno o più server di C2. Rilevare questo tipo di attività ci fornisce un indizio molto importante su cosa sta succedendo nella nostra rete. Riuscire ad identificare eventuali beaconing, specialmente in tempi rapidi, aiuta a capire che abbiamo subito un'intrusione e ci da anche indizi su quale macchina sia stata compromessa.

5.1 Algoritmo di rilevazione

Nel capitolo 2.4 sono state affrontate alcune delle tecniche utilizzate per la rilevazione di periodicità. Alcune utilizzano il ML, altre utilizzano algoritmi basandosi sulla SD, altre ancora mediante sql stabiliscono dei parametri nei quali deve rientrare un certo comportamento. Mentre nei capitoli 3 e 4 sono trattate alcune dei macro comportamenti di questi bot. Questi approfondimenti, sono necessari per capire le similarità che possono avere determinati attacchi, al fine di sviluppare un algoritmo che sia in grado di identificarli. Tuttavia, quello che non è ancora stato ben affrontano nella letteratura, è la possibilità di avere un algoritmo, che si auto regoli in real time, a fronte di cambi di comportamento repentino da parte di un malware. Quello che propongo io è utilizzare il concetto di finestra temporale per valutare la durata di un flusso.
La scelta di utilizzare l'analisi a flussi, invece che quella a pacchetto meglio conosciuta come DPI, è motivata dalla minore quantità di informazioni da dover processare. Eliminare il payload e le informazioni multiple riguardanti gli header di una comunicazione, che prevede lo scambio di molti pacchetti, aiuta a diminuire il

carico di lavoro che deve effettuare la macchina che effettua il calcolo della periodicità. NetFlow aggrega i pacchetti e mantiene solo informazioni generali della comunicazione, scartando la parte del payload. Il lavoro di raggruppamento può essere eseguito da sonde presenti direttamente nei router, ottimizzando ulteriormente l'efficienza della generazione di questi flussi. Dovendo effettuare l'analisi, su un ridotto numero di dati, aiuta considerevolmente a ridurre sia i costi computazionali, (evitando calcoli ripetuti) sia i costi di mantenimento della macchina.

Nel mio algoritmo considero flussi esportati in formato netflow, considero eventuali comunicazioni periodiche solo tra flussi che sono completati. Per il calcolo della finestra temporale non vengono considerati flussi ancora in corso o in idle. Nell'istante in cui ricevo un flusso terminato avrò una tripla <IpSorgente, IpDestinatario, PortaDestinatario>. Per ogni flusso terminato mi segno alcune informazione quali: la prima volta che l'ho visto, l'ultima e la durata della finestra.
Per effettuare il calcolo della finestra temporale ho bisogno di ricevere 2 flussi con la stessa tripla. All'arrivo del secondo flusso la finestra viene così calcolata:

$$TW = F_{2\ firstseen} - F_{1\ lastseen}$$

Una volta ricevuti i primi 2 flussi necessari per inizializzare la TW, si procede al calcolo di questa in modo dinamico. Ad ogni nuovo flusso viene aggiornata la finestra considerando il tempo di arrivo del nuovo con il precedente.
Durante le comunicazioni potrebbero esserci dei ritardi, causati da molti fattori: come ad una momentanea congestione della rete o ad una perdita di pacchetti da parte della sonda e/o collezionatore. Bisogna tenere in considerazione delle tolleranze per cui noi stabiliamo valido un flusso.
Aggiungendo una tolleranza alla TW, in cui ci aspettiamo ricadere una comunicazione periodica, ci assicuriamo che eventuali errori o cambi non troppo repentini di comunicazione, vengano intercettati.

Possiamo fare alcune considerazioni su questo approccio.
La prima riguarda il calcolo della finestra ad ogni nuovo flusso. Con l'attuale implementazione rischiamo di dare troppa importanza ad un valore che si discosta di molto dai precedenti. Se ho segnato come periodici 100 flussi di quella tripla con una TW di 20s ed arriva un nuovo flusso con TW = 3s e successivamente ritorna ai valori precedenti, andrò a perdere il successivo flusso ed avrò la necessità di aspettarne un nuovo ancora per tornare allo stato atteso. Può sembrare una banalità ma se dobbiamo monitorare una periodicità molto bassa con finestre dell'ordine di ore o giorni, avere una strategia così sensibile alla nuova informazione non è ottimale.

Un modo per ovviare a questa eventualità è di cambiare il calcolo, attribuendo un peso maggiore alla media vista fino ad ora. Un'idea potrebbe essere la seguente:

$$TW_{new} = F_{N\ firstseen} - F_{N-1\ lastseen}$$

$$TW = \frac{TW_{OLD} * N_{seen-flows} + TW_{new}}{N_{seen-flows} + 1}$$

In questo modo considerando il numero di volte visto e la media della TW il valore non subirà grandi variazioni in seguito ad un repentino ed elevato valore di cambio di frequenza.

La seconda considerazione riguarda il numero di flussi da osservare per calcolare la TW. Si potrebbe decidere di guardare, non solo l'ultimo ma, gli ultimi N. In questo modo, si potrebbe calcolare una media della finestra calcolata guardando uno storico di N flussi.

Capitolo 6

IMPLEMENTAZIONE

L’implementazione è stata realizzata utilizzando in modo sinergico il risultato dei sistemi di rilevazione burst, slow e periodici. La prima fase ha previsto la creazione di un laboratorio di test per raccogliere dati sui malware. Utilizzando un ambiente virtualizzato, nel nostro caso virtualbox. La seconda fase è stata quella di definire un’architettura che prendesse in input oltre ai soli pacchetti raw, anche flussi netflow. La terza fase ha unito tutti i vari moduli di rilevazioni realizzati, per creare un report finale che utilizzasse quante più metriche possibili.
Per l’analisi della DPI è stato deciso di optare per l’utilizzo di metriche di una facile implementazione e esprimessero informazioni chiare. La scelta è ricaduta su l'analisi di possibili DGA. Sono algoritmi per la generazione di domini. Questi domini essendo generati in un modo prestabilito dall’attaccante, vengono utilizzati dai loro bot per verificare la presenza di nuovi server C2. Andando a generare, con il loro stesso algoritmo, domini validi.
JA3 è un metodo per creare delle firme riguardante l’utilizzo di particolari configurazioni di SSL/TLS. Consente di identificare, dall’analisi delle sole informazioni del Client Hello, combinazioni particolari che sono utilizzate solo da malware. Ovviamente bot sofisticatiz andranno ad emulatre comportamenti il più possibile simili a programmi legittimi come i browser.

