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

Questa è la dedica

Ringraziamenti

Grazie a tutti

Introduzione

Questa è l'introduzione

Indice

Elenco delle figure

Capitolo 1

Esempio di Capitolo

1.1 Questa è una Sezione

1.1.1 Questa è una Sottosezione

Capitolo 2

Altro Esempio di Capitolo

2.1 Questa è una Sezione

2.1.1 Questa è una Sottosezione

Ancora del testo

Come si evince dalle Figure 2.1.a e 2.1.b non si capisce molto.

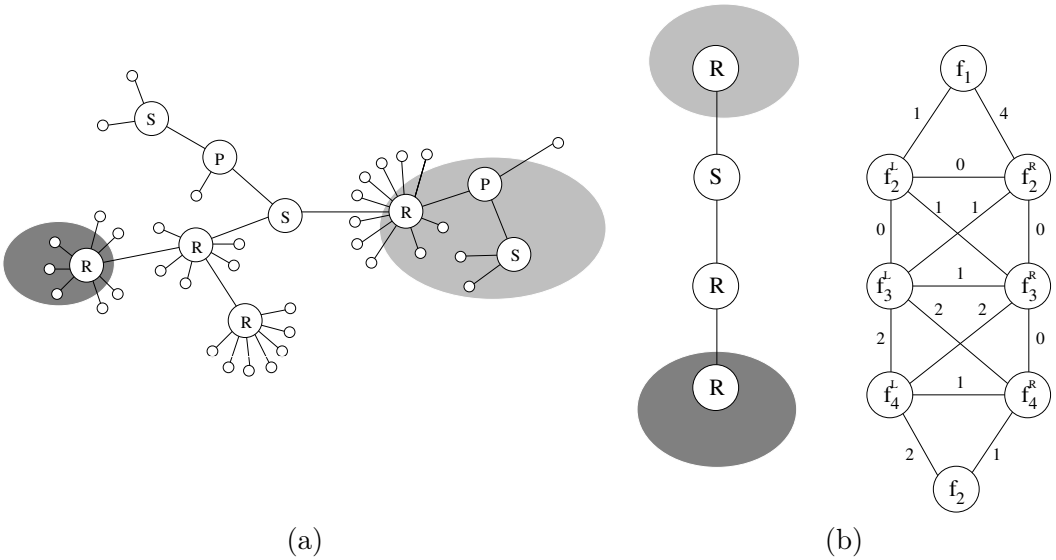

Figura 2.1: SPQR-tree di un grafo. (a) L'albero di allocazione della faccia esterna. (b) Il cammino notevole di cui si parla tanto nella Sezione 2.1.

Capitolo 3

Sicurezza della rete

3.1 Sicurezza e vulnerabilità della rete

3.1.1 Quando un sistema è sicuro?

I dati contenuti in un pc permettono di raggiungere un alto grado di conoscenza circa il contesto in cui esso viene utilizzato (circa il contesto in cui si trova). In un pc privato potrebbero essere memorizzati dati di carte di credito, numeri di conti correnti bancari, fotografie, file personali. In un pc utilizzato da un'azienda potrebbero essere memorizzati i dati dei clienti, rendiconti finanziari, documenti etc... Nel momento in cui un pc entra a far parte di una rete ed è quindi 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 pc, smartphone, tablet ... La sicurezza di un sistema può essere analizzata e descritta a partire da cinque caratteristiche fondamentali (cenni The Security Architecture of the OSI Reference Model (ISO7498-2) [otORMI]):

• Riservatezza: lo scopo è mantenere private le informazioni personali e garantire che siano visibili e accessibili solo agli utenti legittimi, in modo da prevenire la divulgazione non autorizzata. Si ottiene principalmente criptando i dati.

• Integrità: è necessario garantire che i dati siano accurati e affidabili e che non

vengano modificati (operazioni di cancellazione, modifica) in modo errato. L'integrità può essere garantita da meccanismi di checksum, oltre che da meccanismi per il controllo dell'accesso ai dati.

• Disponibilità: le applicazioni e i dati devono essere completamente disponibili quando un utente legittimo ne ha bisogno. I sistemi devono risultare funzionanti con il livello di prestazioni prestabilito e nessuno dovrebbe essere in grado di minacciare il loro funzionamento regolare. Per garantire la disponibilità dei dati bisogna prevenire minacce intenzionali, ma anche fenomeni "naturali" come , cali di tensione, guasti all'hardware etc...

• L'autenticità consiste nel poter identificare in modo certo la provenienza di un messaggio. Dal momento che le informazioni sul mittente previste dai protocolli di comunicazione sono di solito facilmente falsificabili non è detto che sia un requisito facile da soddifare.

• Non ripudio: l'autore di una dichiarazione non può negare la paternità della stessa. Allo stesso modo, chi riceve il messaggio non può negare di averlo ricevuto. In genere, il metodo più utilizzato per la verifica dell'origine dei dati è il passaggio per i certificati digitali. L'integrità e l'autenticità dei dati sono prerequisiti per il Non ripudio.

Questi principi sono un punto di partenza per definire il concetto di rete sicura. Il passo successivo è quello di individuare le minacce più comuni per un host che vive connesso in rete, per poter poi strutturare meccanismi di difesa intelligenti che riconoscano i pattern di attacco e intraprendano delle misure per contrastarli. Quando si realizza un sistema di difesa si procede per strati, 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.

3.1.2 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. Gli attacchi più comuni da cui difendersi sono [sof] :

Virus, worm e altri agenti automatici, che 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 (che come un cavallo di Troia si presenta come un normale programma ma in realtà è stato realizzato per uno scopo malevolo). Il firewall e software antivirus possono essere due strumenti di difesa.

Scan su larga scala alla ricerca indiscriminata di sistemi vulnerabili, sfruttando buchi di sicurezza generalmente noti. Gli scan potrebbero rilevare vulnerabilità soprattutto nei servizi non aggiornati recentemente. Una vulnerabiltà potrebbe essere sfruttata per tentare un 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 accesso come root).

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. Il phishing consente di accedere alle credenziali di accesso, ai dati carte di credito etc... È un tipo di attacco molto popolare e diffuso, fa leva sull'incapacità di molti utenti di non saperlo riconoscere (spesso sembrano email urgenti e quindi si è incentivati ad aprirle, oppure gli url sono molto simili a quelli di siti ufficiali e quindi non è facile capire che sia falso).

Un exploit è un frammento di codice o software 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). Spesso vengono creati dei patch per contrastare gli exploit e correggere le vulnerabilità, quindi è sufficiente aggiornare i servzi installati ed il sistema operativo per limitare questo tipo di attacchi. Un esempio di payload maligno è una backdoor. Le backdoor "porte di servizio" rappresentano degli accessi privilegiati al sistema in grado di superare le procedure di sicurezza attivate. In questo modo autenti non autorizzati sono in grado di accedere a un pc all'insaputa dell'utente.

