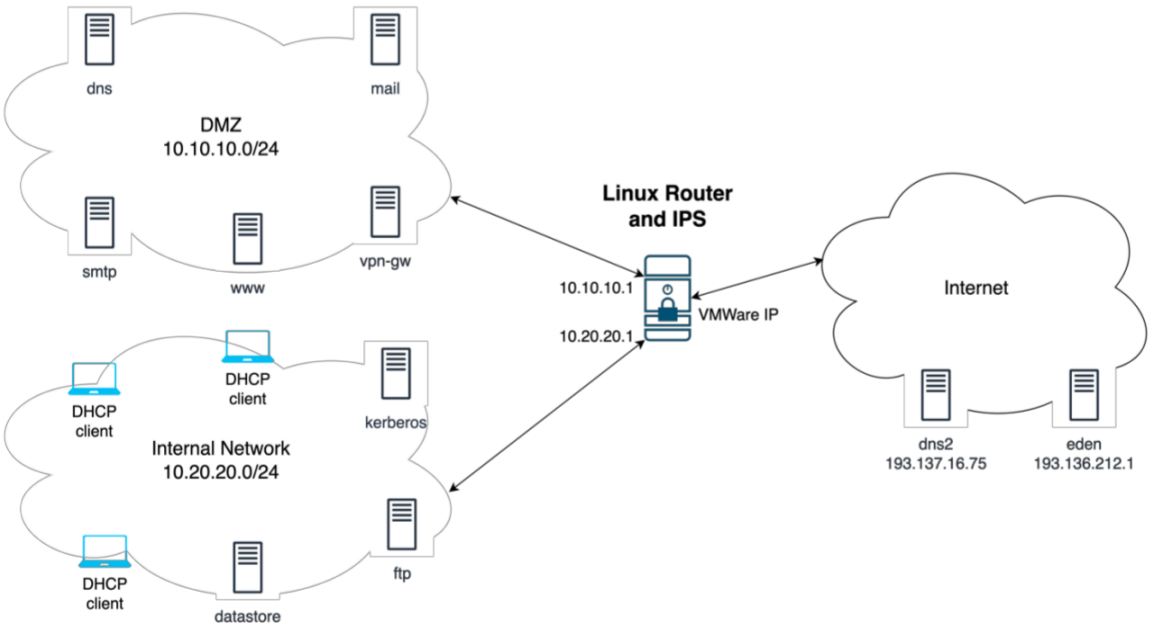
1 2 9 0
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

Segurança em Tecnologias da Informação

Practical Assignment #2



2021/2022

Mestrado em Engenharia Informática

PL2	Joana Brás	2021179983	joanabras@student.dei.uc.pt
PL2	Pedro Rodrigues	2018283166	pedror@student.dei.uc.pt

1. Introdução

O presente relatório aborda uma série de componentes necessários para o segundo trabalho prático da disciplina de Segurança em Tecnologias da Informação. Este tem vários objetivos, sendo os principais:

1. Configuração de um sistema de firewall, usando as funcionalidades disponibilizadas pelo kernel do sistema operativo Linux (`netfilter`), que seja capaz de proteger máquinas posicionadas numa rede protegida (interna) e numa rede que contém serviços públicos (dmz) contra ataques, assegurando o policiamento e filtragem do tráfego que se destina a essas redes. Desta forma pretendemos configurar no sistema operativo funcionalidades de router.
2. Configuração de um sistema de detecção de intrusões (IDS) utilizando a ferramenta Snort, facultando a este router a capacidade de reagir a ataques efetuados por hosts que se encontrem numa rede externa (internet). Os ataques que se pretende que sejam testados no âmbito deste trabalho, pertencem às categorias de: “SQL Injection” (`SQLi`) e “Cross Site Scripting” (`XSS`).

2. Cenário de Rede

Para efeitos de visualização e melhor perceção do cenário que nos foi proposto, construímos este diagrama simplificado que mostra os principais aspetos do trabalho nomeadamente: a implementação do cenário evidenciando as máquinas utilizadas, as redes, os IPs atribuídos às diversas interfaces criadas para o efeito e os serviços que se colocamos a correr em cada uma destas máquinas.

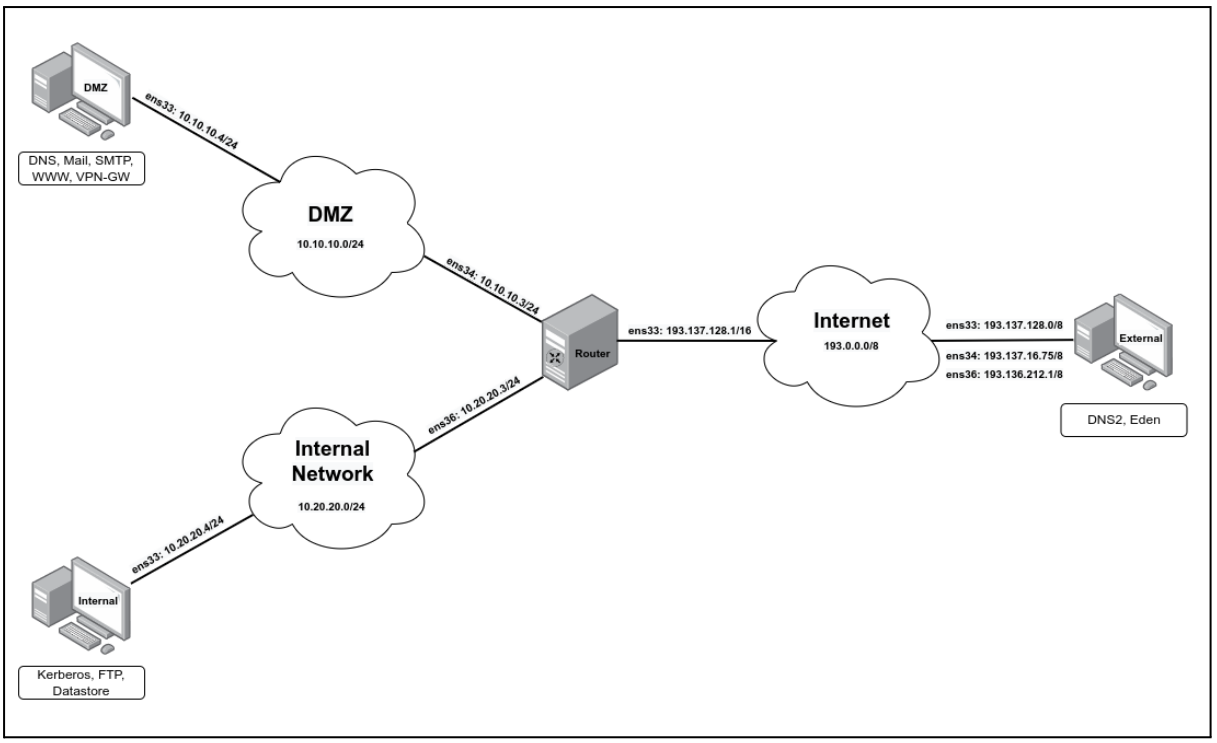


Figura 1 - Cenário de rede

Como podemos observar no diagrama este cenário é então composto por 4 máquinas, simuladas através do software de virtualização "VMware Workstation Pro" e que denominamos de: "Internal", "DMZ", "router" e "External". Também estão presentes no cenário 3 redes distintas: "Internal Network", "DMZ" e "Internet" as quais estas máquinas se vão conectar.

Começando pelas redes presentes no cenário, estas podem ser divididas em duas zonas que denominamos "interna" e "externa".

No caso das redes "DMZ" e "Internal" estas pertencem à zona "interna", Estas, têm uma gama de endereçamento na classe C privada. Assim, não é possível endereçar diretamente a partir da rede "Internet" nenhum dispositivo que esteja numa destas redes, sem que no processo exista algum dispositivo que faça a tradução de endereços públicos em endereços privados (e vice-versa), necessária para que haja comunicação. Por outras palavras é necessário que exista um dispositivo (neste cenário o "router") que tenha a funcionalidade de "NAT" (Network Address Translation) ativa. Para este cenário, a rede "DMZ" foi criada para representar uma onde os serviços publicamente acessíveis da zona "interna" devem estar e a rede "Internal" para albergar os dispositivos que contêm informação mais sensível, daí apresentar maiores restrições na configuração dos filtros para conexões que se podem estabelecer com destino a dispositivos desta rede como será mostrado mais à frente. Em termos de configuração estas redes foram configuradas no "VMware" como sendo redes "Host-Only with no DHCP".