6.1 Ambiente di test

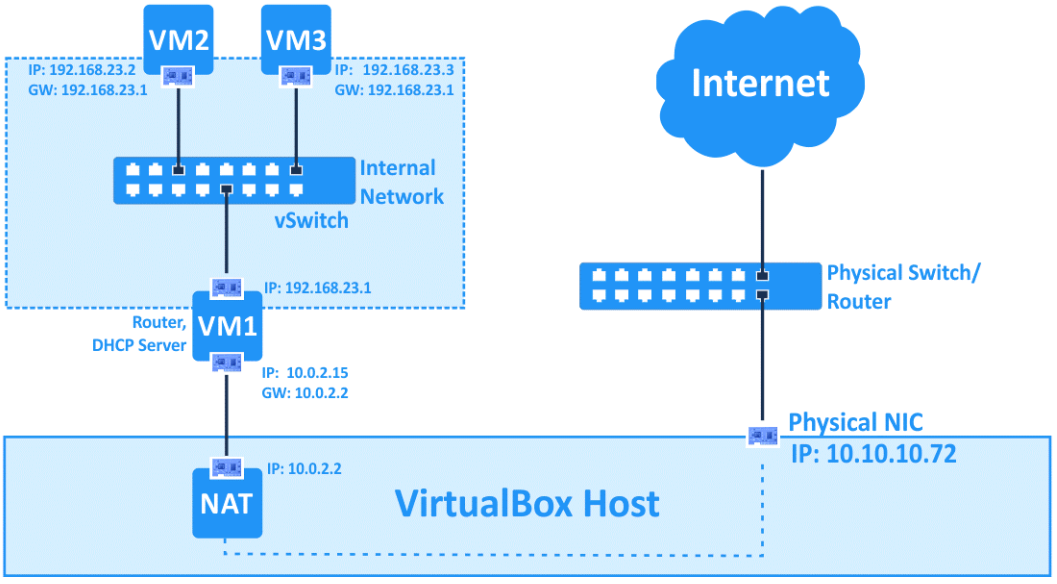
Nella realizzazione della rilevazione periodica dei flussi netflow, è stato necessario realizzare una farm di malware per studiare il comportamento a lungo termine. I dataset presenti in rete, erano per la maggior parte estratti delle caratteristiche più rilevanti del malware. La creazione di queste ambiente di test ci ha consentito di verificare se effettivamente il modulo che rilevasse comportamenti periodici funzionasse in presenza di malware che presentassero comportamenti pseudo-randomici.

6.1.1 Creazione VM

La scelta dell'ambiente virtualizzato è stata quella di VirtualBox, in quanto gratuito e sviluppato da Oracle, che ormai ha dimostrato la sua maturità da molti anni.
Sono state scelte la creazione di macchine virtuali contenenti sistemi operativi differenti, nel nostro caso: ubuntu, debian e windows. Questa decisione è stata presa per verificare che i malware testati sulle macchine windows, tentassero di effettuare operazioni di movimenti orizzontali nella rete, cosa che non abbiamo riscontrato.
Ogni macchina è stata dotata di 2.5GB di RAM e 4 core, per un totale di 2 VM windows, 1 debian ed 1 linux, così da simulare una rete domestica o d'ufficio di piccole dimensioni.

6.1.2 Network vbox

Ogni VM è stata configurata con una scheda di rete di tipo *rete interna*[7] ad eccezione di quella ubuntu che presenta anche la scheda di rete NAT. La macchina ubuntu ha lo scopo di fare da routing per le altre così da creare un layer isolato tra le macchine e l'host.
In virtualbox l'opzione di *rete interna* crea un network tra le macchine che hanno la stessa scheda, che consente esclusivamente di parlare tra loro. La presenza della VM con anche la scheda NAT fa sì che l'eventuale traffico internet venga indirizzato mediante il NAT, tenendo così pulito il network dell'host che risulta invisibile alla rete di vbox.[8]



Come si vede dallo schema precedente ogni vm ha il suo ip assegnato per l'internal network, quando uno delle macchine dovrà andare su internet la VM1, ossia quella che farà da router, passerà il traffico al nat che tramite il virtualbox host passerà nella rete fisica e raggiungerà internet.

6.1.3 VM awareness

I malware più moderni sono dotati di moduli che consentono di capire se sono in un ambiente virtualizzato o controllato [25]. Questo consente di cambiare il loro comportamente disattivando varie funzionalità o diventare completamente innocui.
Durante la fase di analisi in ambienti virtualizzati va presa in considerazione questa loro capacità che può portare a risultati falsati una volta che il malware agisce in una rete reale.
Gli accorgimenti di base che sono stati presi riguardano la disattivazione e rinominazione di vari servizi e file presenti dentro la VM. Tuttavia non vi è un modo che ci dia la consapevolezza di essere pienamente invisibili. La soluzione migliore sarebbe quella di realizzare un network di test con sole macchine fisiche.

6.1.4 Raccolta dati con Wireshark

Wireshark è un network protocol analyzer. Consente di vedere nel dettaglio cosa sta succedendo nel nostro network. È diventato uno standard sia in ambito commerciale e non-profit.
Il suo utilizzo è stato fondamentale per la raccolta e l'analisi del traffico generato dalla VM infette. Per realizzare la cattura del traffico delle differenti macchine, è stato installato sia all'interno di ogni singolo sistema sia nell'host machine.
Avendo realizzato un network interno isolato all'host, per identificare eventuali comportamenti di movimenti orizzontali, è stato necessario avere istanze wireshark aperte in ogni vm.
La procedura di raccolta dati nel laboratorio di test con lo scopo principale di rilevare delle periodicità si è diviso in 3 parti:

