UNIVERSITA' POLITECNICA DELLE MARCHE

FACOLTA' DI INGEGNERIA

Corso di Laurea Magistrale in

Ingegneria Informatica e dell'Automazione

Implementazione infrastruttura sicura

Docente:

Luca Spalazzi

Studenti:

Luca Liberatore

Ciro Maccarone

Anno Accademico 2022-2023

Sommario

Introduzione del progetto

Il progetto prevede lo sviluppo di una rete interna sicura in grado di rifiutare le connessioni non accettate e di allertare il sistema del tentativo. Inoltre, il sistema prevede dei meccanismi di sicurezza per l'integrità del server interno, in grado di segnalare eventuali violazioni di policy prestabilite per l'uso dei dati della macchina.

Il sistema prevede:

- Una macchina esterna (Kali), che deve fungere da attaccante e da utenti appartenenti alla rete esterna. Vedremo come nel capitolo di simulazione cercherà di eseguire diversi tipi di connessioni con le macchine della rete interna e ne osserveremo i risultati.
- Il Router, che deve inoltrare il traffico per lui passante. Fungerà da filtro delle connessioni e ha il compito di monitorare il traffico passante, notificando specifici tentativi di connessione con determinati protocolli.
- Il Bastion Host, che ha il compito di autorizzare o meno le connessioni provenienti dall'esterno verso il server della rete interna. Il suo compito è quello di server proxy.
- Risorsa, è la macchina contenente gli asset da proteggere, vi è installata una applicazione IDS che monitora i dati contenenti all'interno, dando un allarme se alcune policy di integrità dei dati sono violate.

Descrizione delle macchine virtuali

La tipologia di architettura di rete utilizzata per il progetto prevede l'utilizzo di tre macchine Debian, che corrispondono rispettivamente agli asset dell'organizzazione: Router, Bastion Host e Risorsa. L'ambiente in cui sono state implementate le macchine è di tipo virtuale, in modo particolare si è fatto uso dell'hypervisor Oracle VirtualBox al fine di raggiungere gli obiettivi di progetto. Il mondo esterno invece, cioè quello al di fuori della rete aziendale, è costituito unicamente dalla macchina virtuale Kali Linux, la quale rivestirà il ruolo sia di cloud esterno alla rete interna sia di attaccante, nel simulare vari tipi di attacchi e nel comprovare la sicurezza del sistema implementato.

Per motivi computazionali, come già detto, si è scelto di far eseguire le macchine sul sistema operativo Debian, dando ad ognuna delle tre macchine dell'organizzazione 1 Gigabyte di RAM, ovvero il minimo necessario al funzionamento basilare dei software che si andranno ad utilizzare su di esse. La memoria di massa associata ad ogni macchina è di circa 5 GB. Nel caso della macchina esterna, Kali Linux, per motivi di performance sono stati adottati 2 GB di RAM ed una memoria nel disco fisso molto maggiore.

Ogni macchina, escluso il Router, ha una sola scheda di rete abilitata; mentre il Router ha tre schede di rete attive, ognuna delle quali permette di interfacciarsi con i restanti tre elaboratori virtuali e con le loro rispettive interfacce di rete. Al fine di permettere la comunicazione tra le diverse macchine si è optato per impostare le schede di rete in modalità rete interna, secondo le caratteristiche della tabella riportata sotto.

Mode	VM→Host	VM←Host	VM1↔VM2	VM→Net/LAN	VM←Net/LAN
Host-only	+	+	+	–	–
Internal	–	–	+	–	–
Bridged	+	+	+	+	+
NAT	+	Port forward	–	+	Port forward
NATservice	+	Port forward	+	+	Port forward

In questo modo, anche per semplificare le comunicazioni tra le macchine, è garantita la visibilità solo tra le macchine nella stessa rete interna. Nel caso specifico del progetto si è deciso di impostare due reti interne: la prima (*intnet*) che comprende il Bastion Host, la Risorsa e due delle tre interfacce di rete del Router, cioè quelle che si rivolgono rispettivamente a quest'ultime due macchine che sono dentro la rete aziendale; la seconda rete interna (*intnet1*) va a descrivere il mondo esterno alla rete dell'organizzazione. Infatti, in questa rete vi è l'unica interfaccia di rete di Kali e l'unica scheda di rete del Router, che comunica con il mondo esterno. In altri termini, l'idea

che sta alla base di questa configurazione è che ogni macchina, escluso il Router, ha una interfaccia che, nei prossimi capitoli, verrà predisposta per comunicare con il Router stesso; quest'ultimo invece ha a disposizione, come già detto, tre interfacce differenti in modo da poter fungere da tramite tra le diverse macchine.

Per quanto concerne l'installazione delle varie macchine, si sono seguiti i passaggi guidati del sistema operativo e, per semplicità, gli utenti presenti per ogni macchina aziendale sono identificati con il nome della macchina stessa: Router, Bastion Host e Risorsa. La scelta di implementare la rete aziendale con il sistema operativo Debian è dovuta al fatto che è una distribuzione Linux libera, stabile e sicura, che viene frequentemente aggiornata e non presenta elevati requisiti hardware. D'altro canto, Kali Linux è una delle distribuzioni maggiormente utilizzate nel mondo del penetration testing, proprio per questo motivo è stato scelto come strumento per testare la sicurezza dell'implementazione relativa l'infrastruttura aziendale.

Configurazione delle rotte delle macchine

Inizialmente data l'impostazione iniziale tutte le macchine potranno contattare le altre: stabiliamo dunque gli indirizzi IP per tutte le schede di rete delle varie macchine assicurandoci che ognuna di esse possa raggiungere le altre.

Il procedimento è analogo per tutte le macchine: entrando nel file di configurazione di rete delle singole macchine, andremo ad impostare gli IP ed il Gateway per ogni singola scheda di rete.

Entrando nel terminale delle macchine, impartiamo il seguente comando per entrare nel file di configurazione:

sudo nano /etc/network/interfaces

Avendo accesso al file, possiamo impostare gli indirizzi IP per le schede di rete, per conoscere le schede di rete della macchina basta inserire il comando:

ifconfig oppure **ip a**

conoscendo le schede di rete, è possibile ora assegnare gli indirizzi desiderati.

Per Kali scriviamo dentro il file di configurazione:

auto eth0

iface eth0 inet static

address 210.10.10.2

gateway 210.10.10.1

Abbiamo dunque impostato come IP statico della scheda di rete con indirizzo IP 210.10.10.10.2 e gateway 210.10.10.1.

