Universidade da Beira Interior
Faculdade de Engenharia
Departamento de Informática

Segurança Informática

Aula 10

Licenciatura em Engenharia Informática
Licenciatura em Informática Web

Computer Security

Lecture 10

Degree in Computer Science and Engineering
Degree in Web Informatics

Sumário

Arquitetura e funcionamento de uma *firewall*. Discussão dos conceitos de zona desmilitarizada (*DeMilitarized Zone* (DMZ)) e tradução de endereços de rede (*Network Address Translation* (NAT)). Modelos de operação deste tipo de sistemas e serviços que proporciona.
Análise da definição e perfil de intrusões, e discussão acerca da arquitetura e classificação de Sistemas de Deteção de Intrusões (*Intrusion Detection Systems* (IDSs)).

Summary

Architecture and functioning of a firewall. Discussion of the concepts of Demilitarized Zone (DMZ) and Network Address Translation (NAT). Operational models and services provided by this type of systems.
Analysis of the definition and profile of intrusions, and discussion concerning the architecture and classification of of Intrusion Detection Systems (IDSs).

1 Firewalls

Firewalls

1.1 Introdução e Definição

Introduction and Definition

As *firewalls* (palavra por vezes traduzida para *barreira contra fogo* em Português), ou pelo menos a designação pela qual são conhecidas, surgiram na década de 1990. Na sua iteração inicial, uma *firewall* era **apenas definida no contexto de rede de computadores**, mas hoje em dia já é comum **existirem *firewalls* pessoais**.

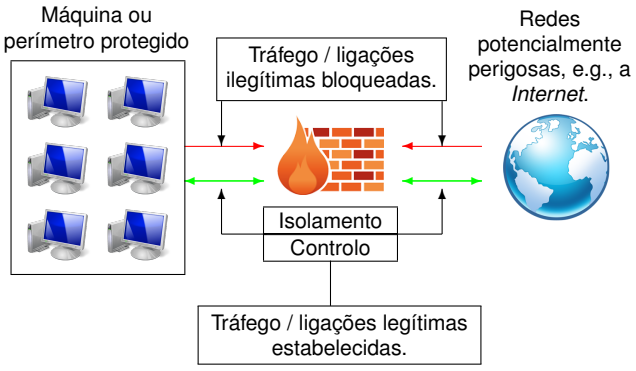
> Uma *firewall* é **um sistema de componentes de *hardware*** (máquinas, redes e equipamentos de interligação como *routers* ou *switches*) **e de *software*** (filtros, monitores, etc.) desenhado com o intuito de **restringir o acesso dentro de uma rede ou entre várias redes**, nomeadamente entre a *Internet* e uma rede privada. As *firewalls* têm, assim, **dois objetivos principais**, ambos relacionados com segurança:
>
> 1. **Proteção por isolamento** de máquinas ligadas à rede;
> 2. **Controlo de interações** entre máquinas.

A *firewall* **concretiza e é parte de uma política de segurança** para **defesa de perímetro**, implementada com o objetivo de proteger os recursos informáticos de uma organização. Por outras palavras, como inicialmente definida, uma *firewall* não é mais do que **um sistema guardião colocado à entrada/saída da rede ou sistema que se quer proteger** e que combina um **conjunto de tecnologias** que permitem estabelecer a defesa de perímetro.

2 Posicionamento e Operação de uma *Firewall*

Positioning and Operation of a Firewall

A figura seguinte ilustra o **posicionamento e responsabilidades ou funcionalidades de uma *firewall***. Esta ilustração aplica-se melhor no contexto de uma **rede de computadores**, embora a parte esquerda possa ser substituída por uma só máquina terminal, transpondo a definição para a de *firewall* pessoal.



Uma *firewall* acumula diversas responsabilidades e funcionalidades perante uma rede ou sistema, nomeadamente (i) as de **implementar políticas de segurança** num **ponto único da rede**; (ii) **monitorizar eventos ou incidentes** relacionados com segurança e fazer o seu registo (***log***); (iii) disponibilizar **meios de autenticação e**

garantia de aplicação de controlo de acesso.