Un attacco DDoS (Distributed Denial of Service) consiste nell'inviare un flusso continuo di traffico falso ai servizi online, come i siti web, fino a quando non si sovraccaricano e crollano. Si distingue dall'attacco DoS (Denial of Service) perché il traffico è proveniente da più fonti e non da un singolo dispositivo. Si può descrivere come un bombardamento di traffico al servizio, il quale non è attrezzato per gestire l'enorme volume di trafficco e collassa. In questo modo gli utenti reali che hanno necessità di accedere ad una risorsa di rete 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, 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.

Questi sono solo alcuni degli attacchi più diffusi. L'obiettivo non è quello di offrire una panoramica completa, bensì uno scorcio che permetta di comprendere la necessità di introdurre degli strumenti che difendano la rete. Inoltre, mostrare degli esempi di minaccia concreti permette di motivare le scelte che vengono prese successivamente durante la realizzazione del sistema di difesa.

3.2 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 in rete. È funzionale a questo scopo introdurre il concetto di firewall.

3.2.1 Firewall

Il firewall è uno strumento di difesa che implementa una policy di sicurezza che permette di determinare quale traffico può transitare. 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. Utilizzando delle regole il firewall filtra il traffico ed esattamente come un muro blocca tutti i pacchetti indesiderati.

3.2.2 Firewall con iptables

Sebbene ci siano diversi modi per realizzare un firewall è oggetto di analisi solo il firewall realizzato con le regole iptables. Come prima cosa bisogna settare la policy, in ACCEPT (se un pacchetto non matcha nessuna regola viene accettato) oppure in DROP (se un pacchetto non matcha nessuna regola viene scartato). Se viene scelta una politica in ACCEPT le regole saranno in DROP, e viceversa. È sconsigliato utilizzare una politica in ACCEPT perchè il firewall potrebbe risultare più "lasco" rispetto ad uno che adotta una politica DROP, tuttavia con una politica DROP bisogna stare attenti a non dimenticarsi di far passare tutto il traffico autorizzato. Ad esempio se ci si connette in ssh ad un server e si modifica il firewall (che utilizza una politica in DROP) dimenticando la rule (in ACCEPT) relativa all'ssh, la connessione con il server viene

repentinamente interrotta. Dopo aver scelto la policy è necessario scrivere le regole specificando se si tratta regole di input-output, forwarding, prerouting o postrouting, il protocollo (TCP o UDP), l'interfaccia, la porta, lo stato in cui è ammessa la connessione.

• Una regola è di INPUT-OUTPUT se riguarda il traffico in ingresso o in uscita dal firewall.

• Una regola è di FORWARDING se i pacchetti hanno come origine internet/lan, come destinazione la lan/internet e sono solo di passaggio per il firewall, che agisce da router e instrada i pacchetti. I pacchetti viaggiano attraverso catene di forwarding, serve una chain per ogni "flusso" di traffico. Ad esempio serve una catena per i pacchetti che viaggiano dalla lan verso internet, e un'altra dalla rete esterna verso quella interna.

• Una regola di prerouting lavora sui pacchetti in entrata al sistema e applica delle regole ancora prima che i pacchetti vengano matchati con le regole di routing.

• Una regola di postrouting lavora sui pacchetti in uscita dal sistema e applica delle regole dopo che i pacchetti sono stati matchati con le regole di routing. Nelle regole di prerouting va specificato DNAT, mentre nelle regole di postrouting SNAT per permettere il mapping degli indirizzi ip.

Per quanto riguarda invece gli stati di una connessione sono ammessi i pacchetti in uno dei seguenti stati:

• 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 (che viene usata dal protocollo HTTPS):

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 input

di tipo ESTABLISHED sull'interfaccia ens160 del firewall, protocollo tcp, source port 443 (ossia permette di accettare risposte da parte di un server https)

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 50.50.50.3:443

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 internet (0/0) e come destinazione l'ip 10.222.111.2 (ip associato all'ens160) verranno forwardati sull'host che ha ip 50.50.50.3 sulla porta 443.

Non c'è in quest'analisi la pretesa di mostrare nei particolari il funzionamento delle regole iptables, ma solo di fornirne un'idea generale. Se si fa rifermineto alla pila ISO-OSI queste permettono un buon grado di dettaglio dei livelli di rete (liv. 3) e di trasporto (liv. 4), ma non di distinguere i livelli applicativi (HTTP, FTP etc...). Il firewall dunque ha dei limiti. Si ambisce alla realizzazione di un apparato più intelligente, che sfrutti la conoscenza dei livelli superiori della pila ISO-OSI per analizzare i pacchetti ed eventualmente intraprendere delle azioni attive a difesa del sistema. Per questo il firewall si accosta spesso a sistemi di rilevazione e prevenzione delle intrusioni, i cosiddetti IDS e IPS.

3.2.3 IDS e IPS

IDS, Intrusion Detection System é 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 verificando 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 degli attacchi viene aggiornato. Quando gli IDS rilevano una violazione della sicurezza la notificano generando degli alert, ed è poi compito dell'amministratore (un utente che ha i privilegi di root) fermare l'attività sospetta. Un dispositivo che richiede meno user assistance poichè prende autonomamente delle decisioni per contrastare possibili attacchi è un IPS, un Intrusion Prevention System, che permette di isolare i pacchetti sospetti interrompendo connessioni e bloccando IP. Questo obiettivo può essere raggiunto riprogrammando la lista di controllo degli accessi del firewall in modo dinamico. Alcuni limiti degli IDS sono l'incapacità di analizzare pacchetti cifrati (non è possibile estrarre il contenuto del pacchetto senza conoscere la chiave) e la necessità (se si basabo su regole) di dover essere costantemente aggiornati per far fronte anche alle nuove minacce.

3.2.4 Cosa deve garantire un buon IDS e un buon IPS

Gli IDS e IPS sfruttano database, librerie e set di regole per decidere se un pacchetto vada bloccato oppure no. 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, con un dovuto intervento vi si può porre rimedio (ad esempio modificando le regole dell'IDS o del firewall). Al contrario tutta l'attività sospetta che supera controlli, analisi dei pacchetti e raggiunge l'ip destinatario indisturbato può generare una quantità di danni variabile, fino a compromettere l'integrità dell'intero sistema. I falsi negativi potrebbero introdurre virus, provocare malfunzionamenti, perdita di dati. Prevenire i casi di falsi negativi è il principale obiettivo di un sistema di difesa.

3.2.5 Tecniche usate

Così come il firewall implementa una policy di sicurezza, anche gli IDS e gli IPS utilizzano delle strategie, in questo caso strategie di detection. Si dividono in tre tipi:

• Signature-based detection, che si basa su signatures "firme". Una firma è un metodo fi rilevamento che si basa sulla presenza di segni o caratteristiche distintive in un exploit. Sono specificamente progettate per rilevare exploit noti poiché contengono segni distintivi, come stringhe , offset fissi, informazioni di debug, che possono essere correlati o meno allo sfruttamento effettivo di una vulnerabilità. 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 sono necessari veri exploit pubblici affinché questo tipo di rilevamento funzioni. Le aziende antivirus utilizzano questo tipo di tecnologia per proteggere i propri clienti dalle epidemie di virus. Come abbiamo visto nel corso degli anni, questo tipo di protezione ha solo capacità di protezione limitate poiché il virus ha già infettato qualcuno prima che sia possibile scrivere una firma.