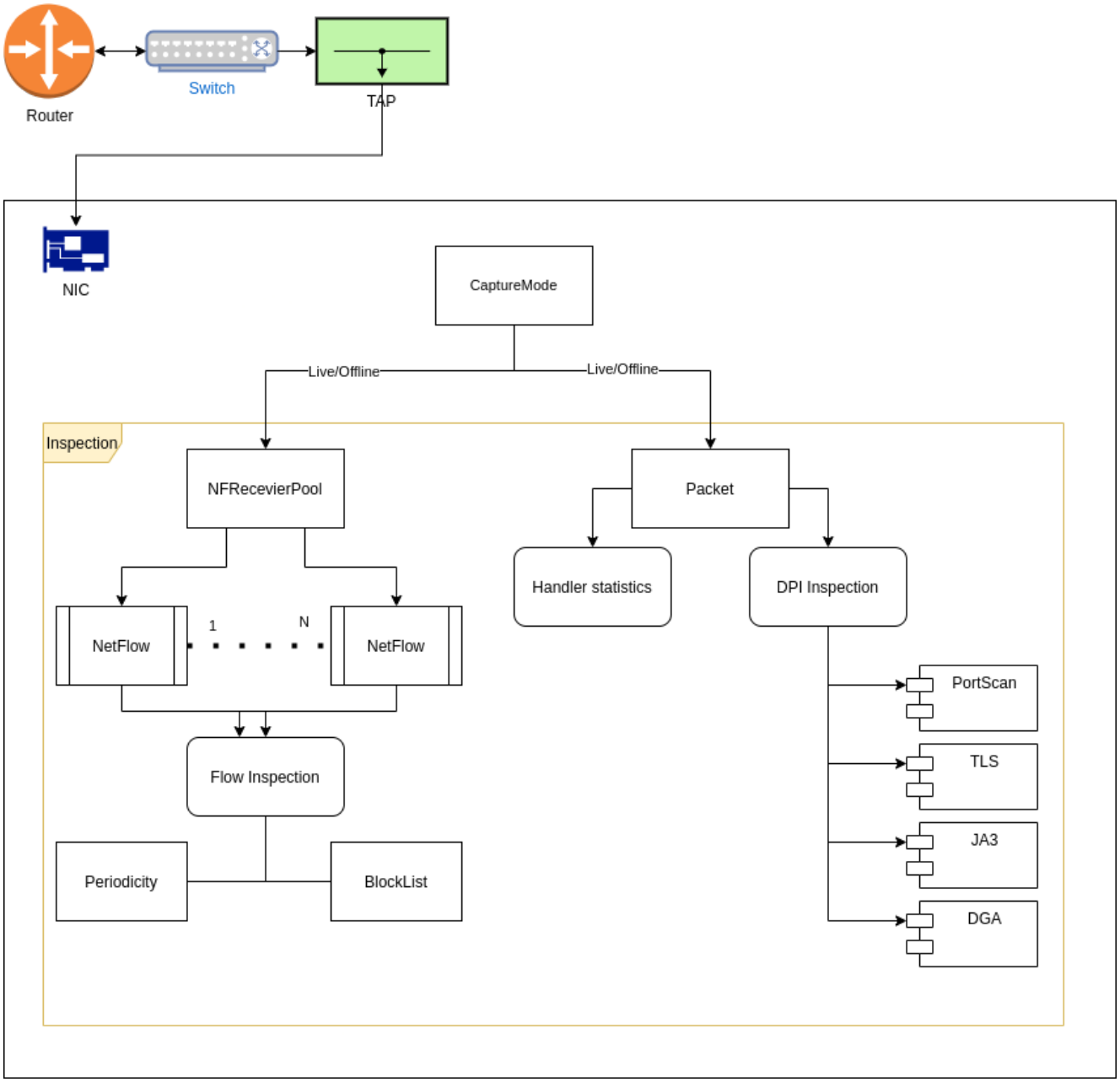
- raccolta del traffico della rete non infetta
- raccolta del traffico della rete infetta
- attivazione del modulo per l'analisi periodica

Prima di infettare la rete, veniva registrato il traffico legittimo delle VM. Una volta raccolte le tracce, si passava alla scelta del malware.
Una volta scelto il malware da analizzare, veniva avviato in una VM windows ed tutte le istanze wireshark pronte per il traffico malevolo.
Raggiunto il punto di avere sufficienti dati da analizzare venivano fermate tutte le catture dei pacchetti e salvati i pcap su disco.

L’ultimo step, era quello di poter effettuare analisi anche offline, per migliorare ed ottimizzare il modulo del calcolo periodico. Fondamentale quindi l’utilizzo di nProbe per generare flussi sia live, sia offline (tramite i pcap).
La raccolta del traffico prima dell’infezione serve per eliminare gran parte del rumore che viene generato dall’OS. Così da agevolare l’analisi e l’eventuale scoperta, delle comunicazioni malevole.
Per ogni analisi di malware è stata attivata la registrazione del traffico interno delle VM e quella provvista di NAT. I dati più significativi sono quelli raccolti dall’istanza con l’accesso ad internet abilitato.

6.2 Architettura

Il software quando lavora in modalità live è suddiviso in 2 goroutine principali: netflow e analisi del pacchetto.



La parte netflow si occupa di ricevere in ingresso flussi netflow v9, con i seguenti parametri:

%IPV4_SRC_ADDR
%IPV4_DST_ADDR
%IPV4_NEXT_HOP
%INPUT_SNMP

%OUTPUT_SNMP
%IN_PKTS
%IN_BYTES
%FIRST_SWITCHED
%LAST_SWITCHED
%L4_SRC_PORT
%L4_DST_PORT
%TCP_FLAGS
%PROTOCOL
%SRC_TOS
%SRC_AS
%DST_AS
%IPV4_SRC_MASK
%IPV4_DST_MASK
%FLOW_END_REASON
%DIRECTION
%BIFLOW_DIRECTION

Sono presenti N worker, che sono in ascolto sulla porta NF per effettuare il parsing del protocollo netflow. Il parser è stato preso dalla libreria realizzata da verizon. Ogni worker una volta letto il flusso manda il contenuto alla funzione *InspectFlow*. L'*InspectFlow* effettua 2 operazioni: controllo degli ip e della periodicità del flusso. Appena arriva un nuovo flusso viene analizzato l'ip e controllato l'eventuale presenza in una blocklist. La seconda operazione effettua il calcolo di arrivo dei flussi. Ogni volta che viene visto un nuovo flusso viene salvato in una mappa che ha come chiava la tripla <IpSorgente,IpDestinatario,PortaDestinatario> e come valore:

```
type FlowInfo struct {
        TWDuration              float64
        ServerPort              uint16
        Client                  string
        PeriodicityCounter      int
        Server                  string
        TimeWindowsExpiresAt time.Time
        TimeWindows             []TimeWindow
        LastSwitched            time.Time
        CurrentlyPeriodic       bool
        Deviation               time.Duration
}
```

Il campo TWDuration indica la frequenza del flusso. La sensibilità della finestra temporale tra l'arrivo di un nuovo flusso ed il precedente. Può essere regolata con un certo delta scelto dall'utente.
Quando arriva il campo NF *EndReason* con valore 0x02 (rfc5102), ossia al verificarsi di un active timeout l'inspection per quel flusso non effettua lookup nella mappa ed esce anticipatamente.
Per evitare di segnalare troppi casi che hanno una bassa frequenza, è possibile scegliere un numero minimo di volte in cui un flusso deve essere visto.

