

1 2



9 0

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE
COIMBRA

Relatório

Trabalho Prático #2 - IPTables and Snort

**Segurança em Tecnologias da Informação
2019/2020**

Duarte Guerreiro 2016231778

João Tomás 2016225021

Índice

Introdução	2
1. Arquitetura do Projeto	3
2. Como executar o projeto	3
3. Configuração das máquinas virtuais	3
4. Implementação	4
4.1 IPTables	4
4.1.1 Firewall configuration to protect the router	4
4.1.2 Firewall configuration to authorize direct communications (without NAT)	5
4.1.3 Firewall configuration for connections to the external IP address of the firewall (using NAT)	5
4.1.4 Firewall configuration for communications from the internal network to the outside (using NAT)	5
4.2 Snort	6
4.2.1 SQL Injections	6
4.2.2 XSS application-layer attacks	8
5. Testes efetuados	9
5.1 IPTables	9
5.2 Snort	9
5.2.1 SQL Injections	10
5.2.2 XSS application-layer attacks	10
Conclusão	11
Bibliografia	11

Introdução

No âmbito do segundo trabalho prático da unidade curricular de Segurança em Tecnologias da Informação implementámos um cenário bastante utilizado por organizações, sendo este constituído por uma rede DMZ, uma rede Interna, um router e a Internet. O router implementa a funcionalidade de firewall, utilizando a tecnologia IPTables, e funciona também como IDS (Intrusion Detection System), o que é possível graças à utilização do Snort.

Ao longo deste documento iremos descrever os passos que nos permitiram que todas estas funcionalidades fossem implementadas, garantindo assim a proteção das redes do nosso cenário.

1. Arquitetura do Projeto

O nosso projeto é constituído por **quatro máquinas virtuais**, sendo estas destinadas à implementação dos diferentes constituintes do nosso trabalho: a rede DMZ (demilitarized zone), a Internet, a rede interna e o router. No router situam-se os ficheiros que permitem implementar as regras das IPTables (iptables.backup) e o ficheiro de configuração do IDS (snort.conf), assim como as diferentes regras associadas ao mesmo, as quais permitem que este detete os ataques pretendidos e reaja aos mesmos.

2. Como executar o projeto

De forma a ser possível executar o trabalho desenvolvido, é necessário colocar as 4 máquinas virtuais em funcionamento:

- Na máquina virtual que serve como **router** é necessário eliminar as regras default que vêm com o iptables e colocar as regras que desenvolvemos, através do comando **iptables-restore < iptables.backup** e é ainda necessário desativar o adaptador NAT, para que não exista qualquer interferência com as regras existentes. Uma vez que é nesta máquina virtual que estará a correr o Snort IDS, é ainda necessário executar o mesmo através do comando **snort -c snort.conf -A console -i enp0s10**. Precisamos de correr este comando com o ficheiro de configuração de forma a que o Snort execute em modo de deteção de intrusões.
- Na máquina virtual que serve como **rede interna** é apenas necessário definir o endereço IP da máquina que pretende simular e testar
- Na máquina virtual destinada à **DMZ** é mais uma vez necessário definir o endereço IP do ponto de comunicação que estamos a testar
- A máquina virtual da **Internet** segue o mesmo princípio, sendo assim apenas necessário atribuir o endereço de IP à mesma consoante o cenário de teste pretendido.

3. Configuração das máquinas virtuais

Como foi referenciado na secção anterior, irão existir quatro máquinas virtuais. De forma a garantir que seria possível estabelecer comunicações entre as mesmas, foi necessário definir os adaptadores das mesmas:

- **Router** - Adaptador NAT e três adaptadores NAT, de nome **intnet**, **intnet2** e **intnet3**, podendo assim estabelecer ligação com as máquinas virtuais da rede interna, DMZ e Internet, respetivamente, e podendo depois coordenar a comunicação entre as mesmas, após configuração das regras das IPTables.
- **Rede interna** - Adaptador NAT e adaptador Internal Network, com o nome **intnet**, para que seja possível estabelecer ligação com o router
- **DMZ** - Adaptador NAT e adaptador Internal Network, com o nome **intnet2**, de forma a que consiga comunicar com o router
- **Internet** - Adaptador NAT e adaptador Internal Network, com o nome **intnet3**, podendo assim estabelecer ligação com o router

Após esta configuração inicial ter sido efetuada, será depois necessário atribuir endereços IP às diferentes máquinas, para que mais tarde possam ser efetuados os diferentes testes com as regras definidas nas IPTables. Esta configuração pode ser efetuada na diretoria `/etc/sysconfig/network-scripts`, no ficheiro correspondente ao nome da devida interface da rede interna (`ifcfg-enp0sx`). De notar que, uma vez que existem diversas máquinas em cada rede, irá ser necessário ir alterando os endereços IP para o de cada uma das diferentes máquinas para que os testes possam ser efetuados, sendo que a única máquina virtual na qual não será necessário alterar quaisquer endereços IP é a que se destina ao router.

Posto isto, terminamos assim o processo de configuração do nosso trabalho prático, ficando com um ambiente como o apresentado na Figura 1.

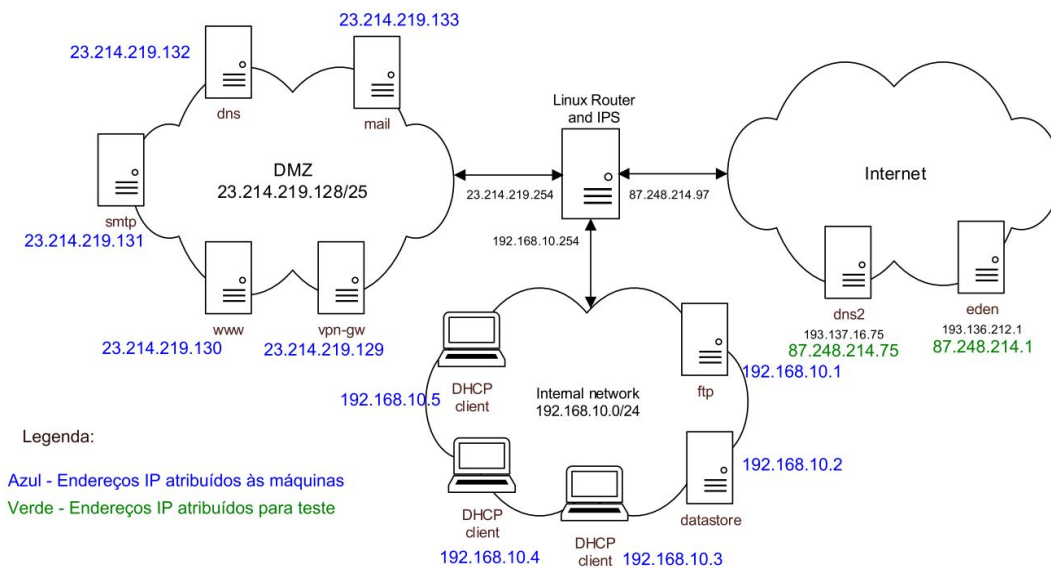


Figura 1 - Esquema do ambiente do nosso trabalho prático

Para que seja possível ao router encaminhar pacotes entre as diferentes máquinas virtuais, é ainda necessário ativar o forwarding de pacotes, através do comando `echo 'net.ipv4.ip_forward = 1' >> /etc/sysctl.conf`. Por fim desativamos a firewall - `systemctl stop firewalld` - para que não exista qualquer interferência com as regras que irão ser posteriormente definidas, podendo assim prosseguir para a devida implementação do nosso projeto.