Para além das responsabilidades acima indicadas, torna-se necessário dizer que uma *firewall* **não tem**, por construção, a responsabilidade de **proteger contra ataques que a rodeiam ou ultrapassam**, ou contra **ameaças internas** à rede protegida (a não ser que a rede esteja dividida em sub-redes, cada uma delas protegidas por outras *firewalls*). I.e., não tem a responsabilidade de proteger contra empregados insatisfeitos, ou um infiltrado a colaborar com um atacante externo à rede. Também não é normalmente da sua responsabilidade **evitar a transferência de ficheiros infetados com vírus**.

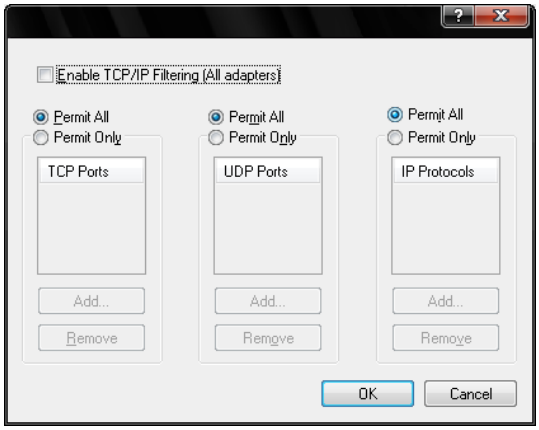
2.1 *Firewall* Pessoal
Personal Firewall

A discussão anterior relativa ao conceito de *firewall* faz mais sentido quando se considera que o perímetro a proteger é uma infra-estrutura complexa formada por diversas máquinas interligadas por redes. Contudo, a massificação do uso do computador pessoal e da sua ligação à Internet levou ao desenvolvimento de ***firewalls* pessoais**, que se **destinam a proteger uma única máquina e fazem parte do sistema dessa máquina** (e.g., estão embutidos no sistema operativo). Neste caso, **o perímetro é o próprio sistema** e **o comprometimento do sistema corresponde ao comprometimento da *firewall***. Estas *firewalls* pessoais não são tipicamente simples de configurar por parte de um utilizador normal.

A partir da secção **??**, a explicação aplica-se, sobretudo, a *firewalls* não pessoais, cujo funcionamento tende a ser bem mais complexo que as mencionadas anteriormente. Sem perda de generalização, algumas das considerações feitas adiante também se aplicam normalmente a essas *firewalls*, embora possam ser particularizadas para esta tipologia. De modo a dar um breve enquadramento a *firewalls* pessoais, e deixar depois o foco livra para as *firewalls* de rede, discute-se, na próxima secção e brevemente, a *firewall* do Windows 2003.

2.2 Caso de Estudo: Sistemas Microsoft Windows
Study Case: Microsoft Windows Systems

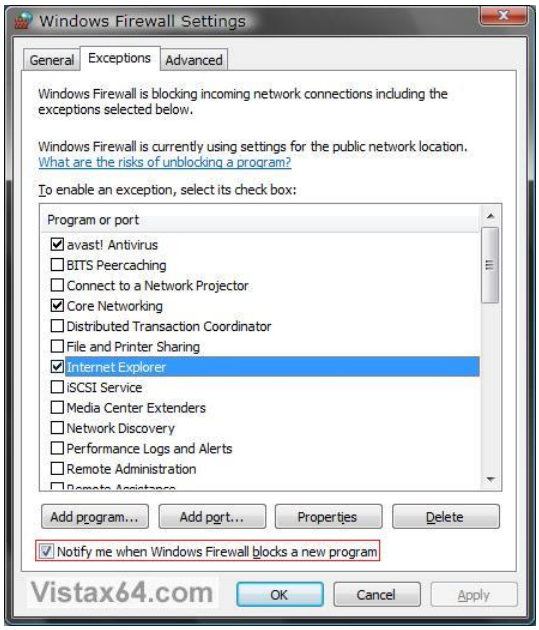
O núcleo Linux suporta nativamente a filtragem de pacotes *Internet Protocol* (IP), cujas regras podem ser configuradas e postas em execução usando o `iptables`, que será estudado nesta aula mais adiante. Estas funcionalidades podem ser usadas em *firewalls* pessoais ou de rede. No caso dos **sistemas operativos da família Windows**, também é possível encontrar *firewalls* (ou funcionalidades de) **desde a versão 2000** (Windows 2000), integradas por necessidade, após vários problemas com a propagação automática de *worms*, nomeadamente os *worms Blaster* e *Sasser*. Estas *firewalls* são suficientes para travar estas ameaças atualmente. A figura seguinte mostra um dos diálogos disponíveis para configurar a *firewall* pessoal do Windows. Como se pode deduzir da