Dopo la modifica salvare le impostazioni e riavviare i servizi di networking con il comando:

sudo systemctl restart networking

Seguendo gli stessi passi per le altre macchine, scriviamo per il Router le tre interfacce di rete:

auto enp0s3

auto iface enp0s3 inet static

address 210.10.10.1

auto enp0s8

auto iface enp0s8 inet static

address 211.11.11.1

auto enp0s9

auto iface enp0s9 inet static

address 192.168.1.1

Per il Bastion Host:

auto enp0s3

iface enp0s3 inet static

address 211.11.11.2

gateway 211.11.11.1

Per la Risorsa:

auto enp0s3

iface enp0s3 inet static

address 192.168.1.2

gateway 192.168.1.1

Per il router è necessario impostare su terminale un altro paio di comandi:

sudo sysctl –w net.ipv4.ip_forward=1

Il comando abiliterà il forwarding dei pacchetti di rete tra le macchine, per verificare sia effettivamente attivo usare il comando:

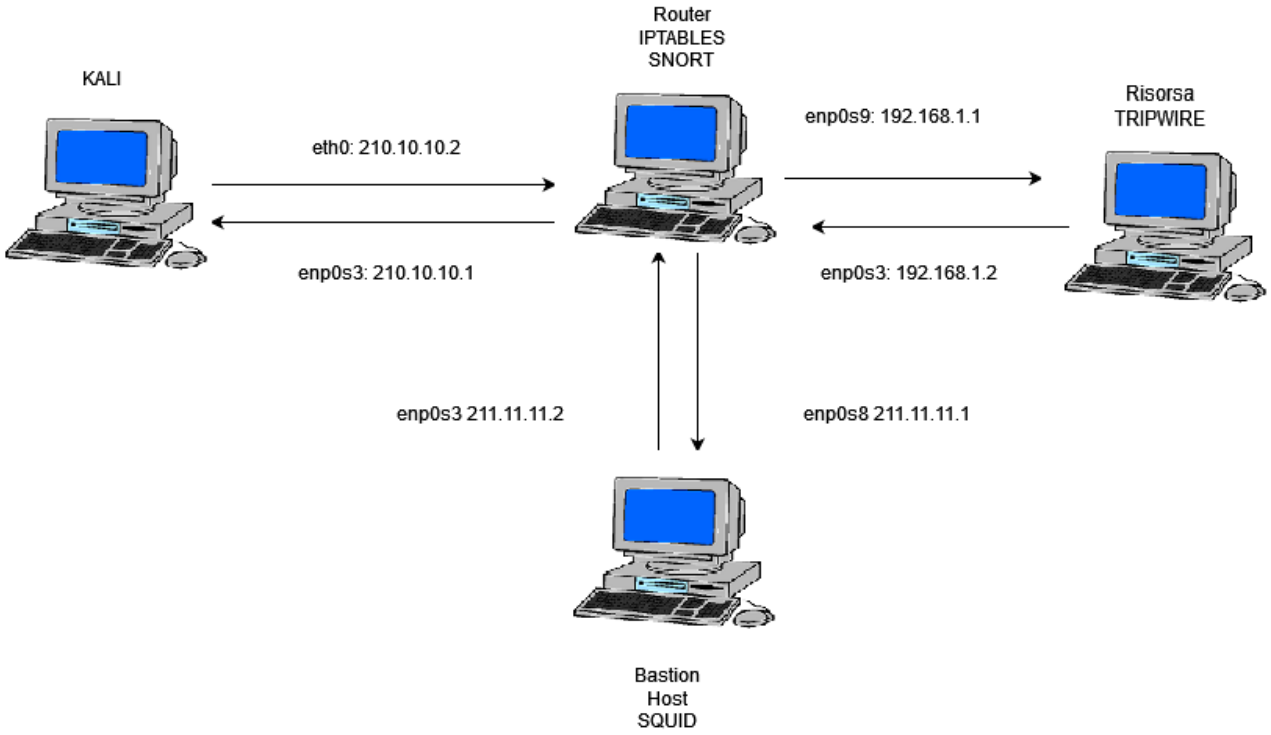
cat /proc/sys/net/ipv4/ip_forward

Impostare ora il traffico tra la macchina esterna kali e il server interno con le risorse:

ip route add 192.168.1.0/24 via 210.10.10.1 dev enp0s3

Importante notare che le macchine assegnano i nomi delle schede di rete di default, modificare i nomi del caso in esame enpos3, eth0 ecc. con i nomi presentati dalla propria macchina.

La situazione finale della configurazione è la seguente:



Implementazione dei sistemi di sicurezza

Nel paragrafo precedente si sono esposte le configurazioni di rete e quindi i vari IP delle macchine, ora ci si occupa della parte di sicurezza dell'infrastruttura. L'idea generale che sta dietro al progetto è quella di proteggere la macchina Risorsa, sia attraverso operazioni di blocco ed instradamento dei pacchetti attraverso il Router (Iptables), che filtraggio del traffico attraverso il proxy server del Bastion Host (Squid). Inoltre, come già anticipato, si farà uso di Intrusion Detection System per il monitoraggio sia del traffico che scorre tra le macchine (Snort), sia per rilevare eventuali modifiche a file ad alta priorità (Tripwire). Ulteriore prerogativa del sistema in questione è quella di permettere unicamente traffico diretto dalla risorsa al mondo esterno, identificato da Kali Linux in questo caso, e non viceversa. Con lo scopo di perseguire questo obiettivo, anche con l'ausilio del proxy Squid che filtra le richieste proveniente dalla risorsa, si è messo in ascolto su Kali un server Apache. La pagina web che compare, una volta inserito l'IP di Kali (210.10.10.2) all'interno della barra di ricerca di Mozilla nella macchina Risorsa, elenca i file presenti dentro la cartella di Kali Linux da cui Apache va a prendere le informazioni necessarie.

Iptables

IpTables è un programma preinstallato su Debian a livello utente che controlla il modulo di rete a livello kernel chiamato Netfilter, che rappresenta il filtro di pacchetti vero e proprio.
Ogni funzione fornita dall'architettura del Netfilter è presentata come una tabella. In particolar modo si hanno tre tabelle:

- filter : questa tabella ha il compito di filtrare i pacchetti
- nat: questa tabella ha il compito di tradurre gli indirizzi IP dei pacchetti
- mangle: questa tabella ha il compito di cambiare il contenuto del pacchetto

Il servizio di Iptables è stato usato nel Router per filtrare i pacchetti in transito per la rete interna. Lo scopo stabilito è quello di impedire la connessione diretta con le macchine di Router e di Bastion Host. Il transito consentito è solo per le connessioni TCP provenienti dalla Risorsa.

Le regole inserite per la catena di INPUT, e quindi per tutti i pacchetti di comunicazione destinati direttamente al Router, prevedono di impedire le richieste dirette dal mondo esterno verso il Router stesso. Inseriamo dunque 3 regole dove rifiutiamo la comunicazione con tutte e tre le interfacce di rete del router come destinatario:

iptables -I INPUT -s 210.10.10.2 -j DROP -d 210.10.10.1

iptables -I INPUT -s 210.10.10.2 -j DROP -d 211.11.11.1

iptables -I INPUT -s 210.10.10.2 -j DROP -d 192.168.1.1

I tre comandi indicano quindi di inserire tre regole, nel quale inseriamo in testa (-I) alla tabella di INPUT della catena FILTER delle regole nel quale la sorgente dei pacchetti (-s) è la macchina Kali esterna alla rete e il destinatario (indicato con -d) le tre interfacce del Router con i rispettivi IP. La regola impartita è quella di scartare tutti i pacchetti (azione DROP indicato con l'opzione -j).