4. Implementação

Nesta secção iremos apresentar os diferentes aspetos relativos à implementação das regras da IPTables e do Snort.

4.1 IPTables

Nas regras configuradas nas IPTables tivemos em atenção que a policy das chains INPUT, OUTPUT e FORWARD estaria a DROP e, como tal, foi necessário adicionar FORWARD's nas regras que utilizam NAT, de forma a garantir que as comunicações continuassem a ser possíveis.

Foi ainda necessário definir uma default gateway nas máquinas virtuais da rede interna, DMZ e Internet, com os endereços 192.168.10.254, 23.214.219.254 e 87.248.214.97, respetivamente. A default gateway é de extrema importância uma vez que irá servir como um host de encaminhamento para outras redes quando nenhuma outra especificação de rota corresponder ao endereço IP de destino do pacote.

4.1.1 Firewall configuration to protect the router

DNS name resolution requests sent to outside servers

```
iptables -A OUTPUT -p udp --dport domain -j ACCEPT
iptables -A INPUT -p udp --sport domain -j ACCEPT
```

Como neste ponto consideramos ligações “diretas”, criámos regras para aceitar comunicações entre o nosso router e outras máquinas desde que fossem dns queries, ou seja, utilizassem o porto 53 e protocolo udp.

SSH connections to the router system, if originated at the internal network or at the VPN gateway (vpn-gw)

```
iptables -A INPUT -s 23.214.219.129 -p tcp --dport ssh -j ACCEPT
iptables -A INPUT -s 192.168.10.0/24 -p tcp --dport ssh -j ACCEPT
iptables -A OUTPUT -d 23.214.219.129 -p tcp --sport ssh -j ACCEPT
iptables -A OUTPUT -d 192.168.10.0/24 -p tcp --sport ssh -j ACCEPT
```

Neste ponto seguimos o mesmo princípio que no conjunto de regras anterior, sendo que desta vez se tratavam de ligações ssh, logo o porto teria que ser o ssh (22) e através do protocolo de comunicação tcp, podendo estas ligações ter origem na rede interna - 192.168.10.0/24 - ou na VPN gateway - 23.214.219.129.

4.1.2 Firewall configuration to authorize direct communications (without NAT)

Relativamente às ligações diretas, mas sem recorrer ao NAT, utilizámos a chain de FORWARD para construir regras que permitissem ligações entre a rede interna e a DMZ e entre a internet e a DMZ. Foi tida especial atenção quanto às regras relacionadas com dns, sendo necessário utilizar o protocolo de transporte udp para queries e tcp para a sincronização entre o dns e o dns2.

Domain name resolutions using the dns server.

```
iptables -A FORWARD -s 192.168.10.0/24 -p udp --dport domain -d 23.214.219.132 -j ACCEPT
iptables -A FORWARD -d 192.168.10.0/24 -p udp --sport domain -s 23.214.219.132 -j ACCEPT
```

Estas regras permitem o encaminhamento de comunicações entre a rede interna e o servidor dns localizado na rede DMZ, as quais são feitas através do protocolo udp, no porto 53 (dns).

The dns server should be able to resolve names using DNS servers on the Internet (dns2 and also others)

```
iptables -A FORWARD -s 23.214.219.132 -p udp --dport domain -o enp0s10 -j ACCEPT
iptables -A FORWARD -d 23.214.219.132 -p udp --sport domain -i enp0s10 -j ACCEPT
```

Este conjunto de regras permite, tal como descrito no título, resolver nomes tanto no dns2 como noutros servidores dns na Internet, logo apenas necessitamos de especificar o porto, protocolo e interface de rede de entrada e de saída (enp0s10 é a interface do router que comunica com a Internet).

The dns and dns2 servers should be able to synchronize the contents of DNS zones

DNS server da DMZ inicia a ligação

```
iptables -A FORWARD -s 23.214.219.132 -p tcp --dport domain -o enp0s10 -d 87.248.214.75 -j ACCEPT
iptables -A FORWARD -d 23.214.219.132 -p tcp --sport domain -i enp0s10 -s 87.248.214.75 -j ACCEPT
```

DNS2 inicia a ligação

```
iptables -A FORWARD -d 23.214.219.132 -p tcp --dport domain -i enp0s10 -s 87.248.214.75 -j ACCEPT
iptables -A FORWARD -s 23.214.219.132 -p tcp --sport domain -o enp0s10 -d 87.248.214.75 -j ACCEPT
```

De forma a sincronizar os conteúdos das zonas DNS, criámos as regras que permitiam a ligação entre os servidores dns e dns2, tendo o cuidado de utilizar o protocolo de transporte tcp, uma vez que é o protocolo utilizado na sincronização de zonas entre servidores, de especificar o porto domain e os respetivos endereços de IP dos servidores dns e dns2. Tivemos ainda em atenção o facto de tanto o servidor dns como dns2 poderem iniciar a ligação.

SMTP connections to the smtp server

```
iptables -A FORWARD -s 192.168.10.0/24 -p tcp --dport smtp -d 23.214.219.131 -j ACCEPT
iptables -A FORWARD -d 192.168.10.0/24 -p tcp --sport smtp -s 23.214.219.131 -j ACCEPT
```

Comunicação com a Internet

```
iptables -A FORWARD -d 23.214.219.131 -p tcp --dport smtp -i enp0s10 -j ACCEPT
iptables -A FORWARD -s 23.214.219.131 -p tcp --sport smtp -o enp0s10 -j ACCEPT
```

Definimos também regras que permitem a comunicação entre a rede interna e o servidor SMTP (Simple Mail Transfer Protocol), localizado na rede DMZ e entre este e a internet. No entanto, para as regras da Internet apenas definimos a interface de entrada e de saída (enp0s10), uma vez que as ligações podem ter origem em qualquer endereço IP, desde que o porto seja o do SMTP.