figura, é **aplicada uma política de**:

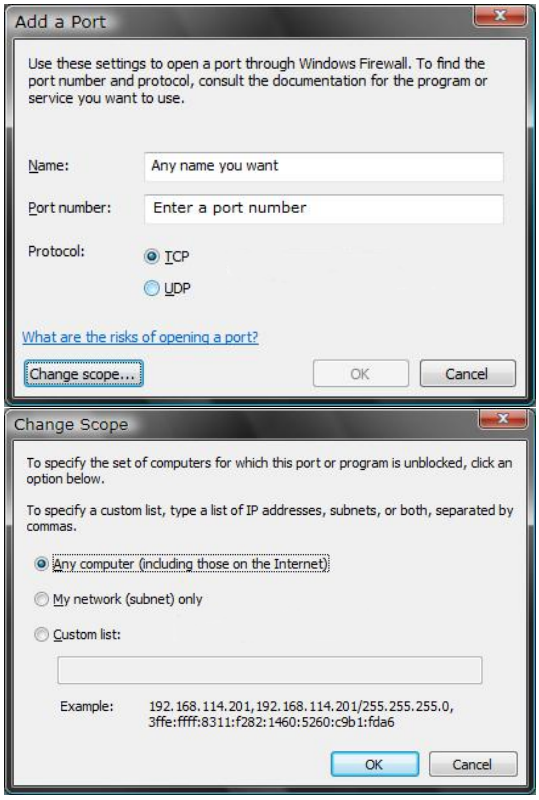
- Ou é **tudo aceite**;
- Ou não é **nada aceite, exceto** as ligações que forem diretamente especificadas como sendo permitidas.

Uma das funcionalidades disponibilizada é a de **limitar o número de portos abertos na máquina**. Esta funcionalidade aplica-se, em muitos dos sistemas operativos desta família, **simultaneamente para todos os *interfaces* de rede** que a máquina possa ter. Para além disso, **só se pode limitar tráfego de entrada (não de saída)**. Adicionalmente, só se podem especificar **portos da camada de transporte locais** (nada pode ser dito acerca dos portos remotos). Também **não se pode fazer filtros para mensagens mal-formadas** (e.g., como as que o `nmap` faz).



Uma segunda funcionalidade (***Internet Connection Firewall*** (ICF)) foi introduzida no sistema Windows XP e nos que lhe seguiram. Este serviço vem **ativo por defeito**, e é mais **simples de configurar**, já que o utilizador pode **escolher as aplicações (em vez das portas)** que podem aceder à rede, conforme se ilustra na figura anterior.

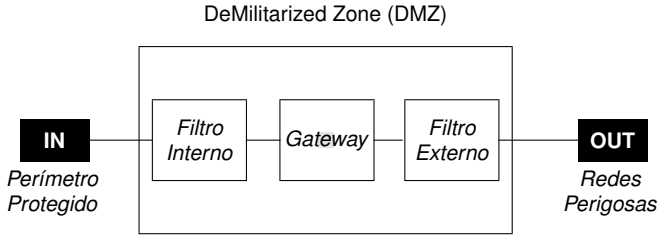
Nesta nova abordagem, continua a ser possível especificar portas *Transmission Control Protocol* (TCP) / *User Datagram Protocol* (UDP) ou até os **endereços IP para os quais os serviços estarão disponíveis**, conforme se ilustra a seguir. Há também **menus de configuração** específicos para datagramas *Internet Control Message Protocol* (**ICMP**) (e.g., para **desativar respostas a PINGs**). A *firewall* também pode guardar um registo (*log*) num ficheiro à escolha.