• 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.

• 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 benigni solo le attività conformi agli standard dei protocolli internazionali.

[cyb].

3.2.6 Tipi di IDS

Un NIDS, Network Intrusion Detection System è un sistema signature-based che monitora il traffico da e verso la lan. I pacchetti che matchano le rules 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. NIDS verificha il payload dei pacchetti (e a volte anche gli header) per stabilire che non siano presenti tracce di codice maligno. Gli exploit, i virus, i worm e altro software maligno generano traffico di rete che se intercettato e studiato, può smascherare l'attacco prima che questo abbia generato danni. Ad esempio una specifica stringa in un payload potrebbe permettere di identificare un exploit, così come specifiche opzioni negli header del TCP/IP potrebbero rivelare un altro genere di attacco. Se viene rilevata l'attività maligna, qualcuno produrrà una "signature". Sulla base di queste signature, il motore di ricerca (uno dei componenti principali dell'IDS che si occupa di elaborare e analizzare i pacchetti applicando le regole) dell'IDS 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].

3.2.7 Tipi di IPS

Un NIPS, Network Intrusion Prevention System che come i NIDS utilizza una strategia di detection signature-based e si occupa di controllare il traffico da e verso la lan. A differenza del NIDS però intraprende azioni attive a discapito dei pacchetti sospetti. Oltre a segnalare attività anomale all'amministratore, un pacchetto può essere dropped, rejected e un NIPS potrebbe bloccare un IP mittente che non considera affidabile. Analogamente agli HIDS esistono gli HIPS, Host-based Intrusion Prevention System,

utilizzano una strategia di detection signature-based e proteggono attivamente il flusso di traffico da un singolo dispositivo endpoint bloccando il transito di pacchetti sospetti. 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, ossia potrebbe non essere grado di distinguere attività benigne da quelle maligne. Un sistema invece 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.

Capitolo 4

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 su 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.

4.1 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 utente che abbia i privilegi da amministratore oppure posso essere utilizzate direttamente le rules offerte dalla community. In entrambi i casi è possibile adattarle alle esigenze 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.

Le componenti principali di Snort sono citate brevemente:

• Decodificatore di pacchetti: raccoglie e prepara i pacchetti per la pre-elaborazione.

• Pre-processori: componente che organizza e modifica i pacchetti per il motore di ricerca.

• Motore di ricerca: Il componente principale che elabora e analizza i pacchetti applicando le regole.

• Logging e alert: Componente per la generazione di log e alert.

4.2 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.

4.2.1 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. A differenza di tcpdump 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 snifferà 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:

Esempio: snort -i ens160 -v.

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

4.2.2 Packet Logger Mode

Il passaggio successivo allo sniffing dei pacchetti è la loro registrazione. Snort in packet logger mode memorizza su disco il traffico rilevato. La registrazione è avviene utilizzando l'opzione -l, seguita dalla directory in cui si desidera archiviare i log. 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 filesystem) 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 si specifica il numero dei pacchetti da leggere.

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

Esempio: snort -v -i ens160 -l /var/log/snort

Questo comando permette di memorizzare i log nella directory 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.

4.2.3 IDS

Nelle modalità viste finora Snort non ha funzionato da IDS/IPS, mancava l'ingrediente fondamentale: le rules. Le rules 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 matcha con 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. Inoltre non dovrebbe essere utilizzata l'opzione -v soprattutto da Bash (Shell di CentOS), perchè considerata lenta e dispendiosa in termini di cicli di CPU, e potrebbe inoltre 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 testa il file di configurazione

• -N vengono disabilitati i log

• -D Snort 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

Esempio: snort -i ens160 -A full -c /etc/snort/snort.conf

– fast: vengono generati degli alert che mostrano ,timestamp, IP sorgente e destinazione, i numeri delle porte. I log vengono registrati in formato tcpdump.log

Esempio: snort -i ens160 -A fast -c /etc/snort/snort.conf

– console: gli alert vengono visualizzati su console. I log vengono registrati in formato tcpdump.log

Esempio: snort -i ens160 -A console -c /etc/snort/snort.conf

– cmg: venono mostrati sulla console dettagli dell'header e del payload del pacchetti in formato di testo e esadecimale.

Esempio: snort -i ens160 -A cmg -c /etc/snort/snort.conf

– none: per disabilitare gli alert. Questa opzione viene utilizzata se si vogliono disabilitare i normali log a vantaggio dei log unified2.

4.2.4 IPS

Oltre alle modalità viste finora Snort ne possiede una quarta, l'inline mode. In 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 alle iptables alcuni pacchetti verrano bloccati. In questa modalitá 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) e SDROP (scarta i pacchetti senza loggare nulla)

Un altro modo per realizzare intrusion prevention è utilizzando FwSnort, un software che permette di tradurre le rules in iptables. Il funzionamento è il seguente: FWSnort si occupa di creare uno script .sh che, se eseguito, provvede ad aggiungere le regole al firewall. Lo script .sh non modificando i file iptables permette facilmente di risalire alla configurazione precedente. 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 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.

4.3 Rules

4.3.1 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. Per fare in modo questo meccanismo funzioni autonomamente, nei sistemi operativi Unix/Unix-like, viene modificato il file crontab (con il comando crontab -e). Crontab é un file gestito da un demone che permette di pianificare l'esecuzione di script. In questo caso viene utilizzato per eseguire PulledPork ogni giorno. PulledPork consente il download automatizzato, l'analisi e la modifica di tutti i set di regole snort. Esegue anche verifiche checksum per tutti i download di regole principali. Genera automaticamente del 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 ('id della rule)

Esempio di come si può modificare il crontab: mm hh * * * /usr/local/bin/pulledpork.pl -c /etc/snort/pulledpork.conf -l

(mm indica il minuto, un numero compreso tra 00 e 59, ed hh indica l'ora, un numero compreso tra 01 e 24):

4.3.2 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". Le regole hanno un campo header e un campo option.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:

– -> si è interessati esclusivamente ai pacchetti che partono dal source ip e arrivano al destination ip

– <> 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.

Le regole sono tanto precise da permettere di filtrare il traffico proveniente da un singolo ip, da un range di ip e da più range di ip. Ad esempio la regola che inizia con alert icmp [192.168.1.0/24, 10.1.1.0/24] any <> .. . genererà un alert per gli ip appartenenti a queste due sottoreti. Usando il !10.1.1.0/24 si può anche escludere un ip o un range di ip dall'analisi dei pacchetti.

È possibile filtrare il traffico per porta o per range di porte. Ad esempio scrivendo 1:1024 nel campo source port è possibile creare degli alert per tutti i pacchetti in transito nell'intervallo di porte dalla 1 alla 1024.