POP and IMAP connections to the mail server

POP (O nome do service pop é pop2)

```
iptables -A FORWARD -s 192.168.10.0/24 -p tcp --dport pop2 -d 23.214.219.133 -j ACCEPT
```

```
iptables -A FORWARD -d 192.168.10.0/24 -p tcp --sport pop2 -s 23.214.219.133 -j ACCEPT
```

POP3

```
iptables -A FORWARD -s 192.168.10.0/24 -p tcp --dport pop3 -d 23.214.219.133 -j ACCEPT
```

```
iptables -A FORWARD -d 192.168.10.0/24 -p tcp --sport pop3 -s 23.214.219.133 -j ACCEPT
```

IMAP

```
iptables -A FORWARD -s 192.168.10.0/24 -p tcp --dport imap -d 23.214.219.133 -j ACCEPT
```

```
iptables -A FORWARD -d 192.168.10.0/24 -p tcp --sport imap -s 23.214.219.133 -j ACCEPT
```

Comunicação com a Internet

```
iptables -A FORWARD -d 23.214.219.133 -p tcp --dport pop2 -i enp0s10 -j ACCEPT
```

```
iptables -A FORWARD -s 23.214.219.133 -p tcp --sport pop2 -o enp0s10 -j ACCEPT
```

```
iptables -A FORWARD -d 23.214.219.133 -p tcp --dport pop3 -i enp0s10 -j ACCEPT
```

```
iptables -A FORWARD -s 23.214.219.133 -p tcp --sport pop3 -o enp0s10 -j ACCEPT
```

```
iptables -A FORWARD -d 23.214.219.133 -p tcp --dport imap -i enp0s10 -j ACCEPT
```

```
iptables -A FORWARD -s 23.214.219.133 -p tcp --sport imap -o enp0s10 -j ACCEPT
```

Neste conjunto de regras definimos as ligações ao servidor mail, as quais tanto podem ser feitas através de IMAP (Internet Message Access Protocol) ou POP (Post Office Protocol), sendo que neste último protocolo tanto considerámos POP2 como POP3. Ambos os protocolos funcionam maioritariamente sobre TCP. Mais uma vez definimos regras que também permitem comunicação entre a rede DMZ e a Internet.

HTTP and HTTPS connections to the www server

```
iptables -A FORWARD -s 192.168.10.0/24 -p tcp --dport http -d 23.214.219.130 -j ACCEPT
```

```
iptables -A FORWARD -d 192.168.10.0/24 -p tcp --sport http -s 23.214.219.130 -j ACCEPT
```

```
iptables -A FORWARD -s 192.168.10.0/24 -p tcp --dport https -d 23.214.219.130 -j ACCEPT
```

```
iptables -A FORWARD -d 192.168.10.0/24 -p tcp --sport https -s 23.214.219.130 -j ACCEPT
```

Comunicação com a Internet

```
iptables -A FORWARD -d 23.214.219.130 -p tcp --dport http -i enp0s10 -j ACCEPT
```

```
iptables -A FORWARD -s 23.214.219.130 -p tcp --sport http -o enp0s10 -j ACCEPT
```

```
iptables -A FORWARD -d 23.214.219.130 -p tcp --dport https -i enp0s10 -j ACCEPT
```

```
iptables -A FORWARD -s 23.214.219.130 -p tcp --sport https -o enp0s10 -j ACCEPT
```

Seguindo o mesmo princípio que no conjunto de regras anteriores, definimos regras que tanto permitem ligação da rede interna ao servidor www da rede DMZ como entre este e a Internet.

OpenVPN connections to the vpn-gw server.

```
iptables -A FORWARD -s 192.168.10.0/24 -p tcp --dport openvpn -d 23.214.219.129 -j ACCEPT
```

```
iptables -A FORWARD -d 192.168.10.0/24 -p tcp --sport openvpn -s 23.214.219.129 -j ACCEPT
```

```
iptables -A FORWARD -s 192.168.10.0/24 -p udp --dport openvpn -d 23.214.219.129 -j ACCEPT
iptables -A FORWARD -d 192.168.10.0/24 -p udp --sport openvpn -s 23.214.219.129 -j ACCEPT
```

Comunicação com a Internet

```
iptables -A FORWARD -d 23.214.219.129 -p tcp --dport openvpn -i enp0s10 -j ACCEPT
iptables -A FORWARD -s 23.214.219.129 -p tcp --sport openvpn -o enp0s10 -j ACCEPT
iptables -A FORWARD -d 23.214.219.129 -p udp --dport openvpn -i enp0s10 -j ACCEPT
iptables -A FORWARD -s 23.214.219.129 -p udp --sport openvpn -o enp0s10 -j ACCEPT
```

De forma a configurar as ligações OpenVPN, considerámos mais uma vez ligações entre a rede interna e a VPN Gateway (23.214.219.129) na rede DMZ e entre esta e a Internet, no porto 1194 (porto reservado para o OpenVPN).

VPN clients connected to the gateway (vpn-gw) should able to connect to the PostgreSQL service on the datastore server

```
iptables -A FORWARD -s 23.214.219.129 -p tcp --dport postgres -d 192.168.10.2 -j ACCEPT
iptables -A FORWARD -d 23.214.219.129 -p tcp --sport postgres -s 192.168.10.2 -j ACCEPT
iptables -A FORWARD -s 23.214.219.129 -p udp --dport postgres -d 192.168.10.2 -j ACCEPT
iptables -A FORWARD -d 23.214.219.129 -p udp --sport postgres -s 192.168.10.2 -j ACCEPT
```

Quanto aos clientes VPN que se encontrem ligados à VPN Gateway (23.214.219.129), estes poderão ter acesso ao serviço Postgres, o qual corre no porto 5432 (porto reservado para o serviço Postgres), no servidor datastore (192.168.10.2) localizado na rede interna, através do conjunto de regras acima definido.

4.1.3 Firewall configuration for connections to the external IP address of the firewall (using NAT)

Visto que as máquinas situadas na internet desconhecem a existência da rede interna e, para que estas redes possam comunicar entre si, procedemos à utilização do **Destination NAT (DNAT)**. Desta forma, hosts que se encontrem na internet, ao tentarem efetuar ligação com o router, num determinado porto, conseguem que a sua comunicação seja redirecionada para uma máquina na rede interna, através da alteração do endereço IP de destino dos pacotes.

Para além disto, e olhando para a sequência que os pacotes percorrem nas chains do router, tornou-se óbvio que seria necessário a utilização de regras na chain de FORWARD. Em primeiro lugar, adicionámos regras para que os pacotes com o novo destino e o retorno destes mesmos fossem aceites. Por fim, ao tratarmos das comunicações FTP, com portos dinâmicos para o channel passivo, utilizámos o módulo de conntrack para

detetar novas ligações, mas que estivessem relacionadas com ligações já estabelecidas à ligação de comando do ftp, como iremos ver de seguida.