Nella catena di FORWARD si inseriscono le regole per tutti i pacchetti in transito nel router e non destinati direttamente ad esso, stabiliamo quindi di scartare tutti i pacchetti destinati direttamente alla Risorsa e al Bastion Host. Inseriamo dunque queste regole:

iptables -A FORWARD -s 210.10.10.2 -j DROP -d 192.168.1.2

iptables -A FORWARD -s 210.10.10.2 -j DROP -d 211.11.11.2

con la -A indichiamo di inserire queste regole nel fondo della catena, in quanto sono le regole applicate quando nessuna altra regola è valida.

Le successive regole sono invece inserite in testa alla catena(opzione -I), e consentono di stabilire connessioni tcp stateful, nel quale si tiene conto anche del continuo cambio di porte per la connessione.

iptables -I FORWARD -s 211.11.11.2 -d 210.10.10.2 -p tcp -m conntrack --ctstate NEW,ESTABLISHED,RELATED -j ACCEPT

iptables -I FORWARD -s 210.10.10.2 -d 211.11.11.2 -p tcp -m conntrack --ctstate NEW,ESTABLISHED,RELATED -j ACCEPT

Lasciamo dunque transitare tutti i pacchetti destinati al Bastion Host (che è il proxy della Risorsa) e alla Risorsa stessa che usino il protocollo tcp (indicato con la lettera -p). L'opzione "-m" ci consente di impostare la condizione **conntrack** che consente appunto di identificare pacchetti di connessioni che si trovano in un particolare stato specificato dalla condizione **ctstate.** Le condizioni specificate della connessione sono:

- **NEW**, nel caso si voglia aprire una nuova sessione di comunicazione tra le macchine
- **ESTABLISHED**, per quei pacchetti che fanno parte di una sessione di comunicazione già aperta
- **RELATED**, si riferisce a tutti quei pacchetti che sono correlati alla connessione specificata dalla regola.

Stabilite le regole, per vedere il risultato finale scrivete il comando:

iptables -L –n –v

viene usato per elencare le regole del firewall iptables in modo dettagliato. Ecco cosa fanno i flag utilizzati:

- "-L" indica l'opzione per elencare le regole.
- "-n" indica l'opzione per visualizzare gli indirizzi IP e i numeri di porta numerici anziché fare la risoluzione inversa dei nomi di dominio.
- "-v" indica l'opzione per mostrare un output dettagliato, inclusi i conteggi dei pacchetti e dei byte per ogni regola.

