UNIVERSITÀ DEGLI STUDI ROMA TRE

Dipartimento di Ingegneria
Corso di Laurea in Ingegneria Informatica

Tesi di Laurea

Realizzazione di un sistema di rilevazione e prevenzione delle intrusioni con Snort

Laureando
Allegra Strippoli
Matricola 548288

Relatore
Prof. Maurizio Patrignani

Correlatore
Federico Lommi

Anno Accademico 2021/2022

A ship in harbor is safe, but that is not what ships are built for. - John Augustus Shedd

Ringraziamenti

Ringrazio il Prof. Maurizio Patrignani, Federico Lommi e l'azienda Bytewise per avermi permesso di vivere quest'esperienza e per avermi accompagnato in questa fase di crescita.

Ai miei genitori e a mia sorella Sabina che per primi mi hanno trasmesso la curiosità e la volontà di conoscere il mondo un passo alla volta.

A Filippo e Lucrezia che mi hanno tenuto per mano tutte le volte che in questi tre anni ho avuto bisogno di un po' di coraggio.

Ai miei nonni che dalla mia nascita sono sempre stati un modello da seguire.

A Erica e Bianca che mi hanno incoraggiato tre anni fa ad iniziare questo percorso trasmettendomi pazienza e dedizione.

A Riccardo, Marco, Andrea, Camilla, Teresa, Matteo, Simona, Andrea e tutti gli amici e colleghi con cui ho avuto la fortuna di condividere la mia quotidianità.

Introduzione

I dati contenuti in un computer permettono di raggiungere un alto grado di conoscenza circa il contesto in cui esso viene utilizzato. In un computer privato potrebbero essere memorizzati dati di carte di credito, numeri di conti correnti bancari, fotografie, file personali. In un computer utilizzato da un'azienda potrebbero essere memorizzati i dati dei clienti, rendiconti finanziari, documenti etc... Nel momento in cui un computer entra a far parte di una rete ed è in grado di comunicare con altri dispositivi collegati direttamente o raggiungibili indirettamente, questi dati vanno protetti affinché non vadano persi, non vengano compromessi o finiscano nelle mani sbagliate. La rapida proliferazione dei dati digitali rende il problema della sicurezza e della protezione dei dati così comune da riguardare chiunque utilizzi computer, smartphone, tablet ...

Nello specifico ci si cala nel contesto di un'azienda come tante, che ha bisogno di esporre dei servizi su internet e teme per la sicurezza della propria rete. La rete nel suo stato primordiale ha due componenti: un router e una macchina virtuale su cui è installato un firewall, un primo strumento di difesa che segue delle politiche di sicurezza e filtra il traffico applicando delle regole. Gli obiettivi da perseguire sono due: è necessario rivedere la morfologia della rete per far si che questa possa ospitare dei web server e file server, ossia possa offrire dei servizi su internet. È necessario aggiungere un altro strato difensivo (oltre al firewall) che renda la rete più sicura. La strategia è quella di introdurre una seconda macchina virtuale che ospiti i servizi richiesti dall'azienda. Le due macchine svolgono ruoli distinti: una si occupa di gestire gli aspetti legati alla sicurezza, l'altra dei servizi. Dividere la rete in moduli, ossia tenere separate aree che svolgono funzioni diverse è una buona norma, il vantaggio è quello di poter correggere, gestire i guasti senza propagare le modifiche per tutta la rete. La macchina del firewall

viene frapposta tra il router e la macchina (o le macchine) dei servizi. In questo modo può filtrare tutto il traffico che proviene da internet ed è diretto all' (o agli) host. Se l'azienda con il tempo decidesse di espandersi e di aggiungere altre macchine, questa configurazione permette di farlo. A valle della macchina che ospita il firewall possono trovarsi tanti host quanti sono gli indirizzi IP disponibili nella LAN. Per questo la rete si dice scalabile.

In questa configurazione la responsabilità di difendere la rete è affidata completamente al firewall. L'impostazione del firewall è compito dell'amministratore di rete, che dovrà progettarlo in modo che siano consentite solo ed esclusivamente le connessioni necessarie al lavoro dell'azienda. Un firewall configurato male, o non configurato proprio, lascerebbe passare traffico indesiderato non svolgendo la sua funzione di difesa. Inoltre non è uno strumento capace di adattarsi autonomamente nel caso in cui si verificassero attacchi. Per risolvere queste problematiche si decide di realizzare un sistema di rilevazione e prevenzione delle intrusioni con un software open source chiamato Snort. I vantaggi sono molteplici: Snort permette di monitorare il traffico, catturare e analizzare pacchetti, sospendere possibili attività sospette, prevenire attacchi. Esso basa la propria strategia di rilevazione su una conoscenza profonda dei protocolli di rete e delle vulnerabilità dei servizi. Un sistema di difesa di questo tipo, che permette di riconoscere attività anomale e di agire per isolarle, eleva gli standard di sicurezza della rete.

Nel primo capitolo vengono elencate alcune delle minacce più comuni in rete, dal phishing agli attacchi DoS e DDoS, vengono poi introdotti alcuni strumenti di difesa come il firewall e i sistemi di rilevazione e prevenzione delle intrusioni (IDS e IPS).

Nel secondo capitolo viene presentato Snort un IDS/IPS network-based. Dopo averne dato la definizione, vengono presentate le varie modalità di esecuzione al fine di esprimerne le potenzialità. Viene posta attenzione sulla strategia di detection rule-based (basata su regole) utilizzata da Snort. Le regole sono uno strumento raffinato e versatile per distinguere i pacchetti autorizzati a transitare da quelli che invece vanno bloccati. Si noti che si tratta di uno strumento di rilevazione e non di prevenzione. Per prevenire le intrusioni è necessario fare in modo che, nel momento in cui Snort rileva un comportamento sospetto, il firewall sia in grado di respingere l'indirizzo IP sorgente del traffico indesiderato. Alla fine del capitolo vengono accennati vari strumenti di appoggio

utilizzati da Snort per la registrazione delle informazioni relative ai pacchetti.

Nel terzo capitolo vengono presentati i requisiti imposti dall'azienda. I vincoli, gli obiettivi, le linee guida da seguire.

Nel quarto capitolo viene presentato il progetto. Gli obiettivi del progetto sono: modificare la struttura della rete rispetto al modello proposto dall'azienda e introdurre il sistema di rilevazione e prevenzione delle intrusioni con Snort.

Nel quinto capitolo si procede con l'installazione e la configurazione di Snort in modo che questo si adegui alle esigenze della rete. Vengono presentati anche PulledPork e FwSnort, il primo è uno script che permette di mantenere aggiornate le regole di Snort, il secondo invece permette la traduzione di regole.

Nel sesto capitolo vengono messe in atto tre verifiche funzionali. Il primo test consiste nel generare del traffico sulle porte di interesse del server e verificare che Snort lo rilevi. Il secondo consiste nell'eseguire uno scan delle porte, e anche in questo caso verificare che Snort lo notifichi. L'ultimo test vede il server vittima di un attacco DoS. Snort deve essere in grado di rilevare e prevenire l'attacco.

Indice

Elenco delle figure

Capitolo 1

Sicurezza della rete

1.1 Sicurezza e vulnerabilità della rete

Quando si realizza un sistema di difesa si procede per livelli, si utilizzano strumenti di difesa passivi e poi, a seconda della necessità, si fa affidamento a tecnologie più evolute (ma più costose) per il controllo del traffico e per la prevenzione di attacchi. Si possono strutturare meccanismi di difesa intelligenti che riconoscano i pattern di attacco e adottino delle misure per contrastarli. Il primo passo è acquisire una conoscenza profonda delle possibili minacce e vulnerabilità della rete. In questo capitolo non viene offerta una panoramica completa, bensì uno scorcio che permetta di comprendere quali siano nel concreto le motivazioni che spingono gli utenti a introdurre sistemi di difesa.

1.1.1 Minacce comuni in rete

Per minacce "comuni" si intendono tutte quelle a cui un utente potrebbe andare incontro con un utilizzo sistematico dei principali servizi in rete. Un utente naviga siti web, gestisce la propria posta elettronica, utilizza pennette USB, esegue il download di file etc... e non sempre è consapevole di esporsi a dei rischi. Di seguito sono illustrati gli attacchi più comuni da cui difendersi [sof].

1.1.1.1 Virus, worm

Virus, worm e altri agenti automatici si diffondono autonomamente tramite meccanismi di infezione. Alla fase di propagazione segue quella di attivazione, causata da un comportamento umano, da un processo schedulato oppure potrebbe avvenire una auto-attivazione. Una volta che un malware ha attaccato un sistema vulnerabile può mostrare i suoi effetti immediatamente oppure può stare nascosto anche per molto tempo prima di causare i danni. Esistono diversi tipi di malware come:

- **Spyware** raccoglie dati e monitora attività dell'utente
- **Ransomware** effettua una crittografia dei dati e li tiene in "ostaggio"
- **Trojan** (cavallo di Troia) si presenta come un normale programma, in realtà è stato realizzato per uno scopo malevolo;

Il firewall e software antivirus possono essere due strumenti di difesa.

1.1.1.2 Scan

Gli Scan su larga scala ricercano indiscriminatamente sistemi vulnerabili e sfruttano buchi di sicurezza generalmente noti. È comune rilevare vulnerabilità nei servizi non aggiornati recentemente, potrebbero essere sfruttate per tentare attacchi. Un esempio è l'RCE (remote code execution) che permette all'autore dell'attacco di eseguire codice da remoto sul computer vittima e tramite un'escalation dei privilegi acquisire il controllo della macchina (ossia l'accesso come root).

1.1.1.3 Phishing

Il phishing consiste nel cercare di acquisire con l'inganno le informazioni personali degli utenti, sfruttando email e siti web con loghi falsificati. Il termine phishing ricorda il termine "fishing", proprio perché come nella pesca vengono utilizzate delle esche per irretire le proprie vittime. È un tipo di attacco molto popolare e diffuso, fa leva sull'incapacità di molti utenti di riconoscerlo. Spesso sembrano email urgenti e quindi gli utenti sono incentivati ad aprirle, oppure gli url sono molto simili a quelli di siti ufficiali e quindi non è facile capire che siano falsi.