FTP connections (in passive and active modes) to the ftp server.

```
iptables -t nat -A PREROUTING -d 87.248.214.97 -i enp0s10 -m state --state NEW  
-p tcp --dport 21 -j DNAT --to-destination 192.168.10.1
```

Command channel

```
iptables -A FORWARD -i enp0s10 -d 192.168.10.1 -p tcp --dport 21 -m state  
--state ESTABLISHED,NEW -j ACCEPT
```

```
iptables -A FORWARD -o enp0s10 -s 192.168.10.1 -p tcp --sport 21 -m state  
--state ESTABLISHED -j ACCEPT
```

Data channel Active

```
iptables -A FORWARD -o enp0s10 -s 192.168.10.1 -p tcp --sport 20 -m state  
--state ESTABLISHED,RELATED -j ACCEPT
```

```
iptables -A FORWARD -i enp0s10 -d 192.168.10.1 -p tcp --dport 20 -m state  
--state ESTABLISHED -j ACCEPT
```

Data channel Passive

```
iptables -A FORWARD -i enp0s10 -d 192.168.10.1 -p tcp -m state --state  
ESTABLISHED,RELATED -j ACCEPT
```

```
iptables -A FORWARD -o enp0s10 -s 192.168.10.1 -p tcp -m state --state  
ESTABLISHED -j ACCEPT
```

No caso do FTP, este é um cenário peculiar, uma vez que este protocolo possui dois modos, sendo um o **ativo** e outro o **passivo**. Como tal, foi necessário utilizar os módulos de connection tracking de forma a permitir que as ligações continuassem a ser estabelecidas mesmo quando não tínhamos conhecimento dos portos a utilizar.

Deste modo, começámos por definir a regra de SNAT, a qual permite ligações ao porto 21 do FTP server, no endereço 192.168.10.1, sendo que o endereço de IP de origem dos pacotes é alterado quando estes chegam à interface do router, passando este a ser 87.248.214.97. Esta ligação faz parte do **command channel** e como tal teremos que definir dois FORWARD's para o mesmo, um de entrada e um de saída, tendo em atenção o uso das flags de estado NEW apenas na primeira ligação, sendo depois utilizada a flag ESTABLISHED, uma vez que a ligação já se encontra estabelecida. No entanto, quanto ao **data channel**, existem diferenças entre os dois modos. No caso do **ativo**, o servidor irá ligar-se, a partir do porto 20, a um porto do cliente cujo número seja acima de 1024, sendo que depois o cliente irá enviar um acknowledge a partir desse mesmo porto, de novo para o porto 20 do servidor (mais uma vez, conseguimos saber quais os portos aos quais a ligação deve ser estabelecida devido aos módulos utilizados). Por outro lado, quando tratamos da ligação ftp em modo **passivo**, será o cliente que se irá ligar a um porto que o servidor irá definir, sendo que o servidor depois irá enviar o acknowledge de que a ligação foi estabelecida (e os dados pretendidos) a partir desse mesmo porto.

SSH connections to the datastore server, but only if originated at the eden or dns2 servers

Eden

```
iptables -t nat -A PREROUTING -i enp0s10 -s 87.248.214.1 -d 87.248.214.97 -p  
tcp --dport ssh -j DNAT --to-destination 192.168.10.2
```

```
iptables -A FORWARD -i enp0s10 -s 87.248.214.1 -d 192.168.10.2 -p tcp --dport
```

```
ssh -j ACCEPT
iptables -A FORWARD -s 192.168.10.2 -d 87.248.214.1 -p tcp --sport ssh -j
ACCEPT
Dns2
iptables -t nat -A PREROUTING -i enp0s10 -s 87.248.214.75 -d 87.248.214.97 -p
tcp --dport ssh -j DNAT --to-destination 192.168.10.2
iptables -A FORWARD -i enp0s10 -s 87.248.214.75 -d 192.168.10.2 -p tcp --dport
ssh -j ACCEPT
iptables -A FORWARD -s 192.168.10.2 -d 87.248.214.75 -p tcp --sport ssh -j
ACCEPT
```

Com este conjunto de regras pretendemos que seja possível comunicar com o datastore, a partir do servidor eden ou dns2, logo iremos mais uma vez utilizar DNAT, sendo que o endereço de destino será traduzido para o endereço do router, sendo depois o pacote corretamente encaminhado para o porto ssh do servidor datastore (192.168.10.2), utilizando o protocolo de transporte tcp.

4.1.4 Firewall configuration for communications from the internal network to the outside (using NAT)

Uma vez que queremos permitir que os hosts na nossa rede interna consigam comunicar com a internet utilizámos **SNAT (Source Network Address Translation)**, o qual nos permite traduzir os endereços IP privados que se encontram na rede interna para o endereço de IP da interface do router, a qual comunica com a internet, para que seja possível comunicar com os serviços que se encontram na Internet.

Domain name resolutions using DNS

```
iptables -t nat -A POSTROUTING -s 192.168.10.0/24 -p udp --dport domain -j SNAT
--to-source 87.248.214.97
iptables -A FORWARD -s 192.168.10.0/24 -p udp --dport domain -o enp0s10 -j
ACCEPT
iptables -A FORWARD -d 192.168.10.0/24 -p udp --sport domain -i enp0s10 -j
ACCEPT
```

Neste primeiro conjunto de regras, começámos por definir a regra na qual os hosts pertencentes à rede interna 192.168.10.0/24 podem efetuar domain name resolutions com destino ao porto domain (porto 53), o qual utiliza o protocolo udp. De seguida, utilizámos a flag --to-source que define qual o endereço para o qual será traduzido o endereço do host da rede interna, sendo este o do router (87.248.214.97).

Por fim, e uma vez que a policy das nossas chains de INPUT, OUTPUT e FORWARD se encontram a DROP foi necessário colocar dois forwards, sendo que o primeiro se destina à ligação de ida e o outro à ligação de volta.

HTTP, HTTPS and SSH connections

```
iptables -t nat -A POSTROUTING -s 192.168.10.0/24 -p tcp --dport ssh -j SNAT
--to-source 87.248.214.97
iptables -t nat -A POSTROUTING -s 192.168.10.0/24 -p tcp --dport http -j SNAT
--to-source 87.248.214.97
```

```
iptables -t nat -A POSTROUTING -s 192.168.10.0/24 -p tcp --dport https -j SNAT
--to-source 87.248.214.97
```

FORWARDS DE SAÍDA

```
iptables -A FORWARD -s 192.168.10.0/24 -p tcp --dport ssh -o enp0s10 -j ACCEPT
iptables -A FORWARD -s 192.168.10.0/24 -p tcp --dport http -o enp0s10 -j ACCEPT
iptables -A FORWARD -s 192.168.10.0/24 -p tcp --dport https -o enp0s10 -j ACCEPT
```

FORWARDS DE ENTRADA

```
iptables -A FORWARD -d 192.168.10.0/24 -p tcp --sport ssh -i enp0s10 -j ACCEPT
iptables -A FORWARD -d 192.168.10.0/24 -p tcp --sport http -i enp0s10 -j ACCEPT
iptables -A FORWARD -d 192.168.10.0/24 -p tcp --sport https -i enp0s10 -j ACCEPT
```

Este conjunto de regras segue o mesmo princípio que o conjunto de regras anterior, com a única diferença que desta vez estamos a tratar de ligações dos serviços http, https e ssh, as quais ocorrem segundo o protocolo de comunicação tcp.