Infine il campo options delle rule 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.

4.4 Alert e log

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

• 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.

• 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.

• 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.

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

Capitolo 5

Requisiti

5.1 Requisiti funzionali

Un'azienda dispone di un server su cui ha installato il virtualizzatore VMware che permette di creare/gestire macchine virtuali. L'azienda ha la necessità di esporre dei servizi su internet. Per farlo vuole utilizzare un router telelcom che ha due schede di rete, una con 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. Tramite virtualizzatore i titolari dell'azienda hanno già creato una prima virtual machine chiamata vm firewall su cui gira il sistema operativo CentOS7, una distro 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. Il router di default è quello telecom. Le risorse assegnate alla vm sono:

- 8 core
- 32 GB ram
- 250 GB hdd

Il router è configurato in modo da forwardare tutti i pacchetti che hanno come destinatario 79.61.138.204 sull'interfaccia ens160 della vm firewall. (Non potendo servirsi di servizi aggiornati?) 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 conservare private le informazioni personali degli utenti che utilizzano i ser-

vizi 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 (e disabilitare firewalld e SELinux utilizzati di default). 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 database server con MariaDB versione 5.5.68
- un file server con Samba versione 4.10.16

Tuttavia all'azienda **NON** interessano i dettagli relativi all'installazione dei servizi e **NON** è oggetto di interesse nemmeno la configurazione del firewall. Bensì richiede i seguenti requisiti funzionali, legati ancora una volta al tema della sicurezza:

- Un utente deve essere in grado di utilizzare i servizi installati sulla macchina in sicurezza. Tanto che 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.
- 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".
- 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.

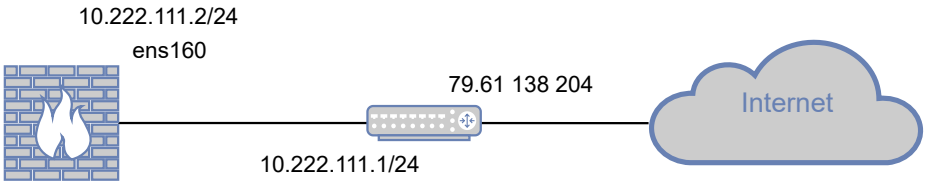

Figura 5.1: Rete nel suo stato primordiale. I componenti sono il router e la macchina virtuale vm firewall.

5.2 Requisiti non funzionali

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

• Modularità. Un sistema modulare prevede la suddivisione delle funzionalità da svolgere in aree distinte. Il vantaggio principale è riuscire a identificare una correzione o riuscire ad apportare un cambiamento anche rilevante, senza dover stravolgere la configurazione precedente, ma riuscendo a individuare l'area interessata e intervenendo solo su questa. Eventuali modifiche ricadranno esclusivamente su tale area e non sull'intero sistema.

• Scalabilità. Un sistema scalabile permette l'aggiunta di funzionalità e ulteriori requisiti a posteriori senza grandi stravolgimenti della 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 in rete, non è consigliabile che riveda l'intera struttura della rete, ma agisca in un contesto molto più ristretto.

• Minor numero possibile di falsi positivi e negativi. Un falso positivo si verifica quando il sistema classifica come maligna un'attività che invece dovrebbe essere autorizzata. Ad esempio se dei pacchetti vengono danneggiati lungo il tragitto potrebbero essere valutati erroneamente come maligni e scartati. Al contrario un falso negativo si verifica quando un'attività sospetta supera controlli, analisi dei pacchetti e raggiunge l'ip destinatario indisturbato. Un falso negativo può generare una quantità di danni variabile, potrebbe compromettere l'integrità dell'intero sistema. Per questo è importante mantenere il minor numero possibile di falsi positivi e negativi.

• Open source. L'azienda richiede che vengano utilizzati software open source. Non solo richiedono bassi costi iniziali, molto frequentemente offrono la possibilità di customizzazione (ossia di adeguare il software alle esigenze dell'utente sfruttando l'aiuto della community), e garantiscono maggiore stabilità e protezione, grazie alla loro popolarità/diffusione e al supporto di sviluppatori esperti che partecipano alla comunità. Inoltre un software di questo tipo libera l'azienda dalla dipendenza da un unico produttore, da un'unica architettura, da un unico protocollo, al contrario permette un alto livello di integrazione. Il codice è continuamente revisionato e corretto.

Capitolo 6

Realizzazione

6.1 Progetto

Dopo aver analizzato i requisiti presentati nel capitolo 3 si inizia a realizzare il progetto.

6.1.1 Configurazione della rete

Si procede come prima cosa con la suddivisione della rete per aree funzionali. La rete deve:

- gestire la sicurezza
- esporre i servizi su internet

Queste aree di lavoro vanno tenute separate per soddisfare due dei requisiti non funzionali indicati dall'azienda: la modularità e la scalabilità. Si sceglie quindi di introdurre una seconda macchina virtuale chiamata vm victim che ospiti web server, database 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 intuisce che la victim non possa avere un ip nell'intervallo 10.222.111.0/24. Se così fosse sarebbe in grado di comunicare direttamente con il router telecom, invece l'obiettivo é che la victim si trovi **a valle**

della vm firewall, in modo che questa possa filtrare il traffico. Si decide di procedere in questo modo:

- Viene aggiunta una seconda scheda di rete ens224 alla vm firewall, con ip 50.50.50.1
- La vm victim appartiene alla lan 50.50.50.0/24, una lan di server, ha un'unica interfaccia ens160 (oltre a quella di loopback) che corrisponde all'indirizzo 50.50.50.3. il router di default è 50.50.50.1.

Il flusso di traffico che arriva al router telecom viene rigirato sull'interfaccia esterna della vm firewall (ens160). Dopo essere stato filtrato/ analizzato la vm firewall svolge il ruolo di router per la lan 50.50.50.0/24, ossia smista i pacchetti tra i vari host seguendo le regole di forwarding. Questa configurazione è perfetta per soddisfare i requisiti di modularità e scalabilità e per permettere il corretto funzionamento della rete. Se la vm victim riscontrasse qualche problema e fosse necessario un intervento, non c'è modo che questo possa interferire o compromettere la sicurezza dell'intera rete, poiché sulla vm victim non risiede alcuno script/ file/ servizio per la gestione della sicurezza. 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! (256 indirizzi disponibili in /24 -1 prefisso -1 broadcast -1 indirizzo router - 1 indirizzo vm victim = 252). 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, macchina virtuale il cui unico scopo è quello di proteggere la lan dei server. Configurare manualmente il firewall risolve solo parte dei problemi legati alla sicurezza. Si limita ad applicare delle regole, senza nessun tipo di consapevolezza sul traffico che sta attraversando la rete in quel momento. È possibile aspirare ad un tipo di intelligenza che sia superiore a quella del firewall, che permetta di effettuare un controllo sul contenuto dei singoli pacchetti, in modo da monitorare continuamente il flusso di traffico e rilevare e prevenire eventuali comportamenti anomali?