No caso da rede “Internet”, por exclusão de partes, vemos que pertence à zona “externa”, e que tendo um endereçamento público na Classe C foi criada para simular uma conexão à internet. Assim é possível testar ligações a serviços que se encontram na zona “internq” (funcionalidades de NAT) sem grande dificuldade. Em termos de configuração no “VMWare” esta rede foi configurada como uma rede “NAT with no DHCP” a fim de possibilitar a conexão com a internet através de um device criado pelo software de virtualização para o efeito.

Em termos de máquinas, como podemos verificar na figura foram criadas 4 com os seguintes propósitos:

- **Router**: Servir de plataforma onde todo o encaminhamento, filtragem e proteção das ligações é feito, bem como a deteção de intrusões. Este dispositivo tem 3 interfaces para conseguir comunicar com as 3 redes para onde deve encaminhar o tráfego.
- **Internal**: Albergar todos os serviços (Kerberos, FTP e Datastore) que podem correr na “Internal Network” facilitando assim o teste das regras de filtragem de tráfego a ser efetuado pelo “router” firewall não existindo assim necessidade de criar entidades separadas. Esta máquina apenas tem uma interface.
- **Dmz**: Albergar todos os serviços (DNS, Mai, SMTP, WWW, VPN-GW) que podem correr na “DMZ” (Demilitarized Zone) facilitando assim, à semelhança da máquina anterior, o teste das regras de filtragem de tráfego a ser efetuado pelo “router” firewall e não existindo assim necessidade de criar entidades separadas. Esta máquina apenas tem uma interface.
- **Externa:** Esta máquina foi criada apenas por uma questão de facilidade. De facto torna-se muito mais cómodo simular os serviços que estão na internet (Eden e DNS2) visto que se encontram a correr na rede interna no DEI e os endereços facultados no enunciado não serem alcançáveis mesmo ligando-nos por VPN (por motivos que não compreendemos). No caso desta máquina (sendo extra ao requisitado no enunciado) tomamos a liberdade de criar 3 interfaces, a fim de facilitar os testes das ligações com origem nos hosts Eden e DNS2 e a partir de outros hosts.

Por fim, no cenário foi necessário configurar então as regras que permitem ao “router” controlar o tráfego que circula entre as zonas “interna” e “externa”. As regras que foram configuradas por nós, por requisito do enunciado, encontram-se resumidas na tabela seguinte:

ID	Origem	Destino	Serviços	Comunicações
1	router	internet	DNS	diretas

ID	Origem	Destino	Serviços	Comunicações
2	internal, vpn-gw (dmz)	router	SSH	diretas
3	internet, internal	dns (dmz)	DNS	NAT, diretas
4	dns (dmz)	dns2(internet), others (internet)	DNS	NAT
5	internet, internal	smtp (dmz)	SMTP	NAT, diretas
6	internet,internal	mail (dmz)	POP3, POP3s	NAT, diretas
7	internet, internal	mail (dmz)	IMAP IMAPs	NAT, diretas
8	internet, internal	www (dmz)	HTTP	NAT, diretas
9	internet, internal	www (dmz)	HTTPS	NAT, diretas
10	internet, internal	vpn-gw (dmz)	OpenVPN	NAT, diretas
11	vpn-gw (dmz)	datastore (internal)	POSTGRES	diretas
12	vpn-gw (dmz)	kerberos	KERBEROS5	diretas
13	external	ftp (internal)	FTP	NAT
14	eden, dns2	datastore (internal)	SSH	NAT
15	internal	dns2, others (internet)	DNS	NAT
16	internal	eden, others (internet)	HTTP	NAT
17	internal	eden, others (internet)	HTTPS	NAT
18	internal	eden, dns2, other(internet)	SSH	NAT
19	internal	eden, others (internet)	FTP	NAT

3. Instalações

Para a implementação das funcionalidades necessárias fizemos uso do seguinte software:

- IPTables (router)
- IPTables-extensions (router)
- snort, libdaq e outras dependências (router)
- vsftpd (internal e outside)
- ftp (internal e outside)
- netcat

Este foi instalado nas diversas *Virtual Machines* usadas para a realização deste projeto. O software de virtualização utilizado para a sua criação (como referido anteriormente) foi o "VMware".

4. Conceitos gerais

4.1. IPTables

IPTables é uma firewall que analisa todos os pacotes de entrada ou saída do sistema e verifica o enquadramento de cada um às regras impostas. As regras, por sua vez, são encadeadas numa tabela. Existem três tipos de tabelas mas no nosso trabalho usámos apenas duas, sendo elas: "`filter`" e "`NAT`". Para ambos há opções para o roteamento de pacotes. Especificamente no nosso caso, quando usámos a tabela "`filter`", foram aplicadas as seguintes instruções:

- "`INPUT`": Os pacotes são localmente distribuídos.
- "`OUTPUT`": Os pacotes são enviados pela máquina
- "`FORWARD`": Os pacotes não se permanecem muito tempo na máquina, já que quando chegam à mesma, são redirecionados para outro local (quer seja um IP ou um porto).

Já para a tabela "`NAT`", aplicaram-se as seguintes instruções:

- "`PREROUTING`": Em vez de lhe ser atribuído um destino ao pacote através da IPTables, este é enviado para um IP destino. É usado para a extensão "`DNAT`".
- "`POSTROUTING`": Após ter sido escolhido um destino pelas IPTables, em vez de ser enviado logo, ainda passará nas regras definidas com "`POSTROUTING`". Com a extensão "`SNAT`", o IP de origem será modificado.

Por último, falta mencionar as ações das regras na tabela relativamente aos pacotes. Neste trabalho usámos três essenciais:

- "`ACCEPT`": Todos os pacotes são aceites e procedem com a sua rota.

- "`DROP`": Todos os pacotes são rejeitados sem enviar mensagem ao utilizador.
- "`LOG`": Ter acesso ao relatório das regras de cada instrução mencionada anteriormente.

4.2. Snort

O Snort é um sistema de prevenção de intrusões (IPS) open source que usa uma série de regras definidas pelo utilizador que permitem deteção e prevenção de ameaças ao sistema ao enviar alertas e, se conjugado com a firewall IPTables, permite a prevenção de ataques. Tem quatro modos de funcionamento essenciais, sendo estes:

- Como "`sniffer`": Analisa as comunicações e imprime a informação na consola.
- Como "`packet logger`": Analisa as comunicações e guarda o seu conteúdo num ficheiro.
- Como "`NIDS`": Analisa as comunicações com base num conjunto predefinido de regras.
- Em modo "`inline`": Tal como no modo "`NIDS`", também são analisadas várias regras, mas são conjugadas com IPTables.

No nosso trabalho usámos o Snort em modo inline. Neste modo, os pacotes são enviados pelo IPTables para o Snort através de IP Queue de forma a que o Snort os possa analisar. Se algum dos pacotes tiver algum caractere ou expressão presente na uma regra definida, o pacote será descartado e o utilizador receberá uma mensagem, que pode ser personalizada na regra a avisar de uma tentativa de intrusão.

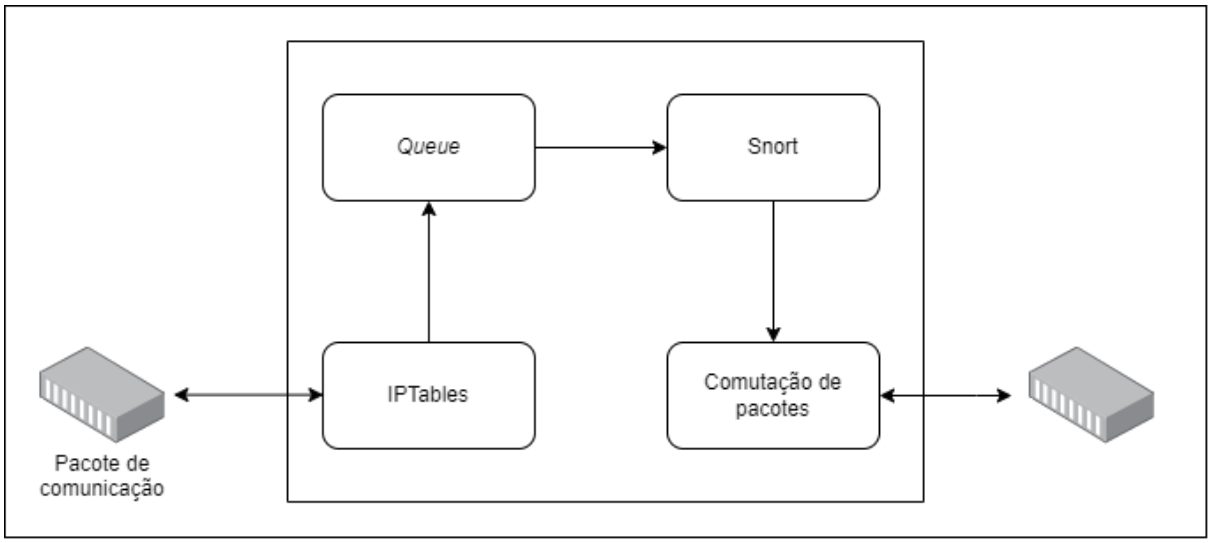


Figura 2 - Funcionamento do Snort em modo inline. (adaptado do livro "Segurança Prática em Sistemas e Redes com Linux, Jorge Granjal")

O diagrama descrito acima pretende ilustrar o funcionamento do Snort em modo “inline”. Como observamos, os pacotes apenas vão para o destino definido nas IPTables a partir do momento em que o Snort os analisa e confirma de que não há uma tentativa de intrusão.