FTP connections (in passive and active modes) to external FTP servers

```
iptables -t nat -A POSTROUTING -s 192.168.10.0/24 -p tcp --dport 21 -j SNAT
--to-source 87.248.214.97
```

FORWARD do Command Channel

```
iptables -A FORWARD -s 192.168.10.0/24 -p tcp --dport 21 -o enp0s10 -m state
--state NEW,ESTABLISHED -j ACCEPT
iptables -A FORWARD -d 192.168.10.0/24 -p tcp --sport 21 -i enp0s10 -m state
--state ESTABLISHED -j ACCEPT
```

ACTIVE do Data Channel

```
iptables -A FORWARD -d 192.168.10.0/24 -p tcp --sport 20 -i enp0s10 -m state
--state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -s 192.168.10.0/24 -p tcp --dport 20 -o enp0s10 -m state
--state ESTABLISHED -j ACCEPT
```

PASSIVE do Data Channel

```
iptables -A FORWARD -s 192.168.10.0/24 -p tcp -o enp0s10 -m state --state
ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -d 192.168.10.0/24 -p tcp -i enp0s10 -m state --state
ESTABLISHED -j ACCEPT
```

Este cenário de ligações FTP segue o mesmo princípio do que foi descrito acima no caso das ligações FTP ao servidor ftp que se encontra na rede interna, com a diferença de que desta vez as ligações são originárias da rede interna (192.168.10.0/24) e têm como destino um servidor FTP que se encontre na Internet. Logo, não teremos que especificar um endereço IP de destino, apenas os portos utilizados pelos ftp e as devidas regras que permitam ao NAT, através dos módulos carregados (ip_nat_ftp e ip_conntrack_ftp), saber quais os portos que o ftp utiliza.

4.2 Snort

De forma a implementar o Snort foi apenas necessário instalar o mesmo e transferir as regras que este disponibiliza, tendo depois configurado e utilizado algumas regras adicionais. Acedendo ao ficheiro de configuração do Snort, de seu nome **snort.conf**, existem certos parâmetros que tiveram que ser alterados. Começámos por definir os endereços IP das redes que estamos a proteger, sendo estas a rede interna e a DMZ - **ipvar HOME_NET [192.168.10.0/24,23.214.219.128/25]** - e os endereços IP externos, os

quais definimos como sendo todos menos os pertencentes à HOME_NET - **ipvar EXTERNAL_NET !\$HOME_NET**. Foi também necessário definir qual a diretoria onde se encontram os ficheiros das regras que serão usadas - **var RULE_PATH /etc/snort/rules/rules**. Por fim, no *Step 7: Customize your rule set*, o qual se destina ao conjunto de regras que vamos utilizar, foi apenas necessário descomentar os *includes* dos ficheiros que continham as regras que irão permitir travar os ataques pretendidos.

Passando agora para as regras utilizadas, iremos detalhar o propósito das mesmas nas duas subsecções que seguem.

4.2.1 SQL Injections

SQL Injections são um dos tipos de ataques mais comuns na Internet e permitem a execução de pesquisas SQL maliciosas. Estas podem permitir aos atacantes evitar as medidas de segurança da base de dados e obter privilégios que lhes permitam retirar certos direitos aos administradores das bases de dados e apagar todos os dados que estas contêm ou, através destas pesquisas, podem conseguir obter informação privada dos utilizadores de uma organização, sem possuírem quaisquer privilégios dentro da base de dados. É ainda importante referir que existem dois grandes tipos de SQL Injections, sendo estas as **In-band SQLi** e as **Out-of-band SQLi**. No primeiro tipo de ataques, o atacante consegue utilizar o mesmo canal de comunicação para atacar e para recolher os resultados. No segundo, isto já não é possível, sendo que o atacante está dependente das funcionalidades que estão ativas na aplicação web, como por exemplo, a capacidade do servidor de base de dados fazer pedidos dns ou http para enviar dados para o atacante.

Regra 1:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (react:block;msg:"SQL 1 = 1 - possible sql injection attempt"; flow:to_server,established; content:"1%3D1"; fast_pattern:only; http_client_body; pcre:"/or\++1%3D1/Pi"; metadata:policy balanced-ips drop, policy max-detect-ips drop, policy security-ips drop, service http; reference:url,attack.mitre.org/techniques/T1190; reference:url,ferruh.mavituna.com/sql-injection-cheatsheet-oku/; classtype:web-application-attack; sid:30040; rev:4;)
```

Esta regra protege contra **String SQL Injections**, **Numerical SQL Injections** e **Blind SQL Injections**, do tipo **Boolean-based**. Se as pesquisas forem construídas dinamicamente através da concatenação de strings podemos estar sujeitos a **String SQL Injections**, visto que se essa mesma string for depois inserida numa pesquisa SQL como parâmetro, podemos tentar modificar a pesquisa original, de forma a ignorar parte desta. Podemos assim tentar colocar uma cláusula que seja avaliada para True, a qual irá retornar todos os dados de uma tabela da base de dados em vez de um único registo em específico. **Numerical SQL Injections** seguem o mesmo propósito, mas com números. Por fim, as **Blind SQL Injections** são um tipo de ataques mais moroso visto que recorrem à inferência, uma vez que não são transferidos quaisquer dados através da aplicação web. Deste modo, o atacante apenas consegue reconstruir a base de dados através do envio de payloads e observando o comportamento da base de dados, nas suas respostas. Nas **Boolean-based** o que acontece é que o atacante envia pesquisas que irão forçar a aplicação a retornar

verdadeiro ou falso, sendo que desta forma o atacante conseguirá compreender se a payload utilizada retornou verdadeiro ou falso.

Assim sendo, esta regra irá permitir procurar por mensagens que contenham `1=1` no pedido http, vindo de qualquer endereço IP que não seja o da HOME_NET e através do porto http e irá bloquear os tipos de ataques referidos.

Regra 2:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (react:block;msg:"SQL union select - possible sql injection attempt - POST parameter";  
flow:to_server,established; content:"union"; fast_pattern:only; http_client_body;  
content:"select"; nocase; http_client_body;  
pcr: "/union(%20|\+)+(all(%20|\+)+)?select(%20|\+)/Pi"; metadata:policy  
max-detect-ips drop, policy security-ips drop, service http;  
reference:url,attack.mitre.org/techniques/T1190; classtype:misc-attack; sid:15874;  
rev:14;)
```

Esta regra destina-se a detetar **Union Based SQL Injections**. Este tipo de ataques permite ao atacante obter informação de outras tabelas da base de dados através da utilização da instrução **Union**, a qual permite adicionar um SELECT adicional (desde que este contenha os mesmos argumentos que o primeiro SELECT) e concatenar os resultados da pesquisa adicional à pesquisa original. Isto pode fazer com que o atacante obtenha informação sobre nomes de utilizadores ou palavras-passe de outras tabelas da base de dados às quais este não deveria ter acesso.

Assim sendo, a forma como esta regra protege contra este tipo de ataques é através da utilização de uma expressão regular que procura por correspondências no pedido http que contenham as keywords union e select de entre os restantes caracteres do pedido.