Vengono elencati brevemente i vari componenti della rete:

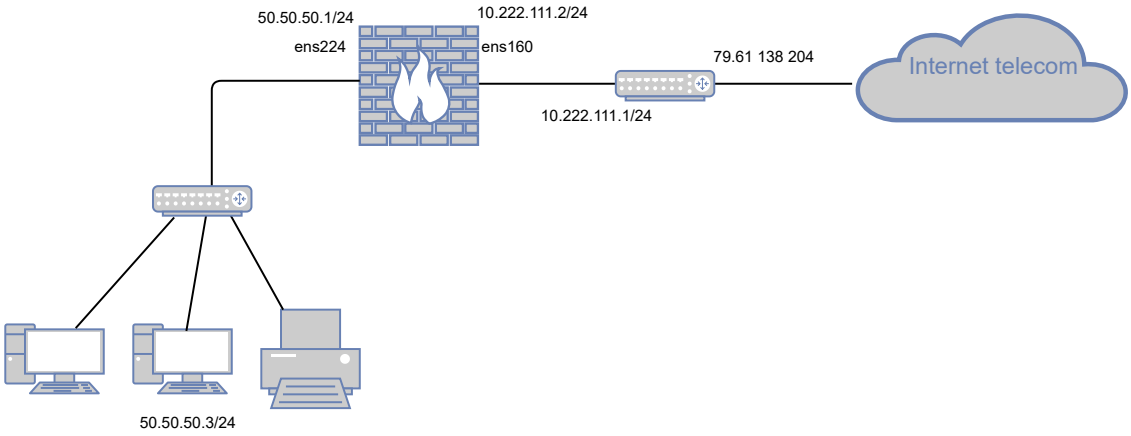

Figura 6.1: Progetto di realizzazione della rete che soddisfi i requisiti imposti nel capitolo 3.

- vm firewall che ospita il firewall,
  - IP 10.222.111.2, ens160, il router di default è 10.222.111.1
  - IP 50.50.50.1, ens224
- vm victim nella lan dei server,
  - IP 50.50.50.3, ens160, il router di default è 50.50.50.1
- Un altro componente della rete è uno switch virtuale

Viene riportato brevemente uno scenario parallelo di insuccesso, per rafforzare ancora di piú la scelta di progettare la rete nel modo illustrato dalla figura. Se la vm victim e la vm firewall fossero due host appartenenti alla stessa lan (e non rispettivamente un host e il router), la vm victim dovrebbe ospitare un firewall per filtrare il traffico proveniente dal router di default (il router telecom).

Capitolo 7

Installazione e configurazione di Snort

Dopo una prima analisi dei requisiti (capitolo 3) è emerso un problema di sicurezza che viene attenuato in parte dalla presenza di un firewall, ma questo non è sufficiente. Per aggiungere un altro livello di sicurezza si vuole fare affidamento ad un IDS/IPS. Questi sistemi 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 (capitolo 2). Si procede quindi con la realizzazione di un sistema IDS/IPS con Snort sulla vm firewall.

7.1 Prerequisiti e installazione

Prima di procedere è necessario installare delle librerie che permettono di catturare i pacchetti. Queste librerie rappresentano un prerequisito e vengono utilizzate per la cattura dei pacchetti che poi verranno analizzati da Snort.

```
yum install -y gcc flex bison zlib libpcap pcre libdnet tcpdump
yum install -y libnghttp2
```

Dopo questi passaggi preliminari si può passare all'effettiva installazione con yum:

```
yum install snort.x86_64
```

7.2 Configurazione

Per adeguare Snort alle esigenze della rete bisogna modificare il file di configurzione, importare le regole, gestire i log. Inizialmente è necessario aggiornare le le librerie con il comando:

```
ldconfig
```

Snort su CentOS viene installato nella directory /usr/local/bin/snort, è buona norma creare un link simbolico a /usr/sbin/snort.

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

Per eseguire Snort su CentOS in modo sicuro senza accesso come root, bisogna creare un nuovo utente senza privilegi e un nuovo gruppo di utenti.

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

Le principali directory che sono state create automaticamente usando il comando yum 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 ma questo è quello di default

Bisogna assicurarsi che lo user snort e il gruppo snort abbiano i permessi di lettura/ scrittura/ esecuzione

```
chmod -R 5775 /etc/snort
chmod -R 5775 /var/log/snort
chmod -R 5775 /usr/local/lib/snort_dynamicrules
chown -R snort:snort /etc/snort
chown -R snort:snort /var/log/snort
```

```
chown -R snort:snort /usr/local/lib/snort_dynamicrules
```

Viene creata uno file per le local.rules: `vi /etc/snort/rules/local.rules`

Ora si arriva al cuore della configurazione. Bisogna modificare il file snort.conf in modo da soddisfare i requisiti della rete. Per farlo viene eseguito il comando `vi /etc/snort/snort.conf` Vengono create due variabili, HOME-NET ed EXTERNAL-NET per distinguere gli indirizzi che appartengono alla lan da tutto il resto (altre lan/ internet). La HOME NET corrisponde all'ip dell'interfaccia esterna della vm firewall, perché è lì che Snort sniffa i pacchetti per analizzarli. L'EXTERNAL-NET invece corrisponde a tutti gli altri ip.

```
ipvar HOME_NET 10.222.111.2/24
ipvar EXTERNAL_NET !$HOME_NET
```

In ogni file di configurazione va specificato il path di dove si trovano le regole.

```
var RULE_PATH /etc/snort/rules
var SO_RULE_PATH /etc/snort/so_rules
var PREPROC_RULE_PATH /etc/snort/preproc_rules
```

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

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

Per includere le local.rules:

```
include RULE_PATH/local.rules
```

Manca un ultimo passaggio: includere le community rules. Per farlo è prima necessario scaricarle con una wget.

7.3 Community Rules

Per scaricare le community rules:

```
wget https://www.snort.org/rules/community -O ~/community.tar.gz
```

Per estrarre le regole e copiarle nella cartella di configurazione:

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

Per commentare tutte le regole diverse da quelle della community

```
sed -i 's/include RULE_PATH/#include RULE_PATH/' /etc/snort/snort.conf
```

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

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

7.4 Pulled Pork

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

```
yum -y install pulledpork
```

Lo script Perl si trova nella directory /usr/local/bin/ e bisogna verfificare che sia eseguibile:

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

È possibile verficare il corretto funzionamento eseguendo PulledPork:

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

Il file di configurazione si trova invece in /etc/snort e si chiama pulledpork.conf. Bisogna effettuare alcune modifiche, quindi si legge il file con il comando `vi /etc/snort/pulledpork.con`

Viene specificato il path delle regole processate da snort.