5. Configurações

5.1. Gerais

Como já referido anteriormente 3 redes tiveram de ser criadas para a realização deste trabalho. Para isso recorremos às configurações do software de virtualização e criamos 2 redes “Host-Only with no DHCP” e 1 rede “NAT with no DHCP”. No caso da rede “DMZ” esta foi criada por forma a que as máquinas presentes nessa rede obtivessem um IP da rede 10.10.10.0 com máscara 255.255.255.0 (/24). No caso da rede “Internal Network” a configuração foi semelhante, mantendo-se a mesma máscara mas com o IP 10.20.20.0. No caso da rede “Internet”, as máquinas associadas ficaram com um IP na gama 193.0.0.0 com máscara 255.0.0.0 (/8). A escolha deveu-se à nossa observação de que as máquinas “Eden” e “DNS2” parecerem encaixar neste esquema de endereçamento. No entanto, estamos cientes que a escolha poderia ter sido outra ou até mesmo deixada ao critério do software de virtualização.

Em termos de interfaces criadas em cada uma das máquinas, os IPs foram atribuídos por forma a garantir dentro do possível a maior compatibilidade com o expresso no enunciado. Não foi possível reproduzir totalmente os IPs propostos visto que o software utilizado para simular o ambiente não o permite. De facto, quer numa rede “Host-Only” quer numa rede “NAT”, o “VMware” reserva o primeiro endereço da rede para atribuir à Host Machine e como no caso de rede /24 (como é o caso da “DMZ” e “Internal Network” o primeiro endereço é na forma X.X.X.1, estando o enunciado a pedir que as interfaces do router tenham esse mesmo endereço vai-se gerar uma incompatibilidade que só pode ser resolvida usando outros endereços que não os propostos.

Na figura seguinte é apresentada uma imagem alusiva ao problema, juntamente com um excerto retirado da documentação do software de virtualização que prova a impossibilidade da configuração ser realizada da forma proposta:

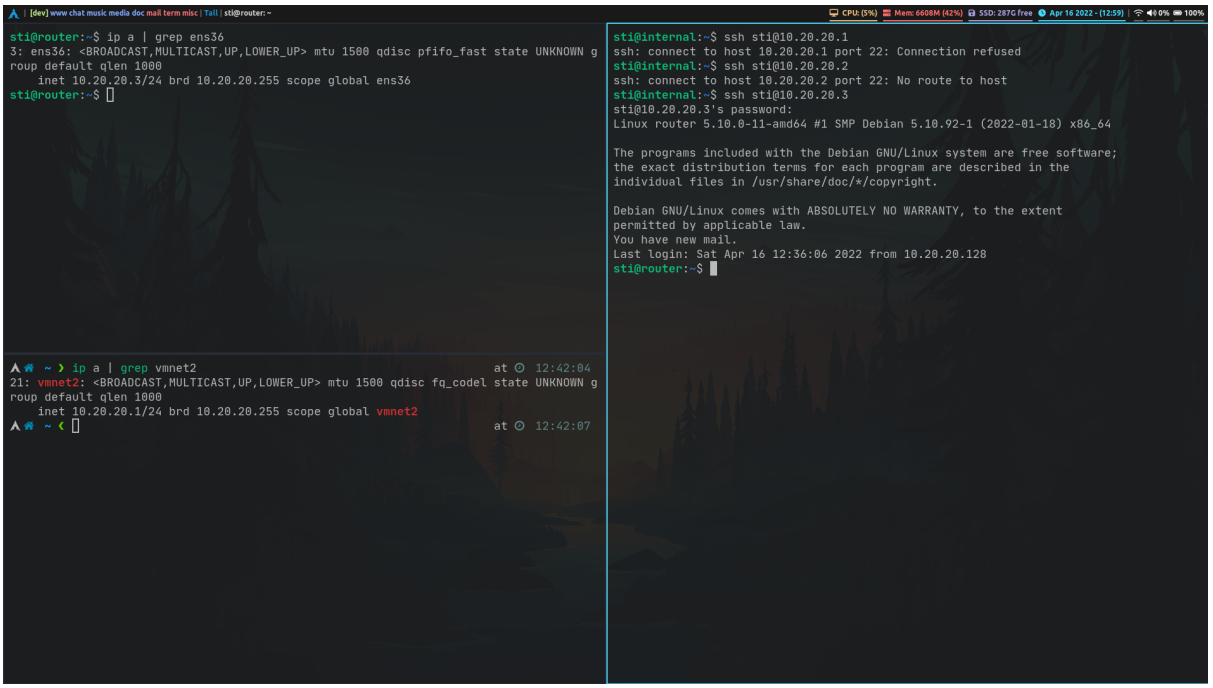


Figura 3 - Prova da impossibilidade da configuração dos IPs conforme o enunciado.

No canto superior esquerdo encontra-se a VM router, onde foi feito um "`grep`" do output do comando "`ip a`" para mostrar a interface do router que está ligada à rede "internal". Como se pode ver foi atribuído estaticamente o IP a esta interface (10.20.20.3 ao contrário do proposto 10.20.20.1 para evidenciar o problema).

No canto inferior esquerdo foi feito o mesmo procedimento, mas desta vez na host machine onde podemos ver que o "VMware" criou um network adapter com o IP 10.20.20.1. O que vai de acordo ao descrito na documentação:

IP Address Use on a Host-Only Network

Range	Address Use	Example
net.1	Host machine	192.168.0.1
net.2–*net*.127	Static addresses	192.168.0.2–192.168.0.127
net.128–*net*.253	DHCP-assigned	192.168.0.128–192.168.0.253
net.254	DHCP server	192.168.0.254
net.255	Broadcasting	192.168.0.255

Figura 4 - Documentação do VMware relativamente à reserva dos IPs da gama X.X.X.1 para a host machine.

No lado direito encontram-se múltiplas tentativas de fazer SSH para os IPs 10.20.20.(1,2,3). Como podemos observar, mesmo o IP 10.20.20.1 não estando a ser atribuído a nenhuma interface das VMs o ssh consegue na mesma encontrar uma route, visto que está de facto a tentar aceder à host machine mas falha por não ter permissões. No caso do ssh para o IP 10.20.20.2 como era espectável nem sequer encontra uma route visto que não temos nenhum ssh service a correr nesse IP. Por fim, no IP 10.20.20.3 finalmente consegue ligar-se à VM router pelo facto de esse ser o IP da interface da mesma, estando o serviço "openssh-server" a correr.