Regra 3:

```
alert tcp any any -> any $HTTP_PORTS (react:block;msg:"Modified regex for  
detection of SQL meta-characters";flow:to_server,established;  
pcr: "/((\%3D)|(\=))[^\n]*((\%27)|(\`)|(\-\\-)|(\%3B)|(\;))/i";classtype:Web-appli  
cation-attack; sid:910000;rev:5;)
```

Esta regra destina-se à deteção de **Error-based SQL Injections** e também de **SQL Query Chaining Injections**. **Error-based SQL Injections** são um tipo de ataques que se aproveitam das mensagens de erro apresentadas pelo servidor de base de dados para tentar compreender qual a pesquisa SQL que é efetuada e qual a estrutura da base de dados. Por outro lado, **SQL Query Chaining Injections** são outro tipo de ataques, no qual é ignorada parte da pesquisa SQL e utilizado o `'` para efetuar pesquisas adicionais na base de dados, as quais podem não só permitir ter acesso a informação de outros utilizadores como também eliminar informação dos mesmos ou conseguir direitos de administrador da base de dados.

Como tal, de forma a proteger contra este tipo de ataques efetuamos a deteção de metacaracteres que sejam inseridos pelo utilizador (tais como, `'`, `--` ou `;`), o que evita assim que a base de dados envie mensagens de erro que poderiam revelar informação sobre a estrutura da mesma e o que evita também que se adicionem pesquisas adicionais à pesquisa original.

Regra 4:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (react:block; msg:"SQL generic sql update injection attempt - POST parameter";  
flow:to_server,established; content:"update+";  
fast_pattern:only;http_client_body; content:"set"; nocase; http_client_body;  
pcr:"/update\[^\&\n\]+?(%20|\+)\set(%20|\+)/Pi"; metadata:policy max-detect-ips drop, policy security-ips drop, service http;  
reference:url,msdn.microsoft.com/en-us/library/ms161953.aspx;  
classtype:web-application-attack; sid:15876; rev:13;)
```

Esta regra é utilizada em conjunto com a regra 3, de forma a impedir que sejam efetuadas quaisquer atualizações (update) das tabelas das bases de dados, de forma a garantir a integridade dos dados dos seus utilizadores.

4.2.2 XSS application-layer attacks

Os ataques de cross-site scripting, também dos mais comuns, são ataques direcionados a páginas web, nas quais são exploradas falhas e falta de proteção/codificação dos inputs dos utilizadores. Isto permite que os atacantes consigam manipular as páginas web, utilizando as mesmas como “ponte” para correr scripts nos browsers de outros utilizadores/vítimas, que o fazem de maneira “involuntária”.

Estes ataques dividem-se em dois grandes ramos: **non-persistent** e **persistent**. No **persistent**, o atacante encontra um site vulnerável, colocando um link no mesmo e levando o utilizador a clicar e a correr o script. Por outro lado, no **non-persistent**, este script está guardado na página web, como, por exemplo, num comentário de um fórum, bastando apenas que o utilizador abra a página para que o script seja executado.

Como é óbvio, após ser possível correr um script num browser de uma vítima, várias informações podem ser roubadas e manipuladas, como, por exemplo, os seus cookies de sessão.

Regra 1:

```
alert tcp any any -> any $HTTP_PORTS (react:block;msg:"NIICross-site scripting attempt";flow:to_server,established;pcr:"/((\%3C)|<)((\%2F)|\/)*[a-z0-9\%]+((\%3E)|>)/i";classtype:web-application-attack; sid:9000; rev:5;)
```

Esta regra destina-se à deteção de scripts na forma de html (na sua forma mais básica), que sejam colocados em pedidos http, bloqueando então um dos vários ataques **XSS Reflected**. Para isso, é preciso o uso de uma expressão regular que contenha todas as variações possíveis deste ataque, ou seja, as várias combinações dos caracteres e até os valores hexadecimais que compõem as opening e closing tags de html.

Regra 2:

```
alert tcp any any -> any $HTTP_PORTS (react:block;msg:"NIICross-site scripting attempt2";flow:to_server,established;pcr:"/((\%3C)|<)((\%69)|i|(\%49))((\%6D)|m|(\%4D))((\%67)|g|(\%47))[\^\\n]+((\%3E)|>)/i"; classtype:web-application-attack; sid:900120; rev:5;)
```

De igual forma, esta regra permite-nos detetar outra variação do Reflected cross-site

scripting, na qual são colocados scripts dentro do elemento html img (<img src=). Mais uma vez, são tidas em consideração as várias combinações possíveis deste ataque na expressão regular, ou seja, os caracteres que fazem parte da palavra img, em formato ASCII e hexadecimal.

5. Testes efetuados

5.1 IPTables

De forma a verificar que as regras implementadas permitem apenas estabelecer as ligações pretendidas, foram efetuados testes utilizando a ferramenta **netcat**. Na máquina que estaria à escuta foi efetuado o comando **nc -lv (-u) <porto>** e na máquina a partir da qual pretendíamos iniciar a ligação utilizámos o comando **nc -v (-u) <endereço IP de destino><porto de destino>**. Dito isto, apresentamos de seguida 4 exemplos deste tipo de testes, sendo 2 relacionados com as ligações FTP passivas e ativas (SNAT e DNAT), outro relacionado com ligações entre a rede interna e a DMZ e outro entre a DMZ e a Internet.

Sendo o FTP um caso à parte, foi necessário instalar o ftp (cliente ftp) e o vsftpd (servidor ftp), tanto na rede interna como na Internet. No nosso caso de teste de ligação ao um servidor FTP externo, colocámos o servidor FTP da internet no endereço 87.248.214.214. Quanto à ligação ao FTP na rede interna, foi apenas necessário efetuar o comando ftp 87.248.214.97 (endereço IP do router), uma vez que através das nossas regras de DNAT, os pacotes serão devidamente encaminhados para o servidor ftp.