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

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

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

I file sid-msg.map si trovano qui:

```
sid_msg=/usr/local/etc/snort/sid-msg.map
```

viene specificato anche il path del file di configurazione

```
config_path=/usr/local/etc/snort/snort.conf
```

Pulled pork aggiorna una blocklist di ip pubblici bloccati dalla comunità

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

7.5 Il demone snortd

Per far si che Snort lavori in background viene attivato il demone snortd. I demoni sono dei programmi eseguiti in background, cioè senza il controllo diretto di un utente. Un sistema di difesa ha l'esigenza di catturare pacchetti e analizzare il traffico durante tutto l'arco di vita della macchina (esposta in rete). 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
```

7.6 FWSnort

FWsnort è uno script perl che traduce le regole di Snort in iptables. Alcune regole non hanno una traduzione diretta, tuttavia circa il 65 per cento delle rules 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.

```
cd /usr/local/src
wget http://www.cipherdyne.org/fwsnort/download/fwsnort-1.0.tar.bz2
wget http://www.cipherdyne.org/fwsnort/download/fwsnort- 1.0.tar.bz2.md5
wget http://www.cipherdyne.org/fwsnort/download/fwsnort- 1.0.tar.bz2.asc

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

Per far partire

```
fwsnort
```

FWSnort se installato su un sistema operativo che offre il supporto per il string match module nel kernel, procederà per la traduzione delle regole in iptables, mostrando da cli quante regole sono state convertite correttamente.

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

FWSnort gestisce tutto il traffico che attraversa la macchina locale ed è diretto verso/proviene dalla lan. Nel file di configurazione vengono definite delle variabili che corrispondo a ip/porte e vengono utilizzate nelle rules.

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

HTTP_SERVERS            $HOME_NET;
```

```
SMTP_SERVERS            $HOME_NET;
DNS_SERVERS             $HOME_NET;
SQL_SERVERS             $HOME_NET;
TELNET_SERVERS          $HOME_NET;

SSH_PORTS               [64022,65022];
HTTP_PORTS              80;
SHELLCODE_PORTS         !80;
ORACLE_PORTS            1521;
```

Un esempio di community rules:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"SERVER-APACHE Apache To
```

Capitolo 8

Verifiche funzionali

È necessario testare il comportamento di Snort nelle sue varie modalità. 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 rispondere alle domande "Snort ha rilevato il pacchetto X?" "Ha generato un alert?" "Ha agito attivamente per prevenire l'attacco Y?". 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.

8.1 Test 1: il generatore di traffico

Scapy

Realizzazione

Il test coinvolge tre macchine:

- Un client in una rete domestica (chiamato per comodità host A) che interagisce con il server remoto e genera traffico. L'IP pubblico del router é noto ed é 87.6.233.6 !!!!
- 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, un programma Python che consente di generare traffico. L'obiettivo è inviare pacchetti sulle porte d'interesse del server. È bene ricordare i servizi installati sulla vm victim e le relative porte:

HTTP	80	TCP
HTTPS	443	TCP
TOMCAT	8080	TCP
TOMCAT	8084	TCP
SAMBA	137	TCP
SAMBA	445	TCP
SAMBA	138	UDP
SAMBA	139	UDP
(a)	(b)	(C)

Figura 8.1: (a) Nome del servizio. (b) Numero di porta. (b)Protocollo.

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.

Viene acceduto il file relativo alle local.rules:

```
vi /etc/snort/rules/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:10000003; rev:001;)
alert TCP any any -> $HOME_NET 443 (msg:"Https detected"; sid:10000004; rev:001;)
alert TCP any any -> $HOME_NET 8080 (msg:"Tomcat detected"; sid:10000005; rev:001;)
alert TCP any any -> $HOME_NET 8443 (msg:"Tomcat detected"; sid:10000006; rev:001;)
alert TCP any any -> $HOME_NET 137 (msg:"SMB detected"; sid:10000007; rev:001;)
alert TCP any any -> $HOME_NET 445 (msg:SMB detected"; sid:10000010; rev:001;)
alert UDP any any -> $HOME_NET 138 (msg:"SMB detected"; sid:10000008; rev:001;)
alert UDP any any -> $HOME_NET 139 (msg:"SMB detected"; sid:10000008; rev:001;)
```

La sintassi viene ricordata brevemente:

Snort creerà un alert per i pacchetti che hanno source-ip: any, source-port: any, destination-ip: HOME-NET (variabile che corrisponde all'IP 10.222.111.2), destination-port: any, 80, 443, a seconda dell'esigenza. Nella seconda parte della regola (che corrisponde al campo option) si trovano msg: messaggio che compare a schermo o nei log, sid: l'id della rule, rev: informazioni sulla modifica o l'update della regola. In questo vale 001 perché regola non è mai stata revisionata.

Bisogna poi verificare che le local rules siano state importate nel file di configurazione, ossia che in `/etc/snort/snort.conf` sia presente la seguente riga:

```
include $RULE_PATH/local.rules
```

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

Quindi si entra nella directory:

```
cd /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 .
```

Da questo momento in poi qualsiasi pacchetto che arriva sull'interfaccia ens160 e che matcha le local.rules comparirà sulla console. È arrivato il momento di utilizzare il software Scapy sulla macchina host A, che generi traffico sulle porte interessate. La shell interattiva di Scapy viene eseguita all'interno di una sessione del terminale. Per avviare ??? la shell bisogna digitare:

```
scapy -H
```

Si decide di aprire tre shell distinte: 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 brevemente:

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 compaiono i seguenti avvisi:

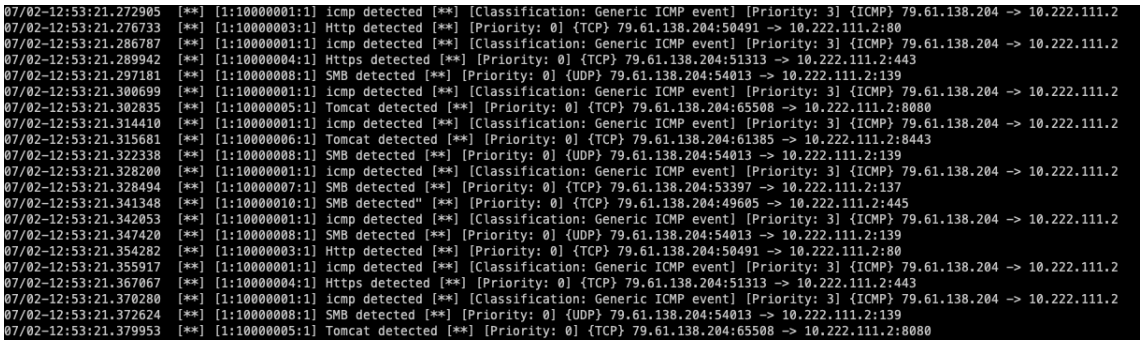

Figura 8.2: visualizzazione sulla console degli alert di Snort

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 delle statistiche:

```
===============================================================================
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
===============================================================================
```

Figura 8.3: tempo di esecuzione di Snort per l'elaborazione dei pacchetti

La figura sopra riportata mostra la durata di esecuzione dell'istanza di Snort. In questo caso il ciclo di vita è stato di 25 minuti e 49 secondi. Sono stati ricevuti 865 pacchetti al secondo.

```
===============================================================================
Packet I/O Totals:
   Received:      1368418
   Analyzed:      1341042 ( 97.999%)
    Dropped:        27370 (  1.961%)
   Filtered:            0 (  0.000%)
Outstanding:        27376 (  2.001%)
   Injected:            0
===============================================================================
```

Figura 8.4: pacchetti ricevuti da Snort sulla ens160

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.