La parte che si occupa di controllare i pacchetti in arrivo è realizzata mediante la libreria *gopacket* basata su libpcap.
Il lavoro principale che svolge questo modulo è quello di: effettuare un'analisi di comportamenti burst, individuare versione e/o cipher TLS deboli e controllare le firme JA3.

Nell'inizializzazione del modulo viene caricata la lista di firme JA3 in blocklist. Terminato la fase di setup, parte la goroutine che si occupa dell'analisi dei pacchetti e in contemporanea parte anche l'handler delle statistiche.
L'handler è una callback che viene richiamata per un quanto di tempo stabilito dall'utente. Questa funzione ha lo scopo di pushare su influxdb o su csv i risultati dell'eventuale port scanning subito. Più sono bassi i valori del quanto di tempo più andremo a cercare comportamenti di tipo burst.
La gorutine che effettua la lettura dei pacchetti passa il contenuto dell'inspection che effettuerà i controlli le rilevazioni che verranno descritte nei successivi capitoli.

6.3 Rilevazione burst

Una rilevazione burst indica un comportamento che viene effettuato in un quantitativo di tempo breve.
Nel nostro caso è stata impiegata una rilevazione per identificare tentativi di portscan.
Il portscan, consiste nel tentativo di aprire una connessione su una porta, al fine di trovare quali servizi sono in ascolto. Esistono vari tipi di portscan, quelli più semplici e veloci prevedono un numero di richieste che tenta di esaurire tutto lo spazio delle porte. È possibile rilevare sia un portscan slow che burst, l'importante è cambiare la finestra temporale in modo opportuno.

6.3.1 Utilizzo di bitmap con funzione hash

Per rappresentare l'intero spazio delle porte (2^{16}= 65535 porte) si è deciso di optare per una bitmap. Questa struttura è stata implementata come un array di uint8.
La bitmap è stata suddivisa in *bin,* ogni bin rappresenta un certo numero di porte in cui vengono mappate dalla nostra funzione hash.
La funzione hash è stata scelta in modo da distribuire in modo sparso tutte le porte nell'intera bitmap.
L'implementazione nel nostro caso è la seguente:

```
func(port uint16) (uint16, uint64) {
        portModuled := (port /NumberOfBin) %SizeBitmap
        index, bit := portModuled/NumberOfBits, portModuled%NumberOfBits
        return index, bit
}
```

L'indice che rappresenta qualche bin selezionare è stata calcolato come:

indice = (porta/NumeroBin % GrandezzaBitmap)/NumeroBits

Il bit da settare all'interno del bit è stato calcolato come:

bit = (porta/NumeroBin % GrandezzaBitmap)%NumeroBits

Il NumeroBits indica quanti bit sono rappresentati nell'intera bitmap.
Prendiamo come esempio il numero di bin =16 e la grandezza di questi di 1024. Riusciamo a rappresentare l'intero spazio delle porte in soli 2048 Byte, contro gli 8192 Byte utilizzati nel caso in cui utilizzassimo un uint8 per rappresentare ogni porta. Un risparmio di spazio ben 3,5 volte.
L'intero codice riguardante il port scanning con l'utilizzo di questa struttura dati, è stata generalizzata da consentire l'utente di utilizzare la propria funzione hash e di scegliere grandezza e numero di bin.

6.3 Rilevazione slow

Una rilevazione slow ha come obiettivo quello di identificare comportamenti anomali effettuati in uno spazio di tempo molto ampio. Come per il port scanning è possibile utilizzare la stessa strategia, semplicemente aumentando la finestra temporale. Questo tipo di rilevazioni è molto complicata in quanto i malware ci danno poche informazioni e dobbiamo utilizzare bene quelle che riusciamo a catturare.
Nel nostro caso abbiamo utilizzato l'analisi della periodicità con finestre temporale dinamiche e l'analisi di DGA, JA3, tls cipher e flag nation.

6.4.1 Nazioni

In un contesto dove abbiamo pochi dati da analizzare è importante fare un buon uso delle informazioni raccolte.
Per questo motivo sono state scelte diverse metriche da prendere in considerazione.
Guardare la provenienza degli ip, con cui comunica la nostra rete, può darci un'idea della presenza di comunicazioni con server da parte di nazioni inusuali.
È stato utilizzato MaxMind GeoLite2 come database delle nazioni. Ad ogni pacchetto viene analizzata la provenienza dell'ip e viene inserita in una map. Ogni occorrenza incrementa il numero di visite da parte di una nazione.
È bene sapere che molte botnet sono presenti dentro i cloud di aws, azure, google, e altri provider. Quindi questa metrica potrebbe rivelarsi altamente fuorviante. Bisogna considerare e analizzare in modo adeguato l'utilizzo normale del nostro network e valutare segnali sospetti in nazionalità che solitamente non serviamo.
Ovviamente se abbiamo un grande traffico proveniente dell'america e prendiamo un malware presente nel cloud aws, sarà difficile capirlo dalla sola nazione.

6.4.2 TLS cipher e version

Nel protocollo TLS abbiamo una vasta scelta degli algoritmi di cifratura e 4 versioni.
La versione 1.0 è insicura. Presenta oltre ad altre criticità anche la vulnerabilità MITM [31]. Da considerare che TLS 1.1 verrà deprecato alla fine del 2020, rappresenta un segnale da prendere in considerazione come possibile problema di sicurezza da segnalare.
Gli algoritmi di cifratura da considerare come insicuri sono:

TLS_ECDHE_RSA_WITH_RC4_128_SHA
TLS_RSA_WITH_RC4_128_SHA
TLS_RSA_WITH_RC4_128_MD5

Quelli da considerarsi deboli sono:

TLS_RSA_WITH_AES_256_GCM_SHA384
TLS_RSA_WITH_AES_256_CBC_SHA256
TLS_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_CAMELLIA_256_CBC_SHA
TLS_RSA_WITH_AES_128_GCM_SHA256
TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_CAMELLIA_128_CBC_SHA
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_SEED_CBC_SHA

TLS_RSA_WITH_IDEA_CBC_SHA

Anche in questo caso l'analisi delle cipher suite e delle versioni, viene effettuato analizzato il payload del pacchetto.