Como foi dito previamente, foram criadas três redes distintas: "`Internal Network`", "`DMZ`" e "`Internet`" em que a "`Internal`" e a "`DMZ`" pertencem à zona que denominamos "`interna`" e a "`Internet`" percebe à zona "`outside`". Todas as ligações são feitas sem DHCP. Dito isto, as configurações da redes são as seguintes:

- A rede "`DMZ`" foi definida na máquina com o mesmo nome. Liga-se à máquina "`DMZ`" pelo IP 10.10.10.4 e ao router pela "`gateway`" com IP 10.10.10.3, como está descrito no cenário de rede. Assim, apenas temos uma interface de rede (tirando o default "`loopback`", descrita no ficheiro "`/etc/networking/interfaces`" que se apresenta posteriormente.

```
auto lo
iface lo inet loopback

auto ens33
iface ens33 inet static
        address 10.10.10.4
        netmask 255.255.255.0
        gateway 10.10.10.3
```

- Da mesma forma foi criada a rede "`Internal Network`", na máquina "`Internal`", em que a rede se liga à máquina pelo IP 10.20.20.3 e ao router pela "`gateway`" 10.20.20.4. Também temos apenas uma interface de rede (excluindo outra vez o "`loopback`") descrita no ficheiro "`interfaces`" na pasta "`/etc/networking`" que se apresenta no seguinte snippet.

```
auto lo
iface lo inet loopback

auto ens33
iface ens33 inet static
        address 10.20.20.4
        netmask 255.255.255.0
        gateway 10.20.20.3
```

- A rede "`Internet`" já é um pouco mais complexa, visto ter uma ligação Eden e uma ligação DNS2. Assim, a interface de redes da máquina "`External`" terá a seguinte configuração de rede:

```
auto lo
iface lo inet loopback

# Interface simulating the dns2 server
auto ens34
iface ens34 inet static
        address 193.137.16.75
        netmask 255.0.0.0

# Interface simulating the eden server
auto ens36
iface ens36 inet static
        address 193.136.212.1
        netmask 255.0.0.0
```

Para haver conexão entre as diferentes máquinas, o "`router`" terá de reconhecer todas as redes para fazer o reencaminhamento de pacotes através da rede. Como tal, o ficheiro de configuração de rede do "`router`" terá de uma interface para cada uma das redes à qual se liga, como é mostrado a seguir.

```
auto lo
iface lo inet loopback

# The interface that connects the router
# to the Internet
auto ens33
iface ens33 inet static
        address 193.137.128.1
        netmask 255.255.255.0

# The interface that connects the router
# to the DMZ
auto ens34
iface ens34 inet static
        address 10.10.10.3
        netmask 255.255.255.0

# The interface that connects the router
# to the Internal Network
auto ens36
iface ens36 inet static
        address 10.20.20.3
        netmask 255.255.255.0
```

De forma a facilitar as ligações por FTP, foi adicionado no ficheiro "`/etc/hosts`" em cada uma das máquinas um alias para os IPs que foram usados. Assim, em vez de correr, por exemplo, a linha de comando "`ftp 10.10.10.3`" para aceder da máquina "`DMZ`" para o "`router`", escreve-se simplesmente "`ftp router`".

Como foi mencionado anteriormente, no ponto do **Cenário de Rede**, na máquina externa temos vários IPs, correspondentes aos diferentes serviços proporcionados pela ligação de outras máquinas à externa, a fim de determinar exatamente a qual serviço se está a aceder e para prevenir ligações por outros meios que não FTP.
Assim, na máquina "`DMZ`", que se liga ao router, à máquina "`internal`" e "`external`" encontra-se o seguinte "`/etc/hosts`":

```
127.0.0.1     localhost
127.0.1.1     debian
10.10.10.3    router
10.20.20.4    internal      datastore     kerberos      ftp
193.137.16.75       dns2
193.136.212.1       eden
```

A máquina "`Internal`", que se liga ao router, à máquina "`DMZ`" e todos os seus serviços associados, e "`external`" tem o seguinte "`/etc/hosts`":

```
127.0.0.1     localhost
127.0.1.1     debian
10.20.20.3    router
10.10.10.4    dmz   dns   smtp   mail   www   vpn-gw
193.137.16.75       dns2
193.136.212.1       eden
```

Na máquina "`router`", como apenas queremos enviar pacotes de máquina para máquina, apenas precisamos mencionar o nome das máquinas que se conectam a si e não os serviços associados a cada um. Como tal, encontra-se o seguinte "`/etc/hosts`":

```
127.0.0.1     localhost
127.0.1.1     debian
10.10.10.4    dmz
10.20.20.4    internal
193.137.0.1   host-machine
193.137.16.75       dns2
193.136.212.1       eden
```

Por último, a máquina "`External`", apenas deve reconhecer o IP do router, já que os IPs das redes interna e DMZ são privados. Como tal, e pela IPTables do "`router`" que será abordada posteriormente, os IPs privados serão substituídos pelo IP do router. Assim, o seu ficheiro "`/etc/hosts`" é o seguinte:

```
127.0.0.1     localhost
127.0.1.1     debian
193.137.128.1       router
193.137.16.75       dns2
193.136.212.1       eden
```

Foi necessário fazer uma última configuração, relativamente ao uso do FTP, nomeadamente ativar os módulos desse serviço. Para isso corremos as seguintes linhas no terminal:

```
sudo modprobe ip_conntrack_ftp
sudo modprobe ip_nat_ftp
```

Como foi explicado no ponto **4.2. Snort**, aos pacotes é atribuído um destino a partir das IPTables. Posteriormente, estes são postos numa lista de espera ("`queue`") ao qual o Snort acede e analisa os pacotes. Se nenhum corresponder às regras declaradas no Snort, os pacotes procedem para o seu destino. O funcionamento do Snort em modo "inline" impossibilita os pacotes de comunicação procederem para o destino sem antes serem analisados pelo Snort. Assim, torna-se crucial ativar o módulo correspondente à lista de espera ao qual o Snort acede. O seguinte snippet de código permite isso mesmo.

```
sudo modprobe nfnetlink_queue
```

Finalmente, de forma a se poder usar o fowarding do IPTables, foi necessário colocar o parâmetro "`ip_forward`" no ficheiro cuja diretoria é "`/proc/sys/net/ipv4/`" com o valor "`1`".

5.2. IPTables

Como foi descrito anteriormente, o IPTables é uma firewall que define o percurso dos pacotes que entram e saem da máquina. Para isso são criadas uma série de regras.
De um modo geral, as regras de "`INPUT`" foram emparelhadas com uma regra de "`OUTPUT`", para permitir a transferência de pacotes para os dois lados. Por questões de comodidade, foram criadas duas regras "`FORWARD`" (descritas no snippet de código abaixo) ao qual foram adicionados os parâmetros "`-m state --state RELATED,ESTABLISHED`" que aceitam qualquer conexão de uma regra "`FORWARD`" desde que a ligação no sentido oposto tenha sido estabelecida, isto é, se tivermos uma ligação `A->B->C`, em que B tem uma regra "`FORWARD`" de A para C, a ligação de C para A, passando por B, pode ser estabelecida ao usar as regras implementadas e evidenciadas no snippet de código abaixo.

```
sudo iptables -A FORWARD -p tcp -m state \
   --state RELATED,ESTABLISHED -j ACCEPT

sudo iptables -A FORWARD -p udp -m state \
   --state RELATED,ESTABLISHED -j ACCEPT
```

Estamos cientes de que os parâmetros descritos anteriormente podiam ter aplicados às regras de "`INPUT`" e de "`OUTPUT`", no entanto queríamos ter descritas as duas possibilidades (com e sem o uso desses parâmetros).

Na tabela presente no ponto **2. Cenário de Rede**, achámos que seria mais claro ter na coluna "**Comunicações**" a distinção de entre ligações que denominámos "diretas" das ligações que requeriam o uso da extensão "NAT". Isto reflete-se também nas regras das IPTables, já que as ligações "diretas" fazem uso apenas da tabela "`filter`" e as ligações "NAT" têm regras em ambas as tabelas que usámos ("`filter`" e "`NAT`"). Posto isto, as ligações "NAT" irão sempre ter uma regra na tabela "`NAT`", que emparelha com uma regra na tabela "`filter`" como será descrito posteriormente.