1.1.1.4 Exploit

Un exploit è un frammento di codice che, sebbene non sia di per sé dannoso, può contenere un payload maligno. Gli exploit sono generalmente ospitati su siti web compromessi, un utente non attento potrebbe finire su un sito non sicuro, oppure potrebbe esservi reindirizzato aprendo un'email sospetta (esempio di phishing). Quando una vulnerabilità in un certo software viene resa nota, viene rilasciata una patch dalla community di sviluppo o dalla casa madre. Questa patch consiste in un aggiornamento che va a risolvere la falla di sicurezza sfruttata dall'exploit. Aggiornando i servzi installati ed il sistema operativo è possibile limitare questo tipo di attacchi. Un esempio di payload maligno è una backdoor. Le backdoor (porte di servizio) rappresentano degli strumenti persistenti che danno accesso al sistema in grado di superare le procedure di sicurezza attivate. In questo modo utenti non autorizzati sono in grado di accedere a un computer all'insaputa dell'utente.

1.1.1.5 DDoS e DoS

Un attacco DDoS (Distributed Denial of Service) consiste nell'inviare un flusso continuo di traffico ai servizi online, come i siti web, fino a quando il server non è più in grado di soddisfare queste richieste e va in crash. Si distingue dall'attacco DoS (Denial of Service) perché il traffico è proveniente da più fonti e non da un singolo dispositivo. In questo modo gli utenti reali che hanno necessità di accedere ad una risorsa di rete ne sono impossibilitati. Più aumenta il numero di fonti da cui parte la minaccia e più è difficile identificare l'autore primario. Esistono diverse tipologie di DDoS: può trattarsi di un attacco indiscriminato, di un attacco mirato a colpire un protocollo o addirittura una specifica vulnerablità di un'applicazione web. Ad accumunarli è il concetto di "flood" (inondazione) e le modalità con cui vengono realizzati, spesso tramite una rete di bot.

1.2 Difesa della rete

Per evitare attacchi, accessi al sistema e connessioni non autorizzate bisogna adottare delle misure che restringano il campo in cui queste intromissioni possano avvenire. Il principale punto di accesso ad un sistema è tramite una porta TCP o UDP aperta, e

quindi è necessario individuare quali porte sia effettivamente necessario tenere aperte e trovare un criterio, uno strumento, che permetta di chiudere qualsiasi altro punto di accesso all'host dalla rete. È funzionale a questo scopo introdurre il concetto di firewall.

1.2.1 Firewall

Il firewall è uno strumento di difesa che implementa delle politiche di sicurezza, permette di bloccare il traffico non autorizzato. I Firewall vengono classificati secondo due tipologie:

- **Host-based**: se controlla tutto il traffico in entrata e in uscita generato da un host.
- **Newtork-based** se filtra tutto il traffico in entrata e in uscita da una sottorete.

Ad esempio se si immaginano due sottoreti, una interna (LAN), una esterna (internet), il firewall viene posto in mezzo, in modo da venire attraversato dal traffico che la LAN e internet si scambiano.

1.2.1.1 Firewall con iptables

È oggetto di analisi iptables, un software firewall utilizzato in ambienti UNIX. Un firewall di questo tipo permette ad un amministratore di scrivere delle regole che moderino il passaggio di pacchetti, affinché tutto il traffico non necessario venga bloccato. Come prima cosa, bisogna settare le politiche di default per le catene di input, output e forward in ACCEPT (tutti i pacchetti vengono accettati) oppure in DROP (nessun pacchetto viene accettato). Se viene scelta una politica in ACCEPT le regole saranno in DROP, e viceversa. Questo perché se di default tutti i pacchetti vengono accettati servono delle regole per bloccarli, al contrario se di default tutti i pacchetti vengono scartati servono delle regole per autorizzarli. È sconsigliato utilizzare una politica in ACCEPT perché il firewall potrebbe risultare più "lasco" rispetto ad uno che adotta una politica DROP. Se arriva un pacchetto su una porta che non corrisponde nessuna regola, viene accettato. Può capitare che l'amministratore si dimentichi di scrivere una regola e quindi che vengano accettati pacchetti senza che ve ne sia controllo.Tuttavia con una politica DROP bisogna stare attenti a non dimenticarsi di far passare tutto

il traffico autorizzato. Ad esempio, se l'amministratore si connette in SSH ad un server e modifica il firewall (che utilizza una politica in DROP) trascurando la regola che consente il traffico SSH, la connessione con il server viene repentinamente interrotta. Dopo aver scelto la policy, è necessario scrivere le regole di input, output, forwarding, prerouting o postrouting, il protocollo (TCP o UDP), l'interfaccia, la porta e lo stato in cui è ammessa la connessione.

- Una regola è di **input** o **output** se riguarda il traffico in ingresso o in uscita dal firewall.
- Una regola è di **forwarding** se i pacchetti sono solo di passaggio per il firewall, che agisce da router, ma sono destinati a un'host interno o esterno alla rete. I pacchetti viaggiano attraverso catene di forwarding. Serve una catena per ogni "flusso" di traffico.
- Una regola di **prerouting** viene applicata ai pacchetti in entrata al sistema, prima che questi vengano confrontati con le regole iptables.
- Una regola di **postrouting** applicata ai pacchetti in uscita dal sistema dopo che questi vengano confrontati con le regole iptables.

Per quanto riguarda gli stati di una connessione, iptables è in grado di determinare se si tratta di:

- **NEW** - una nuova richiesta di connessione da parte di un client (SYN TCP o nuovo pacchetto UDP).
- **ESTABLISHED** - pacchetti relativi a connessioni già stabilite.
- **RELATED** - pacchetti correlati a connessioni esistenti e ESTABILSHED, come ad esempio il traffico FTP

Ecco degli esempi di regole relativi alla porta 443:

```
iptables -A INPUT -p tcp --sport 443 -i ens160 -m state --state
ESTABLISHED -j ACCEPT
```

Questo è un esempio di regola di input permette di accettare i pacchetti in entrata appartenenti a una connessione di tipo ESTABLISHED sull'interfaccia ens160 del firewall, protocollo tcp, source port 443.

```
iptables -A ie -p tcp --dport 443 -m state --state NEW,ESTABLISHED
-j ACCEPT
```

Questo è un esempio di regola di forwarding, "ie" è il nome della catena che consente il transito di pacchetti dalla rete interna verso quella esterna. Come nell'esempio delle regole input output è specificata la porta, questa volta si tratta di una destination port. Il significato della regola è il seguente: è permesso il forward dei pacchetti di tipo NEW ed ESTABLISHED dalla LAN verso internet.

```
iptables -t nat -A PREROUTING -i ens160 -s 0/0 -d 10.222.111.2
-p tcp --dport 443 -j DNAT --to 10.50.50.3:443
```

Questo è un esempio di regola di prerouting. Il significato della regola è: tutti i pacchetti che arrivano sull'interfaccia ens160 del firewall, che hanno come source un indirizzo IP qualsiasi (0/0) e come destinazione l'IP 10.222.111.2 (IP associato all'ens160) verranno inoltrati all'host che ha indirizzo IP 10.50.50.3 sulla porta 443.

Se si fa riferimento alla pila ISO-OSI, le regole di iptables permettono un buon grado di dettaglio nella gestione del traffico ai livelli di rete (liv. 3) e di trasporto (liv. 4), ma non di distinguere i protocolli dei livelli applicativi (HTTP, FTP, SMTP etc...) Si ha dunque la necessità di utilizzare un apparato più intelligente, che abbia la possibilità di analizzare il traffico ai livelli superiori della pila ISO-OSI ed eventualmente intraprendere delle azioni a difesa del sistema. Per questo, il firewall lavora spesso insieme a sistemi di rilevazione e prevenzione delle intrusioni, i cosiddetti IDS e IPS [ext].

1.2.2 Intrusion Detection System

Un Intrusion Detection System (IDS) è un dispositivo software o hardware che viene utilizzato per identificare attività anomale, accessi non autorizzati a un computer o a una rete di computer. L'intercettazione dei pacchetti sospetti avviene principalmente monitorando il traffico e analizzando i log di sistema. Mettere in piedi un meccanismo

di controllo di questo tipo aumenta le possibilità di poter riconoscere pattern di attacco noti, possibili scan e alte minacce, in modo da poter reagire tempestivamente. Gli IDS possono anche sfruttare database, librerie e regole per rilevare le intrusioni. Quando un'attività di rete corrisponde a una regola nota all'IDS, questo segnala il tentativo di intrusione. Il limite principale è che l'affidabilità di questo strumento dipende interamente dalla tempestività con cui il database viene aggiornato. Quando gli IDS rilevano una violazione della sicurezza, la notificano generando degli alert, ed è poi compito dell'amministratore fermare l'attività sospetta.

1.2.3 Intrusion Prevention System

Un dispositivo che richiede meno intervento da parte dell'amministratore è un Intrusion Prevention System (IPS). Prende autonomamente delle decisioni per contrastare possibili attacchi, isolando i pacchetti sospetti interrompendo connessioni e bloccando IP. Questo obiettivo può essere raggiunto aggiornando la lista delle regole del firewall in modo dinamico. Alcuni limiti degli IDS-IPS sono l'incapacità di analizzare pacchetti cifrati (non è possibile estrarre il contenuto del pacchetto senza conoscere la chiave) e la necessità di dover essere costantemente aggiornati per far fronte anche alle nuove minacce.

1.2.4 Cosa deve garantire un buon IDS e un buon IPS

Gli IDS e IPS non sono strumenti infallibili e possono commettere degli errori "di valutazione". La qualità di un buon IDS e IPS dipende dal numero di falsi positivi e falsi negativi che rileva. Un falso positivo si verifica quando un IDS/IPS classifica come maligna un'attività che invece dovrebbe essere autorizzata. Ad esempio i pacchetti danneggiati lungo il percorso o generati da software con bug o problemi possono essere riconosciuti erroneamente come tentativi di intrusione e considerati come pacchetti malevoli. Il problema dei falsi positivi è meno dannoso di quello dei falsi negativi. Tutta l'attività sospetta che supera controlli, analisi dei pacchetti e raggiunge il destinatario può generare una quantità di danni variabile, fino a compromettere l'integrità dell'intero sistema. Prevenire i casi di falsi positivi e negativi è il principale obiettivo di un sistema di difesa.