6.4.3 JA3

JA3 è un metodo creato per SSL/TLS, per realizzare un fingerprint che sia facilmente producibile in ogni piattaforma e condiviso con team di threat intelligence. [30]
Sia TLS che SSL quando inizializzano una connessione mandano in chiaro delle informazioni quali: SSL/TLS Client Hello seguito dal 3-way handshake. Il modo in cui viene generato questo pacchetto dipende da librerie e metodi utilizzati nella fase di build del programma client. Poiché la negoziazione viene effettuata in chiaro è possibile generare un fingerprint sia del lato client (JA3) sia del lato server (JA3S).
I campi con cui viene generata poi l'hash JA3 sono:
SSLVersion,Cipher,SSLExtension,EllipticCurve,EllipticCurvePointFormat
Verrà poi creato un hash md5 dei valori di questi campi.
Nel software è stato implementato il calcolo delle firme JA3 in modo da confrontare gli hash prodotti con quelli della blocklist caricati in memoria.
Ogni qualvolta viene individuato un hash malevole l'ip viene inserito nella struttura dati *PossibleThreat* con relativa motivazione e/o hash JA3/S

6.4.4 DGA

I Domain Generation Algorithm (DGA), sono degli algoritmi in grado di generare nomi di dominio utilizzati da malware e botnet per comunicare con i server C&C. Questo codice è presente nei malware, che per ogni dominio generato tentano di risolvere il loro ip inviando query DNS. Questo sistema consente una grande flessibilità per i server C&C, che possono cambiare ip in modo più frequente, avendo l'accortezza di utilizzare un dominio generato dalla loro DGA.
Gli algoritmi che generano questi nomi non sono tipicamenti parole esistenti. Sono composti da lettere e numeri apparentemente casuali. Delle volte sono presenti dei pattern o prefissi che possono aiutarci all'identificazione.
Le difficoltà, di realizzare contromisure per la loro rilevazione, si nasconde proprio dietro la loro pseudo-casualità.[10] Concentrarsi su ogni singola variante è quasi impossibile. Mentre risulta più semplice rilevare la tecnica utilizzata.
Come metodi per la detection senza l'utilizzo del ML troviamo l'analisi dei bigrammi. Sono sequenze di 2 caratteri adiacenti facenti parti di una stringa. Si confrontano a coppie e si stabilisce con una certa probabilità se il dominio sia stato generato, contando le occorrenze dei bigrammi.

Nell'implementazione è stato utilizzato l'algoritmo di LMS (longest meaningful substring) che assegnando dei pesi ritorna un valore compreso tra 0 e 100. Più basso è questo valore maggiore è la probabilità che sia un dominio generato automaticamente.
Ogni richiesta DNS viene analizzata dal modulo dga, che calcola lo score e se inferiore a 41 viene segnalato come possibile minaccia.
Risulta efficace su nomi maggiori di 10 caratteri. Il modulo è stato testato utilizzando un dizionario di domini malevoli presenti su una repository pubblica di github. [9]

6.5 nProbe come sonda netflow

nProbe è un software sviluppato da *ntop,* che consente di collezionare ed esportare flussi Netflow, IPFIX, sFlow ed una numerosa serie di altre caratteristiche.
Nel progetto è stata utilizzata la funzione di generare flussi NetFlow v9. La scelta di selezionare la versione 9, è dovuta dalla capacità del protocollo di gestire connessioni bidirezionali, ed altri campi opzionali che potrebbero essere prese in considerazione per effettuare analisi su altri dati.
nProbe è stato utilizzato durante tutta la fase di analisi e raccolta dati, essenziale per il funzionamento del modulo della periodicità, basato su NFv9.

Un esempio di utilizzo nel nostro caso prevede il seguento comando:
nprobe -i [network_interface] -n [ip]:2055 -b 2 -V 9 -T "%IPV4_SRC_ADDR %IPV4_DST_ADDR %IPV4_NEXT_HOP %INPUT_SNMP %OUTPUT_SNMP %IN_PKTS %IN_BYTES %FIRST_SWITCHED %LAST_SWITCHED %L4_SRC_PORT %L4_DST_PORT %TCP_FLAGS %PROTOCOL %SRC_TOS %SRC_AS %DST_AS %IPV4_SRC_MASK %IPV4_DST_MASK %FLOW_END_REASON %DIRECTION %BIFLOW_DIRECTION"

Essendo Netflow un protocollo creato da cisco, per il funzionamento della parte specifica del programma, non è necessario l'utilizzo esclusivo di nProbe. È possibili utilizzare una qualsiasi sonda sia in grado di esportare flussi e rispettando il protocollo cisco.

Capitolo 7

VALIDAZIONE

Prima di passare alla validazione sul dataset, sono state effettuate delle prove per verificare il corretto funzionamento dell'algoritmo.
I test hanno coperto la verifica di cambi repentini di periodicità, comunicazioni strettamente costanti e comunicazioni intermittenti. Queste verifiche sono state effettuate utilizzando uno script bash che mandasse X richieste di tipo HTTP GET, ad un server noto, ad un quanto di tempo prestabilito. Confrontando il numero di flussi individuati, con la frequenza corrispondente al delay impostato, abbiamo verificato il corretto funzionamento dell'algoritmo.

L'esempio utilizzato per il test di una comunicazione stabile con cadenza regolare, prevedeva 33 richieste al server web ogni 45 secondi. L'output dell'algoritmo utilizzando una tolleranza del 10%. Sono stati omessi i vari rumori di fondo.


```
"192.168.1.5/192.168.178.20/80": {
   "frequency": 45,
   "server_port": 80,
   "client": "192.168.1.5",
   "num_periodic_loops_accounted": 30,
   "server": "192.168.178.20"
 }
```


Il risultato è quello atteso. Abbiamo individuato correttamente 30 volte la stessa comunicazione, con una frequenza di 45 secondi. Ricordiamo che i primi due flussi servono per calcolare la Time Window, dal terzo in poi, possiamo iniziare a contare l'eventuale ricorrenza. Non rimane altro, che da verificare quanto è preciso l'algoritmo.

