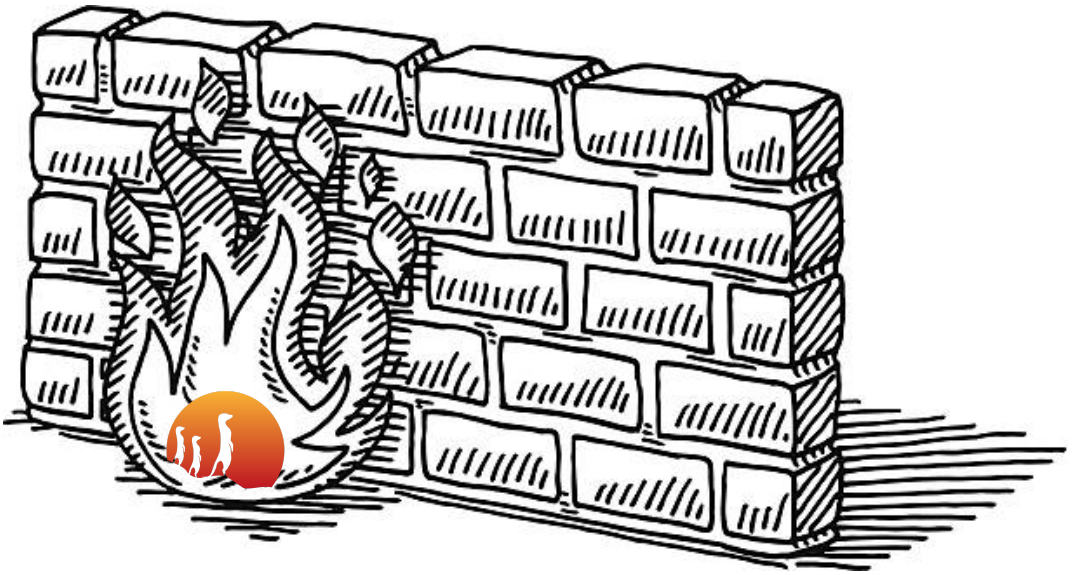
1 2 9 0

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

NFTables + Suricata

Assignment #2



Mestrado em Segurança Informática
Ano letivo 2022/2023

Inês Martins Marçal **Nº: 2019215917**
João Carlos Borges Silva **Nº: 2019216753**

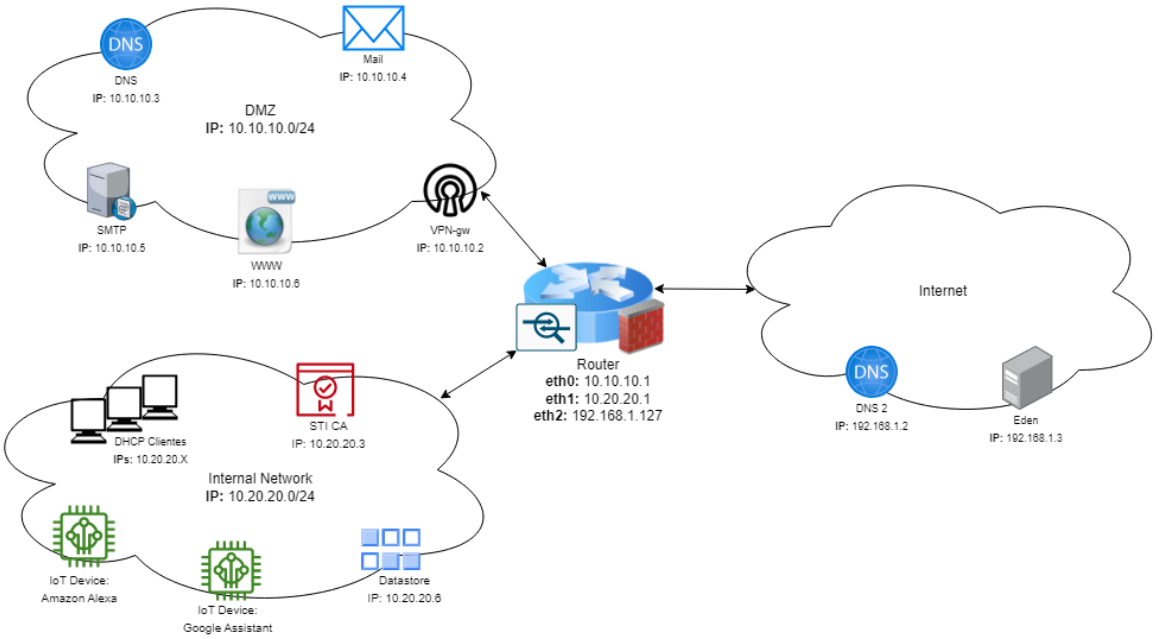
Índice

Introdução

Este trabalho foi realizado no âmbito da cadeira de Segurança em Tecnologias da Informação e visa a implementação de um cenário entre 3(4) máquinas virtuais, aplicando diversos elementos estudados durante as aulas, como *nftables* e *IDS/IPS*. Os objetivos a alcançar por meio da realização deste trabalho são os seguintes:

- Configuração de uma *Firewall* utilizando *nftables*, com recurso à definição de políticas e regras.
- Configuração dos serviços requisitados nas várias redes pertencentes ao cenário.
- Configuração de uma ferramenta *IDS/IPS* para que fosse possível detetar e prevenir ataques do tipo *SQL Injection* e *DoS*.

De modo a facilitar a compreensão do cenário implementado, foi elaborado o seguinte esquema (o mesmo incluí os serviços implementados em cada rede, assim como os respectivos endereços *IP* atribuídos):



Como se pode observar existem 2 pontos conectados pelo router à *internet*, a *Internal Network* e o *DMZ*, podendo representar-se este cenário através de 3 ou 4 Máquinas Virtuais. Para a máquina virtual *DMZ* é, assim, utilizado maioritariamente o *IP* 10.10.10.2, enquanto que para a máquina virtual *Internal Network* assumiu-se o *IP* 10.20.20.3. Já para simular a *Internet* poderia ter sido apenas usado o nosso *host* de virtualização (o nosso computador), mas preferiu-se utilizar uma máquina virtual para o efeito.

Finalmente, foi criada uma máquina virtual para o *Router*, onde o mesmo contém 3 interfaces, cada uma referente a uma rede diferente ou *Internet*. O cenário apresentado envolve, portanto, 4 máquinas virtuais.

Software utilizado

À semelhança do primeiro trabalho prático, recorreu-se a *software* tanto a nível da virtualização das máquinas virtuais, como para a instalação de algumas ferramentas nas mesmas. Relativamente à virtualização das máquinas virtuais, optou-se pela utilização (tal como no primeiro trabalho) da *Oracle VM Box*, tendo sido escolhida a distribuição *Kali*

Rolling 2022.4 (e a *Lubuntu 22.04* de modo a simular a Internet). Como tal, os comandos mencionados ao longo deste trabalho seguirão estas mesmas distribuições.

De modo a atingir o objetivo e o cenário propostos foi instalado o seguinte software:

- Router
  - nftables (já instalado por *default*): constituiu, na sua essência, a *firewall* presente no router que, juntamente com as regras e políticas especificadas, permitirá gerir o tipo de tráfego que é permitido trocar entre todas as redes do cenário.
  - suricata (já instalado por *default*: mecanismo de segurança *IDS/IPS* utilizado. O mesmo, em conjunto com as *nftables*, permite detetar e bloquear o tráfego (segundo as regras definidas no *suricata*).