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

Figura 8.5: pacchetti che hanno generato log e avvisi

Infine Snort mostra quanti pacchetti abbiano generati log e alert: in questo caso 276257. 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. Come mai? Si spiega semplicemente: questo 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. Come mai Snort ne ha loggati solo 276257? Mancano 71165 pacchetti all'appello. La spiegazione più probabile è che una parte sia rimasta nel buffer e sia stata droppata, mentre una parte non sia stata registrata a causa di dei vincoli del mondo reale, ad esempio limiti sul tempo di elaborazione, o limiti sulla memoria.

tcpdump.log

In questo caso è possibile leggere il file tcpdump.log.1656762820 con il comando riportato a seguire. Con l'opzione -n 10 vengono stampati a schermo gli ultimi dieci pacchetti.

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

Qui ne viene riportato uno solo:

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
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
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

L'amministratore potrebbe ricavare diverse informazioni come:

• time to live TTL:46

• type of service (TOS) specifica la priorità di un datagram 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.

merged.log

È anche possibile leggere il file merged.log.

```
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
```

8.2 Test 2: scan delle porte con Nmap

Il test coinvolge due macchine:

- Un host in una rete domestica, l'ip pubblico del router é noto ed é 87.6.233.6
- La vm firewall che ha ip 10.222.111.2 La prima fa uno scan delle porte, la seconda lo rileva utilizzando Snort.

8.3 Test 3: simulazione attacco DoS

Il test coinvolge due macchine:

- Un host in una rete domestica, l'ip pubblico del router é noto ed é 87.6.233.6
- La vm firewall che ha ip 10.222.111.

```
alert tcp any any -> $HOME_NET any (msg:"TCP SYN flood"; flags:!A;
```

```
> sudo nmap 79.61.138.204
Password:
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-02 18:32 CEST
Nmap scan report for 204.138.61.79.in-addr.arpa (79.61.138.204)
Host is up (0.011s latency).
Not shown: 996 filtered tcp ports (no-response)
PORT     STATE  SERVICE
80/tcp   open   http
443/tcp  open   https
3000/tcp open   ppp
5001/tcp closed commplex-link

Nmap done: 1 IP address (1 host up) scanned in 6.05 seconds
```

Figura 8.6: scan tcp con nmap

```
          Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
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
07/02-18:32:30.004174  [**] [1:10000004:1] Https scan detected [**] [Priority: 0] {TCP} 87.6.233.6:64366 -> 10.222.111.2:443
07/02-18:32:30.013831  [**] [1:10000010:1] SMB scan detected" [**] [Priority: 0] {TCP} 87.6.233.6:64622 -> 10.222.111.2:445
07/02-18:32:30.014234  [**] [1:10000005:1] Tomcat scan detected [**] [Priority: 0] {TCP} 87.6.233.6:64622 -> 10.222.111.2:8080
07/02-18:32:31.123379  [**] [1:10000005:1] Tomcat scan detected [**] [Priority: 0] {TCP} 87.6.233.6:64624 -> 10.222.111.2:8080
07/02-18:32:31.124796  [**] [1:10000010:1] SMB scan detected" [**] [Priority: 0] {TCP} 87.6.233.6:64624 -> 10.222.111.2:445
07/02-18:32:31.233973  [**] [1:10000004:1] Https scan detected [**] [Priority: 0] {TCP} 87.6.233.6:64622 -> 10.222.111.2:443
07/02-18:32:31.234010  [**] [1:10000003:1] Http scan detected [**] [Priority: 0] {TCP} 87.6.233.6:64622 -> 10.222.111.2:80
07/02-18:32:31.244579  [**] [1:10000003:1] Http scan detected [**] [Priority: 0] {TCP} 87.6.233.6:64622 -> 10.222.111.2:80
07/02-18:32:31.245754  [**] [1:10000004:1] Https scan detected [**] [Priority: 0] {TCP} 87.6.233.6:64622 -> 10.222.111.2:443
07/02-18:32:32.504396  [**] [1:10000003:1] Http scan detected [**] [Priority: 0] {TCP} 87.6.233.6:64627 -> 10.222.111.2:80
07/02-18:32:32.515730  [**] [1:10000003:1] Http scan detected [**] [Priority: 0] {TCP} 87.6.233.6:64627 -> 10.222.111.2:80
07/02-18:32:32.515740  [**] [1:100000014:1] ppp scan detected [**] [Priority: 0] {TCP} 87.6.233.6:64622 -> 10.222.111.2:3000
07/02-18:32:32.526554  [**] [1:100000014:1] ppp scan detected [**] [Priority: 0] {TCP} 87.6.233.6:64622 -> 10.222.111.2:3000
07/02-18:33:19.232869  [**] [1:10000008:1] SMB scan detected [**] [Priority: 0] {UDP} 10.222.111.254:138 -> 10.222.111.255:138
07/02-18:33:19.233014  [**] [1:10000008:1] SMB scan detected [**] [Priority: 0] {UDP} 10.222.111.254:138 -> 10.222.111.255:138
```

Figura 8.7: rilevazione del primo scan da parte di Snort

```
~
> sudo nmap -sU 79.61.138.204
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-02 18:33 CEST
Nmap scan report for 204.138.61.79.in-addr.arpa (79.61.138.204)
Host is up (0.021s latency).
Not shown: 999 open|filtered udp ports (no-response)
PORT   STATE  SERVICE
53/udp closed domain

Nmap done: 1 IP address (1 host up) scanned in 7.04 seconds
~
```

Figura 8.8: scan udp con nmap

`flow: stateless; detection_filter: track by_dst, count 70, seconds 10; sid:2000003;)`

Per ripristinare le regole iptables del firewall:

```
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
07/02-18:33:44.150664  [**] [1:10000011:1] DNS scan detected [**] [Priority: 0] {UDP} 87.6.233.6:37169 -> 10.222.111.2:53
07/02-18:33:45.414388  [**] [1:10000011:1] DNS scan detected [**] [Priority: 0] {UDP} 87.6.233.6:37174 -> 10.222.111.2:53
07/02-18:33:45.414490  [**] [1:10000011:1] DNS scan detected [**] [Priority: 0] {UDP} 87.6.233.6:37174 -> 10.222.111.2:53
07/02-18:33:46.416682  [**] [1:10000009:1] SMB scan detected [**] [Priority: 0] {UDP} 87.6.233.6:37169 -> 10.222.111.2:139
07/02-18:33:48.089410  [**] [1:10000011:1] DNS scan detected [**] [Priority: 0] {UDP} 87.6.233.6:37178 -> 10.222.111.2:53
07/02-18:33:48.089569  [**] [1:10000011:1] DNS scan detected [**] [Priority: 0] {UDP} 87.6.233.6:37178 -> 10.222.111.2:53
```