```
[root@localhost ~]# ftp 87.248.214.97
Connected to 87.248.214.97 (87.248.214.97).
220 (vsFTPd 3.0.2)
Name (87.248.214.97:root): joaotomas
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
227 Entering Passive Mode (87,248,214,97,36,236).
150 Here comes the directory listing.
drwxr-xr-x  4 1000    1000    96 Mar 23 12:11 Desktop
drwxr-xr-x  2 1000    1000    6 Feb 12 21:59 Documents
drwxr-xr-x  2 1000    1000    89 Mar 23 12:07 Downloads
drwxr-xr-x  2 1000    1000    6 Feb 12 21:59 Music
drwxr-xr-x  2 1000    1000    6 Feb 12 21:59 Pictures
drwxr-xr-x  2 1000    1000    6 Feb 12 21:59 Public
drwxr-xr-x  2 1000    1000    6 Feb 12 21:59 Templates
drwxr-xr-x  2 1000    1000    6 Feb 12 21:59 Videos
226 Directory send OK.
ftp> passive
Passive mode off.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxr-xr-x  4 1000    1000    96 Mar 23 12:11 Desktop
drwxr-xr-x  2 1000    1000    6 Feb 12 21:59 Documents
drwxr-xr-x  2 1000    1000    89 Mar 23 12:07 Downloads
drwxr-xr-x  2 1000    1000    6 Feb 12 21:59 Music
drwxr-xr-x  2 1000    1000    6 Feb 12 21:59 Pictures
drwxr-xr-x  2 1000    1000    6 Feb 12 21:59 Public
drwxr-xr-x  2 1000    1000    6 Feb 12 21:59 Templates
drwxr-xr-x  2 1000    1000    6 Feb 12 21:59 Videos
226 Directory send OK.
ftp>
```

Figura 2 - Ligação FTP ao servidor ftp na rede interna

```
[root@localhost ~]# ftp 87.248.214.214
Connected to 87.248.214.214 (87.248.214.214).
220 (vsFTPd 3.0.2)
Name (87.248.214.214:root): joaotomas
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
227 Entering Passive Mode (87,248,214,214,126,224).
150 Here comes the directory listing.
drwxr-xr-x  4 1000    1000    96 Mar 23 12:11 Desktop
drwxr-xr-x  2 1000    1000    6 Feb 12 21:59 Documents
drwxr-xr-x  2 1000    1000    89 Mar 23 12:07 Downloads
drwxr-xr-x  2 1000    1000    6 Feb 12 21:59 Music
drwxr-xr-x  2 1000    1000    6 Feb 12 21:59 Pictures
drwxr-xr-x  2 1000    1000    6 Feb 12 21:59 Public
drwxr-xr-x  2 1000    1000    6 Feb 12 21:59 Templates
drwxr-xr-x  2 1000    1000    6 Feb 12 21:59 Videos
226 Directory send OK.
ftp> passive
Passive mode off.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxr-xr-x  4 1000    1000    96 Mar 23 12:11 Desktop
drwxr-xr-x  2 1000    1000    6 Feb 12 21:59 Documents
drwxr-xr-x  2 1000    1000    89 Mar 23 12:07 Downloads
drwxr-xr-x  2 1000    1000    6 Feb 12 21:59 Music
drwxr-xr-x  2 1000    1000    6 Feb 12 21:59 Pictures
drwxr-xr-x  2 1000    1000    6 Feb 12 21:59 Public
drwxr-xr-x  2 1000    1000    6 Feb 12 21:59 Templates
drwxr-xr-x  2 1000    1000    6 Feb 12 21:59 Videos
226 Directory send OK.
ftp>
```

Figura 3 - Ligação FTP a um servidor ftp na Internet

```
[root@localhost network-scripts]# nc -zv 192.168.10.2 5432
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Connected to 192.168.10.2:5432.
Ncat: 0 bytes sent, 0 bytes received in 0.02 seconds.
```

```
[root@localhost network-scripts]# nc -lv 5432
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Listening on :::5432
Ncat: Listening on 0.0.0.0:5432
Ncat: Connection from 23.214.219.129.
Ncat: Connection from 23.214.219.129:53546.
```

Figuras 4 e 5 - Ligação ao serviço PostgreSQL na datastore a partir da VPN gateway

```
[root@localhost ~]# nc -uv 87.248.214.75 53
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Connected to 87.248.214.75:53.
teste
```

```
[root@localhost ~]# nc -uv 87.248.214.75 53
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Connected to 87.248.214.75:53.
teste
```

Figuras 6 e 7 - Teste de comunicação entre o dns e o dns2

5.2 Snort

Para que fosse possível testar as regras utilizadas e de forma a verificar se o Sistema de Detecção de Intrusões **Snort** conseguia identificar corretamente as mesmas e travar os diferentes ataques aos quais foi exposto, definimos um cenário no qual temos um servidor web com vulnerabilidades propositadas (WebGoat) a correr no endereço IP correspondente ao do servidor www da rede DMZ. Os ataques foram efetuados através de um browser na máquina virtual referente à Internet. Para que fosse possível efetuar estes ataques sem que a firewall travasse os mesmos, tomámos partido do envio de pedidos http pelo porto 80, a partir do browser até ao servidor web referido. Decidimos usar este servidor web visto que possui as vulnerabilidades que pretendemos testar no âmbito do nosso projeto (SQL injections e XSS application-layer attacks), permitindo assim compreender como tomar partido das mesmas e também como mitigá-las. No meio deste cenário irá estar então o Snort a correr no router, estando atento aos pacotes trocados entre as máquinas, para que possa detetar os diferentes ataques que se encontrem em curso e bloqueá-los.

5.2.1 SQL Injections

Para que fosse possível averiguar se o nosso IDS estava a travar corretamente os ataques, começámos por efetuar os diferentes tipos de ataques (nas *lessons* que o WebGoat nos disponibiliza sobre SQL Injections) que referimos quando definimos as regras na secção 4.2.1, sem este estar em execução, conseguindo fazer os diversos ataques com sucesso. De seguida, ao tentar executar as mesmas instruções, mas desta vez com o IDS em pleno funcionamento reparámos que os nossos ataques foram travados e que foram acionadas as regras devidas, consoante o diferente tipo de ataque.

Ataque do tipo Union based SQL Injection

Ao colocarmos a seguinte pesquisa ' or 1=1 union select userid, user_name, password, null, null, null, 0 from user_system_data-- , iremos conseguir retirar dados sobre dados dos utilizadores que se encontrem na tabela user_system_data, como podemos ver na Figura 8. No entanto, com o Snort a correr, este ataque é travado, como podemos ver na Figura 9.

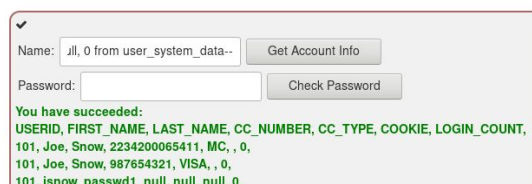


Figura 8 - Representação do ataque do tipo Union based SQL Injection

```
04/25-22:46:37.709739  [**] [1:910000:5] Modified regex for detection of SQL meta-characters [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 87.248.214.214:40014 -> 23.214.219.130:80
04/25-22:46:37.709739  [**] [1:30040:4] SQL 1 = 1 - possible sql injection attempt [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 87.248.214.214:40014 -> 23.214.219.130:80
04/25-22:46:37.709739  [**] [1:15874:14] SQL union select - possible sql injection attempt - POST parameter [**] [Classification: Misc Attack] [Priority: 2] {TCP} 87.248.214.214:40014 -> 23.214.219.130:80
```

Figura 9 - Linha de comandos com as diferentes regras despoletadas pelo ataque

Ataque do tipo SQL Query Chaining Injection

Efetuada a seguinte pesquisa `' ;update employees set salary=1000000 where last_name='Smith' --` no campo “Employee name” (Figura 10) não iremos conseguir atualizar o salário do mesmo, uma vez que este ataque é detetado como apresentado na Figura 11.

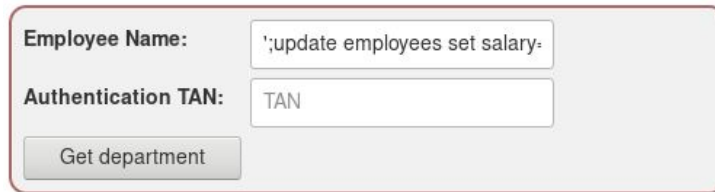


Figura 10 - Representação do ataque do tipo SQL Query Chaining Injection