1.2.5 Tecniche usate

Così come il firewall implementa politiche di sicurezza, anche gli IDS e gli IPS utilizzano delle strategie, in questo caso strategie di detection. Si dividono in tre tipi. [cyb].

1.2.5.1 Signature-based detection

Una signature (firma) è un metodo di rilevamento basato sulla presenza di segni o caratteristiche distintive in un exploit. Le signature devono essere preconfigurate e sono statiche, ovvero non cambiano se non vengono aggiornate. Si tratta del metodo più semplice ed efficace per rilevare gli attacchi noti. Questo tipo di rilevamento è in genere classificato come "day after detection", poiché il virus deve aver infettato qualcuno (deve essere noto) affinché sia possibile scrivere una firma. Le aziende antivirus utilizzano i sistemi signature-based per proteggere i propri clienti dalle epidemie di virus.

1.2.5.2 Behaviour-based detection

Behaviour-based detection, per cui il sistema è in grado di confrontare il comportamento noto/normale con quello sconosciuto/anormale. Il vantaggio è poter rilevare efficacemente nuovi attacchi e vulnerabilità non viste prima. È meno dipendente dai protocolli e dal sistema operativo, e facilita la rilevazione di abusi di privilegi. Tuttavia bisogna superare la difficoltà di dover istruire il sistema in modo corretto.

1.2.5.3 Stateful protocol analysis detection

Stateful protocol analysis detection (o policy-based detection). Stateful significa che viene considerato e tracciato lo stato dei protocolli. I pacchetti quindi vengono analizzati nell'insieme, e non solo singolarmente. Questo metodo identifica le anomalie degli stati del protocollo, e considera benigne solo le attività conformi agli standard dei protocolli internazionali

1.2.6 Tipi di IDS: NIDS e HIDS

Un **NIDS**, Network Intrusion Detection System è un sistema che monitora il traffico da e verso la LAN. I pacchetti che corrispondono a delle signatures vengono registrati

in alert generati dal sistema. Un NIDS configura la scheda di rete per funzionare in modo "promiscuo", il che significa che saranno fatti passare i pacchetti attraverso lo stack di rete indipendentemente dal fatto che siano o meno indirizzati a una particolare macchina. Un NIDS verificha il payload dei pacchetti (e a volte anche gli header) per stabilire se siano presenti o meno tracce di codice maligno. Exploit, virus, worm generano traffico di rete che, se opportunamente intercettato e studiato, può smascherare l'attacco prima che questo abbia generato danni. Una specifica stringa in un payload potrebbe permettere di identificare un exploit maligno, così come specifiche opzioni negli header del TCP/IP potrebbero rivelare un altro genere di attacco. Sulla base delle signatures, il motore di ricerca, uno dei componenti principali dell'IDS che si occupa di elaborare e analizzare i pacchetti applicando le regole, deciderà se contrassegnare un pacchetto come potenzialmente maligno. Al contrario di un NIDS, un **HIDS**, Host-based Intrusion Detection System indaga il traffico di un singolo dispositivo endpoint, e non dell'intera sottorete, per il resto il funzionamento si può considerare analogo al NIDS [zer].

1.2.7 Tipi di IPS: NIPS, HIPS, NBA, WIPS

Un **NIPS**, Network Intrusion Prevention System come i NIDS si occupa di controllare il traffico da e verso la LAN. A differenza di questo però, intraprende azioni a discapito dei pacchetti sospetti. Oltre a segnalare attività anomale all'amministratore, un pacchetto può essere dropped o rejected. Un NIPS potrebbe anche bloccare un IP mittente che non considera affidabile. Analogamente agli HIDS esistono gli **HIPS**, che bloccano il traffico sospetto di un singolo dispositivo endpoint. Una strategia behaviour-based è invece implementata dall'**NBA**, Network Behaviour Analysis. A differenza di un NIPS richiede un periodo di formazione (noto anche come baseline) per apprendere il traffico normale ed essere così in grado di distinguerlo dal traffico anomalo. Identificare flussi di traffico insoliti può essere un ottimo modo per prevenire attacchi DoS e DDoS. Uno dei parametri per confrontare i vari sistemi di prevenzione delle intrusione è in base al numero di falsi postivi e falsi negativi. Un sistema NBA addestrato potrebbe essere più performante di un normale IPS. Tuttavia un NBA mal istruito potrebbe raggiungere un risultato opposto, potrebbe non essere grado di distinguere attività benigne da quelle

maligne. Infine un sistema che sfrutta l'analisi dei protocolli (stateful protocol analysis detection) è il **WIPS**, Wireless Intrusion Prevention System. Lo scopo principale di un WIPS è impedire l'accesso non autorizzato alle reti locali da parte di dispositivi wireless [try].

Capitolo 2

Snort: un IDS/IPS network-based

Utilizzare sistemi IDS e IPS in una rete risulta vantaggioso per diverse ragioni. Gli IDS/IPS hanno un buon grado di conoscenza dei livelli applicativi, (a differenza del firewall la cui competenza si ferma ai livelli 3 e 4 della pila ISO-OSI) possono investigare i pacchetti e sono istruiti in modo da riconoscere eventuali pattern di attacco noti sfruttando delle strategie di detection signature-based, behaviour-based e policy-based. Viene ora posta l'attenzione su un particolare NIDS/NIPS chiamato Snort, un software nato nel 1998 diventato molto popolare e diffuso grazie alla sua usabilità e al fatto di essere open source. Sfrutta una strategia di detection rule-based (basato su regole), leggermente differente rispetto alle tecniche descritte in precedenza. A differenza delle firme, le regole si basano sull'identificazione della vulnerabilità effettiva, non di un exploit. Questo tipo di rilevamento è in genere classificato come "0-day detection", perché non è necessario che il virus abbia infettato qualcuno per scrivere una regola, bensì è sufficiente una comprensione approfondita delle vulnerabilità. Snort ha una community vasta e questo permette di aggiornare continuamente le rules. Non appena viene scritta una nuova regola tutta la community ne potrà beneficiare semplicemente facendo un aggiornamento [sno].

2.1 Definizione

Snort è un software open source rule-based che permette di rilevare e prevenire intrusioni in una LAN. A seconda di come configurato può generare alert che avvisano gli utenti,

oppure può intraprendere azioni attive sui pacchetti sospetti bloccandone il transito. Le regole possono essere scritte/modificate da un amministratore oppure possono essere utilizzate direttamente le rules offerte dalla community. In entrambi i casi è possibile adattarle alle esigenze e ai servizi della propria rete. Inoltre Snort è in grado di combinare più strategie di detection. Oltre alla gestione delle rules può eseguire analisi di protocollo (policy-based detection) o studiare le anomalie (behaviour-based detection). Un esempio può essere l'analisi del protocollo TCP. Se dei pacchetti di SYN contenessero dei dati, o venissero ricevuti dei dati al di fuori della finestra di ricezione, allora verrebbe segnalata un'attività sospetta all'amministratore.

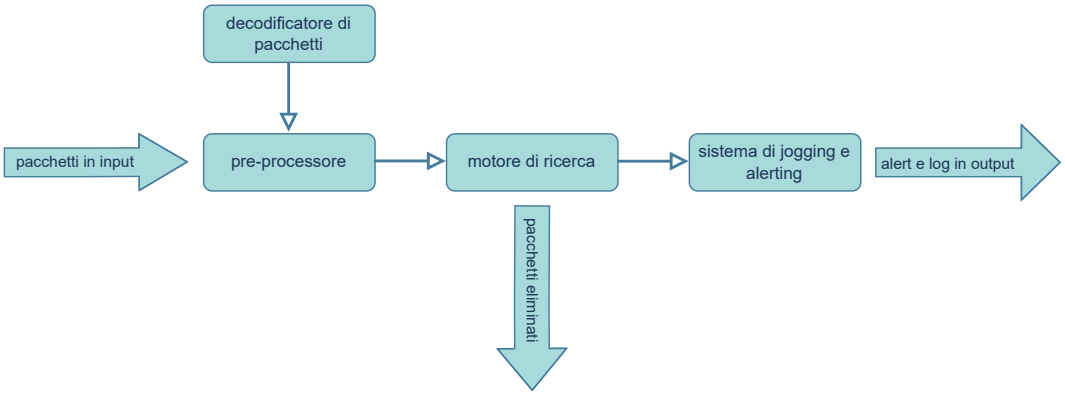

Figura 2.1: Componenti di Snort

I componenti principali di Snort sono citati brevemente:

- **Decodificatore di pacchetti** preleva i pacchetti da varie interfacce di rete e li prepara per la pre-elaborazione.
- **Pre-processori** identifica gli header dei protocolli al livello di datalink, di rete, di trasporto e di applicazione. Si occupa del riassemblaggio dei pacchetti e del flusso TCP.
- **Motore di ricerca** componente principale che elabora e analizza i pacchetti applicando le regole. Confronta ogni pacchetto con un set di regole per vedere se esiste una corrispondenza.
- **Logging e alerting** sistema che si occupa della generazione di log e alert.

2.2 Modalità di base

Il comportamento di Snort nei confronti dei pacchetti varia a seconda della sua configurazione. Esistono diverse modalità di esecuzione che vengono presentate a seguire.

2.2.1 Sniffer Mode

Snort in sniffer mode ha un funzionamento analogo a tcpdump: sniffa il traffico che transita su un'interfaccia e lo stampa su console. Offre il riepilogo del traffico di rete alla fine dell'acquisizione. In questo modo vengono stampati con diversi livelli di dettaglio i contenuti dei pacchetti della suite TCP/IP, è possibile decidere di mostrare l'header, il payload o tutto il pacchetto. È importante sottolineare che per ora non sono state ancora introdotte regole, quindi Snort analizzerà e stamperà a schermo indiscriminatamente tutto il traffico presente sull'interfaccia.

I parametri che possono essere utilizzati sono:

- **-i** per specificare l'interfaccia da sniffare.
- **-v** verbose.
- **-d** mostra il payload del pacchetto.
- **-e** mostra l'header del pacchetto.
- **-X** per fare il display di tutto il contenuto del pacchetto.

Un esempio di utilizzo è il seguente:

```
snort -i ens160 -v.
```

In questo modo partirà un'istanza di Snort che analizzerà il traffico sull'interfaccia ens160 e lo stamperà su console in verbose mode.