La situazione finale delle regole impostate su iptables dovrebbe essere la seguente:

```
root@l0:/home/router# /sbin/iptables -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  210.10.10.2          192.168.1.1
DROP       all  --  210.10.10.2          211.11.11.1
DROP       all  --  210.10.10.2          210.10.10.1

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  211.11.11.2          210.10.10.2          ctstate NEW,RELATED,ESTABLISHED
ACCEPT     tcp  --  210.10.10.2          211.11.11.2          ctstate NEW,RELATED,ESTABLISHED
DROP       all  --  210.10.10.2          211.11.11.2
DROP       all  --  210.10.10.2          192.168.1.2

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Come da immagine si può vedere che la policy di default è quella di accettare tutte le connessioni, anche questo volendo è possibile modificarlo con il comando:

iptables -P <catena> <azione>

Dove:

- **<catena**> rappresenta il nome della catena di iptables, ad esempio INPUT, OUTPUT o FORWARD.
- **<azione>** indica l'azione da applicare alla catena, che può essere ACCEPT (accettare), DROP (scartare) o altre azioni come REJECT o LOG.

Snort

Snort è un Intrusion Detection System (IDS), ma che può fungere anche da Intrusion Prevention System (IPS), da analizzatore di pacchetti e può anche essere adoperato come strumento di analisi forense. Nel nostro caso abbiamo utilizzato l'applicativo unicamente per le sue funzionalità di base come IDS. Le modalità con le quali Snort è in grado di rilevare o meno un attacco si basano sul modello "misuse detection", ovvero riesce ad individuare un attacco se risulta simile ad un insieme di attacchi già noti che ha a disposizione. Entrando maggiormente nel dettaglio, Snort estrae delle features dai pacchetti che vede transitare e, se individua dei pattern simili a quelli di attacchi noti (*pattern matching*), genera un segnare di allarme. Il modo in cui Snort è in grado di rilevare attacchi noti è per mezzo delle cosiddette "rules", che se violate generano un segnale di allarme. La difficoltà di Snort, non risiedono tanto nello scrivere le singole regole, che ad ogni modo possono essere generate tramite dei tool online[1], ma nel configurarlo correttamente nel suo insieme in modo da ottenere davvero il comportamento desiderato per proteggere l'infrastruttura target.

Nel caso specifico di questo progetto si è scelto di installare Snort all'interno della macchina Router, in modo da poter analizzare tutto il traffico rivolto verso le due interfacce che comunicano rispettivamente con il Bastion Host e con la Risorsa. Snort è già presente nei repository di Debian, dunque è possibile installarlo sulla macchina virtuale lanciando il comando:

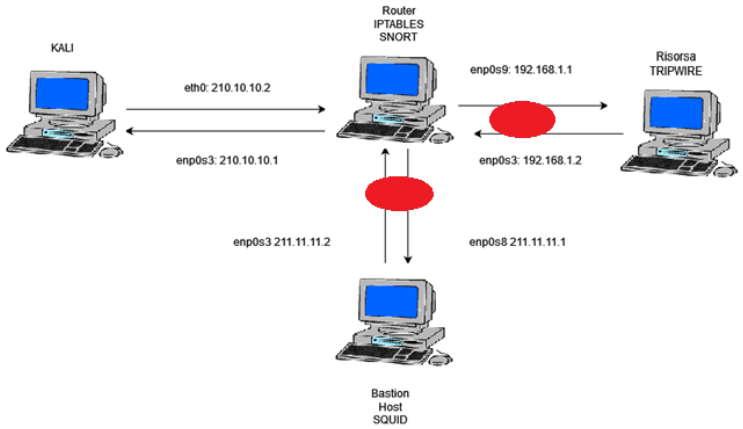
apt-get install Snort -y

Inoltre, si consiglia di attivare la modalità promiscua, all'interno di Oracle Virtual Box, per quelle schede di rete del Router rivolte verso le interfacce delle macchine da proteggere, cioè Bastion Host e Risorsa. Il file di configurazione di Snort è presente nel path "/etc/snort/snort.conf"; nel nostro caso abbiamo creato un file di copia sul quale andare a scrivere le nostre regole. Questa scelta è dovuta a due motivazioni principali: innanzitutto, Snort richiede una mano esperta e quindi si è voluto tenere un file di configurazione originale come backup; inoltre, dato che come già anticipato in precedenza, per motivi computazionali, le macchine Debian hanno una memoria RAM limitata ad 1 GB, utilizzare tutte le regole di default di Snort sarebbe troppo oneroso in termini di complessità computazionale per una macchina con così poche risorse hardware. Ad ogni modo, nel file di configurazione "snort2.conf" si è stabilito di far analizzare il traffico per le due sottoreti che comprendono il Bastion Host e la macchina Risorsa, questo è possibile farlo andando a settare la variabile HOME_NET:

[1] http://snorpy.cyb3rs3c.net/

ipvar HOME_NET [211.11.11.0/24,192.168.1.0/24]

I due punti di controllo su cui va ad agire Snort sono evidenziati in rosso in Figura.



Successivamente, come anticipato, si commentano tutte le regole di community già presenti nel file “snort2.conf”, essendo una copia della sua versione originale. Questa prima prova è stata svolta unicamente per l’interfaccia di rete del Router rivolta verso il Bastion Host, dunque, per controllare che il file di configurazione sia implementato correttamente si digita il comando:

snort -i enp0s8 –c –T /etc/snort/snort2.conf

Di seguito si dà una breve spiegazione dei flag e degli argomenti utilizzati:

- "-i enp0s8": specifica l'interfaccia di rete su cui Snort deve catturare il traffico, in questo caso la scheda di rete specificata è quella rivolta verso il Bastion Host;
- "-c": specifica il percorso del file di configurazione di Snort;
- "-T": esegue una prova di rilevamento simulata senza intraprendere alcuna azione reale.

Dunque, si sono tenute esclusivamente quelle regole esplicitamente scritte da noi e che Snort va a pescare dal file “/etc/snort/rules/local.rules”. In definitiva, si sono volute scrivere tre regole di esempio, la prima delle quali verrà testata nei paragrafi successivi. Le tre regole riguardano rispettivamente: tentativi di ping diretti verso le due reti da proteggere, tentativi di accesso SSH e richieste di connessione FTP. Le tre regole sono espresse nella seguente formulazione:

1. **alert icmp any any -> $HOME_NET any (msg:"ICMP Ping Rilevato"; sid:100; rev:1;)**
2. **alert tcp any any -> $HOME_NET 22 (msg:"Tentivo Autenticazione SSH"; sid:101; rev:1;)**
3. **alert tcp any any -> $HOME_NET 21 (msg:"Tentativo Connessione FTP"; sid:102; rev:1;)**

La prima regola esprime sostanzialmente il seguente concetto: la regola genera un allarme (**alert**) ad ogni richiesta di tipo **icmp** proveniente da qualsiasi ip (**any**) e da qualsiasi porta (**any**) esterna verso la nostra rete da proteggere (**$HOME_NET**) ed ogni sua porta interna (**any**). Come configurazione opzionale, l'alert genera un messaggio da mostrare (**ICMP Ping Rilevato**), a cui si associa un id (**100**) ed una versione (**1**).

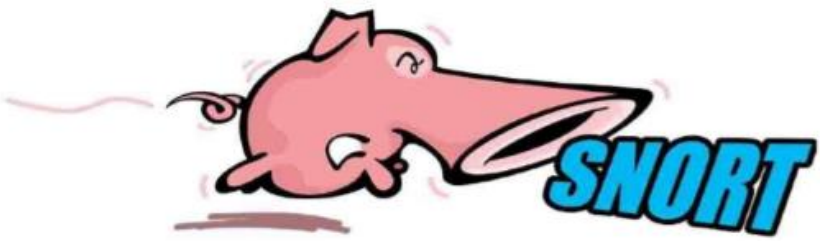
Seguendo lo stesso schema logico di interpretazione, si può affermare che le altre due regole fanno riferimento entrambe all'utilizzo del protocollo tcp e considerano qualsiasi tipo di porta e IP in ingresso; mentre, in questo caso, come porte di destinazione della rete interna, si fa riferimento a quelle associate ai tentativi di connessione SSH (**22**) ed FTP (**21**). Le due regole hanno quindi due identificativi differenti e due messaggi di allarme che ne descrivono la violazione.

Infine, per visualizzare in tempo reale l'andamento delle policy violate e prontamente segnalate da Snort si utilizza il comando:

snort -q -l /var/log/snort -i enp0s3 -A console -c /etc/snort/snort2.conf

La descrizione di ogni flag utilizzato è la seguente:

- "-q": modalità silenziosa, in modo da ridurre l'output verboso di Snort;
- "-l /var/log/snort": specifica la directory in cui verranno memorizzati i file di log generati da Snort;
- "-i enp0s3": specifica l'interfaccia di rete su cui Snort deve catturare il traffico;
- "-A console": specifica l'output di Snort sulla console, dove verranno visualizzati gli avvisi e le informazioni rilevate. In alternativa, si può utilizzare il flag "-A fast" che non permette di visualizzare i log in tempo reale su console, ma salva le informazioni nel file "/var/log/snort/alert";
- "-c /etc/snort/snort2.conf": specifica il percorso del file di configurazione di Snort da utilizzare.



Squid

Squid è un Server Web Proxy che fornisce delle funzionalità di caching e proxing. Esso consente di stabilire delle regole per l'utilizzo delle risorse in rete:

- Policy di accesso per delle classi di indirizzi IP
- Restrizioni per gruppi di utenti o singoli individui
- Stabilire un database di siti web accessibili o proibiti
- Definire una lista di parole chiave che possono identificare i siti da rendere irraggiungibili
- Controllo degli accessi
- Controllo del contenuto

Il firewall non è installato di default nel sistema del Bastion Host. Per installarlo, aprire il terminale e inserire il comando:
apt install squid

Provvederà a scaricare e ad installare il programma.
Una volta installato, posizionarsi nella cartella di installazione da terminale
cd /etc/squid/
e aprire il file di configurazione di squid con il comando:
nano squid.conf

Troveremo all'interno già delle regole commentate sulle loro funzionalità.
Il formato usato da Squid è:

- **acl aclname acltype string | file**: denota un nome al quale associamo un particolare file
- **acl aclname src ip_address | ip_mask**: denota un nome al quale associamo una serie di indirizzo/i
- **http_access allow | deny aclname** * : per consentire/negare i diritti di accesso

Nel nostro caso, posizioniamoci dopo la sezione commentata "INSERT YOUR OWN RULE(S) HERE" e inseriamo le regole.
Con il comando **include /etc/squid/conf.d/***impartiamo che tutti i file di configurazione nel path specificato devono essere considerati per le regole di firewall, specifico la rete interna da proteggere, (nel nostro caso il server Risorsa) e scrivo i comandi:

acl localnet src 192.168.1.0/24
http access allow localnet

Con la prima coppia di comandi abbiamo associato alla sottorete 192.168.1.0/24 il nome localnet e abbiamo consentito l'accesso alla comunicazione http. Mentre con la

coppia successiva di comandi abbiamo assegnato al nome badurl il contenuto del file txt url contenente una serie di nomi di host al quale verrà negato l'accesso.

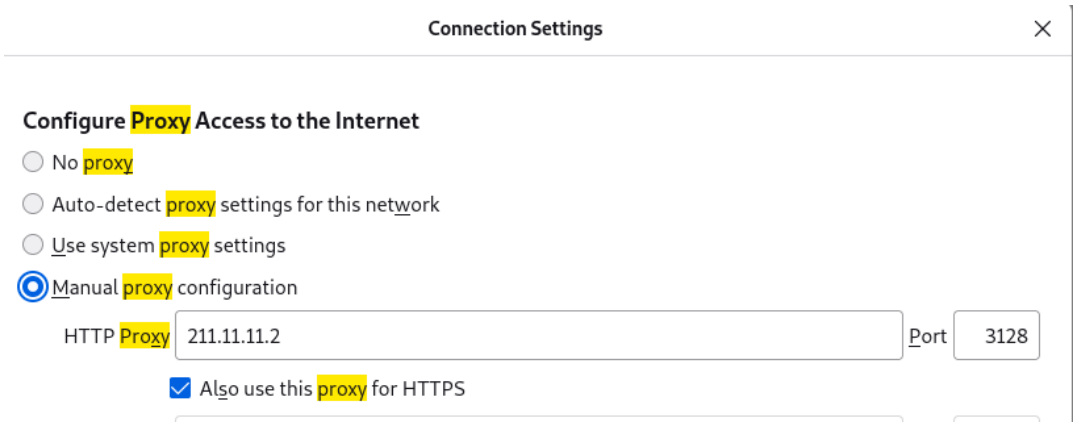
acl badurl url_regex "/etc/squid/url.txt"
http_access deny badurl

La situazione finale del file di configurazione dovrebbe essere la seguente:

```
  GNU nano 5.4

# Deny CONNECT to other than secure SSL ports
http_access deny CONNECT !SSL_ports

# Only allow cachemgr access from localhost
http_access allow localhost manager
http_access deny manager

# We strongly recommend the following be uncommented to protect innocen
# web applications running on the proxy server who think the only
# one who can access services on "localhost" is a local user
#http_access deny to_localhost

#
# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS
#
include /etc/squid/conf.d/*

acl localnet src 192.168.1.0/24
http_access allow localnet

acl badurl url_regex "/etc/squid/url.txt"
http_access deny badurl

# Example rule allowing access from your local networks.
# Adapt localnet in the ACL section to list your (internal) IP networks
# from where browsing should be allowed
#http_access allow localnet
http_access allow localhost

# And finally allow all other access to this proxy
http_access allow all
```

Una volta salvato il file possiamo impartire il comando per riavviare Squid:
systemctl restart squid

In questo modo Squid potrà applicare le nuove regole impostate. Posizionarsi sulla macchina sul quale impostare il Proxy (nel nostro caso sulla Risorsa). Nel browser Mozilla settare nelle impostazioni di Proxy l'IP della macchina che ha il servizio di Squid: si configura il proxy in modo manuale inserendo l'IP del Bastion Host (211.11.11.2) e la porta di default che utilizza Squid (3128). Inoltre, si spunta anche la casella che permette di gestire anche il traffico HTTPS, e non solamente quello di tipo non crittografato.



È possibile osservare i log generati da Squid durante la connessione della macchina al quale abbiamo impostato la macchina come proxy con il comando:
tail -f /var/log/squid/access.log

Qualora qualcosa non andasse bene e si volesse riprovare con una nuova configurazione, è bene cancellare la cache in modo da evitare che riporti i risultati del test precedente. Arrestare quindi Squid con il comando
systemctl stop squid
Dopodichè cancellare la sua cache con il comando
rm -rf /var/spool/squid/*
Effettuare le modifiche e avviare nuovamente Squid, che è pronto ad un nuovo test. Ad ogni modo, per controllare lo stato di funzionamento di Squid si può utilizzare il classico comando per visualizzare lo stato dei servizi:
systemctl status squid

Una precisazione che non è stata effettuata fino a questo punto, perché data per scontata, riguarda il fatto che per gestire tutte queste configurazioni e per lanciare la maggior parte dei comandi si ha bisogno delle credenziali da amministratore all'interno delle varie macchine Linux adoperate.
Inoltre, essendo le macchine collegate in una rete interna priva di accesso ad Internet, il file con gli indirizzi URL da bloccare è puramente a scopo dimostrativo, dato che non può essere effettivamente testato.

Tripwire

Si tratta di un Target-Based IDS, che monitora il file system e gli accessi che vengono fatti su di esso. Esso è di tipo “misuse detection” basato sulle policy di sicurezza. In tal caso si stabiliscono le policy di accesso ai vari file e si controlla se c’è una violazione di esse.
Anche Tripwire può essere usato con finalità e obiettivi leggermente diversi, infatti si può usare come strumento di intrusion detection, di analisi forense (per capire qual è stato l’impatto di un attacco) e per tenere traccia delle modifiche fatte al file system.
Le regole stesse usate da Tripwire vengono criptate in modo che un attaccante non riesca a modificarle. Quindi, esso genera una baseline prendendo uno snapshot di file e directory, confronta con la baseline contenuta nel suo database. A questo punto segnala le modifiche, le aggiunte o le cancellazioni.
Nel nostro caso abbiamo installato l’app nella Risorsa, per verificare che i suoi asset non fossero stati violati.

Posizionarsi sul terminale ed installare Tripwire con il comando:

apt-get install tripwire

Durante l’installazione verranno richieste di inserire due chiavi, queste serviranno per crittografare le regole di configurazione, è importante ricordarsele altrimenti sarà necessario reinstallare il programma.
Controllare che la configurazione di base non abbiano regole di default per file inesistenti nel sistema(che causano falsi positivi di violazione).
Entriamo dentro il file di configurazione con il comando:
nano /ect/tripwire/twpol.txt

e commentiamo tutti i file mancanti che vengono generati da questo file di configurazione e impostiamo le nostre regole personalizzate. Nel nostro caso abbiamo creato due nuovi file del quale ad uno non è possibile modificare i permessi dentro la directory “cartella segreta”, che ha importanza (*severity*) il cui valore SIG_MED è impostato nel file di configurazione di default con un valore numerico (*66*). Mentre nell’altro file “leggi_solo.txt” creato ad hoc per la simulazione sono consentite solo le letture. Il risultato dei comandi inseriti nel file dovrebbe apparire in questo modo:

```
#
# Commonly accessed directories that should remain static with regards
# to owner and group
#
(
  rulename = "Invariant Directories",
  severity = $(SIG_MED)
)
{
        /               -> $(SEC_INVARIANT) (recurse = 0) ;
        /home           -> $(SEC_INVARIANT) (recurse = 0) ;
        /tmp            -> $(SEC_INVARIANT) (recurse = 0) ;
        /usr            -> $(SEC_INVARIANT) (recurse = 0) ;
        /var            -> $(SEC_INVARIANT) (recurse = 0) ;
        /var/tmp        -> $(SEC_INVARIANT) (recurse = 0) ;
        /home/risorsa/Documents/cartella_segreta/ -> $(SEC_INVARIANT) (recurse = 0) ;
}