No ponto **4.1. IPTables** é mencionado que fizemos uso de três ações distintas: "`ACCEPT`", "`DROP`" e "`LOG`".
Declarámos cinco regras referentes à ação "`LOG`". Este adiciona a um ficheiro chamado "`iptables.log`" a informação referente aos pacotes que são rejeitados para cada tipo de instrução que usámos: "`INPUT`", "`OUTPUT`", "`FORWARD`", "`PREROUTING`" e "`POSTROUTING`". Para isso corre-se o seguinte código:

```
echo "kern.warning /var/log/iptables.log" | sudo tee -a /etc/rsyslog.conf
```

As regras referentes à ação "`LOG`", que se encontram abaixo descritas, estão colocadas após todas as regras de "`ACCEPT`", precedendo apenas a regra "`DROP`".

```
#Tabela INPUT
sudo iptables -A INPUT -j LOG -m limit --limit 5/min\
     --log-level 4 --log-prefix 'IP INPUT DROP: '

#Tabela OUTPUT
sudo iptables -A OUTPUT -j LOG -m limit --limit 5/min\
    --log-level 4 --log-prefix 'IP OUTPUT DROP: '

#Tabela FORWARD
sudo iptables -A FORWARD -j LOG -m limit --limit 5/min\
    --log-level 4 --log-prefix 'IP FORWARD DROP: '

#Tabela POSTROUTING
sudo iptables -t nat -A POSTROUTING -j LOG -m limit --limit 5/min\
     --log-level 4 --log-prefix 'POSTROUTING EVENT: '

#Tabela PREROUTING
sudo iptables -t nat -A PREROUTING -j LOG -m limit --limit 5/min \
    --log-level 4 --log-prefix 'PREROUTING EVENT: '
```

O "`DROP`", por sua vez, foi declarado numa regra para cada tipo de instrução da tabela "`filter`" ("`INPUT`", "`OUTPUT`" e "`FORWARD`"), colocadas estrategicamente após declaradas todas as regras desse tipo. O IPTables é, no fundo, uma lista de regras encadeadas e sempre que um pacote entra ou sai do sistema, conforme o serviço que representa (por exemplo ssh, open-vpn, etc), vai percorrer a lista por ordem até encontrar a primeira à qual corresponde. Se a regra "`DROP`" estiver no início, pela forma como descrevemos as regras (snippet abaixo), como corresponderá à primeira - "`DROP`" - o pacote será rejeitado, independentemente de

haver uma regra posterior à qual também corresponde ou não. Assim, como essas três regras se encontram no fim da tabela, impossibilitam qualquer tipo de ligação que não tenha sido declarada anteriormente. Isto salvaguarda de qualquer tentativa de intrusão no sistema por outros portos que não os declarados previamente pelas regras do IPTables. Essas três regras encontram-se no snippet de código a seguir.

```
#Tabela INPUT
sudo iptables -P INPUT DROP

#Tabela OUTPUT
sudo iptables -P OUTPUT DROP

#Tabela FORWARD
sudo iptables -P FORWARD DROP
```

Por questões de facilidade foi adicionada uma coluna na tabela presente no ponto **2. Cenário de Rede** para identificação de cada uma das transferências de pacotes que foram propostas para implementação. Além disso, para todas as regras foram usados o nome dos portos e não os seus números, de forma a ser mais intuitivo a que serviço a regra se destina.

5.2.1. Ligação 1 (DNS):

A ligação é feita entre o router e a internet. Ambas têm uma interface para a mesma rede, descrita anteriormente, chamada "`ens33`" no "`router`" e "`ens34`" na máquina "`Internet`". Como tal, é uma ligação direta, em que apenas é necessária a declaração da abertura do porto 53 (com alias "`domain`") para permitir que o serviço DNS passe para a máquina "`Internet`". Como foi referido anteriormente, declara-se que se pretende que os pacotes sejam localmente distribuídos, coloca-se na regra a referência a ser um "`INPUT`", com ambos os protocolos "`udp`" e "`tcp`", no porto 53, e a ação é aceitar os pacotes. Da mesma forma que os pacotes são recebidos pelo router, também são enviados por este, no mesmo porto. Daí estar emparelhado com uma regra "`OUTPUT`", que em tudo é igual ao seu par.

```
sudo iptables -A INPUT -p udp --dport domain -j ACCEPT
sudo iptables -A INPUT -p tcp --dport domain -j ACCEPT

sudo iptables -A OUTPUT -p udp --dport domain -j ACCEPT
sudo iptables -A OUTPUT -p tcp --dport domain -j ACCEPT
```

5.2.2. Ligação 2 (SSH):

Da mesma forma que a ligação anterior foi feita, a ligação por ssh terá características semelhantes à ligação 1. Igualmente como no ponto anterior, a ligação é feita diretamente do router para a máquina "`DMZ`" e "`Internal`", pelo router se encontrar ligado à mesma rede que ambas as máquinas. Neste caso, em vez de se declarar o porto que iria fazer a ligação, optou-se por pôr qual o IP da

rede "DMZ". Já para a ligação com a máquina "Internal", optou-se por pôr a interface de rede que liga o "router" a esta última máquina referida, através do parâmetro "-i". Mais uma vez usa-se uma expressão em vez do número do porto. "ssh" é um alias para o porto 22. Também temos neste caso, semelhante ao anterior, o emparelhamento de uma regra "INPUT" com uma regra de "OUTPUT" para permitir a transferência de pacotes em ambos os sentidos da ligação.

```
sudo iptables -A INPUT -p tcp -s 10.10.10.4 --dport ssh -j ACCEPT
sudo iptables -A INPUT -p tcp -i ens36 --dport ssh -j ACCEPT

sudo iptables  -A OUTPUT -p tcp -d 10.10.10.4 --dport ssh -j ACCEPT
sudo iptables -A OUTPUT -p tcp -p tcp -o ens36 --dport ssh -j ACCEPT
```

5.2.3. Ligação 3 (DNS):

Na ligação 3 vemos pela primeira vez a necessidade de uma ligação "NAT". Como foi mencionado previamente, a máquina "External" não reconhece os IPs privados dos clientes que lhe acedem. Num cenário real, quando uma pessoa faz o acesso à Internet, o seu IP privado nunca é acedido. Em vez disso, o router que temos em casa transforma o nosso IP privado num IP que pode ser acedido pela Internet. É essa mesma situação que se pretende simular com esta ligação "NAT". A ligação 3 tem dois pontos de origem, sendo que o primeiro é originado na máquina "External" (que simula a Internet) e o segundo na máquina "Internal". Ambas as ligações são intermediadas pelo "router". A principal diferença reside na necessidade ou não de estabelecer uma regra pertencente à tabela "NAT". A grande semelhança é que ambas têm uma regra correspondente ao forwarding dos pacotes de um IP para outro.
A ligação originada na máquina "Internal" e com destino à máquina "DMZ" é mais simples. Contrariamente às ligações anteriores, em que os pacotes eram transferidos e mantidos no "router" ou que tinham origem no "router" para outra máquina, neste caso os pacotes são temporários nessa máquina, apenas se mantendo na mesma desde o momento em que os pacotes são transferidos de uma máquina para o "router" e reencaminhados para outra máquina diferente. É o que acontece com esta mesma ligação. Assim sendo, apenas precisamos de duas regras, sendo que uma foi referida anteriormente (com os parâmetros "-m state --state RELATED,ESTABLISHED") e as seguintes:

```
sudo iptables -A FORWARD -p tcp -d 10.10.10.4 -o ens34 \
    --dport domain -j ACCEPT

sudo iptables -A FORWARD -p udp -d 10.10.10.4 -o ens34 \
    --dport domain -j ACCEPT
```

Estas regras destinam-se ao IP da rede "DMZ", pela interface "ens34" (correspondente à rede "DMZ") e pelo port 53 (com alias "domain").

Passando à ligação originada na máquina "External", esta requer duas regras adicionais, correspondente às regras "NAT". Temos uma regra da tabela "filter", que tem como destino o IP da rede "DMZ", na interface "ens34" e origem no IP "193.137.16.75", cuja interface é "ens33". O port usado é novamente o 53.

```
sudo iptables -A FORWARD -p tcp --dport domain \
    -d 10.10.10.4 -o ens34 -s 193.137.16.75 -i ens33 -j ACCEPT
```

Como foi descrito previamente, os IPs privados têm de ser escondidos antes de poder aceder à internet, tanto por razões de segurança e privacidade, como por questões de reconhecimento (a Internet não conhece IPs privados). Como os IPs são substituídos pelo IP do router antes de aceder à Internet, é preciso que o "router" tenha capacidade de fazer o oposto, isto é, quando recebe um pacote da Internet, tem de poder redirecionar esse pacote para a máquina ao qual se destina. Para isso temos as regras "DNAT", que transformam o IP público no IP privado da máquina ao qual está associado. Só depois dessa troca ser feita é que o pacote percorre as regras do IPTables para saber exatamente para que máquina é encaminhado. Assim, as regras correspondentes ao "DNAT" encontram-se no snippet seguinte. As seguintes regras definem que se um pacote "tcp" ou "udp" provenientes da interface "ens33" com destino ao IP "193.137.128.1" no porto 53 (alias "domain"), o seu destino será a máquina ligada à rede "DMZ" (IP "10.10.10.4").

```
sudo iptables -t nat -A PREROUTING -p tcp -i ens33 -d 193.137.128.1 \
    --dport domain -j DNAT --to-destination 10.10.10.4

sudo iptables -t nat -A PREROUTING -p udp -i ens33 -d 193.137.128.1 \
    --dport domain -j DNAT --to-destination 10.10.10.4
```