2.2.2 Packet Logger Mode

Il passaggio successivo allo sniffing dei pacchetti è la loro registrazione. Snort eseguito in packet logger mode memorizza su disco il traffico rilevato. La directory predefinita

di Snort è `/var/log/snort`. Se i log venissero memorizzati solo in formato ASCII la directory di registrazione potrebbe diventare molto congestionata nel tempo, a causa del numero sempre crescente di directory e file. Registrare il traffico di una rete molto attiva potrebbe portare ad esaurire gli inode (una struttura dati Unix che limita il numero totale di file in un file system) molto prima di esaurire lo spazio di archiviazione. Questa esplosione di file può mettere a dura prova la macchina e potrebbe finire per trasformarsi in un vero e proprio attacco DoS. Per questo i log vengono registrati in binario e poi all'occorrenza vengono convertiti in formato ASCII per essere letti dall'amministratore.

I parametri che possono essere utilizzati sono:

- **-l** permette di specificare la directory dove si vogliono memorizzare i log.
- **-k ASCII** i log vengono memorizzati nel formato più leggibile per l'utente. Quando Snort viene eseguito in questa modalità, raccoglie ogni pacchetto che vede e lo colloca in una gerarchia di directory basata sull'indirizzo IP degli host.
- **-r** opzione per leggere i log memorizzati in binario.
- **-n** specifica il numero dei pacchetti da leggere.

I vari parametri possono essere combinati a quelli dello sniffer mode.

```
snort -v -i ens160 -l /var/log/snort
```

Terminando l'esecuzione e accedendo alla directory specificata sarà possibile vedere che è stato generato un file binario dal nome snort.log.*. Per leggerlo è possibile utilizzare l'opzione -r in questo modo:

```
snort -r snort.log.1655428986 -n 10
```

L'amministratore sarà in grado di leggere gli ultimi 10 pacchetti in formato ASCII.

2.2.3 IDS Mode

Nelle modalità viste finora Snort non ha funzionato da IDS/IPS, mancava l'ingrediente fondamentale: le regole. Le regole vanno specificate nel file di configurazione il cui path

predefinito di solito è `/etc/snort/snort.conf`. È possibile avere più file snort1.conf, snort2.conf etc... e decidere quale utilizzare nel momento in cui si lancia una nuova istanza di Snort. Se un pacchetto corrisponde ad una regola viene registrato e si genera un alert, altrimenti viene eliminato. Alla creazione di un alert l'amministratore di rete viene avvisato (anche tramite email) e può intervenire tempestivamente. Dovendo funzionare in real time, per stare al passo con la velocità della rete si utilizza unified2, che registra in forma binaria il più velocemente possibile. Non dovrebbe essere utilizzata l'opzione -v soprattutto da Bash (Shell di CentOS), perché considerata lenta e dispendiosa in termini di cicli di CPU, e potrebbe comportare la perdita di alcuni pacchetti.

I parametri che possono essere utilizzati sono:

- **-c** permette di definire il file di configurazione a cui fa riferimento una specifica istanza di Snort.
- **-T** per testare il file di configurazione.
- **-N** funziona da demone (in backgound).
- **-A** alert mode.

Esistono diversi tipi di alert mode:

- **full alert mode** fornisce tutte le possibili informazioni sull'alert. Viene usato come mode di default. I log vengono registrati in formato tcpdump.log.
- **fast** vengono generati degli alert che mostrano timestamp, IP sorgente e destinazione, i numeri delle porte. I log vengono registrati in formato tcpdump.log.
- **console** gli alert vengono visualizzati su console. I log vengono registrati in formato tcpdump.log.
- **cmg** venono mostrati sulla console dettagli dell'header e del payload del pacchetti in formato di testo e esadecimale.
- **none** per disabilitare gli alert. Questa opzione viene utilizzata se si vogliono disabilitare i normali log a vantaggio dei log unified2.

Esempio di utilizzo:

```
snort -i ens160 -A console -c /etc/snort/snort.conf
```

2.2.4 Inline mode

Oltre alle modalità viste finora Snort ne possiede una quarta, l'inline mode. In IDS mode non viene bloccato in nessun modo il traffico, si utilizzano delle librerie pcap per catturare i pacchetti, che poi vengono analizzati dal motore di ricerca e se si rivela necessario vengono generat deli alert. Snort-inline invece utilizza un modulo ip-queue e delle librerie libipq (alternative a pcap) che permettono di incanalare i pacchetti in una coda e applicare le regole iptables direttamente a quella coda. In base ad esse alcuni pacchetti verrano bloccati. In questa modalità sono presenti tre tipi di regole:

- **DROP** i pacchetti vengono scartati e loggati.
- **REJECT** i pacchetti vengono scartati, loggati e viene chiusa la connessione TCP, oppure viene inviato un ICMP port-unreachable.
- **SDROP** scarta i pacchetti senza registrare nulla.

2.2.5 FwSnort

Un altro modo per realizzare intrusion prevention è utilizzare FwSnort, un software che permette di tradurre le regole di Snort in regole iptables. Il funzionamento è il seguente: FwSnort si occupa di creare uno script che, se eseguito, provvede ad aggiungere le regole al firewall. Normalmente il path di default utilizzato per leggere le regole è `/etc/fwsnort/snort-rules`. L'unico limite di questa configurazione è il fatto di non lavorare in real-time, le regole iptables non vengono aggiornate in tempo reale e quindi non mettono al riparo da attacchi non previsti dal set di regole utilizzate in quel momento.

2.3 Rules

2.3.1 Community Rules e PulledPork

Le regole della community sono tutte le regole che sono state presentate dai membri della comunità open source. Vengono aggiornate ogni giorno ma è possibile che la frequenza vari a seconda dell'urgenza delle vulnerabilità rilevate. Per sollevare l'amministratore dell'onere di aggiornare manualmente le rules a cadenza regolare si utilizza PulledPork, un plugin che permette di scaricare automaticamente pacchetti di regole. Esegue anche verifiche checksum per tutti i download di regole principali. Genera automaticamente dei file sid-msg.map aggiornati che permettono la mappatura tra i nomi degli avvisi msg (il messaggio che compare a schermo/nei log) e i sid (l'id della regola)

2.3.2 Local Rules

Per realizzare un sistema di difesa con Snort potrebbe essere utile all'amministratore scrivere personalmente delle regole. Le rules che non vengono scaricate dalla comunità, ma che vengono scritte da utenti autorizzati sono chiamate "local rules". Hanno un campo header e un campo option.

Action	Protocol	Source IP	Source Port	Direction	Destination IP	Destination Port	Options
Alert Drop Reject	TCP UDP ICMP	ANY	ANY	<>	ANY	ANY	Msg Reference Sid Rev
Rule Header							Rule Options

Figura 2.2: Regola di Snort

L'header comprende a sua volta diversi campi:

- **Action** alert, drop (blocca e logga), reject (blocca logga e termina la connessione)
- **Protocol** tcp,udp,icmp
- **Source IP** any, un intervallo di ip, uno specifico ip
- **Source port** any, un range di porte,una porta

- **Direction** -> se si è interessati esclusivamente ai pacchetti che partono dal source ip e arrivano al destination ip, <> se si è interessati ai traffico in ambi i sensi.
- **Destination IP** any, un intervallo di ip, uno specifico ip.
- **Destination port** any, un range di porte,una porta.

L'options invece:

- **msg** è il messaggio che compare a schermo o nei log.
- **reference** riferimento a un CVE (Common Vulnerabilities and Exposures).
- **sid** l'id della rule. Deve essere un numero > di 1000000, le regole al di sotto sono riservate.
- **rev** informazioni sulla modifica o l'update della regola. Se rev:1 la regola non è mai stata revisionata.

Esempio:

```
alert icmp ant any -> 10.222.111.2 any (msg:"ICMP test detected";
sid:1000001; rev:001;)
```

Snort manda un alert per i pacchetti che hanno source-ip: any, source-port: any, destination-ip: 10.222.111.2, destination-port: any.

Infine il campo options delle regole può contenere un payload, oppure dei flags per distinguere i FIN, SYN, ACK

Ad esempio:

```
alert tcp any any <> any 80 (msg: "GET Request Found"; content:"GET";
sid: 100001; rev:1;)
```

Crea un alert per le GET http.

```
alert tcp any any <> any any (msg: "FLAG TEST"; flags:S; sid: 100001; rev:1;)
```

Crea un alert per i SYN.

2.4 Alert e log

Comprendere il meccanismo di alerting e logging di Snort può facilitare l'apprendimento dell'intero funzionamento del sistema di rilevazione. A seguire sono mostrati alcuni dei principali strumenti di appoggio che permettono la corretta creazione dei file di log e registrazione dei pacchetti.

2.4.1 Libpcap, Libnet, tcpdump

Libpcap, Libnet, tcpdump sono API che permettono di costruire iniettare e gestire pacchetti di rete, svolgono il ruolo di packet analyzer. Vengono utilizzate per acquisire o inviare pacchetti da un dispositivo di rete live o da un file.

2.4.2 DAQ

Snort introduce il DAQ, Data Acquisition System. Il DAQ sostituisce le chiamate dirette alle funzioni libpcap che si occupano della cattura dei pacchetti, per conentire un livello di astrazione che faciliti il funzionamento di Snort.

2.4.3 Pcap

Pcap è la libreria DAQ predefinita. Se snort viene eseguito senza argomenti DAQ, funzionerà utilizzando questo modulo. Pcap si occupa della cattura dei pacchetti che verranno poi successivamente analizzati.

2.4.4 Unified2

Uno dei formati di file binario in cui possono essere registrati i pacchetti si chiama "Unified2". Unified2 è anche il nome del parser che consente di elaborare i log in oggetti python. Lo scopo principale è estrarre i dati di un pacchetto dal log [MRR20] [man] [man]..