7.1 Dataset

Per validare l'algoritmo, che utilizza il calcolo delle finestre temporali, è stato utilizzato il dataset Botnet2014 [18] del CIC (Canadian Institute for Cybersecurity). Questo dataset contiene informazioni riguardanti numerose botnet. È stato costruito unendo 3 dataset: ISOT [19], ISCX 2012 IDS [20] e Malware Capture Facility Project [21].
Abbiamo così una collezione di traffico legittimo con al suo interno comportamenti anomali effettuati dai vari malware e botnet. Tuttavia il dataset non fornisce indicazioni sulla periodicità di una comunicazione. Si è reso necessario effettuare un labeling dei flussi che risultavano periodici. Facendomi guidare dall'algoritmo con tolleranze molto lasche, si è costruito un dataset. Tutti i flussi che apparivano 1 sola volta e tutti quelli che apparivano come idle venivano scartati. Questo ha consentito di pulire molto rumore di fondo e semplificare il processo di verifica manuale di eventuali periodicità.
Per non invalidare la creazione del dataset, ed incappare in un possibile overfitting, non è stato utilizzato esclusivamente il risultato, della pulizia del rumore, di una singola esecuzione dell'algoritmo. È stato eseguita una media dei risultati ottenuti da ogni singola iterazione. Ogni esecuzione è stata effettuata cambiando la tolleranza e il taglio da effettuare, per valutare possibili falsi positivi e falsi negativi.
La decisione di scegliere, da quale quantità in poi è necessario considerare cosa è falso positivo o falso negativo, è stata guidata da un training. Effettuato cambiando i range in cui realizzare il taglio di TN, TP, FP, FN. L'accuratezza migliore è stata raggiunta dalla seguente combinazione:

$$Pivot = lowerbound(\frac{n}{2})$$
$$TN = [0, Pivot - 1]$$
$$FP = [Pivot, Pivot]$$
$$FN = [Pivot + 1, Pivot + 1]$$
$$TP = [Pivot + 2, n)$$

Un risultato prevedibile. In quanto i valori che pesano di più, nel calcolo della precision e recall, sono esattamente i falsi positivi ed i falsi negativi. Andando a restringere l'intervallo dei valori falsi si ottiene un'accuratezza maggiore.

Esempio: a partire dal nostro dataset iniziale, vogliamo effettuare una media, su 11 tolleranze diverse, dei flussi che sono periodici con il cambiare della tolleranza.
Andremo a creare una mappa globale. Ad ogni flusso (chiave) è associato un contatore (valore). Questo contatore viene incrementato, per ogni flusso periodico trovato, ad ognuna delle 10 esecuzioni. Una volta stabilita: la tolleranza di partenza, il delta di incremento della tolleranza ed il numero di esecuzioni da effettuare, si procede ad analizzare la nostra mappa di contatori.

Avremo una situazione in cui un flusso non periodico, potenzialmente visto solo una volta, avrà come contatore 0. Una comunicazione periodica, che appare in ogni singola esecuzione, avrà come valore massimo 11. Possiamo dire con certezza che i flussi con valore 0 saranno sicuramente TN (true negative) mentre quelli con valore massimo 11 saranno sicuramente TP (true positive). Il taglio verrà effettuato nella seguente maniera:

TN = 0 a 4
FP = 5 a 5
FN = 6 a 6
TP = 7 a 11

In questo modo avremo un'idea, su quali flussi effettuare un ulteriore controllo manuale. Andremo a realizzare il labelling delle periodicità, del dataset originale, prestando particolare attenzione ai risultati incerti. Dovremo controllare noi e decidere trai i risultati di FP e FN cosa è realmente periodico e cosa no.

7.2 Confronto

Partendo dal dataset Botnet2014, opportunamente etichettato per validare il calcolo della periodicità, effettuiamo delle analisi per vedere come performa il nostro algoritmo. La scelta della tolleranza è stata calcolata sul dataset utilizzato per fare il training. Con il risultato migliore, che ha raggiunto un più alto valore di precision and recall, è stato testato e quindi validato nel testset. Andremo a vedere nel primo test, utilizzando il valore ottimale della tolleranza, trovato nello step precedente. I successivi serviranno per avanzare delle riflessioni su cosa succede ad aumentare in modo spropositata la tolleranza.

Test1
Scegliamo come tolleranza il valore di 12%. Ossia se il flusso dura 60" potrà variare di ±7" quindi sarà considerato periodico se lo rivediamo tra 53" e 67".

True Positive (flussi periodici)	675
True Negative (flussi aperiodici)	166877
False Positive (flussi aperiodici segnati come periodici)	45
False Negative (flussi periodici segnati come aperiodici)	61

Otteniamo una precision del 94% ed una recall del 92%.
Come ci aspettavamo il numero dei veri negativi (TN) sono la maggioranza. Questo

perché il nostro dataset, non essendo stato creato in moto artificiale, riesce a riflettere bene la natura aperiodica nelle interazioni, che ha l'utente con la normale navigazione. Solo la minoranza dei flussi risulta periodico.
La periodicità della rete è costituita anche da traffico legittimo di vari programmi che effettuano polling come i client di posta che ogni x minuti chiedono se ci sono nuove email da scaricare.

Test2
Scegliamo come tolleranza il valore di 50%. Se la TW è di 60" questa potrà variare tra 30"-90".

True Positive (flussi periodici)	675
True Negative (flussi aperiodici)	166087
False Positive (flussi aperiodici segnati come periodici)	835
False Negative (flussi periodici segnati come aperiodici)	61

La precision ammonta al 45% e la recall al 92%.
Possiamo notare come la precisione crolla ed aumentano in modo significativo i FP da 45 a 835. L'aumento è giustificato da un'eccessiva bontà nel determinare comportamenti periodici. Questa bontà consente molto più rumore. Il rischio è quello di considerare comunicazioni perturbate come qualcosa di periodico. Vedremo come nel prossimo test questo comportamento andrà a peggiorare significativamente.

Test3
In questo caso avremo tolleranza del 1000%. Quindi se la TW è di 60" questa potrà variare tra 3" (che il nostro minimo) e 660".

True Positive (flussi periodici)	675
True Negative (flussi aperiodici)	162316
False Positive (flussi aperiodici segnati come periodici)	4606
False Negative (flussi periodici segnati come aperiodici)	61

La precision crolla al 13% e la recall rimane al 92%.
Valgono le stesse considerazioni del secondo test. L'aumento eccessivo di questa tolleranza ha portato eccessivo rumore, portando i FP a 4606.