```
04/25-22:54:56.950450  [**] [1:910000:5] Modified regex for detection of SQL meta-characters [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 87.248.214.214:40028 -> 23.214.219.130:80
04/25-22:54:56.950450  [**] [1:15876:13] SQL generic sql update injection attempt - POST parameter [*] [Classification: Web Application Attack] [Priority: 1] {TCP} 87.248.214.214:40028 -> 23.214.219.130:80
```

Figura 11 - Regras acionadas em resposta ao ataque efetuado

Ataque do tipo Numerical SQL Injection

Neste tipo de ataque, colocamos uma pesquisa que será sempre avaliada como verdadeiro, o que nos permitirá obter informação de outras tabelas da base de dados. No entanto, como vemos na Figura 13, conseguimos detetar este ataque com sucesso e bloqueá-lo.

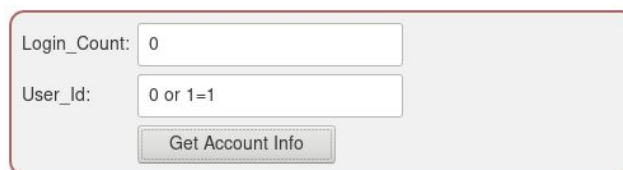


Figura 12 - Representação do ataque do tipo Numerical SQL Injection

```
04/25-22:57:11.233890  [**] [1:30040:4] SQL 1 = 1 - possible sql injection attempt [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 87.248.214.214:40030 -> 23.214.219.130:80
```

Figura 13 - Regras acionadas em resposta à tentativa de intrusão

Ataque do tipo Error-based SQL Injection

Como podemos observar na Figura 14, ao colocar um metacaracter o input field “User_Id”, a aplicação revela um erro que não deveria apresentar, no qual nos revela informação crucial para saber qual a estrutura da pesquisa SQL, podendo assim tirar partido desta vulnerabilidade. No entanto, com o Snort em execução, conseguimos tratar este ataque com sucesso, como podemos ver na Figura 15.

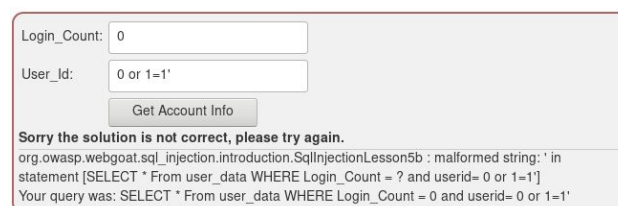


Figura 14 - Representação do ataque do tipo Error-based SQL Injection

```
04/26-18:54:21.324547  [**] [1:910000:5] Modified regex for detection of SQL meta-characters [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 87.248.214.75:47682 -> 23.214.219.130:80
04/26-18:54:21.324547  [**] [1:30040:4] SQL 1 = 1 - possible sql injection attempt [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 87.248.214.75:47682 -> 23.214.219.130:80
```

Figura 15 - Regras acionadas em resposta à tentativa de ataque efetuada

5.2.2 XSS application-layer attacks

Quanto aos ataques de cross-site scripting, foi seguida a mesma abordagem que foi referida para as SQL Injections, só que neste caso realizámos os ataques nas *lessons* destinadas aos ataques de cross-site scripting. Mais uma vez conseguimos detetar com sucesso os diferentes tipos de ataques que pretendíamos travar, como podemos observar nas figuras que seguem.

Basic XSS Test Without Filter Evasion

Ao tentarmos executar o seguinte script no input field destinado ao número de cartão de crédito - `<script>alert("XSS Test")</script>` - conseguimos obter o resultado observado na Figura 17. No entanto, tendo o Snort em execução, o ataque é travado com sucesso (Figura 18).

Figura 16 - Representação do ataque XSS

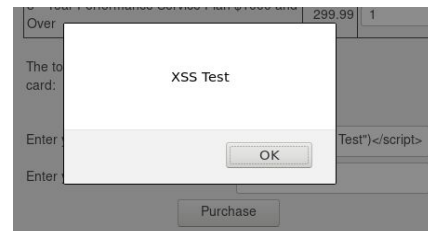


Figura 17 - Sucesso do ataque XSS

```
04/26-00:12:46.110698  [**] [1:9000:5] NIICross-site scripting attempt [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 87.248.214.214:40068 -> 23.214.219.130:80
```

Figura 18 - Regra do Snort acionada face ao ataque em curso

Image XSS using on error alert

Tomando partido do mesmo input field utilizado no ataque anterior, podemos também utilizar a `img` tag do HTML para executar o seguinte ataque - `` - obtendo o resultado apresentado na Figura 20. No entanto, o Snort consegue detetar o mesmo e travar este ataque, como apresentado na Figura 21.

Figura 19 - Representação do ataque XSS



Figura 20 - Sucesso do ataque XSS

```
04/26-00:17:02.529926  [**] [1:900120:5] NIICross-site scripting attempt2 [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 87.248.214.214:40076 -> 23.214.219.130:80
```

Figura 21 - Regra do Snort que permite travar o ataque XSS em questão

Conclusão

Com a realização deste trabalho prático conseguimos adquirir uma visão mais aprofundada sobre como deve ser utilizada uma firewall para detetar e reagir a ataques destinados às redes que esta protege, utilizando IPTables e um sistema de deteção de intrusões, o Snort, de forma a permitir a passagem das comunicações estritamente necessárias, reduzindo assim a possibilidade de comunicações não desejadas. Compreendemos também o quão fácil será atacar um website que não valide corretamente as inputs dos utilizadores, ficando cientes da importância da existência de um IDS que consiga garantir a proteção dos sistemas fulcrais de uma organização (mesmo quando estes se encontram vulneráveis) contra ataques não desejados, os quais podem não só comprometer o correto funcionamento da mesma, como também pôr em causa os seus dados confidenciais ou os dos seus clientes.

Bibliografia

- [1] <https://zupzup.org/websecurity-with-webgoat/>
- [2] http://www.tcpipguide.com/free/t_TCPIPClientEphemeralPortsandClientServerApplication-3.htm
- [3] <https://slacksite.com/other/ftp.html>
- [4] https://www.karlrupp.net/en/computer/nat_tutorial
- [5] <https://netfilter.org/documentation/HOWTO/pt/NAT-HOWTO-7.html>
- [6] <https://github.com/WebGoat/WebGoat/releases>