Para o efeito, foi utilizado o seguinte comando:

```
┌──(root㉿kali)-[/home/kali]
└─# sudo apt install nftables suricata
```

- Internet
  - bind9: permite a criação de um servidor *DNS* (com recurso a outros servidores já existentes, como o servidor da google 8.8.8.8).

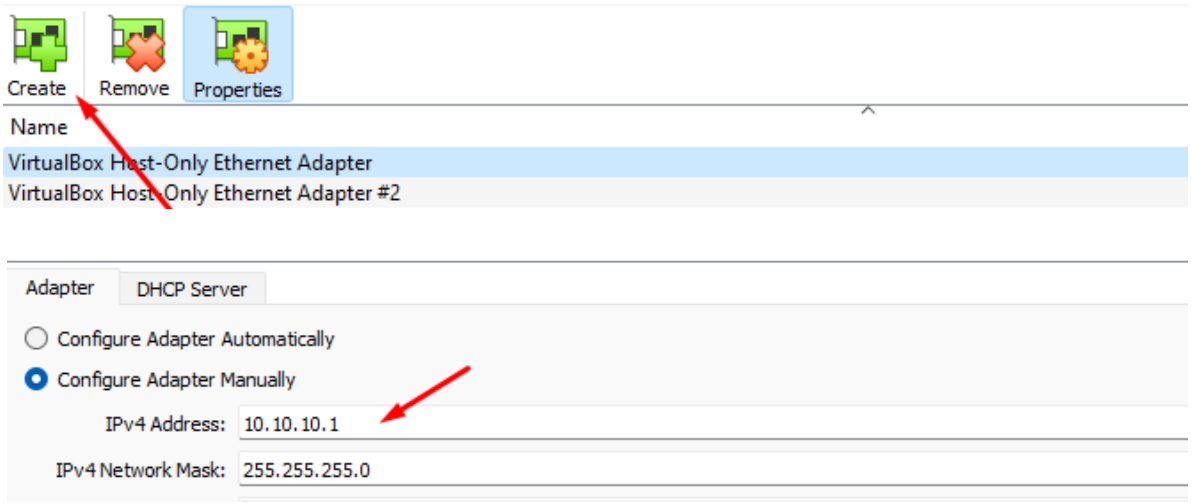
Para tal, foi utilizado o seguinte comando:

```
┌──(root㉿kali)-[/home/kali]
└─#  bind9
```

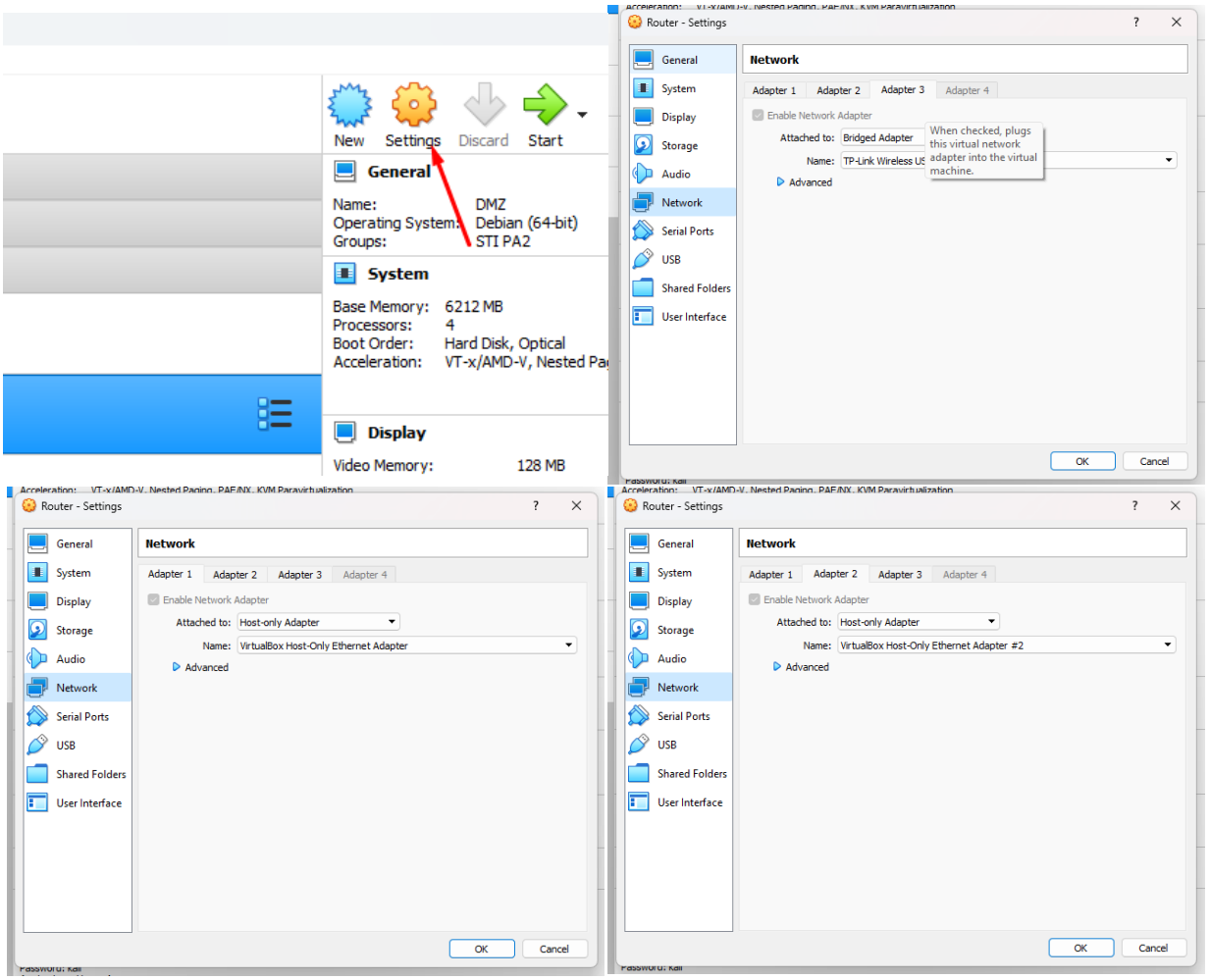
- DMZ: bind9

Configuração de *IPs* das Máquinas Virtuais

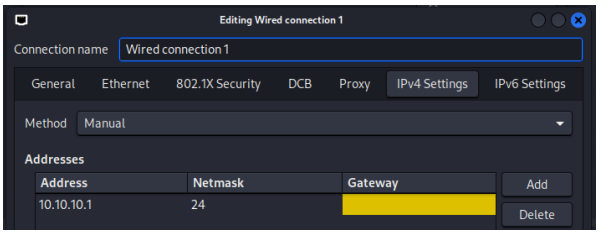
De modo a assegurar que todas as Máquinas Virtuais possuíam endereços IP adequados, foram necessárias algumas configurações ao nível de adaptadores de rede. Assim, em primeiro lugar, procedeu-se à criação dos seguintes tipos de adaptadores: *Bridged Adapter* (com vista a estabelecer ligação ao *WiFi* e conseguir instalar o *software* necessário) e *Host-Only* (para estabelecer as restantes ligações às redes). Inicialmente foram criados 2 adaptadores *Host-Only*, utilizados não só para abranger os IPs das redes *DMZ* e *Internal Network*, bem como os IPs das interfaces do router que estão ligadas às mesmas.