Test4
Vogliamo vedere cosa succede se utilizziamo una tolleranza estremamente bassa. Utilizzando un valore pari al 0.1% andremo ad oscillare di 6ms, per una TW di 60" oscillerà tra 59,94" e 60,06".

True Positive (flussi periodici)	318
True Negative (flussi aperiodici)	166877
False Positive (flussi aperiodici segnati come periodici)	45
False Negative (flussi periodici segnati come aperiodici)	418

Precision dell'88% e recall del 43%
Qui succede l'esatto contrario dei test 2 e 3. Scegliendo valori estremamenti piccoli, andiamo a ridurre il rumore. Ma eventuali oscillazioni nella periodicità non verranno considerate, scartando così molti più casi. Questo approccio molto miope non considera eventuali ritardi di trasmissione da parte della sonda. Quindi per TW piccole avremo molti flussi che verranno scartati.

7.3 Verifica su malware samples

Per verificare il corretto funzionamento sul campo, è stato utilizzato un dataset contenente il tracciato del malware Zeus della durata di 24h, fornito da *activecountermeasures*[36].
L'analisi che viene fornita, riguarda la problematica del malware beaconing. Il report ci offre informazioni riguardanti il tipo di malware, il tipo di piattaforma C2, l'ip della vittima, del server di C&C ed la durata del Beaconing.
Possiamo rappresentare le loro informazioni nella seguente tabella:

Malware	Zeus
Traffic Type	Crimeware
C2 Platform	Cobalt Strike
Connection Type	Reverse HTTP
Host Payload Delivery Method	Powershell one-liner
Target Host/Victim	192.168.99.53 – Windows 10 x64
C2 Server	67.207.93.135 – Ubuntu 18.04.3
Beacon Timing	30s

Jitter	5%

Sono stati individuate 615 connessioni con un intervallo di 29 secondi, 1771 connessioni con 30 secondi d'intervallo e 491 con 31 secondi. Con una variazione del 5% rispetto ai 30 secondi attesi.

Effettuando l'analisi del tracciato con il nostro algoritmo, impostando una tolleranza del 10% otteniamo i seguenti flussi periodici:


```
{
 "192.168.99.10/192.168.99.53/54527": {
  "frequency": 120,
  "server_port": 54527,
  "client": "192.168.99.10",
  "num_periodic_loops_accounted": 16,
  "server": "192.168.99.53"
 },
 "192.168.99.53/192.168.99.1/67": {
  "frequency": 300,
  "server_port": 67,
  "client": "192.168.99.53",
  "num_periodic_loops_accounted": 226,
  "server": "192.168.99.1"
 },
 "192.168.99.53/52.177.165.30/443": {
  "frequency": 60,
  "server_port": 443,
  "client": "192.168.99.53",
  "num_periodic_loops_accounted": 744,
  "server": "52.177.165.30"
 },
 "192.168.99.53/52.179.224.121/443": {
  "frequency": 60,
  "server_port": 443,
  "client": "192.168.99.53",
  "num_periodic_loops_accounted": 654,
  "server": "52.179.224.121"
 },
 "192.168.99.53/67.207.93.135/80": {
  "frequency": 29,
  "server_port": 80,
  "client": "192.168.99.53",
```

```
  "num_periodic_loops_accounted": 1864,
  "server": "67.207.93.135"
 },
 "192.168.99.55/192.168.99.53/7680": {
  "frequency": 41,
  "server_port": 7680,
  "client": "192.168.99.55",
  "num_periodic_loops_accounted": 16,
  "server": "192.168.99.53"
 }
}
```


Otteniamo 6 flussi totali. Tra i risultati notiamo che siamo riusciti ad individuare correttamente il flusso associato al beaconing.


```
 "192.168.99.53/67.207.93.135/80": {
  "frequency": 29,
  "server_port": 80,
  "client": "192.168.99.53",
  "num_periodic_loops_accounted": 1864,
  "server": "67.207.93.135"
 }
```


Nella tripla e' presente la comunicazione tra la vittima (192.168.99.53) e il server di C2 (67.207.93.135) con una periodicità di circa 29 secondi. Individuata per ben 1864 volte, è un risultato evidente che siamo in presenza di un malware beaconing. Possiamo anche notare come in un altra pubblicazioni quale: "Uncovering Periodic Network Signals of Cyber Attacks"[32].
Abbiamo così confermato che il nostro algoritmo è in grado di rilevare con successo il beaconing.

Capitolo 8

CONCLUSIONI

Abbiamo discusso di quali sono i tipi di malware più comuni. I comportamenti che questi possono assumere, di tipo burst o slow. Come implementazioni semplici tendono ad essere facilmente rilevabili nella rete. La grande dinamicità di malware più complessi e pericolosi quali emotet, mostra come non è semplice stabilire se la comunicazione sia legittima o meno. I numerosi moduli di cui sono dotati questi software malevoli, rendono estremamente difficile la loro analisi. Durante il nostro testlab abbiamo verificato che i sample di malware utilizzati non hanno prodotto evidente traffico con i loro botmaster, questi a fronte di una vm-awareness [25] che consente loro di cambiare il comportamente e rimanere in uno stato silenzioso per un tempo non ben determinato. Infine la realizzazione dell'algoritmo che crea dinamicamente finestre temporali all'arrivo di flussi netflow dimostra essere capace di rilevare le comunicazioni periodiche. Tuttavia è necessario effettuare un tuning dei parametri per ottimizzare il suo funzionamento, in base al tipo di traffico che una rete genera. Infine utilizzare le informazioni che ci fornisce una sonda netflow, aiuta la macchina, in cui si trova il collezionatore, ad evitare pesanti calcoli di ogni singolo pacchetti e di liberare le risorse per effettuare altre operazioni e di poter richiedere esclusivamente i pacchetti delle comunicazioni che risultano essere periodiche per effettuare un'analisi più approfondita.
Questo algoritmo è un indicatore che va utilizzato in sinergia con altre informazioni che ci può dare una DPI per avere un ulteriore riscontro di un potenziale traffico malevolo.