5.2.4 Ligação 4 (DNS):

A ligação 4 refere-se à ligação inversa da descrita anteriormente. Dentro do "router", o encaminhamento de pacotes já está a ser feito pela regra com os parâmetros "-m state --state RELATED,ESTABLISHED". Assim sendo, falta apenas abordar as regras "SNAT". Foi descrito na ligação 3 que para uma máquina aceder à Internet, terá de ter um IP público a cobrir o seu IP privado. Esse IP público será o IP do "router". Antes de se mudar o IP da source do pacote, este passa pela IPTables. Imediatamente antes de ser enviado, as regras "SNAT" são cumpridas. O que o seguinte snippet de código indica é que as ligações "tcp" ou "udp" enviadas pelo source IP "10.10.10.4", na interface "ens33", pelo port 53 terão o seu IP de origem transformado do IP do "router" ("193.137.128.1").

```
sudo iptables -t nat -A POSTROUTING -p tcp -o ens33 -s 10.10.10.4 \
   --dport domain -j SNAT --to-source 193.137.128.1

sudo iptables -t nat -A POSTROUTING -p udp -o ens33 -s 10.10.10.4 \
   --dport domain -j SNAT --to-source 193.137.128.1
```

5.2.5. Ligação 5 (SMTP):

A ligação 5 torna-se semelhante à ligação 3. A maior diferença é no porto em que a ligação se dá. Também temos duas origens diferentes, em que uma requer o uso de tabelas "NAT" e outra não.
Assim sendo a ligação SMTP é feita da seguinte forma:

```
sudo iptables -A FORWARD -p tcp -d 10.10.10.4 -o ens34 \
    --dport smtp -j ACCEPT
```

Esta regra significa que todos os pacotes recebidos referentes ao serviço SMTP são encaminhados para o IP "10.10.10.4" - rede "DMZ". Devido à regra que estabelece ligações no sentido oposto ao estabelecido anteriormente (regra dos parâmetros "-m state --state RELATED,ESTABLISHED"), qualquer que seja a origem do pacote SMTP, a ligação da máquina "DMZ" para essa máquina fica estabelecida.
Mais uma vez o número do porto foi substituído pelo seu alias. O número que ao qual o alias "SMTP" corresponde é o 25.
A regra "NAT" presente, já que a regra de "FORWARD" está no sentido "Internal"/"Internet" -> "DMZ", tem de também corresponder a esta ligação. Como tal, o pacote vai ter o IP do "router" como destino. É a regra "DNAT" que transforma o IP do "router" no IP da máquina "DMZ".

```
sudo iptables -t nat -A PREROUTING -p tcp -i ens33 -d 193.137.128.1 \
    --dport smtp -j DNAT --to-destination 10.10.10.4
```

É de notar que a partir do momento em que esta ligação é feita, o sentido oposto também passa a ser executável. No entanto, ao contrário do expectável, não é necessário criar uma regra "SNAT" que seja semelhante à regra descrita. Isto deve-se às próprias características do Linux, que quando recebe ou envia um pacote com endereço alterado, consegue operar no sentido oposto e alterar ou ler o endereço de forma automática.

5.2.6. Ligação 6 (POP3, POP3s):

A ligação 6 é bastante semelhante à ligação 5, mas aplicada aos serviços POP3 e POP3s, sendo que por isso muda o porto (neste caso 110 e 995 respetivamente), com destino à máquina "DMZ".

```
sudo iptables -A FORWARD -p tcp -d 10.10.10.4 -o ens34 \
        --dport pop3 -j ACCEPT

sudo iptables -A FORWARD -p tcp -d 10.10.10.4 -o ens34 \
        --dport pop3s -j ACCEPT
```

Da mesma forma que anteriormente só foi necessário implementar a regra num sentido, neste caso é semelhante. Assim declarámos as regras "DNAT" para os dois serviços descritos anteriormente.

```
sudo iptables -t nat -A PREROUTING -p tcp -i ens33 -d 193.137.128.1 \
        --dport pop3 -j DNAT --to-destination 10.10.10.4

sudo iptables -t nat -A PREROUTING -p tcp -i ens33 -d 193.137.128.1 \
        --dport pop3s -j DNAT --to-destination 10.10.10.4
```

5.2.7. Ligação 7 (IMAP):

A ligação 7 é também bastante semelhante à ligação 5, mas aplicada aos serviços IMAP e IMAPs, sendo que por isso muda o porto (neste caso 143 e 993 respetivamente), com destino à máquina "DMZ".

```
sudo iptables -A FORWARD -p tcp -d 10.10.10.4 -o ens34 \
        --dport imap -j ACCEPT

sudo iptables -A FORWARD -p tcp -d 10.10.10.4 -o ens34 \
        --dport imaps -j ACCEPT
```

Da mesma forma que anteriormente só foi necessário implementar a regra num sentido, neste caso é semelhante. Assim declarámos as regras "DNAT" para os dois serviços descritos anteriormente.

```
sudo iptables -t nat -A PREROUTING -p tcp -i ens33 -d 193.137.128.1 \
        --dport imap -j DNAT --to-destination 10.10.10.4

sudo iptables -t nat -A PREROUTING -p tcp -i ens33 -d 193.137.128.1 \
        --dport imaps -j DNAT --to-destination 10.10.10.4
```

5.2.8. Ligação 8 (HTTP):

A ligação 8 tem como destino a máquina "DMZ", com pacotes provenientes da máquina "External". O alias "http" corresponde ao porto 80. A regra em si tem vários parâmetros em comum com as regras previamente mostradas.

```
sudo iptables -A FORWARD -p tcp -d 10.10.10.4 -o ens34 \
    --dport http -j ACCEPT
```

Da mesma forma que anteriormente só foi necessário implementar a regra num sentido, neste caso é semelhante. Assim declarámos a regra "DNAT" para o serviço HTTP.

```
sudo iptables -t nat -A PREROUTING -p tcp -i ens33 -d 193.137.128.1 \
        --dport http-j DNAT --to-destination 10.10.10.4
```

5.2.9. Ligação 9 (HTTPS):

A ligação 9 é também bastante semelhante à ligação 5, mas aplicada ao serviço HTTPS, com destino à máquina "DMZ". O HTTPS (porto 443) pode funcionar com o protocolo "tcp" ou "udp", que estão descritos nas duas regras a seguir:

```
sudo iptables -A FORWARD -p udp -d 10.10.10.4 -o ens34 \
        --dport https -j ACCEPT

sudo iptables -A FORWARD -p tcp -d 10.10.10.4 -o ens34 \
        --dport https -j ACCEPT
```

Da mesma forma que anteriormente só foi necessário implementar a regra num sentido, neste caso é semelhante. Assim declarámos as regras "DNAT" usando o protocolo "udp" e "tcp".

```
sudo iptables -t nat -A PREROUTING -p udp -i ens33 -d 193.137.128.1 \
        --dport https -j DNAT --to-destination 10.10.10.4

sudo iptables -t nat -A PREROUTING -p tcp -i ens33 -d 193.137.128.1 \
        --dport https -j DNAT --to-destination 10.10.10.4
```

5.2.10. Ligação 10 (OpenVPN):

A ligação 10 tem como destino a máquina "DMZ". O serviço funciona em ambos os protocolos "udp" e "tcp", que têm regras associadas a cada um destes.

```
sudo iptables -A FORWARD -p udp -d 10.10.10.4 -o ens34 \
        --dport openvpn -j ACCEPT

sudo iptables -A FORWARD -p tcp -d 10.10.10.4 -o ens34 \
        --dport openvpn -j ACCEPT
```

Da mesma forma que anteriormente só foi necessário implementar a regra num sentido, neste caso é semelhante. Assim declarámos as regras "DNAT" usando o protocolo "udp" e "tcp".

```
sudo iptables -t nat -A PREROUTING -p udp -i ens33 -d 193.137.128.1 \
        --dport openvpn -j DNAT --to-destination 10.10.10.4

sudo iptables -t nat -A PREROUTING -p tcp -i ens33 -d 193.137.128.1 \
        --dport openvpn -j DNAT --to-destination 10.10.10.4
```

5.2.11. Ligação 11 (POSTGRES):

A ligação 11 é a primeira ligação que criámos cujo destino é a máquina "Internal". No entanto, as regras não diferem muito das estabelecidas anteriormente. As diferenças residem nos serviços providenciados pela máquina, a interface de contacto com o "router" e o IP de destino.