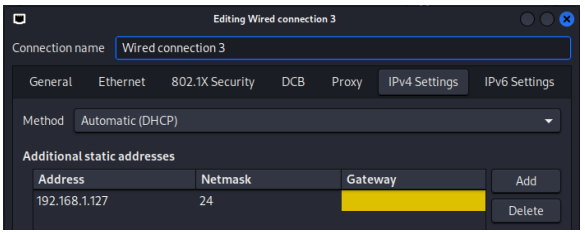


De seguida, define-se cada adaptador da seguinte forma:



Finalmente, já dentro de cada Máquina Virtual, configura-se o *IP* desejado. É importante notar, que os IPs dos adaptadores *Host-Only* devem ser configurados com a opção Manual selecionada, enquanto que IPs dos adaptadores *Bridged Adapter* devem ser configurados com a opção *Automatic (DHCP)* selecionada.





Nos adaptadores Host-only das máquinas virtuais *DMZ* e *Internal Network*, além de ser necessário especificar o IP e a máscara, deverá indicar-se a respetiva gateway (10.10.10.1 no caso da *DMZ*, 10.20.20.1 no caso da *Internal Network* 192.168.1.127 no caso da *Internet*). Isto permitirá estabelecer uma rota entre estas duas máquinas virtuais.

Configurações iniciais nas máquinas virtuais

De modo a assegurar que todas as máquinas virtuais não têm mais nenhuma *firewall* a ser executada, foram verificadas as regras presentes nas *nftables* e *iptables* de cada máquina

virtual, constatando-se que ambas as tabelas encontravam-se vazias ou continham apenas políticas de *accept*. Para efetuar tal verificação, executou-se os seguintes comandos:

```
┌──(root㉿kali)-[/home/kali]
└─# iptables -L -v -n
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

┌──(root㉿kali)-[/home/kali]
└─#   nft list ruleset
```

Relativamente às máquinas virtuais que continham servidores DNS, foi necessário proceder-se a algumas configurações nas mesmas. Seguem-se os passos realizados:

```
kali@kali:~$ sudo nano /etc/bind/named.conf.options
```

```
options {
        directory "/var/cache/bind";

        // If there is a firewall between you and nameservers you want
        // to talk to, you may need to fix the firewall to allow multiple
        // ports to talk.  See http://www.kb.cert.org/vuls/id/800113

        // If your ISP provided one or more IP addresses for stable
        // nameservers, you probably want to use them as forwarders.
        // Uncomment the following block, and insert the addresses replacing
        // the all-0's placeholder.

        recursion yes;
        // allow-query { any };

        forwarders {
                //193.136.212.1;
                8.8.8.8;
        };
        dnssec-validation auto;

        listen-on-v6 { any; };
};
```

Foi, portanto, adicionado o ip do servidor DNS público da *google*, de modo a que o servidor DNS criado pelas respectivas VMs consiga resolver nomes. Para que as restantes máquinas virtuais, posteriormente, pudessem utilizar o servidor DNS da Internet, foi adicionada a seguinte linha de configuração no ficheiro "/etc/resolv.conf":

```
┌──(kali㉿kali)-[/home/kali]
└─# sudo nano /etc/resolv.conf

# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
# 127.0.0.53 is the systemd-resolved stub resolver.
# run "resolvectl status" to see details about the actual nameservers.

nameserver 192.168.1.2
```

Os servidores DNS são iniciados da seguintes forma:

```
root@kali:/home/kali# systemctl start bind9
```

Configurações gerais das nftables

Após todas as configurações iniciais, poderá prosseguir-se para a configuração das regras e políticas da *firewall* a implementar com *nftables*. Considerando que os requisitos exigidos pelo enunciado recorrem aos protocolos de comunicações TCP ou UDP, as tabelas a serem criadas serão da família *inet* (isto é *ipv4*). É de relembrar que as configurações das *nftables* podem ser aplicadas através de um ficheiro de configurações *.nft*, neste caso *router.nft*. Começa-se, assim, por definir uma tabela *inet* com apenas 3 *chains*, uma com *hook input*, outra com *hook output* e ainda uma com *hook forward*:

```
        chain output {
                type filter hook output priority 0; policy drop;
        }

        chain forward {
                type filter hook forward priority 0; policy drop;
        }
}
```

Como se pode observar, além das três *chains* criadas (e, por sua vez, as três *hooks*), foi definida uma política *drop*, significando que qualquer tráfego de rede que não corresponda a uma das regras definidas na *firewall* será descartado ao passar pelo *router*.

Apesar de não ter sido pedido no enunciado, foi adicionada uma *chain* com a *hook prerouting* para que fosse possível efetuar *debug* da *firewall*:

```
chain debug {
        type filter hook prerouting priority -301;
        ip protocol { tcp,udp } meta nftrace set 1;
}
```

Por fim, foram definidas variáveis de modo a guardar os valores dos IPs a utilizar durante a configuração das *nftables*. Assim sendo, no início do ficheiro *router.nft* adicionou-se as seguintes linhas:

```
define vpn-gw_dmz = {10.10.10.2}
define network_dmz = {10.10.10.0/24}
define dns_dmz = {10.10.10.3}
define mail_dmz = {10.10.10.4}
define smtp_dmz = {10.10.10.5}
define www_dmz = {10.10.10.6}

define network_internal-network = {10.20.20.0/24}
define sti-ca_internal-network = {10.20.20.3}
define datastore_internal-network = {10.20.20.6}

define network_internet = {192.168.1.0/24}
define dns2_internet = {192.168.1.2}
define eden_internet = {192.168.1.3}

define interface_dmz = {10.10.10.1}
define interface_internal-network = {10.20.20.1}
define interface_internet = {192.168.1.127}
```

Configurações de tráfego de entrada no router das nftables

Efetuadas todas as configurações gerais, procedeu-se à definição das regras solicitadas nas *nftables*. Relativamente ao tráfego que entra diretamente no *router*, isto é, pelas interfaces do mesmo, pretende-se o seguinte:

1. Pedidos de resolução de nomes aos servidores DNS existentes;
2. Conexões SSH iniciadas pela *VPN-gw*, localizada no DMZ ou por qualquer IP da rede interna. Estas deverão estar protegidas com o mecanismo de *Port Knocking*, onde é solicitado pelo menos 5 *port knocks*.

Para tal, foram definidas as seguintes regras:

1.
```
ip saddr { $dns_dmz, $dns2_internet } tcp sport { domain } ct state new,established,related counter accept;
ip saddr { $dns_dmz, $dns2_internet } udp sport { domain } ct state new,established,related counter accept;

ip daddr { $dns_dmz, $dns2_internet } tcp dport { domain } ct state new,established,related counter accept;
ip daddr { $dns_dmz, $dns2_internet } udp dport { domain } ct state new,established,related counter accept;
```

   As 2 primeiras regras são definidas na *chain* de *input* e permitem que seja possível receber mensagens dos servidores DNS existentes, provenientes da porta para este mesmo fim (porta 53, correspondente ao nome *domain*). Já as últimas 2 encontram-se definidas na *chain* de *output,* possibilitando, assim, efetuar pedidos de resolução de nomes aos servidores DNS através da porta 53.

