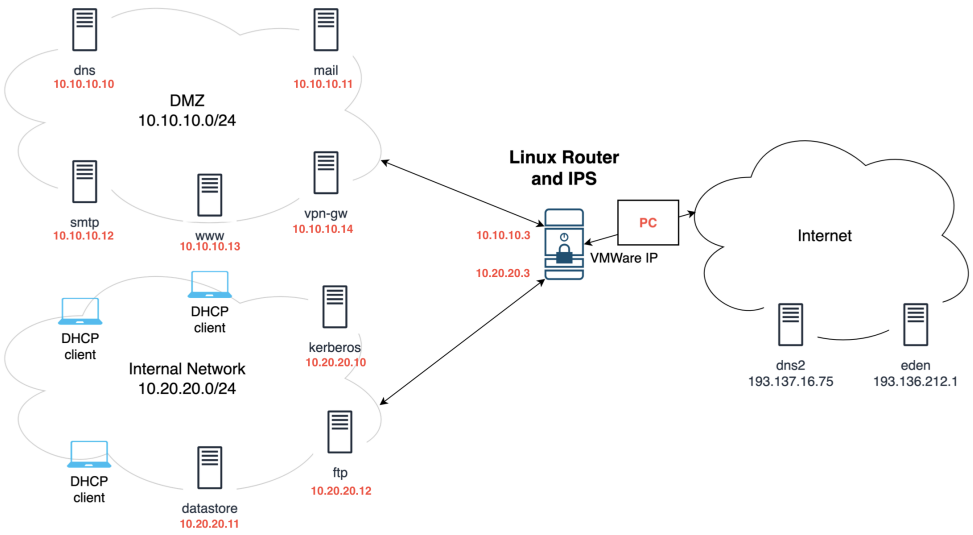
Configuração de um Router

Trabalho Prático 2

Bruno Faria e Dylan Perdigão
2018295474, 2018233092
{brunofaria, dgp}@student.dei.uc.pt
Departamento de Engenharia Informática, Universidade de Coimbra

Abril 2022



Conteúdo

1 Introdução

O objetivo do seguinte trabalho prático é efetuar a configuração de uma *network firewall* com o uso de *IPTables*, criando regras com filtragem, NAT e integrção com o *Snort*. De seguida, o alvo é a configuração do *Snort* para deteção e reação/prevenção de intrusões.

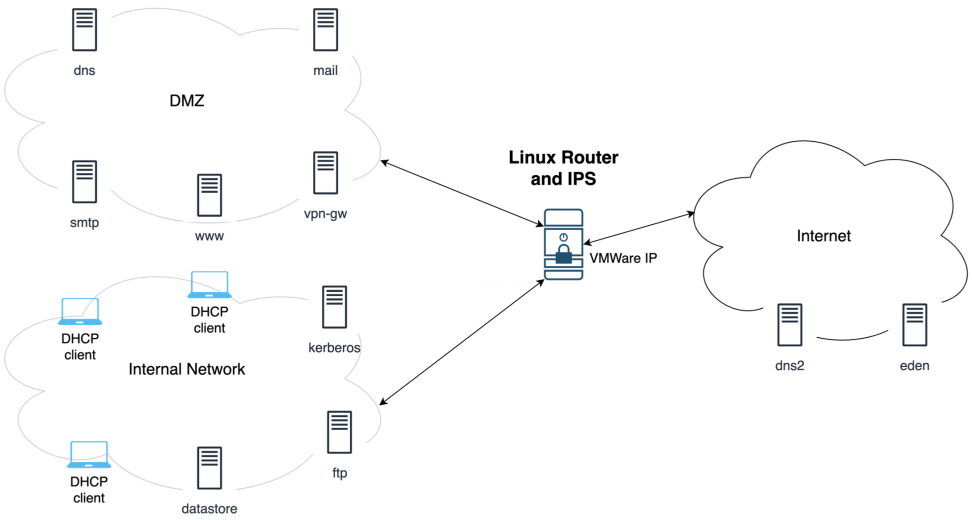


Figura 1: Esquema geral implementado

O cenário experimental é composto por 3 máquinas virtuais (DMZ, Internal, Router) cujo os endereços IPv4 das redes e das máquinas podem ser verificados na figura anterior.