Capitolo 3

Requisiti

3.1 Requisiti funzionali

Un'azienda dispone di un server su cui ha installato il virtualizzatore VMware che permette di creare/gestire macchine virtuali. Una macchina virtuale ha le stesse funzionalità di un computer fisico ma utilizza componenti hardware astratte. L'azienda ne vuole utilizzare una per esporre dei servizi su internet. Per farlo si serve un router che ha due schede di rete, una con indirizzo IP pubblico 79.61.138.204 e l'altra con IP privato 10.222.111.1 che si affaccia sulla LAN 10.222.111.0/24. L'ambiente è composto da una prima macchina virtuale chiamata vm firewall su cui gira il sistema operativo CentOS 7, una distribuzione linux. La macchina dispone di un'unica scheda di rete (escludendo l'interfaccia di loopback) la ens160, corrispondente all'indirizzo IP 10.222.111.2. Le risorse assegnate alla vm sono:

- 8 core
- 32 GB ram
- 250 GB hdd

Il router è configurato in modo da inoltrare tutti i pacchetti che hanno come destinatario 79.61.138.204 sull'interfaccia ens160 della vm firewall.

L'azienda ha l'esigenza di utilizzare dei servizi non recenti, non sempre aggiornati e teme per la sicurezza della propria rete. Il suo obiettivo è garantire riservatezza,

integrità, disponibilità dei dati in qualsiasi momento. Se ad esempio non riuscisse a mantenere private le informazioni personali degli utenti che utilizzano i servizi offerti, andrebbe incontro a gravi ripercussioni (anche legali). Allo stesso modo, se non riuscisse a mantenere attivo il servizio a causa di malfunzionamenti potrebbe subire perdite economiche. Allo scopo di proteggere la rete vuole configurare manualmente il firewall con iptables.

I servizi che vuole esporre in rete sono:

- un web server usando il protocollo applicativo HTTP (sulla porta 80) e HTTPS (sulla porta 443) pubblicato utilizzando il servizio Http Apache versione 2.4.6
- un web server con Tomcat versione 7.0.76
- un file server con Samba versione 4.10.16

L'azienda richiede i seguenti requisiti funzionali, legati ancora una volta al tema della sicurezza.

3.1.1 Requisito 1: monitoraggio del traffico

Un utente deve essere in grado di utilizzare i servizi installati sulla macchina in sicurezza. Quando arriva del traffico sulle porte relative ai servizi installati, deve essere notificato e registrato, in modo che un amministratore di rete possa successivamente investigare il contenuto dei pacchetti se lo ritiene necessario. Deve essere possibile poter effettuare dei controlli mirati sui pacchetti, ad esempio analizzare solo le GET HTTP, solo i pacchetti provenienti da un IP o da un range di IP, su una porta o su un range di porte.

3.1.2 Requisito 2: rilevazione di attacchi

Devono poter essere rilevati possibili attacchi in rete. Se avviene una rilevazione deve essere notificata all'amministratore, in modo che possa intervenire per contrastarla. L'avviso deve essere chiaro e far riferimento all'attacco. Ad esempio un volume non indifferente di traffico rilevato in modo continuo deve essere identificato come "possible DoS attack" e non "UDP packet detected" o "TCP packet detected".

3.1.3 Requisito 3: prevenzione di attacchi

Devono essere prese delle misure contro possibili attacchi in rete. Non è sufficiente rilevare la minaccia, il sistema deve essere in grado di prevenire l'attacco in modo da da esonerare l'amministratore da un intervento repentino (qualora possibile). Si richiede quindi l'uso di una tecnologia capace di intervenire attivamente per eliminare pacchetti che sono ritenuti sospetti.

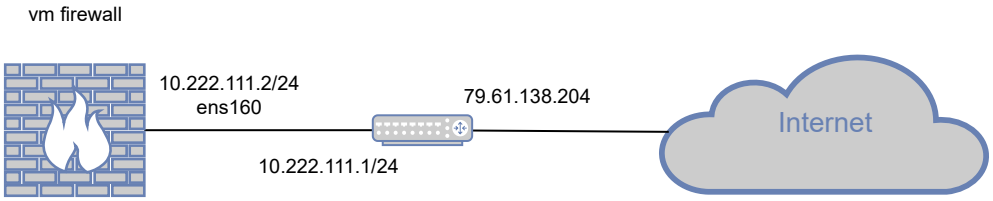

Figura 3.1: Rete nel suo stato primordiale.

3.2 Requisiti non funzionali

L'azienda richiede anche che venga rispettata una serie di requisiti non funzionali.

3.2.1 Modularità

Un sistema modulare separa funzionalità distinte in aree distinte. Il vantaggio di un sistema di questo tipo è riuscire a identificare una correzione o riuscire ad apportare un cambiamento, senza dover stravolgere la configurazione precedente, individuando l'area interessata e intervenendo localmente. Le modifiche ricadranno esclusivamente su di essa e non si propagheranno sull'intero sistema.

3.2.2 Scalabilità

Un sistema scalabile permette l'aggiunta di funzionalità e ulteriori requisiti a posteriori senza stravolgerne la struttura, ma solo applicando le nuove caratteristiche richieste nell'area di competenza. Se ad esempio nel prossimo futuro l'azienda decidesse di espandersi e di esporre nuovi servizi, potrebbe farlo senza riorganizzare l'intera struttura della rete, agendo in un contesto molto più ristretto.

3.2.3 Open source

L'azienda richiede che vengano utilizzati software open source. Non solo perché non richiedono costi iniziali, ma perché garantiscono maggiore stabilità e protezione, grazie alla loro diffusione e al supporto di sviluppatori esperti che partecipano alla comunità. Un software di questo tipo libera l'azienda dal dipendere da un unico produttore, un'unica architettura, un unico protocollo, al contrario permette un alto livello di integrazione. Il codice open source è continuamente revisionato e corretto. Nel caso di sistemi di rilevazione e prevenzione delle intrusioni utilizzare codice affidabile e aggiornato garantisce di raggiungere un secondo obiettivo: avere un basso numero di falsi positivi e negativi.

Capitolo 4

Realizzazione

4.1 Progetto

Dopo aver analizzato i requisiti presentati nel capitolo 3 si inizia a realizzare il progetto. Si procede come prima cosa con la suddivisione della rete per aree funzionali. La rete deve:

- mantenere elevati standard di sicurezza
- esporre i servizi su internet

Queste aree di lavoro vanno tenute separate per soddisfare due dei requisiti indicati dall'azienda: la modularità e la scalabilità. Per perseguire tale scopo si sceglie di introdurre una seconda macchina virtuale chiamata vm victim che ospiti dei web server e file server. Il nome 'victim' è giustificato dal fatto che ogni possibile attacco in rete sarà direzionato verso questa macchina virtuale, poiché espone dei servizi su internet. Si intende assegnare alla vm firewall ogni responsabilità relativa alla gestione della sicurezza dell'intera rete (firewall e altri strumenti di difesa...), mentre alla vm victim ogni responsabilità circa la gestione di dati e servizi. Si decide di procedere così:

- Si dedica un intervallo di indirizzi IP 10.50.50.0/24 ad una LAN, chiamata LAN dei server.
- Alla vm firewall viene aggiunta una seconda scheda di rete ens224, con ip 10.50.50.1.

- La vm victim appartiene alla LAN dei server, ha un'unica interfaccia ens160 (oltre a quella di loopback) che corrisponde all'indirizzo 10.50.50.3. Il router di default è 10.50.50.1.

In questo modo la vm firewall si trova tra il router e la vm victim. Il router è configurato in modo da rigirare tutto il traffico che proviene da internet sull'interfaccia esterna della vm firewall (ens160). La vm firewall dopo aver filtrato e analizzato i pacchetti li smista tra i vari host della LAN dei server seguendo le regole di forwarding, svolgendo così il ruolo di router per la 10.50.50.0/24. Questa configurazione è perfetta per soddisfare il requisito di modularità e per permettere il corretto funzionamento della rete. Le macchine virtuali sono separate per ruoli, e questo permette di non propagare modifiche e correzioni, bensì qualsiasi intervento ricadrà unicamente nell'aria di competenza. Una rete realizzata in questo modo è anche scalabile. Se nel prossimo futuro l'azienda decidesse di espandersi e avesse bisogno di creare altre macchine virtuali, potrebbe popolare la LAN dei server con ben altri 252 dispositivi. Il cuore dell'intera rete è la LAN dei server, su cui vengono installati dei servizi che sono esposti su internet. Per evitare perdita e compromissione dei dati, tentativi di connessione non autorizzati, injection di codice, e altra attività malevola bisogna lavorare sulla vm firewall.

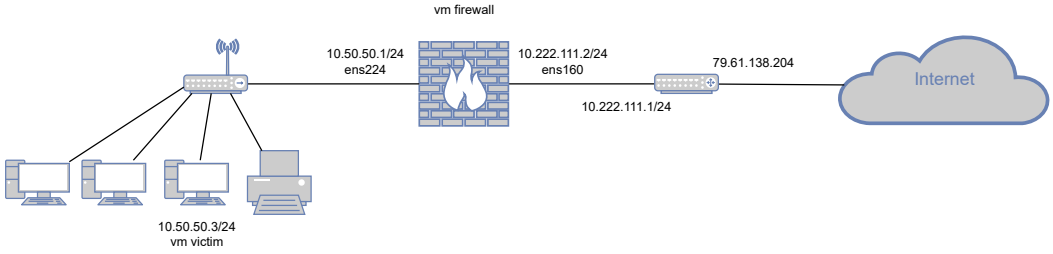

Figura 4.1: Progetto di realizzazione della rete che soddisfi i requisiti imposti nel capitolo 3.

Vengono elencati brevemente i vari componenti della rete da sinistra verso destra:

- **vm victim** nella LAN dei server con una scheda di rete:

– ens160 con IP 10.50.50.3, il router di default è 10.50.50.1

- **uno switch virtuale**
- **vm firewall** che ospita il firewall e Snort, con due schede di rete:
  – ens160 con IP 10.222.111.2, il router di default è 10.222.111.1
  – ens224 con IP 10.50.50.1
- **il router** con due schede di rete:
  – IP privato 10.222.111.1
  – IP pubblico 79.61.138.204

Per la realizzazione della macchina virtuale vm victim viene seguita la linea definita dall'azienda e viene utilizzato il sistema operativo CentOS 7. La macchina viene configurata in questo modo:

- viene settato l'IP statico 10.50.50.3.
- il DNS di riferimento è uno dei DNS server di Google con IP 8.8.8.8.
- vengono installati i web server, file server richiesti.

Il passo successivo è mostrare nel dettaglio la realizzazione del sistema di rilevazione e prevenzione delle intrusioni con Snort.