2.
```
ip saddr { $network_internal-network, $vpn-gw_dmz } tcp dport { ssh } ct state new,established,related counter accept;
```

Esta regra é definida na *chain* de *input*, permitindo conexões *SSH* provenientes de IPs que pertençam à *Internal Network* ou à *VPN-gw* da rede DMZ, originados na respetiva porta 22.
Este tipo de ligação deverá ainda ser protegido pelo mecanismo de *Port Knocking*, o qual é definido da seguinte forma:

```
table inet portknock {
        chain debug{
                type filter hook prerouting priority -301;
                # meta nftrace set 1 # For everything
                ip protocol {tcp} meta nftrace set 1 # Only for TCP and ICMP packets
        }

        set clients_ipv4 {
                type ipv4_addr
                flags timeout
                counter
        }

        set candidates_ipv4 {
                type ipv4_addr . inet_service
                flags timeout
                counter
        }

        chain input {
                type filter hook input priority -10; policy accept;

                iifname "lo" return

                tcp dport 123 add @candidates_ipv4 {ip  saddr . 234 timeout 60s}
                tcp dport 234 ip saddr . tcp dport @candidates_ipv4 add @candidates_ipv4 {ip saddr . 345 timeout 60s}
                tcp dport 345 ip saddr . tcp dport @candidates_ipv4 add @candidates_ipv4 {ip saddr . 456 timeout 60s}
                tcp dport 456 ip saddr . tcp dport @candidates_ipv4 add @candidates_ipv4 {ip saddr . 567 timeout 60s}
                tcp dport 567 ip saddr . tcp dport @candidates_ipv4 add @clients_ipv4 {ip saddr timeout 60s} log prefix "OK"

        #Allow SSH
                tcp dport { ssh } ip  saddr @clients_ipv4 counter accept
                tcp dport { ssh } ct state established,related counter accept

        # Reject the remaining
                tcp dport { ssh } counter reject with tcp reset
        }
}
```

Com vista a implementar este mecanismo procedeu-se à criação de uma tabela à parte. São inicialmente definidos 2 grupos distintos: *clients_ipv4* e *candidates_ipv4*. O primeiro é constituído pelo conjunto de *IPs:portas* que realmente poderão efetuar uma conexão SSH e já efetuaram o 5 *port knocks* pela sequência correta. O segundo representa o conjunto de *IPs:portas* que estão a seguir a sequência de *port knocks* estabelecida. Na *chain* de *input* é definida esta mesma sequência de portas da seguinte forma:

- As comunicações SSH são apenas permitidas para IPs que estejam no grupo *clients_ipv4*, como tal não será possível efetuar, inicialmente, este tipo de ligações.

- A primeira ligação deverá portanto ser feita em direção ao porto 123 e caso esta regra seja cumprida o *IP:porta* é adicionado ao conjunto de *candidates_ipv4*.
- O passo anterior é repetido para as restantes portas, excepto para a última. Assumindo que foi seguida a sequência de *port knocks* definida, efetuado o último na porta 567, o *IP:porta* usados vão ser adicionados ao grupo *clients_ipv4*.
- Finalmente será possível efetuar uma conexão SSH. É importante mencionar que cada *port knock* possui um *timeout* de 1 minuto, sendo que ao fim deste tempo o IP:porta são removidos dos respectivos grupos.

Configurações de tráfego entre redes das nftables

Quanto ao tráfego entre as redes *DMZ* e *Internal Network* são apenas permitidas as seguintes comunicações:

1. Pedidos de resolução de nomes ao servidor DNS existente na rede *DMZ*;
2. O servidor DNS na rede *DMZ* deve ser capaz de efetuar resolução de nomes a partir do servidor DNS da *Internet*.
3. Os servidores DNS da rede *DMZ* e *Internet* devem ser capazes de sincronizar os conteúdos das zonas DNS
4. Ligações ao servidor *SMTP* da rede *DMZ* pela porta *SMTP*
   Ligações ao servidor Mail da rede *DMZ* pelas portas *POP* e *IMAP*
   Ligações ao servidor *WWW* da rede *DMZ* pelas portas *HTTP* e *HTTPS*
   Ligações à *VPN-gw* da rede *DMZ* pela porta *OpenVPN*.

Foram, assim, definidas as seguintes regras:

1. ```
   ip daddr { $dns_dmz } tcp dport { domain } ct state new,established,related counter accept;
   ip daddr { $dns_dmz } udp dport { domain } ct state new,established,related counter accept;
   ip saddr { $dns_dmz } tcp sport { domain } ct state new,established,related counter accept;
   ip saddr { $dns_dmz } udp sport { domain } ct state new,established,related counter accept;
   ```

   Estas foram integradas na *chain forward*, visto estar-se perante informação redirecionada pelo router entre redes. As regras acima apresentadas permitem tanto saída como entrada de tráfego para o servidor DNS da rede *DMZ* através da porta 53. É ainda de mencionar que há 2 regras semelhantes, diferindo apenas nos protocolos utilizados, já que as comunicações DNS podem recorrer a estes mesmos 2 protocolos.

2. ```
   ip daddr { $dns_dmz } ip saddr { $dns2_internet } tcp dport { domain } ct state new,established,related counter accept;
   ip daddr { $dns_dmz } ip saddr { $dns2_internet } udp dport { domain } ct state new,established,related counter accept;
   ip daddr { $dns2_internet } ip saddr { $dns_dmz } tcp dport { domain } ct state new,established,related counter accept;
   ip daddr { $dns2_internet } ip saddr { $dns_dmz } udp dport { domain } ct state new,established,related counter accept;
   ```

   O conjunto de regras apresentado encontra-se novamente definido na *chain forward*, de modo a permitir que o servidor DNS da rede DMZ consiga efetuar a resolução de nomes através do servidor DNS da *Internet*. Este requisito é satisfeito, já que é indicado que tanto o tráfego no sentido [servidor DNS da rede *DMZ* -> servidor DNS da internet], como no sentido [servidor DNS da internet -> servidor DNS da rede *DMZ*] é aceite, novamente através das porta 53.
3. Este tipo de comunicação já é abrangida pelas regras anteriormente definidas.

4.
```
ip daddr { $smtp_dmz } tcp dport { smtp } ct state new,established,related counter accept;
ip daddr { $mail_dmz } tcp dport { pop3, imap } ct state new,established,related counter accept;
ip daddr { $www_dmz } tcp dport { http, https } ct state new,established,related counter accept;
ip daddr { $vpn-gw_dmz } tcp dport { openvpn } ct state new,established,related counter accept;
ip daddr { $vpn-gw_dmz } udp dport { openvpn } ct state new,established,related counter accept;
```

A definição destes últimos requisitos mostrou-se bastante simples, já que apenas foi necessário especificar o IP destino e o porto de destino correspondente a um determinado serviço. É de notar que, para conexões com a *VPN-gw* da rede *DMZ* por meio da porta *OpenVPN,* é utilizada a mesma regra para os protocolos TCP e UDP, já que o serviço OpenVPN pode usar qualquer um destes 2 protocolos.

Configurações de tráfego no sentido Internet -> Internal Network das nftables

Até agora as regras definidas aplicam-se apenas a conexões diretas do cenário. Para ligações indiretas, como *Internet* <-> *Internal Network*, além da *chain forward* também é necessário definir regras NAT. Estas encontram-se apresentadas de seguida:

1. Ligações SSH originadas pela *Internet* em direção à porta 2021 do *router* devem ser redirecionadas para a porta SSH do servidor *datastore*, localizado na *Internal Network*. Foi estabelecido um limite máximo de duas conexões simultâneas.
2. Qualquer ligação em direção à porta 2022 do router devem ser redirecionadas para a porta 2022 do servidor STI CA, localizado na *Internal Network*. Para o caso, apenas um IP, da *Internet*, é que poderá estabelecer esta ligação, pelo que o IP escolhido foi o do servidor Eden.