Este relatório está organizado da seguinte forma. A secção 2 descreve o cenário experimental assim como a configuração das VMs criadas. A secção 3 mostra as regras defenidas com o IPTables de forma a responder ao pedido no enunciado. A secção 4 começa por apresentar quais os tipos de intrusões iremos detectar e e bloquear, bem como as regras definidas no *Snort* para a deteção e prevenção desses mesmos ataques. Por fim temos a secção 5 onde tiramos as notas finais sobre o projeto.

2 Configuração Inicial das Maquinas Virtuais

As Maquinas virtuais foram configuradas no *VMWare* com vários *Networks Adapaters* de forma a suportarem ligações com os seguintes IPs.

	Global	Local
DMZ	-	10.10.10.10/24
Rede Interna	-	10.20.20.10/24
Router	192.168.93.158/24	10.20.20.1/24 e 10.10.10.1/24

Tabela 1: Endereços IP das máquinas virtuais

Nas configurações de rede do Debian na VM, é possivél criar perfis de rede definindo manualmente na secção "*IPv4*"os endereços IP da rede local e global para cada maquina como mostra a figura 1. Foi então efetuada estas configurações ficando com os ips da tabela 1.

3 Configuraçãos de IPTables

Numa primeira fase foi necessário ativar o encaminhamento de pacotes no router e reinicializar as IPTables do mesmo (código 1).

```
echo 1 > /proc/sys/net/ipv4/ip_forward
sudo modprobe ip_conntrack_ftp

sudo iptables -t nat -F
sudo iptables -t mangle -F
sudo iptables -F
sudo iptables -X
```

Código 1: Ativaç ão de encaminhamento de pacotes e reinitialização das regras

Na segunda fase foram criadas políticas (policies) para descartar quaisqueres pacotes que não foram configurados nas IPTables (figura 2).

```
sudo iptables -P INPUT DROP
sudo iptables -P FORWARD DROP
sudo iptables -P OUTPUT DROP
```

Código 2: Definição das policies

Após efetuar os comandos anteriores, foi necessário aceitar ligações nos portes necesssários como feito no código 3.

```
sudo iptables -A FORWARD -d $VPN -p tcp --dport openvpn -j ACCEPT
sudo iptables -A FORWARD -d $VPN -p udp --dport openvpn -j ACCEPT
```

Código 3: Autorização do Openvpn por TCP e UDP

No caso de limitar as ligações foi necessário adicionar o argumento "connlimit-upto"para 10 como mostrado no código 4

```
sudo iptables -A FORWARD -d $KERBEROS -s $VPN -p tcp --dport kerberos -m connlimit --
    connlimit-upto 10 -j ACCEPT
sudo iptables -A FORWARD -d $KERBEROS -s $VPN -p udp --dport kerberos -m connlimit --
    connlimit-upto 10 -j ACCEPT
```

Código 4: Autorização do Kerberos com limite de 10 conexões

No caso de ligações a partir do exterior é preciso efetuar um prerouting para as redes internas do router como se pode observar no código5.

```
sudo iptables -t nat -A PREROUTING -d $ROUTER_VM -s $EDEN -p tcp --dport ssh -j DNAT
    --to-destination $DATASTORE
sudo iptables -A FORWARD -d $DATASTORE -s $EDEN -p tcp --dport ssh -j ACCEPT
```

Código 5: Definição do prerouting para SSH

4 Deteção e Prevenção de Intrusões

Para a prevenção de intrusões usamos o Snort com as rules do Código 6

```
    ipvar VAR [10.10.10.0/24,10.10.20.0/24]

    #[action] [protocol] [sourceIP] [sourceport] -> [destIP] [destport] ( [Rule
    options] )

    #SQL
    drop tcp any any -> $VAR any (msg:" SQL Injection Based on or TRUE "; sid
    :1000000; rev:1; content:"or"; nocase;flow:to_server,established;)
    drop tcp any any -> $VAR any (msg:" SQL Injection Based on DROP "; sid:1000001;
    rev:1; content:"drop"; nocase;flow:to_server,established;)

    #XSS
    drop tcp any any -> $VAR any (msg:" XSS Attack <...> "; sid:1000002; rev:1;
    content:"<script>"; nocase;flow:to_server,established;)
```

```
11  drop tcp any any -> $VAR any (msg:" XSS Attack <img ...> "; sid:1000003; rev:1;
    content:"<img"; nocase; flow:to_server,established;)
```

Código 6: Conteúdo do ficheiro *local.rules*

Assim como o pedido no enunciado, tratamos dois tipos de intrusões de SQL, bem como de dois tipos de ataques de XSS.