Capitolo 5

Installazione e configurazione di Snort

5.1 Prerequisiti e installazione

Come prerequisito è necessario installare delle librerie che permettono di catturare pacchetti: gcc, flex, bison, zlib, libpcap, pcre, libdnet, tcpdump e libnghttp2.

Poi viene installato Snort con il package manager della distribuzione utilizzata.

Snort su CentOS viene installato nella directory `/usr/local/bin/snort`, è buona norma creare un link simbolico nel path `/usr/sbin/snort`. `/usr/sbin` è dedicata a programmi di gestione del sistema (normalmente non utilizzati dagli utenti ordinari).

```
ln -s /usr/local/bin/snort /usr/sbin/snort
```

Per eseguire Snort in modo sicuro senza che esegua con i privilegi di root, bisogna creare un nuovo utente e un nuovo gruppo di utenti.

```
groupadd snort
useradd snort -r -s /sbin/nologin -c SNORT_IDS -g snort
```

5.2 Configurazione

Per adeguare Snort alle esigenze della rete bisogna:

- modificare il file di configurazione.
- importare le regole.
- gestire i log.

Le principali directory utilizzate da Snort sono:

`/etc/snort`: in questa directory si trova il file di configurazione snort.conf e la sottodirectory che contiene le rules.

`/var/log/snort`: in questa directory vengono memorizzati i log, si possono specificare anche altri path, questo è quello di default.

Bisogna assicurarsi che lo user snort e il gruppo snort abbiano i permessi di lettura/ scrittura/ esecuzione. Viene creato un file per le local.rules, ossia per tutte quelle regole customizzate dall'amministratore.

Ora si arriva al cuore della configurazione. Bisogna modificare il file snort.conf in modo da soddisfare i requisiti della rete. Snort deve catturare e analizzare i pacchetti provenienti dall'interfaccia esterna della vm firewall. Per fare questo bisogna definire la variabile HOME_NET e assegnargli il range di IP da proteggere 10.222.111.0/24. Viene creata anche una seconda variabile EXTERNAL_NET per indicare la rete esterna, in questo caso rappresentante tutti gli indirizzi IP non compresi nella HOME_NET

```
ipvar HOME_NET 10.222.111.0/24
ipvar EXTERNAL_NET !$HOME_NET
```

Qui viene specificato il formato dei log, in questo caso utilizzando unified2 e tcpdump

```
# unified2
output unified2: filename merged.log, limit 128, nostamp
# pcap
output log_tcpdump: tcpdump.log
```

Per includere le local.rules:

```
include /etc/snort/rules/local.rules
```

Manca un ultimo passaggio: includere le community rules.

5.2.1 Community Rules

È possibile trovare le Community Rules sul sito ufficiale di Snort.

Per estrarre le regole e copiarle nella directory dove si trovano le rules:

```
tar -xvf ~/community.tar.gz -C ~/
cp ~/community-rules/* /etc/snort/rules
```

Per includere tutte le regole della community in `/etc/snort/snort.conf`

```
include $RULE_PATH/snort.rules
```

5.3 Pulled Pork

Ora è possibile procedere con l'installazione di PulledPork, il plugin che permette di scaricare automaticamente le nuove regole della community.

Viene installato PulledPork con il package manager della distribuzione utilizzata.

Lo script si trova nella directory `/usr/local/bin/` e viene reso eseguibile con il comando:

```
chmod +x /usr/local/bin/pulledpork.pl
```

Per verificare il corretto funzionamento:

```
/usr/local/bin/pulledpork.pl -V
```

Il file di configurazione si trova in `/etc/snort`, si chiama pulledpork.conf e va modificato.

Viene specificato il path delle community rules da aggiornare:

```
rule_path=/usr/local/etc/snort/rules/snort.rules
```

Pulled pork richiede anche il path delle local.rules per costruire dei file sid-msg.map che contengono informazioni sui metadati (msg) delle local.rules

```
local_rules=/usr/local/etc/snort/rules/local.rules
```

Pulled pork aggiorna una blocklist di indirizzi IP bloccati dalla comunità:

```
block_list=/usr/local/etc/snort/rules/iplists/default.blocklist
```

5.4 Il demone snortd

Per far sì che Snort lavori in background, viene eseguito come demone. Un sistema di difesa ha l'esigenza di catturare pacchetti e analizzare il traffico durante tutto il periodo di attività della macchina. Per questo viene largamente sfruttata questa possibilità.

```
systemctl daemon-reload
systemctl start snortd
```

Per avere la conferma che l'avvio del demone sia andata a buon fine:

```
systemctl status snortd
```

5.5 FwSnort

FwSnort è uno script che traduce le regole di Snort in regole iptables. Alcune non hanno una traduzione diretta, tuttavia circa il 65% delle regole possono essere tradotte con successo utilizzando l'iptables string match module (modulo che confronta l'inizio della stringa che compone la regola con la chiave di una mappa. Se viene trovata una corrispondenza restituisce l'oggetto corrispondete, un iptables). FwSnort analizza anche la policy iptables in esecuzione sulla macchina per determinare quali regole Snort sono applicabili.

È possibile scaricare il codice sorgente dal sito ufficiale.

Per estrarre ed eseguire il file a partire dal pacchetto bz2:

```
tar xfj fwsnort-1.0.tar.bz2
cd /usr/local/src/fwsnort-1.0
./install.pl
```

Il file di configurazione si trova nella directory `/etc/fwsnort/fwsnort.conf`

Nel file di configurazione vengono definite delle variabili che corrispondono a indirizzi IP o a porte e vengono utilizzate nelle regole.

```
HOME_NET                10.222.111.0/24;
EXTERNAL_NET            !$HOME_NET;

HTTP_SERVERS            $HOME_NET;
DNS_SERVERS             $HOME_NET;

SSH_PORTS               [64022,65022];
HTTP_PORTS              80;
SHELLCODE_PORTS         !80;
```

Si noti che è stato scelto un range di porte SSH al posto della porta di default 22. Questo per motivi di sicurezza: leggendo i log SSH nella directory `/var/log/secure` e i log registrati dallo stesso Snort, si è visto che esistono degli script automatizzati che tentano continuamente di instaurare una connessione SSH con il server. Per eliminare il problema, SSH è stato reso disponibile su un altre porte.

Un esempio di regola facente parte delle community rules che usa le variabili specificate durante la configurazione:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg:"SERVER-APACHE Apache Tomcat view source attempt";
 flow:to_server,established; content:"%252ejsp"; http_uri;
metadata:ruleset community, service http;
reference:bugtraq,2527; reference:cve,2001-0590;
classtype:web-application-attack; sid:1056; rev:16;)
```

[Ras07] [tec].

Capitolo 6

Verifiche funzionali

Questo capitolo è dedicato alla realizzazione di verifiche funzionali che permettano di attestare quale sia il grado di difesa che Snort sia in grado di garantire. Il metodo più semplice per dimostrare l'efficacia di un sistema di rilevazione e prevenzione delle intrusioni è determinare se:

- Snort abbia rilevato pacchetti.
- Abbia generato alert.
- Abbia agito attivamente per prevenire l'attacco.

Un altro criterio che verrà utilizzato è confrontare il numero di pacchetti inviati con il numero di pacchetti elaborati da Snort, se questi numeri sono simili vuol dire che ogni attività viene opportunamente individuata ed elaborata senza grosso margine di errore.

6.1 Test 1: il generatore di traffico

6.1.1 Scapy

Per effettuare i test è stata utilizzata Scapy, una libreria che consente di generare traffico. Scapy fa principalmente due cose: inviare pacchetti e ricevere risposte. In questo caso verrà utilizzato solo nella prima modalità di esecuzione, come strumento per generare un flusso di traffico continuo che arrivi all'indirizzo IP destinazione 79.61.138.204.

L'installazione è molto rapida e non viene mostrata nella sua interezza. È sufficiente avere Python 2.7 sulla macchina e utilizzare il comando `pip install scapy`. [sca]

6.1.2 Realizzazione

Il test coinvolge tre macchine:

- Un client in una rete domestica che ha IP 87.6.233.6 (chiamato per comodità host A) che interagisce con il server remoto e genera traffico.
- La vm firewall che ha IP 10.222.111.2, su cui è in esecuzione Snort.
- La vm victim, su cui sono in esecuzione web server, file server.

L'host A utilizza Scapy. L'obiettivo è inviare pacchetti sulle porte d'interesse del server. È bene ricordare i servizi installati sulla vm victim e le relative porte:

Nome del servizio	Numero di porta	Protocollo
HTTP	80	TCP
HTTPS	443	TCP
TOMCAT	8080	TCP
TOMCAT	8084	TCP
SAMBA	137	TCP
SAMBA	445	TCP
SAMBA	138	UDP
SAMBA	139	UDP

Dal lato server invece bisogna mettere Snort in condizione di rilevare tutti i pacchetti generati e inviati. Dunque, per tutti i servizi sopra citati viene scritta una corrispondente regola di Snort nelle local rules.

Vengono scritte le seguenti regole:

```
alert icmp any any -> $HOME_NET any (msg:"icmp detected"; GID:1;
sid:10000001; rev:001; classtype:icmp-event;)
alert TCP any any -> $HOME_NET 80 (msg:"Http detected"; sid: rev: ;)
alert TCP any any -> $HOME_NET 443 (msg:"Https detected"; sid: rev: ;)
```

```
alert TCP any any -> $HOME_NET 8080 (msg:"Tomcat detected"; sid: rev: ;)
alert TCP any any -> $HOME_NET 8443 (msg:"Tomcat detected"; sid: rev: ;)
alert TCP any any -> $HOME_NET 137 (msg:"SMB detected"; sid: rev: ;)
alert TCP any any -> $HOME_NET 445 (msg:SMB detected"; sid: rev: ;)
alert UDP any any -> $HOME_NET 138 (msg:"SMB detected"; sid: rev: ;)
alert UDP any any -> $HOME_NET 139 (msg:"SMB detected"; sid: rev: ;)
```

La sintassi viene ricordata brevemente:
Snort creerà un alert per i pacchetti che hanno:

- **source-ip** any.
- **source-port** any.
- **destination-ip** HOME_NET
- **destination-port** any, 80, 443, a seconda dell'esigenza.

Si ricorda inoltre il significato dei seguenti campi:

- **msg** rappresenta il messaggio che compare a schermo o nei log.
- **sid** è l'id della regola
- **rev** fornisce informazioni sulla modifica o l'update della regola. In questo vale 001 perché regola non è mai stata revisionata.

Bisogna verificare che le local rules siano state importate nel file di configurazione.
È inoltre necessario riavviare il servizio:

```
systemctl restart snortd
```

Prima di iniziare a generare traffico sono state prese queste due scelte per la visualizzare gli alert:

- Snort stamperà su console gli avvisi con il comando -A console.
- Snort memorizzerà i log nella directory /var/log/snort

E si procede per far partire un'istanza di Snort come IDS:

```
snort -i ens160 -c /etc/snort/snort.conf -A console -l /var/log/snort
```

Da questo momento in poi qualsiasi pacchetto che arriva sull'interfaccia ens160 e che corrisponde ad una delle local.rules comparirà sulla console.

Sull'host A si decide di aprire tre istanze distinte di Scapy: una per il traffico TCP, una per l'UDP e una per l'ICMP. Vengono eseguiti questi tre comandi in parallelo (nonostante qui vengano mostrati in sequenza):

```
send(IP(dst="79.61.138.204")/TCP(dport=[80,443,8080,8443,137,445]),
loop=1,inter=0.01)
send(IP(dst="79.61.138.204")/UDP(dport=[138,139]),loop=1,inter=0.01)
send(IP(dst="79.61.138.204")/ICMP(),loop=1,inter=0.01)
```

Anche in questo caso la sintassi viene ricordata:

Scapy invierà a intervalli di 0.01 secondi un pacchetto all'IP destinazione 79.61.138.204 nelle porte specificate nell'opzione dport.

Appena viene lanciato il comando sulla console della vm firewall Snort genera i seguenti avvisi:

```
07/02-12:53:21.272905 [1:10000001:1] icmp detected [**] [Classification:
Generic ICMP event] [Priority: 3] {ICMP} 87.6.233.6 -> 10.222.111.2
Http detected [**] [Priority: 0] {TCP} 87.6.233.6:50491 ->
10.222.111.2:80
icmp detected [**] [Classification: Generic ICMP event] [Priority: 3]
{ICMP} 87.6.233.6 -> 10.222.111.2
Https detected [**] [Priority: 0] {TCP} 87.6.233.6:51313 ->
10.222.111.2:443
SMB detected [**] [Priority: 0] {UDP} 87.6.233.6:54013 ->
10.222.111.2:139
Tomcat detected [**] [Priority: 0]  {TCP} 87.6.233.6:65508 ->
10.222.111.2:8080
```

```
Tomcat detected [**] [Priority: 0] {TCP} 87.6.233.6:61385 ->
10.222.111.2:8443
SMB detected [**] (Priority: 0]  {UDP} 87.6.233.6:54013 ->
10.222.111.2:139
```

Il flusso è continuo e gli alert si leggono in modo chiaro. Un amministratore interessato ad approfondire il contenuto dei pacchetti potrebbe utilizzare i comandi:

- snort -r.
- u2spewfoo.

per leggere i log che si sono generati nella directory indicata. Dopo circa 25 minuti viene interrotto il flusso di traffico. Scapy mostra dei messaggi che indica il numero di pacchetti inviati:

Per il traffico TCP: `Sent 108761 packets.`

Per il traffico ICMP: `Sent 118540 packets.`

Per il traffico UDP: `Sent 120121 packets.`

Snort mostra sulla console la durata di esecuzione dell'istanza di Snort, il numero di pacchetti ricevuto al minuto e al secondo, il numero di pacchetti ricevuti e analizzati:

```
===============================================================
Run time for packet processing was 1549.301984 seconds
Snort processed 1341041 packets.
Snort ran for 0 days 0 hours 25 minutes 49 seconds
Pkts/min:           53641
Pkts/sec:           865
===============================================================
Packet I/O Totals:
Received:           1368418
Analyzed:           1341042 ( 97.999%)
Dropped:            27370   ( 1.961%)
Filtered:               0   ( 0.000%)
Outstanding:        27376   ( 2.001%)
```

```
Iniected:               0    ( 0.000%)
===================================================================
```

Snort in totale ha ricevuto più di un milione di pacchetti. La voce **"Outstanding"** indica quanti pacchetti sono memorizzati nel buffer in attesa di elaborazione. Questo risultato dipende dalla specifica implementazione del DAQ. La voce **"Drop"** indica quanti pacchetti sono stati persi, ossia quanti pacchetti Snort non ha elaborato. Se questi due valori coincidono vuol dire che il numero di pacchetti persi è causato dalla repentina interruzione dell'istanza di Snort.

Infine Snort mostra quanti pacchetti abbiano generati log e alert:

```
===================================================================
Action Stats:
Alerts:      276257 ( 20.600%)
Logged:      276257 ( 20.600%)
Passed:      0 (0.000%)
===================================================================
```

Bisogna fare due considerazioni.

- Il numero di pacchetti ricevuti (più di un milione) è di molto superiore rispetto a quello che ha generato un alert. Questo perché il test è stato effettuato in connessione SSH con la vm firewall. La maggior parte del traffico è riconducibile ai pacchetti SSH che non sono stati registrati.

- Scapy aveva comunicato che i pacchetti inviati fossero:

  108761 + 118540 + 120121 = 347422.

  Snort ne ha loggati solo 276257, quindi mancano 71165 pacchetti. La spiegazione più probabile è che una parte sia rimasta nel buffer e non sia stata elaborata, mentre una parte non sia stata registrata a causa di vincoli del mondo reale, ad esempio limiti sul tempo di elaborazione, limiti sulla memoria, oppure problemi di connessione.

Il primo test si può definire concluso. È stato dimostrato il corretto funzionamento di Snort in modalità IDS. Snort ha analizzato più di un milione di pacchetti, di cui più di trecentomila hanno generato avvisi. Gli alert sono stati generati correttamente così come i log, in modo da permettere ad un amministratore di monitorare il traffico e studiare il contenuto dei pacchetti. Snort è riuscito ad elaborare e processare un numero considerevole di pacchetti, soddisfacendo in questo modo i requisiti imposti nei capitoli precedenti. Prima di passare al secondo test viene mostrato il modo di procedere di un amministratore che intende andare più a fondo nell'analisi di pacchetti. Non si limita a visualizzare gli avvisi sulla console, bensì vuole leggere il contenuto di un pacchetto.

6.1.3 tcpdump.log

La directory `/var/log/snort` si popola di log. È stato generato il file `tcpdump.log.1656762820`. È possibile leggere il contenuto con il comando `snort -r`. Con l'opzione `-n 10` vengono stampati a schermo solo gli ultimi dieci pacchetti.

```
snort -r tcpdump.log.1656762820 -n 10
```

Un amministratore che legge i log in formato tcpdump per un pacchetto TCP visualizzerà un contenuto simile a questo:

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
07/02-12:53:47.224263 87.6.233.6:52571 -> 10.222.111.2:443
TCP TTL:46 TOS:0x0 ID:24849 IpLen:20 DgmLen:44
S Seq: 0xF0988237  Ack: 0x0  Win: 0x400  TcpLen: 24
TCP Options (1) => MSS: 1452
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Da cui potrebbe ricavare diverse informazioni come:

- **time to live** TTL:46.
- **type of service (TOS)** specifica la priorità di un datagramma rispetto a un altro.

- **lunghezza dell'header** IpLen:20.
- **lunghezza del pacchetto header + data** DgmLen:44.
- **IP packet ID** intero che identifica un datagramma, se due frammenti hanno lo stesso id riassemblati nello stesso pacchetto.
- **sequence number Seq** in formato esadecimale.
- **il campo window Win** in formato esadecimale.

6.1.4 merged.log

È anche possibile leggere il file merged.log con il comando u2spewfoo.

```
u2spewfoo merged.log
```

Viene mostrato il payload di un pacchetto:

```
Packet
sensor id: 0 event id: 1 event second: 1656757856
packet second: 1656757856 packet microsecond: 501909
linktype: 1 packet_length: 66
[    0] 00 0C 29 BE C1 97 30 91 8F 4A 25 14 08 00 45 00  ..)...0..J%...E.
[   16] 00 34 36 68 40 00 71 06 9C 28 67 62 55 F1 0A DE  .46h@.q..(gbU...
[   32] 6F 02 EE 26 01 BD 25 50 91 32 00 00 00 00 80 02  o..&..%P.2......
[   48] 20 00 71 84 00 00 02 04 05 AC 01 03 03 02 01 01   .q.............
[   64] 04 02
```

[Reh]

6.2 Test 2: scan delle porte con Nmap

6.2.1 Nmap

Nmap è un tool Linux per l'esplorazione della rete e il controllo della sicurezza. Viene utilizzato per ricavare:

- informazioni in tempo reale di una rete.
- informazioni dettagliate di tutti gli IP visibili in una rete.
- l'elenco degli host attivi.
- una scansione di porte, sistemi operativi e host.

In questo test viene utilizzato per ottenere informazioni sul server remoto. Eseguire uno scan delle porte è una pratica preliminare comune se si decide di intraprendere un attacco. Non per forza viene utilizzata a questo scopo, tuttavia è bene che un sistema di rilevazione e prevenzione delle intrusioni lo rilevi [gee] [fra].

6.2.2 Realizzazione

L'host A esegue lo scan delle prime 10.000 porte TCP con il comando Nmap:

```
> sudo nmap 79.61.138.204
Starting map 7.92 ( https://nmap.org )
Host is up (0.011s latency).
Not shown: 996 filtered tc ports (no-response)
PORT        STATE       SERVICE
80/tcp      open        http
443/tcp     open        https
Nmap done: 1 IP address (1 host up) scanned in 6.05 seconds
```

Il report di Nmap mostra che le porte aperte sono 80, 443. Simultaneamente sono comparsi gli alert di Snort:

```
Commencing packet processing (pid=20613)
07/02-18:32:29.993754 [**] [1:10000004:1] Https scan detected [**]
[Priority: 0] {TCP} 87.6.233.6:64366 -> 10.222.111.2:443
Https scan detected [**] [Priority: 0] {TCP} 87.6.233.6:64366 ->
10.222.111.2:443
SMB scan detected [**] [Priority: 0] {TCP} 87.6.233.6:64622 ->
10.222.111.2:445
```

```
Tomcat scan detected [**] [Priority: 0] {TCP} 87.6.233.6:64622 ->
 10.222.111.2:8080
Tomcat scan detected [**] [Priority: 0] {TCP} 87.6.233.6:64624 ->
10.222.111.2:8080
SMB scan detected [**] [Priority: 0] {TCP} 87.6.233.6:64624 ->
10.222.111.2:445
Https scan detected [**] (Priority: 0] {TCP} 87.6.233.6:64622 ->
10.222.111.2:443
Http scan detected [**] [Priority: 0] {TCP} 87.6.233.6:64622 ->
10.222.111.2:80
Http scan detected [**] [Priority: 0] {TCP} 87.6.233.6:64622 ->
10.222.111.2:80
Https scan detected [**] [Priority: 0] {TCP} 87.6.233.6:64622 ->
10.222.111.2:443
Http scan detected [**] [Priority: 0] {TCP} 87.6.233.6:64627 ->
10.222.111.2:80
Http scan detected [**] [Priority: 0] {TCP} 87.6.233.6:64627 ->
10.222.111.2:80
```

L'host A esegue un secondo scan per rilevare le porte utilizzate dal protocollo UDP. Viene utilizzata l'opzione -sU.