De seguida, são definidas as regras implementadas para que fosse possível concretizar o solicitado:

1.
```
ip saddr { $network_internet } ip daddr { $datastore_internal-network } tcp dport { ssh } ct count over 2 counter drop;
ip saddr { $network_internet } ip daddr { $datastore_internal-network } tcp dport { ssh } ct state new,established,related
counter accept;
```

Na *chain forward*, a primeira regra definida permite a entrada de tráfego da *Internet* para o endereço IP do servidor *datastore* na porta 22, localizado na *Internal Network*. É de notar que a primeira regra deverá anteceder obrigatoriamente a segunda, já que estas são verificadas sequencialmente. Desta forma, se já estiverem a ser estabelecidas 2 conexões, não será possível efetuar a próxima ligação SSH, pois primeiro é verificado se o número de ligações ativas é igual a 2 e, em caso afirmativo, a comunicação é logo interrompida.

2.
```
ip saddr { $eden_internet } ip daddr { $sti-ca_internal-network } tcp dport { 2022 } ct state new,established,related counter
accept;
```

Novamente, esta regra encontra-se definida na *chain forward* e irá permitir que o tráfego originado pelo servidor *Eden* (IP escolhido) e com destino ao servidor *STI CA* na porta 2022 seja aceite.

Finalmente, de modo a permitir a tradução de endereços de IPs provenientes da *Internet* para IPs da *Internal Network* deverá ser criada uma tabela NAT, sendo definida dentro da mesma uma *hook* de *prerouting*:

```
table nat {
        chain prerouting {
                type nat hook prerouting priority -100;
                iif { eth2 } ip daddr { $interface_internet } tcp dport { 2021 } ct state new,established,related dnat to
$datastore_internal-network:22;
```

```
                ip saddr { $eden_internet } iif { eth2 } ip daddr { $interface_internet } tcp dport { 2022 } ct state new,established,related
dnat to $sti-ca_internal-network:2022;
        }
}
```

Na tabela criada foram, assim, definidas 2 regras que permitem efetuar a tradução de IPs requisitada. É de notar que na segunda regra, a qual é referente ao ponto 2 deste capítulo, apenas é traduzido 1 endereço IP (servidor *Eden*).

Configurações de tráfego no sentido Internal Network -> Internet das nftables

Configuradas as ligações NAT no sentido *Internet -> Internal Network*, deverão ser definidas as regras de modo a filtrar o tráfego contrário. Os requisitos são os seguintes:

1. Pedidos de resolução de nomes ao servidor DNS existente na Internet;
2. Ligações HTTP, HTTPS e SSH originado por dispositivos com IP dinâmico (clientes DHCP).
3. Ligações HTTPS através do dispositivo *IoT Amazon Alexa* ao servidor público *alexa.amazon.com*
   Ligações HTTPS através do dispositivo *IoT Google Assistant* ao servidor público *assistant.google.com*

Com base no solicitado, foram definidas as seguintes regras:

1 e 2.

```
ip saddr { $network_internal-network } ip daddr { $network_internet } tcp dport { ssh, domain, http, https }
ct state new,established,related counter accept;
ip saddr { $network_internal-network } ip daddr { $network_internet } udp dport { domain }
ct state new,established,related counter accept;
```

Estas regras foram definidas na *chain forward* da tabela *inet router*, permitindo qualquer tráfego proveniente da *Internal Network* entrar na *Internet* através dos serviços de HTTP, HTTPS, SSH e DNS. No enunciado é referido que apenas os clientes com IP dinâmico podem realizar este tipo de ligações, mas para facilitar posteriormente os testes foi definido que qualquer IP da *Internal Network* pode efetuar estas comunicações.

3.

```
ip saddr { $network_internal-network } ip daddr { ocsp.pki.goog, pki.goog, crls.pki.goog, crl.pki.goog, assistant.google.com } tcp dport {
http, https } ct state new,established,related counter accept;
ip saddr { $network_internal-network } ip daddr { ocsp.digicert.com, cacerts.digicert.com, www.digicert.com, crl3.digicert.com,
crl4.digicert.com, alexa.amazon.com } tcp dport { http, https } ct state new,established,related counter accept;
```

A partir das regras acima estabelecidas torna-se possível que IPs da *Internal Network* consigam aceder ao *website alexa.amazon.com* e *assistant.google.com*. É de mencionar que, além dos nomes dos *websites*, foram também incluídos os nomes dos servidores OCSP, CA, CRL dos mesmos, de modo a que fosse possível aceder a estes por meio do *browser*.

É ainda necessário, novamente, a definição de regras para que seja possível a tradução de IPs utilizando o NAT:

```
chain postrouting {
        type nat hook postrouting priority 100;
        ip saddr { $network_internal-network } oif { eth2 } tcp dport { ssh, domain, http, https } ct state new,established,related snat to
        $interface_internet
        ip saddr { $network_internal-network } oif { eth2 } udp dport { domain } ct state new,established,related snat to $interface_internet
}
```

Desta forma, comunicações que tenham origem na *Internal Network* e saiam pelo *router* na interface *eth2* (interface ligada à *Internet*) serão traduzidos num IP da própria *Internet*, permitindo comunicação com a mesma.

IDS/IPS - Suricata

Após a definição das principais regras das *nftables* é necessário configurar um mecanismo IDS/IPS de modo a complementar esta mesma *firewall*, neste caso o Suricata. A configuração do Suricata requer apenas a definição de regras que possibilitem detetar os ataques: *SQL Injection* e *DoS*. Estas foram, portanto, adicionadas no ficheiro *suricata.rules*, devendo proceder-se à referência do mesmo no ficheiro */etc/suricata/suricata.yml* da seguinte forma:

```
##
## Configure Suricata to load Suricata-Update managed rules.
##

default-rule-path: /etc/suricata/rules

rule-files:
  - suricata.rules
```

Assim, quando o suricata estiver a ser executado, este saberá que tipo de regras deve detetar e, consequentemente, bloquear. Antes de se proceder à especificação das mesmas, é importante realçar que nas *chains input*, *output* e *forward,* a política de cada uma destas deve de conter a regra "queue" escrita abaixo, possibilitando assim o reencaminhamento do tráfego para o Suricata.

```
chain input {
        type filter hook input priority 0; policy drop;
        queue;
        ....
}

chain output {
        type filter hook output priority 0; policy drop;
        queue;
        ....
}

chain forward {
        type filter hook forward priority 0; policy drop;
        queue;
        ....
}
```

Finalmente são explicitadas as regras a usar no mecanismo de IDS/IPS:

- SQL Injection: tem como objetivo a execução de ataques do tipo SQL Injection, como ' OR 1==1 --, ou seja, SQL Injections que tentem usar um padrão com uma plica ou aspa para completar a *query* e a utilização de 2 traços para comentar as restantes *queries* que possam vir a ser executadas.

- DoS: para evitar *Denial-of-Services*, foi desenvolvida uma regra que ao detetar uma quantidade de 2500 pacotes vinda de um host seria interrompida a sua ligação por 5 segundos, tentando evitar *flooding* das redes.