Esta ligação é feita sem necessidade de recorrer à tabela "NAT". Isto porque não há a necessidade de esconder os IPs das máquinas, por estas fazerem parte da zona "interna". Assim, apenas temos uma regra de forwarding entre as interfaces das redes às quais as máquinas "DMZ" e "Internal" estão ligadas, no porto 5432:

```
sudo iptables -A FORWARD -p tcp -d 10.20.20.4 -o ens36 \
    -s 10.10.10.4 -i ens34 --dport postgres -j ACCEPT
```

5.2.12. Ligação 12 (KERBEROS5):

Da mesma forma que a anterior, este serviço funciona entre as duas máquinas da zona "interna". Assim, na regra, a única mudança é o porto (88) e a adição de um limite de conexões simultâneas (é permitido 10 conexões simultâneas). Além disso, este serviço funciona em ambos os protocolos "udp" e "tcp".

```
sudo  iptables -A FORWARD -p tcp -m connlimit --connlimit-upto 10 \
    -d 10.20.20.4 -o ens36 -s 10.10.10.4 -i ens34 \
    --dport kerberos5 -j ACCEPT

sudo  iptables -A FORWARD -p tcp -m connlimit --connlimit-upto 10 \
    -d 10.20.20.4 -o ens36 -s 10.10.10.4 -i ens34 \
    --dport kerberos5 -j ACCEPT
```

5.2.13. Ligação 13 (FTP):

O serviço FTP tem dois modos de funcionamento: ativo e passivo. O modo vai determinar como a conexão de dados é estabelecida. No modo ativo, o cliente envia para o servidor o endereço IP e o número da porta na qual ele irá ouvir e então o servidor inicia a conexão. O modo passivo ocorre quando o cliente não tem contacto direto com o servidor. Ambos os modos foram implementados com duas regras. A conexão por FTP pode ser realizada por dois portos: o porto com alias "ftp-data" (20) e "ftp" (21). Ambos estão declarados nas regras seguintes:

```
sudo iptables -A FORWARD -p tcp -i ens33 \
    -d 10.20.20.4 -o ens36 --dport ftp -j ACCEPT

sudo iptables -A FORWARD -p tcp -s 10.20.20.4 -i ens36 \
    -o ens33 --dport ftp-data -j ACCEPT
```

Estamos cientes que numa ligação passiva o cliente liga-se ao servidor usando portos dinâmicos por isso cabe ao sistema de connection tracking to IPTables que ligações deixar passar.

Esta ligação conecta a máquina "Internal" com destino a "External". Como tal, é preciso haver uma regra "NAT" associada, Como a ligação é feita do exterior para a rede interna, é preciso ter uma regra "PREROUTING" que permita a tradução do IP do router para o IP da máquina "Internal". Essa mesma está definidas posteriormente:

```
${SUDO} iptables -t nat -A PREROUTING -p tcp -i ens33 -d 193.137.128.1 \
   --dport ftp -j DNAT --to-destination 10.20.20.4
```

5.2.14. Ligação 14 (SSH):

Para esta ligação foi feito um emparelhamento de uma regra "INPUT" com uma regra "OUTPUT". O porto SSH, como foi descrito anteriormente, é o número 22.

Nestas regras foi usada a interface de rede correspondente à ligação com a máquina "Internal", como pode ser descrita no snippet de código abaixo.

```
sudo iptables -A INPUT -p tcp -i ens36 --dport ssh -j ACCEPT
sudo iptables -A OUTPUT -p tcp --dport ssh -o ens36 -j ACCEPT
```

Já que se trata de uma conexão entre zonas diferentes, há necessidade de alterar o IP privado da máquina "Internal" para o IP do "router". Para reverter essa mudança, usa-se uma regra "PREROUTING" para quando os pacotes são provenientes da máquina "External" para a "Internal".

```
sudo iptables -t nat -A PREROUTING -p tcp -i ens33 -d 193.137.128.1 \
    --dport ssh -j DNAT --to-destination 10.20.20.20.4
```

5.2.15. Ligação 15 (DNS):

Esta é uma ligação em que os pacotes são provenientes da máquina "Internal" e têm destino na máquina "External". O serviço DNS permite uso do protocolo "udp" e "tcp", por isso estão descritas duas regras para diferenciar essas duas opções. Sendo assim, as regras foram definidas da seguinte forma:

```
sudo iptables -A FORWARD -p tcp -s 10.20.20.4 -i ens36 \
    -o ens33 --dport domain -j ACCEPT

sudo iptables -A FORWARD -p udp -s 10.20.20.4 -i ens36 \
    -o ens33 --dport domain -j ACCEPT
```

Já que se trata de uma conexão entre zonas diferentes, há necessidade de alterar o IP privado da máquina "Internal" para o IP do "router" para tal, usa-se uma regra "POSTROUTING" para quando os pacotes são provenientes da máquina "Internal" para a "External".

```
sudo iptables -t nat -A POSTROUTING -p tcp -o ens33 -s 10.20.20.4 \
      --dport domain -j SNAT --to-source 193.137.128.1

sudo iptables -t nat -A POSTROUTING -p udp -o ens33 -s 10.20.20.4 \
      --dport domain -j SNAT --to-source 193.137.128.1
```

5.2.16. Ligação 16 (HTTP):

A ligação 16 tem como destino a máquina "Externa", com pacotes provenientes da máquina "Interna". O alias "http" corresponde ao porto 80. A regra em si tem vários parâmetros em comum com as regras previamente mostradas.

```
sudo iptables -A FORWARD -p tcp -d 10.20.20.4 -o ens36 \
    --dport http -j ACCEPT
```

Mais uma vez, como envolve a rede "externa", é necessário implementar uma regra para mudar o IP privado da máquina para o IP do "router". Assim declarámos a regra "SNAT" para o serviço HTTP.

```
sudo iptables -t nat -A POSTROUTING -p tcp -o ens33 -s 10.20.20.4\
    --dport http -j SNAT --to-source 193.137.128.1
```

5.2.17. Ligação 17 (HTTPS):

A ligação 17 é também bastante semelhante à HTTPS mencionada previamente, mas com destino à máquina "External" e proveniente da máquina "Internal". O HTTPS (porto 443) pode funcionar com o protocolo "tcp" ou "udp", que estão descritos nas duas regras a seguir:

```
sudo iptables -A FORWARD -p udp -d 10.20.20.4 -o ens36 \
        --dport https -j ACCEPT

sudo iptables -A FORWARD -p tcp -d 10.20.20.4 -o ens36 \
        --dport https -j ACCEPT
```

Da mesma forma que anteriormente só foi necessário implementar a regra num sentido, neste caso é semelhante. Assim declarámos as regras "DNAT" usando o protocolo "udp" e "tcp".

```
sudo iptables -t nat -A POSTROUTING -p udp -i ens33 -d 10.20.20.4 \
        --dport https -j SNAT --to-destination 193.137.128.1

sudo iptables -t nat -A POSTROUTING -p tcp -i ens33 -d 10.20.20.4 \
        --dport https -j SNAT --to-destination 193.137.128.1
```

5.2.18. Ligação 18 (SSH):

Esta é uma ligação em que os pacotes são provenientes da máquina "Internal" e têm destino na máquina "External". Como tal, houve necessidade de especificar que a conexão entre as máquinas é possível através de um forward dos pacotes provenientes do servidor Eden ou DNS2 (ambos na máquina "External"). Sendo assim, a regra foi definida da seguinte forma:

```
sudo iptables -A FORWARD -p tcp -s 10.20.20.4 -i ens36 \
    -o ens33 --dport ssh -j ACCEPT
```

Já que se trata de uma conexão entre zonas diferentes, há necessidade de alterar o IP privado da máquina "Internal" para o IP do "router" para tal, usa-se a seguinte regra quando os pacotes são originados na máquina "Internal" para o exterior.

```
sudo iptables -t nat -A POSTROUTING -p tcp -o ens33 -s 10.20.20.20.4 \
    --dport ssh -j SNAT --to-source 193.137.128.1
```

5.2.19. Ligação 19 (FTP):

Esta ligação fica estabelecida a partir do momento em que a ligação 13 foi feita. Como tal a única diferença entre as duas é apenas na regra "`NAT`", já que o sentido da ligação se altera. Esta encontra-se descrita no seguinte snippet:

```
sudo iptables -t nat -A POSTROUTING -p tcp -o ens33 -s 10.20.20.4 \
    --dport ftp -j SNAT --to-source 193.137.128.1
```