Capitolo 9

LAVORI FUTURI

Il lavoro proposto termina con la presentazione e l'implementazione dell'algoritmo della periodicità basato su una finestra temporale dinamica. Manca la parte di verifica del traffico periodico se sia malevolo o meno. Un buon punto di partenza potrebbe essere quella di: implementare un sistema in cui a fronte di una comunicazione ricorrente, chiede di ricevere su una macchina le informazioni dei singoli pacchetti, per effettuare una DPI ed avere una visione d'insieme maggiore. Estendere anche il numero di tecniche ed indicatori utilizzati nella DPI è un buon modo per rilevare più specificità possibili. Un esempio è quello di utilizzare il concetto di "top talkers" per identificare un ip con quanti altri ip diversi sta parlando. Una possibile implementazione di questo, potrebbe essere quello di utilizzare una struttura dati chiamata HyperLogLog per memorizzare in modo efficiente, la cardinalità dei talkers.
Effettuare un'analisi ed un confronto con altri sistemi presenti sul mercato dell'implementazione proposta. Verificando così i punti di forza e debolezza dell'implementazione.

Capitolo 10

Codice sorgente

Il codice sorgente ed il file README.md che mostra la compilazione, la lista dei comandi e l'utilizzo del software, è reperibile al seguente link: https://github.com/alessio-perugini/stanislav

Riferimenti

[1] Bottazzi G., Italiano G.F., Rutigliano G.G. (2018) A New Scalable Botnet Detection Method in the Frequency Domain. In: Jahankhani H. (eds) Cyber Criminology. Advanced Sciences and Technologies for Security Applications. Springer, Cham. https://doi.org/10.1007/978-3-319-97181-0_7
[2] B. AsSadhan, J. M. F. Moura and D. Lapsley, "Periodic Behavior in Botnet Command and Control Channels Traffic," GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference, Honolulu, HI, 2009, pp. 1-6, doi: 10.1109/GLOCOM.2009.5426172.
[3]. QASSRAWI, M. T.; ZHANG, Hongli. Client honeypots: Approaches and challenges. In: 4th International Conference on New Trends in Information Science and Service Science. 2010, pp. 19–25.
[4] “Malware detection based on periodic behavior”, Bc. Martin Korec
[5] Farnaaz, Nabila & Akhil, Jabbar. (2016). Random Forest Modeling for Network Intrusion Detection System. Procedia Computer Science. 89. 213-217. 10.1016/j.procs.2016.06.047.
[6] Y. Y. Aung and M. M. Min, "An analysis of random forest algorithm based network intrusion detection system," 2017 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), Kanazawa, 2017, pp. 127-132, doi: 10.1109/SNPD.2017.8022711.
[7] Creating a Malware Analysis Lab and Basic Malware Analysis Joseph Peppers Iowa State University
[8] https://www.nakivo.com/blog/virtualbox-network-setting-guide/
[9] https://github.com/andrewaeva/DGA
[10] https://www.cybereason.com/blog/what-are-domain-generation-algorithms-dga
[11] Pande, Sagar & Khamparia, Aditya & Gupta, Deepak & Thanh, Dang. (2020). DDOS Detection Using Machine Learning Technique. 10.1007/978-981-15-8469-5_5.
[12]Analysis of L4 DoS/DDoS Attacks and Mitigation Techniques for DNS Reflection Attack, Zaid Al-Jarrah
[13] FlowSummary: Summarizing Network Flows for Communication Periodicity Detection Neminath Hubballi and Deepanshu Goyal

[14] Modeling, Analysis, and Characterization of Periodic Traffic on a Campus Edge Network
[15] B. AsSadhan and J. Moura, “An Efficient Method to Detect Periodic Behavior in Botnet Traffic by Analyzing Control Plane Traffic”, Journal of Advanced Research, Vol. 5, No. 4, pp. 435–448, July 2014
[16] G. Bartlett, J. Heidemann, and C. Papadopoulos, “Low-rate, Flow-level Periodicity Detection”, Proceedings of the 14th IEEE Global Internet Symposium, pp. 804–809, Shanghai, China, April 2011.
[17] Low-Rate, Flow-Level Periodicity Detection, Genevieve Bartlett, John Heidemann, Christos Papadopoulos
[18] https://www.unb.ca/cic/datasets/botnet.html
[19] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, and D. Garant,“Botnet detection based on traffic behavior analysis and flow intervals,”Computers & Security, 2013.
[20] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, “Toward developing a systematic approach to generate benchmark datasets for intrusion detection,” Computers & Security, vol. 31, no. 3, pp. 357–374, 2012.
[21] S. Garcia, “Malware capture facility project, retrieved July 03, 2013. university,”
[22] Using Machine Learning Techniques to Identify Botnet Traffic, Carl Livadas, Robert Walsh, David Lapsley, W. Timothy Strayer, Internetwork Research Department BBN Technologies
[23] Visualizing Automatically Detected Periodic Network Activity
[24] Analysis of Network Traffic Flows for Centralized Botnet Detection, Pedram Amini, Reza Azmi, and Muhammad Amin Araghizadeh
[25] Anti-Analysis Trends in Banking Malware, Paul Black, Joseph Opacki
[26] Malware Detection Based on Mining API Calls
[27] On Periodic Behavior of Malware: Experiments, Opportunities and Challenges
[28] A. Makandar and A. Patrot, "Malware class recognition using image processing techniques," 2017 International Conference on Data Management, Analytics and Innovation (ICDMAI), Pune, 2017, pp. 76-80, doi: 10.1109/ICDMAI.2017.8073489.
[29] Sathyanarayan V.S., Kohli P., Bruhadeshwar B. (2008) Signature Generation and Detection of Malware Families. In: Mu Y., Susilo W., Seberry J. (eds) Information Security and Privacy. ACISP 2008. Lecture Notes in Computer Science, vol 5107. Springer, Berlin, Heidelberg.
[30] https://engineering.salesforce.com/tls-fingerprinting-with-ja3-and-ja3s-247362855967
[31] Manik Lal Das, Navkar Samdaria, On the security of SSL/TLS-enabled applications, Applied Computing and Informatics, Volume 10, Issues 1–2, 2014, Pages 68-81

[32] Uncovering Periodic Network Signals of Cyber Attacks, Ngoc Anh Huynh, Wee Keong Ng, Alex Ulmer, Jörn Kohlhammer
[33] https://stellarcyber.ai/solarwinds-sunburst-backdoor-dga-and-infected-domain-analysis/
[34] https://www.deepinstinct.com/2020/12/28/why-the-sunburst-malware-was-so-unique-and-what-weve-learnt-from-it/
[35] https://github.com/2igosha/sunburst_dga
[36] https://www.activecountermeasures.com/malware-of-the-day-zeus/