Segue-se a definição e explicação das mesmas:

- SQL Injection:

```
drop tcp any any -> any any (msg:"Possible SQL Injection with apostrophe and comments";
                    pcre:"/[a-z0-9 ]*('|")+[a-z0-9 =]*--/i"; sid:99999998; rev:1;)
```

Através da regra acima definida, podemos concluir que a mesma "dropa" pacotes (*drop*) tcp que venham de qualquer *host* (*any any*) em direção a qualquer *host* (*any any*) e que sigam a seguinte expressão regular: [a-z0-9]*('|")+[a-z0-9 =]*--. Explicando a expressão regular por partes:

- [a-z0-9]*: qualquer caracter que seja uma letra ou número ou espaço e apareça 0 ou mais vezes;
- ('|")+: apareça 1 ou mais plicas ou aspas;
- [a-z0-9 =]*: qualquer caracter que seja uma letra ou número ou espaço ou sinal de igual e apareça 0 ou mais vezes;
- --: apareça 2 hífens seguidos.

- DoS:

```
drop tcp any any -> any any (msg:"Possible DoS Attack with SYN scans"; flags:S; flow:stateless;
                    threshold: type both, track by_dst, count 2500,seconds 5; sid:99999999; rev:1;)
```

A regra acima estabelecida permite "dropar" pacotes (*drop*) tcp que venham de qualquer *host* (*any any*) em direção a qualquer *host* (*any any*) e que sigam o seguinte padrão: na presença de um grande fluxo de pacotes, mais precisamente 2500, que contenham a *flag* S, correspondente a um *SYN scan*, o tráfego será interrompido por 5 segundos, de modo a evitar o *flood* da rede.

Execução da firewall e Suricata

Definidas todas as regras das *nftables* no ficheiro *router.nft,* é necessário colocá-las em vigor para que seja possível proteger o cenário implementado. É importante destacar que, para as regras da nftables poderem vir a ser aplicadas, o Suricata terá de ser executado em *inline mode*, ou seja, em modo IPS (já que por *default* o mesmo é executado como IDS). Seguem-se os comandos para a execução da *firewall* e Suricata:

```
┌──(root㉿kali)-[/home/kali]
└─# suricata -q 0
```

```
┌──(root㉿kali)-[/home/kali]
└─# nft -f router.nft
```

É ainda necessário ativar o *ipv4_forward* no *router* através do seguinte comando:

```
┌──(root㉿kali)-[/home/kali]
└─# sysctl -w net.ipv4.ip_forward=1
```

Testes

Após a implementação do cenário inicialmente proposto, foram efetuados alguns testes de modo a perceber se toda a comunicação entre as diferentes entidades e serviços estava a ser efetuada como esperado.

Configurações de tráfego de entrada no *router* das *nftables*

1. Pedidos de resolução de nomes aos servidores DNS existentes;

```
┌──(root㉿kali)-[/home/kali]
└─# nslookup alexa.amazon.com 192.168.1.2
Server:         192.168.1.2
Address:        192.168.1.2#53

Non-authoritative answer:
alexa.amazon.com        canonical name = pitangui.amazon.com.
pitangui.amazon.com     canonical name = tp.5fd53c725-frontier.amazon.com.
tp.5fd53c725-frontier.amazon.com        canonical name = d1wg1w6p5q8555.cloudfront.net.
Name:   d1wg1w6p5q8555.cloudfront.net
Address: 13.33.232.191
```

```
┌──(root㉿kali)-[/home/kali]
└─# nslookup alexa.amazon.com 10.10.10.3
Server:         10.10.10.3
Address:        10.10.10.3#53

Non-authoritative answer:
alexa.amazon.com        canonical name = pitangui.amazon.com.
pitangui.amazon.com     canonical name = tp.5fd53c725-frontier.amazon.com.
tp.5fd53c725-frontier.amazon.com        canonical name = d1wg1w6p5q8555.cloudfront.net.
Name:   d1wg1w6p5q8555.cloudfront.net
Address: 13.33.232.191
```

2. Conexões SSH iniciadas pela *VPN-gw*, localizada no DMZ ou por qualquer IP da rede interna. Estas deverão estar protegidas com o mecanismo de *Port Knocking*, onde é solicitado pelo menos 5 *port knocks*.

```
┌──(root㉿kali)-[/home/kali]
└─# nc 10.10.10.1 123 -w 1

┌──(root㉿kali)-[/home/kali]
└─# nc 10.10.10.1 234 -w 1

┌──(root㉿kali)-[/home/kali]
└─# nc 10.10.10.1 345 -w 1

┌──(root㉿kali)-[/home/kali]
└─# nc 10.10.10.1 567 -w 1

┌──(root㉿kali)-[/home/kali]
└─# ssh kali@10.10.10.1
^C

┌──(root㉿kali)-[/home/kali]
└─# nc 10.10.10.1 456 -w 1

┌──(root㉿kali)-[/home/kali]
└─# nc 10.10.10.1 567 -w 1

┌──(root㉿kali)-[/home/kali]
└─# ssh kali@10.10.10.1
kali@10.10.10.1's password:
Linux kali 6.1.0-kali7-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.20-2kali1 (2023-04-18) x86_64

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Apr 30 08:07:14 2023 from 10.10.10.2
┌──(kali㉿kali)-[~]
└─$
```

Configurações de tráfego entre redes das nftables

1. Pedidos de resolução de nomes ao servidor DNS existente na rede *DMZ*;
2. O servidor DNS na rede *DMZ* deve ser capaz de efetuar resolução de nomes a partir do servidor DNS da *Internet*.
3. Os servidores DNS da rede *DMZ* e *Internet* devem ser capazes de sincronizar os conteúdos das zonas DNS

```
root@kali:/home/kali# nslookup alexa.amazon.com 10.10.10.3
Server:         10.10.10.3
Address:        10.10.10.3#53

Non-authoritative answer:
alexa.amazon.com        canonical name = pitangui.amazon.com.
pitangui.amazon.com     canonical name = tp.5fd53c725-frontier.amazon.com.
tp.5fd53c725-frontier.amazon.com        canonical name = d1wg1w6p5q8555.cloudfront.net.
Name:   d1wg1w6p5q8555.cloudfront.net
Address: 13.33.232.191
```

```
┌──(root㉿kali)-[/home/kali]
└─# nslookup alexa.amazon.com 192.168.1.2
Server:         192.168.1.2
Address:        192.168.1.2#53

Non-authoritative answer:
alexa.amazon.com        canonical name = pitangui.amazon.com.
pitangui.amazon.com     canonical name = tp.5fd53c725-frontier.amazon.com.
tp.5fd53c725-frontier.amazon.com        canonical name = d1wg1w6p5q8555.cloudfront.net.
Name:   d1wg1w6p5q8555.cloudfront.net
Address: 13.33.232.191
```

4. Ligações ao servidor *SMTP* da rede *DMZ* pela porta *SMTP*

```
┌──(root㉿kali)-[/home/kali]
└─# nc 10.10.10.5 25
ola
tudo
bem
```

```
┌──(root㉿kali)-[/home/kali]
└─# nc -l 25
ola
tudo
bem
```

Ligações ao servidor Mail da rede *DMZ* pelas portas *POP* e *IMAP*

```
┌──(root㉿kali)-[/home/kali]
└─# nc 10.10.10.4 110
ola
tudo
bem
^[[D^C

┌──(root㉿kali)-[/home/kali]
└─# nc 10.10.10.4 143
ola
tudo
bem
```

```
┌──(root㉿kali)-[/home/kali]
└─# nc -l 110
ola
tudo
bem

┌──(root㉿kali)-[/home/kali]
└─# nc -l 143
ola
tudo
bem
```