# nuova regola
(
 rulename = "Varie",
 severity = $(SIG_LOW)
)
{
        /home/risorsa/Documents/cartella_pubblica/leggi_solo.txt -> $(ReadOnly)  ;
}
```

Salviamo le modifiche applicate con il comando:

/sbin/twadmin --create-polfile -S site.key /etc/tripwire/twpol.txt

Inizializziamo il database con il quale Tripwire farà il confronto di integrità con il comando:

/sbin/tripwire -–init

Impostato tutto, è possibile effettuare un controllo del sistema con il comando:

/sbin/tripwire --check

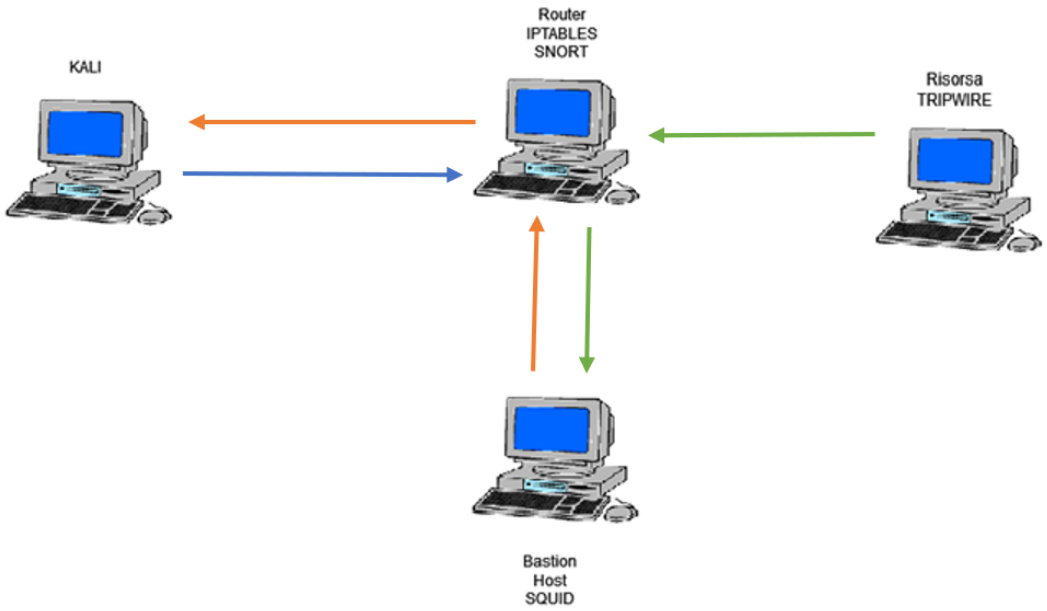
che produrrà un file di log con i risultati. I file di log sono situati nel path **“/var/lib/tripwire/report**” ed è possibile visualizzarli con il comando:

/sbin/twprint -m r --twrfile /var/lib/tripwire/report/10-20230530-174538.twr

Simulazione d'uso

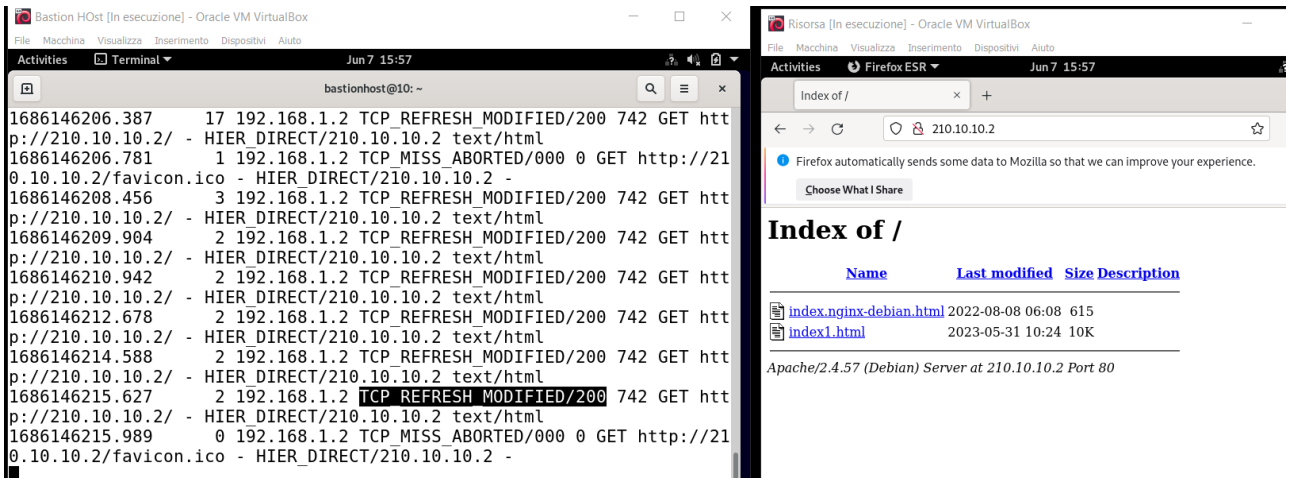
Connessione della macchina risorsa vero l'esterno della rete aziendale

Al fine di visualizzare il corretto funzionamento dell'infrastruttura, si vanno ad effettuare delle richieste di collegamento da parte della macchina Risorsa verso il mondo esterno. In questo caso ci si collega, tramite il web browser, all'indirizzo IP di Kali Linux. L'obiettivo è quello di assicurarsi che il traffico di rete parta correttamente dalla Risorsa, passi per il proxy server Squid (Bastion Host) attraverso il Router, e successivamente esca, sempre tramite il Router, dalla rete aziendale per effettuare la richiesta della pagina web alla macchina Kali Linux; infine può tornare indietro, da Kali Linux, alla Risorsa come risposta alla richiesta della GET. Il traffico dei pacchetti è riassunto in maniera semplice nella Figura sottostante: la Risorsa fa la richiesta della pagina web di Kali (prima freccia verde sulla dx), ma può comunicare esclusivamente con il Router, il quale dunque inoltra il pacchetto al Bastion Host (seconda freccia verde). Quest'ultimo, per mezzo del proxy Squid, monitora il traffico e lo instrada nuovamente verso il Router, il quale è l'unico capace di comunicare direttamente con il mondo esterno (frecce arancioni). Kali, una volta ricevuta la richiesta, risponde direttamente al Router, facendo eseguire ai pacchetti lo stesso percorso ma con il verso opposto dell'andata (freccia blu).

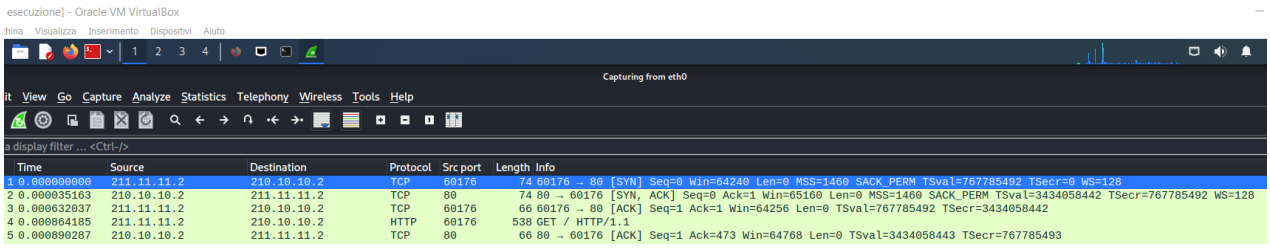


Essendo una connessione stateful si vanno quindi ad applicare le policy per cui Iptables è in grado di ricordarsi dell'instaurazione di tale connessione e permette il passaggio del traffico, dato che Squid cambia in modo casuale la porta di uscita

attraverso cui fa transitare il traffico verso Kali Linux. Innanzitutto, si può osservare come, dalla Figura sottostante a destra, la macchina Risorsa è in grado di visualizzare correttamente il contenuto della pagina web di Kali: sono infatti presenti due file che rappresentano le configurazioni sei servizi web (Apache e Nginx). Nella stessa Figura è inoltre possibile osservare come, una volta lanciato il comando di visualizzazione dei log in tempo reale di Squid all'interno del Bastion Host, effettivamente le richieste di connessione della Risorsa passano per il proxy server.



Per essere ulteriormente sicuri che il traffico dei pacchetti transiti correttamente per Squid per poi giungere su Kali, si è utilizzato uno strumento di analisi dei pacchetti, quale WhireShark, che risulta essere preinstallato su Kali. Effettivamente, come si può osservare dalla Figura riportata sotto, la richiesta di connessione (SYN, SYN-ACK e ACK) avviene da parte del Bastion Host (211.11.11.2) verso la macchina Kali Linux (210.10.10.2). Dunque, il proxy server Squid svolge correttamente il proprio compito, nascondendo alle proprie spalle la macchina Risorsa da proteggere.



Allo stesso tempo, grazie alle policy di Iptables, che lascia passare il traffico tra Bastion Host e Kali per quelle connessioni stabilite e nuove, la Risorsa può effettivamente contattare la pagina web di Kali Linux come appena dimostrato. Infatti, se si fa riferimento alla Figura sottostante si può osservare come le due regole per la

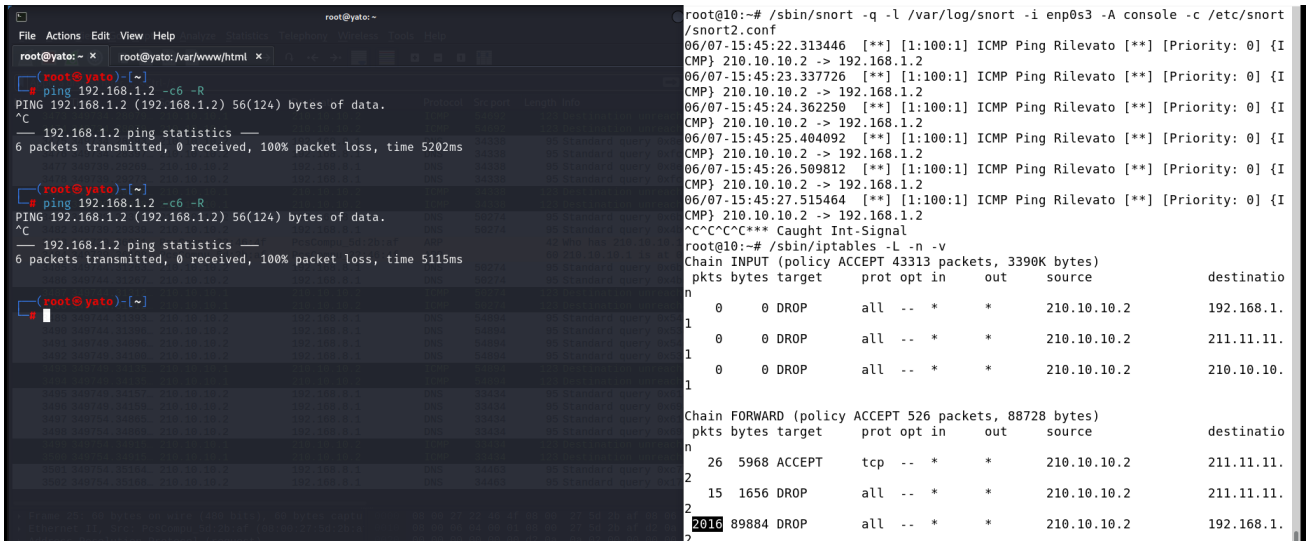
connessione stateful siano state effettivamente applicate: prima della richiesta HTTP si hanno zero pacchetti che interessano le due policy, mentre, a seguito della richiesta web, si può osservare come le policy siano applicate a 5 pacchetti che transitano per il Router.