4.1 SQL Injection

4.1.1 SQL Injection Based on 'or True'

Este tipo de ataque pode pode levar a que o atacante consiga ter acesso a informação sobre os utilizadores, incluíndo, por vezes, até mesmo as suas passwords.

Consideremos o exemplo desprotegido onde é pedido o userID. No caso do utilizador inserir *"123 or 1=1"* isto poderia gerar o comando apresentado no Código 7.

```
1  SELECT * FROM Users WHERE UserId = 123 OR 1=1;
```

Código 7: Exemplo de SQL Injection - Tipo 1

Isto levaria a um comando válido, que devolveria todas as linhas com todas as informações da tabela users.

Para colmatar este problema foi inserida a regra da linha 6 do Código 6, onde notifica e rejeita as comunicações com "or"no seu conteúdo.

4.1.2 SQL Injection Based on Batched SQL Statements

A maior parte das bases de dados suportam o uso de vários comandos separados por ';'. Neste exemplo em concreto, no caso do comando *DROP*, o ataque pode pode levar a perda permanente de dados, assim como a problemas de funcionamento no sistema.

Analisemos novamente o exemplo anterior com o *userID*. No caso de o utilizador inserir *"123; DROP TABLE Suppliers"*, poderia ser gerado o comando apresentado no Código 8.

```
1  SELECT * FROM Users WHERE UserId = 123; DROP TABLE Users;
```

Código 8: Exemplo de SQL Injection - Tipo 2

Isto levaria a um comando válido, que eliminaria a tabela *Users*, levando à perda de toda a informação nela armazenada.

Para colmatar este problema foi inserida a regra da linha 7 do Código 6 onde notifica e rejeita as comunicações com "drop"no seu conteúdo.

4.2 XSS Attacks

4.2.1 Reflected cross-site scripting

Reflected XSS acontece quando uma aplicação recebe um request HTTP incluíndo código malicioso.

Como exemplo deste tipo de ataque temos o Código 9

```
1  <p>Status: <script>/* Bad stuff here... */</script></p>
```

Código 9: Exemplo de ataque XSS - Tipo 1

Se o utilizador compilar esse código o *script* do atacante será executado, podendo roubar informações do utilizador e etc.

Para colmatar este problema foi inserida a regra da linha 10 do Código 6 onde notifica e rejeita as comunicações com `"<script>"` no seu conteúdo.

4.2.2 DOM-based cross-site scripting

Analogamente ao ataque anterior, neste o atacante também consegue inserir o seu próprio código, como no exemplo do Código 10, onde este entra sob a forma de uma imagem. If the attacker can control the value of the input field, they can easily construct a malicious value that causes their own script to execute:

Como exemplo deste tipo de ataque temos o Código 10

```
1    <img src=1 onerror='/* Bad stuff here... */'>
```

Código 10: Exemplo de ataque XSS - Tipo 2

Para colmatar este problema foi inserida a regra da linha 11 do Código 6 onde notifica e rejeita as comunicações com `"<img"` no seu conteúdo.

4.3 Testes

Estes ataques foram simulados com dois terminais netcat. O primeiro na VM DMZ a escutar no porto em causa, e o segundo a enviar patindo da VM Internal. Simultaneamente tínhamos o Snort a correr na VM router. Assim sendo, verificamos que os pacotes com *"or"*, *"drop"*, *≪script¿"* e *≪img"* eram travados pelo Snort, enquanto todos os outros passavam.

5 Conclusão

O presente trabalho prático proporcionou uma aprendizagem mais aprofundada sobre a configuração de um *Firewall*, usando regras de *IPTables*, bem como o uso do *Snort* para deteção e prevenção de intrusões.