Ligações ao servidor *WWW* da rede *DMZ* pelas portas *HTTP* e *HTTPS*

```
┌──(root㉿kali)-[/home/kali]
└─# nc 10.10.10.6 80
ola
tudo
bem
^C

┌──(root㉿kali)-[/home/kali]
└─# nc 10.10.10.6 443
ola
tudo
bem
```

```
┌──(root㉿kali)-[/home/kali]
└─# nc -l 80
ola
tudo
bem

┌──(root㉿kali)-[/home/kali]
└─# nc -l 443
ola
tudo
bem
```

Ligações à *VPN-gw* da rede *DMZ* pela porta *OpenVPN*.

```
┌──(root㉿kali)-[/home/kali]
└─# nc 10.10.10.2 1194
ola
tudo
bem
```

```
┌──(root㉿kali)-[/home/kali]
└─# nc -l 1194
ola
tudo
bem
```

Configurações de tráfego no sentido Internet -> Internal Network das nftables

1. Ligações SSH originadas pela *Internet* em direção à porta 2021 do *router* devem ser redirecionadas para a porta SSH do servidor *datastore*, localizado na *Internal Network*. Foi estabelecido um limite máximo de duas conexões simultâneas.

```
root@kali:/home/kali# ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.2  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 2001:818:ea9d:f700:4c2d:c0f4:d6f3:3854  prefixlen 64  scopeid 0x0<global>
        inet6 2001:818:ea9d:f700:956f:f628:78ab:7e44  prefixlen 64  scopeid 0x0<global>
        inet6 fe80::75b5:a649:fdf9:558d  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:65:9c:d2  txqueuelen 1000  (Ethernet)
        RX packets 30089  bytes 9135094 (9.1 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 82920  bytes 5107983 (5.1 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 247  bytes 31408 (31.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 247  bytes 31408 (31.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@kali:/home/kali# ssh kali@192.168.1.127 -p 2021
kali@192.168.1.127's password:
Linux kali 6.1.0-kali7-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.20-2kali1 (2023-04-18) x86_64

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Apr 30 14:53:43 2023 from 192.168.1.2
```

```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.20.20.6  netmask 255.255.255.0  broadcast 10.20.20.255
        inet6 fe80::e403:6b11:671d:8a38  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:bd:b3:8b  txqueuelen 1000  (Ethernet)
        RX packets 43000  bytes 2592118 (2.4 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 25673  bytes 1558038 (1.4 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 9  bytes 744 (744.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 9  bytes 744 (744.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0


┌──(kali㉿kali)-[~]
└─$
```

2. Qualquer ligação em direção à porta 2022 do router devem ser redirecionadas para a porta 2022 do servidor STI CA, localizado na *Internal Network*. Para o caso, apenas um IP, da *Internet*, é que poderá estabelecer esta ligação, pelo que o IP escolhido foi o do servidor Eden.

```
root@kali:/home/kali# nc 192.168.1.127 2022
ola
tudo
bem
```

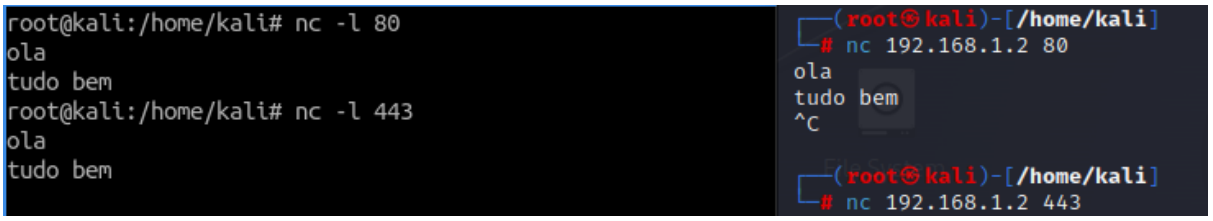
```
┌──(root㉿kali)-[/home/kali]
└─# nc -l 2022
ola
tudo
bem
```

<u>Configurações de tráfego no sentido Internal Network -> Internet das nftables</u>

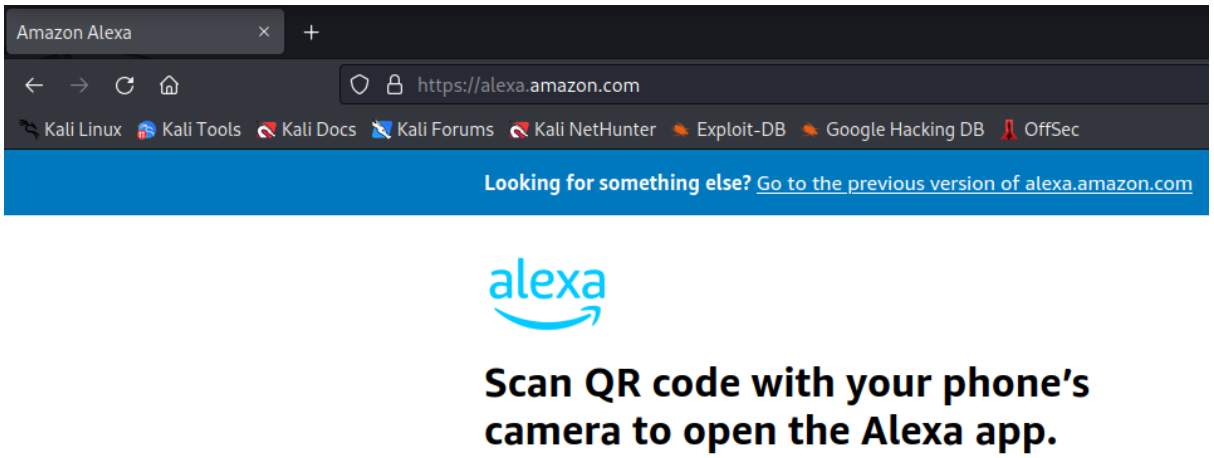
1. Pedidos de resolução de nomes ao servidor DNS existente na Internet;

```
┌──(root㉿kali)-[/home/kali]
└─# nslookup assistant.google.com 192.168.1.2
Server:         192.168.1.2
Address:        192.168.1.2#53

Non-authoritative answer:
Name:   assistant.google.com
Address: 172.217.168.174
Name:   assistant.google.com
Address: 2a00:1450:4003:80a::200e


┌──(root㉿kali)-[/home/kali]
└─# nslookup alexa.amazon.com 192.168.1.2
Server:         192.168.1.2
Address:        192.168.1.2#53

Non-authoritative answer:
alexa.amazon.com        canonical name = pitangui.amazon.com.
pitangui.amazon.com     canonical name = tp.5fd53c725-frontier.amazon.com.
tp.5fd53c725-frontier.amazon.com        canonical name = d1wg1w6p5q8555.cloudfront.net.
Name:   d1wg1w6p5q8555.cloudfront.net
Address: 13.33.232.191
```