```
root@10:/home/router# /sbin/iptables -L -n -v
Chain INPUT (policy ACCEPT 1792 packets, 140K bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DROP       all  --  *      *       210.10.10.2          192.168.1.1
    0     0 DROP       all  --  *      *       210.10.10.2          211.11.11.1
    0     0 DROP       all  --  *      *       210.10.10.2          210.10.10.1

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     tcp  --  *      *       211.11.11.2          210.10.10.2          ctstate NEW,RELATED,ESTABLISHED
    0     0 ACCEPT     tcp  --  *      *       210.10.10.2          211.11.11.2          ctstate NEW,RELATED,ESTABLISHED
    0     0 DROP       all  --  *      *       210.10.10.2          211.11.11.2
    0     0 DROP       all  --  *      *       210.10.10.2          192.168.1.2

Chain OUTPUT (policy ACCEPT 1550 packets, 135K bytes)
 pkts bytes target     prot opt in     out     source               destination
root@10:/home/router# /sbin/iptables -L -n -v
Chain INPUT (policy ACCEPT 2382 packets, 186K bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DROP       all  --  *      *       210.10.10.2          192.168.1.1
    0     0 DROP       all  --  *      *       210.10.10.2          211.11.11.1
    0     0 DROP       all  --  *      *       210.10.10.2          210.10.10.1

Chain FORWARD (policy ACCEPT 10 packets, 1636 bytes)
 pkts bytes target     prot opt in     out     source               destination
    5   690 ACCEPT     tcp  --  *      *       211.11.11.2          210.10.10.2          ctstate NEW,RELATED,ESTABLISHED
    5   959 ACCEPT     tcp  --  *      *       210.10.10.2          211.11.11.2          ctstate NEW,RELATED,ESTABLISHED
    1    56 DROP       all  --  *      *       210.10.10.2          211.11.11.2
    0     0 DROP       all  --  *      *       210.10.10.2          192.168.1.2