```
> sudo map -sU 79.61.138.204
Starting Nmap 7.92 ( https://nmap.org )
Host is up (0.021s latency).
Not shown: 999 open|filtered up ports (no-response)
PORT    STATE   SERVICE
53/udp  closed  domain
Nmap done: 1 IP address (1 host up)
scanned in 7.04 seconds
```

Il report di Nmap trova la porta 53 utilizzata per il DNS, ma il servizio in ascolto non accetta connessioni e quindi risulta come chiusa.

Snort genera gli avvisi per il secondo scan:

```
07/02-18:33:44.150544 [**] [1:10000011:1] DNS scan detected [**]
[Priority: 0] {UDP} 87.6.233.6:37169 -> 10.222.111.2:53
DNS scan detected [**] [Priority: 0] {UDP} 87.6.233.6:37169 ->
10.222.111.2:53
DNS scan detected [**] [Priority: 0] {UDP} 87.6.233.6:37174 ->
10.222.111.2:53
DNS scan detected [**] [Priority: 0] {UDP} 87.6.233.6:37174 ->
10.222.111.2:53
[**] (1:10000009:1] SMB scan detected [**] [Priority: 0] {UDP}
87.6.233.6:37169 -> 10.222.111.2:139
DNS scan detected [**] [Priority: 0] {UDP} 87.6.233.6:37178 ->
10.222.111.53
```

Si conclude in questo modo il test 2: lo scan é stato rilevato da Snort con successo.

6.3 Test 3: simulazione attacco DoS

Si intende simulare un attacco DoS in modo da osservare il comportamento di Snort in modalità IDS e IPS. Il dispositivo che lancia l'attacco è l'host A, utilizzando Scapy, lo stesso software utilizzato nel test 1.

Dal lato server la vm firewall deve intraprendere un'azione di difesa. La prima decisione che va presa è la seguente: quand'è che Snort deve generare un alert per un possibile attacco DoS? Ossia, come distinguere un flusso di traffico normale da quello anomalo? Le regole di Snort permettono di stabilire una soglia di pacchetti al secondo oltre la quale viene segnalato un possibile attacco. Questa soglia dipende dalla capacità del server di gestire i pacchetti. In questo esempio però non è oggetto di interesse studiare le performance del software, bensì testare la capacità di Snort di rilevare e prevernire intrusioni. Quindi viene stabilita una soglia "simbolica" di 70 pacchetti ogni 10 secondi, sufficienti per indurre Snort a sospettare di un possibile attacco DoS. Viene scritta una regola di Snort nelle local rules:

```
alert tcp any any -> $HOME_NET any (msg:"TCP SYN flood"; flags:!A;
flow: stateless; detection_filter: track by_dst, count 70, seconds 10;
sid:2000003;)
```

[cyv]

Oltre ai normali campi msg, sid, compare il campo detection-filter che consente di specificare la soglia stabilità.

È possibile lanciare l'attacco dall'host A, che invierà un flusso di pacchetti TCP sulla porta 443:

```
>>> send (IP(dst="79.61.138.204")/TCP(dport=[443]),Loop=1,inter=0.01)
.............................................................
Sent 175 packets.
```

L'intervallo tra un pacchetto e il successivo è solo di 0.01 secondi. In poco tempo viene superata la soglia dei 70 pacchetti, ed iniziano a comparire gli alert sulla console generati da Snort:

```
Commencing packet processing (pid=13661)
07/02-19:55:07.920807 [**] (1:2000003:0] TCP SYN flood [**]
[Prioritv: 0] {TCP} 87.6.233.6:53015 -> 10.222.111.2:443
TCP SYN flood [**] [Priority: 0] {TCP} 87.6.233.6:53015 -> 10.222.111.2:443
TCP SYN flood [**] [Priority: 0] {TCP} 87.6.233.6:53015 -> 10.222.111.2:443
TCP SYN flood [**] [Priority: 0] {TCP} 87.6.233.6:53015 -> 10.222.111.2:443
TCP SYN flood [**] [Priority: 0] {TCP} 87.6.233.6:53015 -> 10.222.111.2:443
```

Sarebbe stato possibile prevenire l'attacco utilizzando Snort come IPS. FwSnort permette di convertire regole specifiche per l'attacco DoS di Snort in iptables.

FwSnort attinge alle rules nel path `/etc/fwsnort/snort.rules` e le converte in regole iptables. È bene sottolineare che FwSnort non sta convertendo le local rules, bensì delle regole nel file snort.rules specifiche per l'attacco DoS.

FwSnort traduce le regole di Snort in regole iptables:

```
root@firewall ~ fwsnort
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
Snort Rules File        Success        Fail       Total
[+] dos.rules              9             7           16
```

FwSnort genera uno script `/var/lib/fwsnort/fwsnort.sh`. Una volta eseguito le regole iptables per l'attacco DoS saranno in funzione.

Per verificare che la generazione delle regole sia avvenuta con successo si può utilizzare il comando `iptables -L | grep DOS` in modo da visualizzare tutte le regole che contengono la stringa DOS:

```
root@firewall ~ iptables -L | grep DOS
LOG        icmp -- !10.222.111.0/24      10.222.111.0/24
icmp echo-request STRING match  "+++ath" ALGO name bm TO 65535 ICASE;
msg:{DOS} ath; classtype  attempted-dos; reference:arachnids,
264; rev:5; FWS:1.6.8;
LOG        udp  -- !10.222.111.0/24      10.222.111.0/24
msg:{DOS Ascend Route}; classtype:attempted-dos;
LOG        tcp  -- !10.222.111.0/24      10.222.111.0/24
msg:{DOS Real Audio Server}; classtype:attempted-dos;
LOG        tcp  -- !10.222.111.0/24      10.222.111.0/24
msg:{DOS Real Server template.html}; classtype:attempted-dos;
LOG        tcp  -- !10.222.111.0/24      10.222.111.0/24
msg:{DOS arkiea backup}; classtype:attempted-dos;
LOG        tcp  -- !10.222.111.0/24      10.222.111.0/24
msg:{DOS MSDTC attempt}; classtype:attempted-dos;
```

Invece per ripristinare le regole iptables del firewall:

```
service iptables restart
```

È stato dimostrato come Snort reagisca tempestivamente anche nel caso di un attacco DoS, generando log,alert e facendo in modo che il firewall sia in grado di bloccare il traffico indesiderato.

Conclusioni e sviluppi futuri

Ecco alcuni spunti per risolvere criticità rimaste e per portare avanti il lavoro:

La rete è stata progettata soddisfacendo i requisiti di modularità e scalabilità. È ancora possibile aumentare il livello di modularità aggiungendo una terza macchina dedicata a Snort. Snort è per definizione un IDS/IPS network-based, può lavorare in modalità promiscua e intercettare indifferentemente tutti i pacchetti che transitano in rete, senza dover per forza analizzare i pacchetti sull'interfaccia esterna del firewall. Separare la macchina del firewall da quella che ospita il sistema di rilevazione e prevenzione delle intrusioni può risultare vantaggioso nello sfortunato caso in cui un host interno all'azienda attaccasse la vm firewall. Nell'ipotesi in cui la vm firewall subisse dei danni, Snort sarebbe comunque attivo su un'altra macchina.

Nella realizzazione dell'IDS/IPS la responsabilità di prevenire gli attacchi è stata affidata a FWsnort. Fin da subito è emerso un limite di questa configurazione: FWsnort non lavora in real-time, le iptables non vengono aggiornate in tempo reale e quindi non mettono al riparo da attacchi non previsti dal set di regole utilizzate in quel momento. Una soluzione più efficiente sarebbe utilizzare una modalità di esecuzione chiamata inline, che permette di aggiornare le regole iptables in modo dinamico.

Le tre verifiche descritte nel capitolo 6 hanno permesso di mostrare alcune delle funzionalità di Snort, ma c'è ampio margine per espandere il panorama di test. Sarebbe interessante studiare il comportamento in caso di un exploit con payload maligno, per vedere se le regole scaricate siano sufficienti per individuare stringhe e caratteri peculiari di un attacco.

Bibliografia

[cyb] https://www.cybersecurity360.it/soluzioni-aziendali/.

[cyv] https://cyvatar.ai/write-configure-snort-rules/.

[ext] https://www.extraordy.com/.

[fra] https://frankfu.click/security/ids/how-to-detect-nmap-scan-using-snort/.

[gee] https://www.geeksforgeeks.org/nmap-command-in-linux-with-examples/.

[man] http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node17.html/.

[MRR20] Shatel group of companies October 13 Milad Rezaei RD. *Snort 2.9.16.1 on Centos 8*. 2020.

[Ras07] Michael Rash. *Linux Firewalls*. 2007.

[Reh] Rafeeq Ur Rehman. *Intrusion Detection Systems with Snort: Advanced IDS Techniques Using Snort, Apache, MySQL, PHP, ACID*.

[sca] https://scapy.readthedocs.io/en/latest/.

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

[sof] https://softwarelab.org/it/exploit/.

[tec] https://www.techrepublic.com/article/using-snort-for-intrusion-detection/.

[try] https://tryhackme.com/room/snort/.

[zer] https://www.zerounoweb.it/cloud-computing/.