Figura 8.9: rilevazione del secondo scan da parte di Snort

```
>>> send(IP(dst="79.61.138.204")/TCP(dport=[445]),loop=1,inter=0.01)
.....................................................................
Sent 175 packets.
```

Figura 8.10: flusso di pacchetti TCP sulla porta 443

```
Commencing packet processing (pid=13661)
07/02-19:55:07.920807  [**] [1:2000003:0] TCP SYN flood [**] [Priority: 0] {TCP} 79.61.138.204:53015 -> 10.222.111.2:445
07/02-19:55:07.933789  [**] [1:2000003:0] TCP SYN flood [**] [Priority: 0] {TCP} 79.61.138.204:53015 -> 10.222.111.2:445
07/02-19:55:07.946807  [**] [1:2000003:0] TCP SYN flood [**] [Priority: 0] {TCP} 79.61.138.204:53015 -> 10.222.111.2:445
07/02-19:55:07.959732  [**] [1:2000003:0] TCP SYN flood [**] [Priority: 0] {TCP} 79.61.138.204:53015 -> 10.222.111.2:445
07/02-19:55:07.972701  [**] [1:2000003:0] TCP SYN flood [**] [Priority: 0] {TCP} 79.61.138.204:53015 -> 10.222.111.2:445
07/02-19:55:07.986051  [**] [1:2000003:0] TCP SYN flood [**] [Priority: 0] {TCP} 79.61.138.204:53015 -> 10.222.111.2:445
07/02-19:55:07.999011  [**] [1:2000003:0] TCP SYN flood [**] [Priority: 0] {TCP} 79.61.138.204:53015 -> 10.222.111.2:445
07/02-19:55:08.012002  [**] [1:2000003:0] TCP SYN flood [**] [Priority: 0] {TCP} 79.61.138.204:53015 -> 10.222.111.2:445
07/02-19:55:08.024964  [**] [1:2000003:0] TCP SYN flood [**] [Priority: 0] {TCP} 79.61.138.204:53015 -> 10.222.111.2:445
07/02-19:55:08.037957  [**] [1:2000003:0] TCP SYN flood [**] [Priority: 0] {TCP} 79.61.138.204:53015 -> 10.222.111.2:445
07/02-19:55:08.050853  [**] [1:2000003:0] TCP SYN flood [**] [Priority: 0] {TCP} 79.61.138.204:53015 -> 10.222.111.2:445
07/02-19:55:08.063839  [**] [1:2000003:0] TCP SYN flood [**] [Priority: 0] {TCP} 79.61.138.204:53015 -> 10.222.111.2:445
07/02-19:55:08.076848  [**] [1:2000003:0] TCP SYN flood [**] [Priority: 0] {TCP} 79.61.138.204:53015 -> 10.222.111.2:445
07/02-19:55:08.089758  [**] [1:2000003:0] TCP SYN flood [**] [Priority: 0] {TCP} 79.61.138.204:53015 -> 10.222.111.2:445
07/02-19:55:08.102775  [**] [1:2000003:0] TCP SYN flood [**] [Priority: 0] {TCP} 79.61.138.204:53015 -> 10.222.111.2:445
07/02-19:55:08.115704  [**] [1:2000003:0] TCP SYN flood [**] [Priority: 0] {TCP} 79.61.138.204:53015 -> 10.222.111.2:445
07/02-19:55:08.128677  [**] [1:2000003:0] TCP SYN flood [**] [Priority: 0] {TCP} 79.61.138.204:53015 -> 10.222.111.2:445
07/02-19:55:08.142038  [**] [1:2000003:0] TCP SYN flood [**] [Priority: 0] {TCP} 79.61.138.204:53015 -> 10.222.111.2:445
```

Figura 8.11: rilevazione del TCP FLOOD da parte di Snort

```
service iptables restart
```

8.3.1 Snort, tcpdump e wireshark a confronto

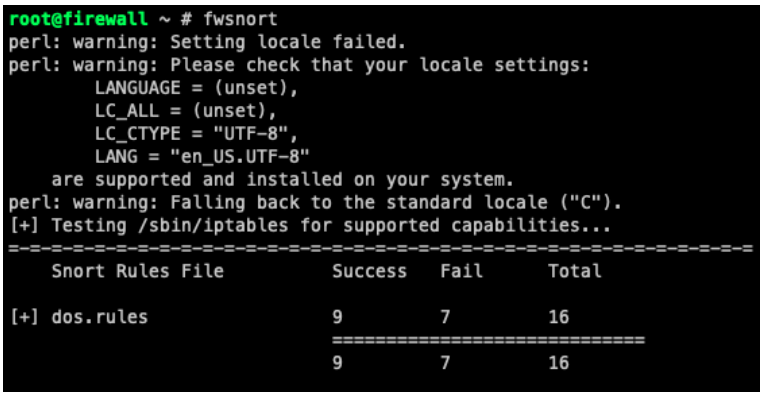

Figura 8.12: fwsnort

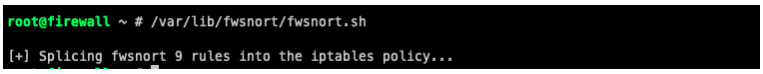

Figura 8.13: fwsnort

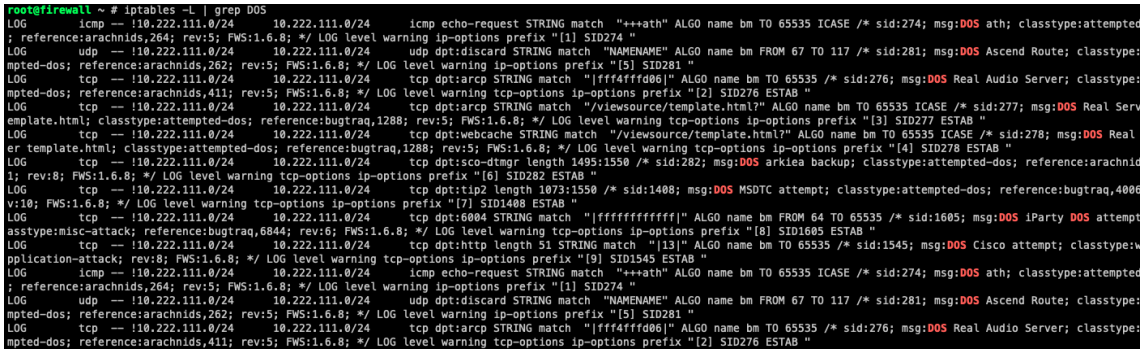

Figura 8.14: fwsnort

Conclusioni e sviluppi futuri

La tesi è finita

Bibliografia

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

[otORMI] The Security Architecture of the OSI Reference Model (ISO7498-2). https://www.sciencedirect.com/topics/computer-science/security-architecture.

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

[zer] https://www.zerounoweb.it/cloud-computing/intrusion-detection-ecco-come-funzionano-gli-ids/.