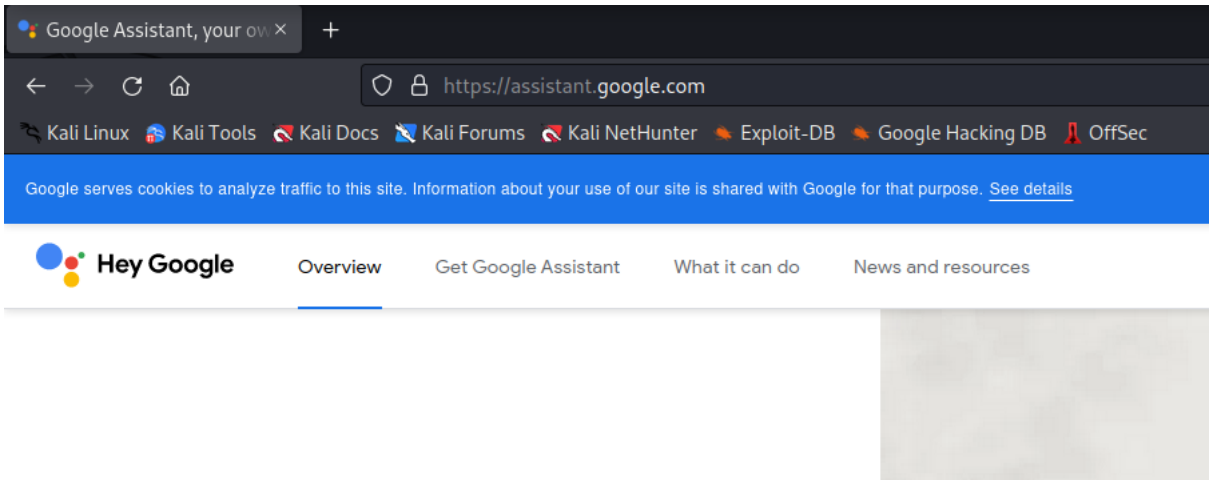
2. Ligações HTTP, HTTPS e SSH originado por dispositivos com IP dinâmico (clientes DHCP).



3. Ligações HTTPS através do dispositivo *IoT Amazon Alexa* ao servidor público *alexa.amazon.com*

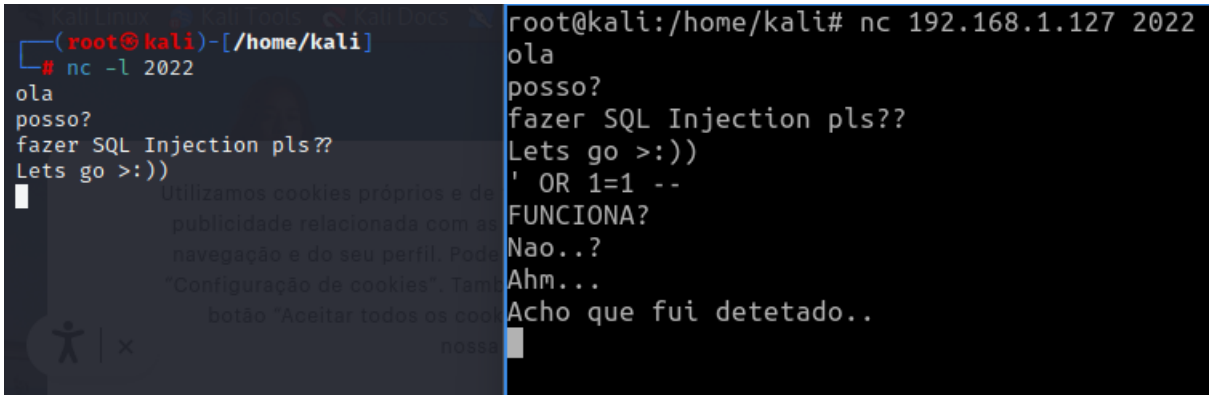


Ligações HTTPS através do dispositivo *IoT Google Assistant* ao servidor público *assistant.google.com*



IDS/IPS - Suricata

1. SQL Injection:



2. DoS:

```
root@kali:/home/kali# hping3 -S --faster -V -p 2022 192.168.1.127
using enp0s3, addr: 192.168.1.3, MTU: 1500
HPING 192.168.1.127 (enp0s3 192.168.1.127): S set, 40 headers + 0 data bytes
len=46 ip=192.168.1.127 ttl=63 DF id=0 tos=0 iplen=44
sport=2022 flags=SA seq=0 win=64240 rtt=27.0 ms
seq=1868862281 ack=1557006006 sum=10a0 urp=0

len=46 ip=192.168.1.127 ttl=63 DF id=0 tos=0 iplen=44
sport=2022 flags=SA seq=1 win=64240 rtt=140.0 ms
seq=4059210529 ack=1818179257 sum=14ce urp=0

len=46 ip=192.168.1.127 ttl=63 DF id=0 tos=0 iplen=44
sport=2022 flags=SA seq=2 win=64240 rtt=201.6 ms
seq=3670706170 ack=383879135 sum=bb04 urp=0

len=46 ip=192.168.1.127 ttl=63 DF id=0 tos=0 iplen=44
sport=2022 flags=SA seq=3 win=64240 rtt=206.9 ms
seq=1275618264 ack=1144722587 sum=88fd urp=0
```

```
04/30/2023-16:22:36.604157  [Drop] [**] [1:99999998:1] Possible SQL Injection with apostrophe and comments [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.1.3:54978 → 10.20.20.3:2022
04/30/2023-16:24:49.599443  [Drop] [**] [1:99999999:1] Possible DoS Attack with SYN scans [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.1.3:4635 → 10.20.20.3:2022
04/30/2023-16:24:54.424178  [Drop] [**] [1:99999999:1] Possible DoS Attack with SYN scans [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.1.3:21192 → 10.20.20.3:2022
04/30/2023-16:24:59.426372  [Drop] [**] [1:99999999:1] Possible DoS Attack with SYN scans [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.1.3:37559 → 10.20.20.3:2022
04/30/2023-16:25:04.428888  [Drop] [**] [1:99999999:1] Possible DoS Attack with SYN scans [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.1.3:54644 → 10.20.20.3:2022
04/30/2023-16:25:09.426308  [Drop] [**] [1:99999999:1] Possible DoS Attack with SYN scans [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.1.3:5602 → 10.20.20.3:2022
```

Conclusão

Através do presente trabalho tornou-se possível uma melhor compreensão do modo de funcionamento das *Firewalls* por meio de *nftables* e da importância que estas apresentam na regulação do tráfego das redes. Ao longo deste projeto foi ainda implementado mecanismos de *Port Knocking,* o que permitiu, neste caso, proporcionar uma maior segurança nas comunicações *SSH* estabelecidas com o *router*. Foi ainda possível aprofundar um pouco melhor o modo de funcionamento de serviços, como servidores *DNS*, e a limitar a resolução de nomes dos mesmos. Por fim, procedeu-se à implementação de um sistema *IDS/IPS* de modo a que este funcionasse em conjunto com a *Firewall*, dando a conhecer, um pouco, o tipo de regras que podem vir a ser integradas no mesmo.

Apesar de todos os imprevistos encontrados, foi possível efetuar a implementação do cenário proposto, ainda que tenham, novamente, surgido alguns problemas relativamente ao *software* utilizado.

Referências

- Slides da disciplina de Segurança em Tecnologias da Informação
- https://suricata.readthedocs.io/en/suricata-6.0.1/setting-up-ipsinline-for-linux.html
- https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Pcre_(Perl_Compatible_Regular_Expressions)
- https://medium.com/@mshulkhan/detection-attack-using-suricata-1-5ea7b2f62551
- https://ravi73079.medium.com/attacks-to-be-performed-using-hping3-packet-crafting-98bc25584745
- https://linux.die.net/man/8/hping3
- https://linuxhint.com/hping3/
- https://wiki.nftables.org/wiki-nftables/index.php/Performing_Network_Address_Translation_(NAT)
- https://wiki.nftables.org/wiki-nftables/index.php/Main_Page