Chain OUTPUT (policy ACCEPT 2040 packets, 178K bytes)
 pkts bytes target     prot opt in     out     source               destination
```

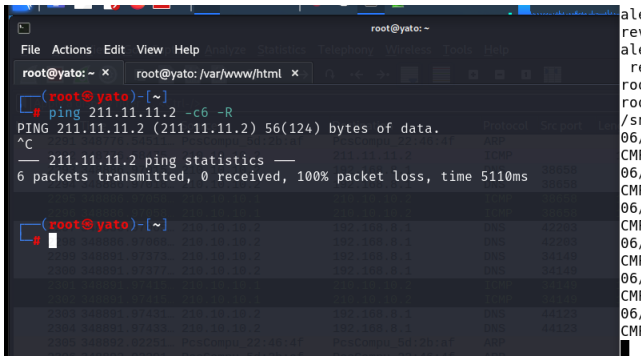
Blocco del tentativo di ping da parte di Kali

In questo paragrafo, invece, si vogliono andare ad analizzare sempre le regole di FORWARD di Iptables, che bloccano tentativi di accesso diretto alla Risorsa e al Bastion Host. In aggiunta, in questo scenario di attacco, che può essere paragonato ad un tentativo di Information Gathering con strumenti di scansione della rete, come Nmap, si va a testare il funzionamento dell'IDS Snort nel rilevare questo tentativo di scanning delle macchine. La simulazione di attacco viene replicata sia per la macchina Risorsa che per il Bastion Host. Nel primo caso si procede ad effettuare un tentativo di ping da parte di Kali con 6 pacchetti (**-c 6**) e tenendo traccia del percorso degli stessi (**-R**). L'infrastruttura risulta configurata correttamente, infatti, il ping non va a buon fine e sia su Snort che su Iptables si ha riscontro di tale tentativo. Snort, seguendo la prima policy che è stata inserita, è in grado di rilevare il tentativo di ping.

Infatti, sulla console Snort del Router appaiono 6 messaggi che confermano i tentativi di ping effettuati dalla macchina Kali Linux. Mentre, si osservano, nel riquadro in basso a destra relativo a iptables, come effettivamente siano stati bloccati dei pacchetti rispetto al caso precedente.



Lo stesso comportamento si può osservare se Kali prova ad effettuare il tentativo di ping nei confronti del Bastion Host. Il risultato è riportato nella Figura sottostante. Come per il caso appena descritto, sulla sinistra della Figura si riporta il tentativo di ping, sulla destra il comportamento di Snort e sotto il risultato dall'applicazione delle policy da parte di Iptables: si passa da 9 a 15 pacchetti droppati.




```
alert tcp any any -> $HOME_NET 22 (msg:"Tentivo Autenticazione SSH"; sid:101;
rev:1;)
alert tcp any any -> $HOME_NET 21 ( msg:"Tentativo Connessione FTP";  sid:102;
 rev:1;)
root@10:~# man snort
root@10:~# /sbin/snort -q -l /var/log/snort -i enp0s3 -A console -c /etc/snort
/snort2.conf
06/07-15:30:53.979759  [**] [1:100:1] ICMP Ping Rilevato [**] [Priority: 0] {I
CMP} 210.10.10.2 -> 211.11.11.2
06/07-15:30:54.987500  [**] [1:100:1] ICMP Ping Rilevato [**] [Priority: 0] {I
CMP} 210.10.10.2 -> 211.11.11.2
06/07-15:30:56.010285  [**] [1:100:1] ICMP Ping Rilevato [**] [Priority: 0] {I
CMP} 210.10.10.2 -> 211.11.11.2
06/07-15:30:57.043093  [**] [1:100:1] ICMP Ping Rilevato [**] [Priority: 0] {I
CMP} 210.10.10.2 -> 211.11.11.2
06/07-15:30:58.063521  [**] [1:100:1] ICMP Ping Rilevato [**] [Priority: 0] {I
CMP} 210.10.10.2 -> 211.11.11.2
06/07-15:30:59.089655  [**] [1:100:1] ICMP Ping Rilevato [**] [Priority: 0] {I
CMP} 210.10.10.2 -> 211.11.11.2
```

Regola prima del ping:

```
    9   912 DROP       all  --  *      *       210.10.10.2          211.11.11.2
```

Regola dopo il ping:

```
   15  1656 DROP       all  --  *      *       210.10.10.2          211.11.11.2
```

Controllo integrità dei file nella macchina Risorsa

L'ultima analisi riguarda il controllo dell'integrità dei file da parte di Tripwire. Nell'esempio in questione si simula lo scenario in cui un attaccante, prendendo possesso della macchina, vada a modificare alcuni file che in realtà sono monitorati dall'IDS. In particolare, nel nostro caso si effettua un tentativo di modifica del file "leggi_solo.txt" che ha come policy quella di sola lettura. Una volta effettuata la modifica del file si ha una violazione della policy. Questa violazione può essere rilevata se si fa una operazione di Check con Tripwire. Infatti, come riportato nella Figura sottostante, la violazione viene prontamente rilevata: si osserva un'anomalia nel confronto tra gli hash salvati di Tripwire e quelli calcolati dal check, parimenti si evidenzia anche un aumento della dimensione del file.

```
---------------------------------------------------------------------------
  Section: Unix File System
---------------------------------------------------------------------------

  Rule Name                       Severity Level    Added    Removed  Modified
  ---------                       --------------    -----    -------  --------
  Other binaries                  66                0        0        0
  Tripwire Binaries               100               0        0        0
  Other libraries                 66                0        0        0
  Root file-system executables    100               0        0        0
  Tripwire Data Files             100               0        0        0
  System boot changes             100               0        0        0
  (/var/log)
  Root file-system libraries      100               0        0        0
  (/lib)
  Critical system boot files      100               0        0        0
  Other configuration files       66                0        0        0
  (/etc)
  Boot Scripts                    100               0        0        0
  (/etc/init.d)
  Security Control                66                0        0        0
  Root config files               100               0        0        0
  Devices & Kernel information    100               0        0        0
  (/dev)
  Invariant Directories           66                0        0        0
* Varie                           33                0        0        1
  (/home/risorsa/Documents/cartella_pubblica/leggi_solo.txt)

Total objects scanned:  39774
Total violations found:  1

===========================================================================
Object Detail:
===========================================================================


---------------------------------------------------------------------------
  Section: Unix File System
---------------------------------------------------------------------------

---------------------------------------------------------------------------
Rule Name: Varie (/home/risorsa/Documents/cartella_pubblica/leggi_solo.txt)
Severity Level: 33
---------------------------------------------------------------------------
  ---------------------------------------
  Modified Objects: 1
  ---------------------------------------

Modified object name:  /home/risorsa/Documents/cartella_pubblica/leggi_solo.txt

  Property:            Expected                   Observed
  -------------        -----------                -----------
* Size                 26                         30
* Modify Time          Tue 30 May 2023 05:22:07 PM CEST
                                                  Tue 30 May 2023 05:45:31 PM CEST
* CRC32                Br4ciN                     BnSvfs
* MD5                  CV6BGisJKOF0UFF189FPWM     DBQ/q2ql2vEW4R1wOy3XKy
```

TroubleShooting

IpTables

È bene salvare le configurazioni impostate su IpTables su file con il comando

/sbin/iptables-save > /home/router/Documents/iptables_finale

Dopo aver resettato le regole con il comando

iptables –F

è possibile ripristinare la configurazione salvata su “iptables_finale” con il comando:

/sbin/iptables-restore < /home/router/Documents/iptables_finale.

Per eliminare una regola specifica, possiamo sfruttare l’enumerazione delle regole fornita da Iptables stesso, visualizziamo dunque le regole con i relativi numeri con il comando:

iptables -n -L –line-numbers

Grazie a questo comando possiamo dunque conoscere il loro numero e scrivere il comando

iptables -D <catena> <numero_regola>