5.3. Snort

O Snort, como foi dito previamente, foi posto em modo "`inline`". Num ficheiro chamado "`local.rules`", na pasta "`/etc/snort/rules`", foram colocadas quatro regras, em que duas correspondem à deteção e prevenção de ataques "`SQLi`" e outras duas à deteção e prevenção de ataques do tipo "`XSS`", descritas posteriormente:

- "`SQLi`":

```
drop tcp any any -> 10.10.10.4 [80,443] (msg: "SQLi OR ATTACK!"; \
        pcre:"/w*((%27)|('))((%6F)|o|(%4F))((%72)|r|(%52))/i"; sid:1000001;)

drop tcp any any -> 10.10.10.4 [80,443] (msg: "SQLi UNION ATTACK!"; pcre:"/((\%27)|(\'))union/i" ; sid:1000005; )
```

- "`XSS`":

```
drop tcp any any -> 10.10.10.4 [80,443] (msg: "XSS TAG ATTACK!"; \
        pcre:"/((\%3C)|<)((\%2F)|V)*[a-z0-9\%]+((\%3E)|>)/i"; sid:1000006;)

drop tcp any any -> 10.10.10.4 [80,443] (msg: "XSS IMAGE ATTACK!"; \
        pcre:"/((\%3C)|<)((\%69)|i|(\%49))((\%6D)|m|(\%4D))((\%67)|g|(\%47))[^\n]+((\%3E)|>)/i" ;sid:1000007; )
```

Logo à partida dá para ver alguns pontos comuns nas quatro regras, nomeadamente:

- "`drop`": Ação da regra. O Snort rejeita qualquer pacote que tenha as características da regra.
- "`tcp`": Protocolo da ligação.
- "`any`": IP de origem. O Snort olha para todos os IPs.
- "any": Porto de origem. Snort olha para todos os portos.
- "`->`": Direção do pacote. Do lado esquerdo é a origem, do lado direito o destino.
- "`10.10.10.4`": IP de destino.
- "`[80,443]`" Portos de destino. Estes portos (80 e 443) estão reservados para "`http`" e "`https`" respetivamente.

- “msg:”: Mensagem definida entre aspas que aparece quando o Snort deteta aquele tipo de intrusão.
- “sid:”: Identificação da regra. Os números até 1000000 estão reservados. As regras podem ser definidas a partir do número 1000001.
- “/i”: Ignora se o texto está escrito em maiúsculas ou minúsculas.

Analisando uma a uma, a primeira tem alguns aspetos a salientar, nomeadamente:
- “pcre:” (Perl Compatible Regular Expressions): semelhante à função “grep” do Linux. Serve para identificar expressões semelhantes.
- “\w*”: Para identificar zeros ou mais alfanuméricos ou underscores.
- “((\%27)|\(’))”: Para identificar uma pelica sozinha ou o seu equivalente hexadecimal.
- “((%6F)|o|(%4F))((%72)|r|(%52))”: Para encontrar a palavra “or” com variantes de maiúscula e minúsculas e correspondente hexadecimal.

Já a segunda, para além do que foi referenciado anteriormente, ainda tem a componente “union”, que vai detetar quando se usa essa expressão. Esta é uma palavra comumente usada quando se pretende fazer este tipo de intrusões. Como tem o “\i”, deteta a palavra com maiúsculas ou minúsculas.

Para a primeira regra de Cross-site script injection refere outros aspetos que importam mencionar, como:
- “((\%3C)|<)”: Identifica um parênteses de abertura ou um equivalente hexadecimal.
- “((\%2F)|V)*”: Identifica um “/” ou o equivalente hexadecimal.
- “*[a-z0-9\%]+”: Identifica uma string alfanumérica dentro de uma tag ou o equivalente hexadecimal.
- “((\%3E)|>)”: Identifica um parênteses de fecho ou um equivalente hexadecimal.

Por útlimo, na última regra vemos também alguns parâmetros de identificação de caracteres diferentes dos mencionados anteriormente, nomeadamente:
- “((\%69)|i|(\%49))((\%6D)|m|(\%4D))((\%67)|g|(\%47))”: Diferentes formas de escrever “img” em ASCII, com variantes em maiúsculas e minúsculas.
- “[^\n]+”: Qualquer caractere que não seja uma nova linha que esteja posterior a “img”.

6. Testes

Origem	Destino	Serviços	Estado	Ferramenta
router	internet	DNS	OK	Netcat
internal	router	SSH	OK	Netcat
vpn-gw (dmz)	router	SSH	OK	Netcat
internet	dns (dmz)	DNS	OK	Netcat
internal	dns (dmz)	DNS	OK	Netcat
dns (dmz)	dns2(internet)	DNS	OK	Netcat
dns (dmz)	others (internet)	DNS	OK	Netcat
internet	smtp (dmz)	SMTP	OK	Netcat
internal	smtp (dmz)	SMTP	OK	Netcat
internet	mail (dmz)	POP3, POP3s	OK	Netcat
internal	mail (dmz)	POP3, POP3s	OK	Netcat
internet	mail (dmz)	IMAP IMAPs	OK	Netcat
internal	mail (dmz)	IMAP IMAPs	OK	Netcat
internet	www (dmz)	HTTP	OK	Netcat
internal	www (dmz)	HTTP	OK	Netcat
internet	www (dmz)	HTTPS	OK	Netcat
internal	www (dmz)	HTTPS	OK	Netcat
internet	vpn-gw (dmz)	OpenVPN	OK	Netcat
internal	vpn-gw (dmz)	OpenVPN	OK	Netcat
vpn-gw (dmz)	datastore (internal)	POSTGRES	OK	Netcat

Origem	Destino	Serviços	Estado	Ferramenta
vpn-gw (dmz)	kerberos	KERBEROS 5	OK	Netcat
external	ftp (internal)	FTP	OK	ftp,vsftp,vsftpd
eden	datastore (internal)	SSH	OK	SSH
dns2	datastore (internal)	SSH	OK	SSH
internal	dns2	DNS	OK	Netcat
internal	others (internet)	DNS	OK	Netcat
internal	eden	HTTP	OK	Netcat
internal	others (internet)	HTTP	OK	Netcat
internal	eden	HTTPS	OK	Netcat
internal	others (internet)	HTTPS	OK	Netcat
internal	eden	SSH	OK	SSH
internal	dns2	SSH	OK	SSH
internal	other(internet)	SHH	OK	SHH
internal	eden	FTP	OK	ftp,vsftp,vsftpd
internal	others (internet)	FTP	OK	ftp,vsftp,vsftpd

Além do teste de ligações, foi também testado o Snort. Usámos o comando "`netcat -l https -v`" na consola da máquina "`DMZ`" e no lado da máquina "`External`" escrevia-se o comando "`Netcat 193.137.128.1 http`". Depois foi possível enviar queries da máquina "`External`" para a "`DMZ`". Quando se escreveu

a expressão “`union`”, foi possível ver no Snort a mensagem “`SQLi INJECTION ATTACK!`”, confirmando, de facto, que o Snort detetou esta ameaça.

7. Conclusões

Neste trabalho conseguimos reproduzir o cenário de rede que nos foi proposto estando todos os serviços funcionais. Inicialmente foram propostas apenas a implementação de três máquinas, no entanto, de forma a facilitar testes, criámos uma quarta para simular a Internet.

Tivemos a oportunidade de estudar como o router presente em todas as casas funciona, além de termos conseguido implementar um de raíz.

Mais uma vez, a documentação foi de difícil compreensão e os artigos e manuais que melhor elucidam para a resolução deste problema de implementação de um router foram bastante difíceis de encontrar. No entanto, o trabalho foi concluído com sucesso.

8. Referências

- Linux Man Pages: snort, iPTables, iPTables-extensions
- Segurança Prática em Sistemas e Redes com Linux, Jorge Granjal, editor: FCA, isbn: 9789727228652
- https://www.blackhat.com/presentations/bh-usa-04/bh-us-04-mookhey/old/bh-us-04-mookhey_whitepaper.pdf
- https://www.snort.org/
- https://docs.vmware.com/en/VMware-Workstation-Pro/16.0/com.vmware.ws.using.doc/GUID-9831F49E-1A83-4881-BB8A-D4573F2C6D91.html.