3 Arquitetura e Topologias de uma *Firewall*

Architecture and Topologies of a Firewall

De um modo generalista e básico, pode-se dizer que **uma *firewall* é composta por uma *gateway*, dois filtros e uma rede de interligação de todas estas componentes**, normalmente designada por zona desmilitarizada (mais conhecida pela designação inglesa *DeMilitarized Zone* (DMZ)).



Uma das designações que mais se usa e é enfatizada neste contexto é a de *Gateway*. **Uma *Gateway* não é mais do que uma ou mais máquinas com a função de controlar e encaminhar corretamente a comunicação IN-OUT**. Os filtros, que também se ilustram na figura, destinam-se a fazer **alguma filtragem do tráfego autorizado a passar pela *firewall***, mas **sobretudo a proteger** esta *gateway*, para que não possa ser contactada diretamente ou ficar demasiado exposta. **Caso** a *gateway* esteja **diretamente exposta a ataques**, então chama-se **máquina-bastião**.

3.1 Topologias de *Firewalls*

Firewalls Topologies

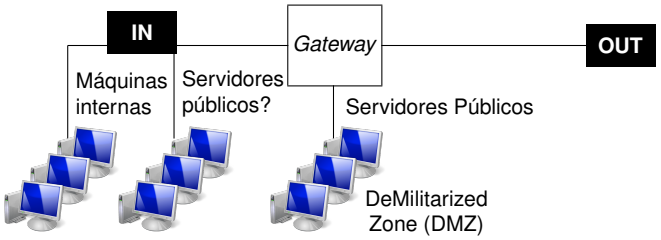
Na literatura da especialidade, descrevem-se normalmente **3 topologias elementares** para *firewalls*, embora não se apliquem necessariamente e em toda a sua plenitude:

1. Na topologia ***Gateway* simples** (*dual-homed gateway*), a *firewall* é **uma só máquina com duas ou mais placas de rede**, **devidamente robustecida** com *software* de *firewall* e **despida de serviços** desnecessários.
2. Na topologia **Máquina escondida** (*screened host*), a *firewall* é constituída por **um *router*** que liga a rede protegida ao exterior e por **uma *gateway* ligada à rede interior** por uma única placa de rede.
3. Na topologia **Sub-rede escondida** (*screened subnet*), a *firewall* é constituída por **dois *routers***, um que liga a rede protegida a uma DMZ e outro que liga a DMZ ao exterior, e por **uma *gateway***.

3.2 Topologia *Gateway* Simples

Topology Dual-Homed Gateway

A topologia de *Gateway* simples **congrega todos os elementos** ilustrados na figura anterior **numa única máquina** e encontra-se ilustrada em baixo.

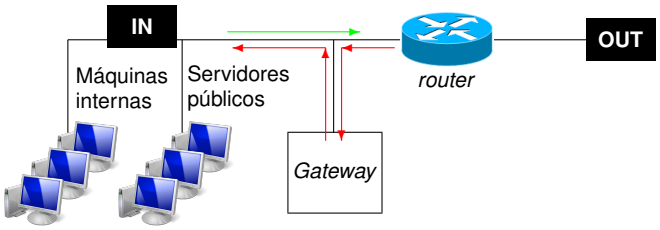


Esta topologia é uma das mais populares para este tipo de sistema, sobretudo por ser a **mais simples** e **mais barata** de implementar. O laboratório de redes na sala 6.27 implementa esta topologia. As grandes **desvantagens** associadas a esta topologia é que **o comprometimento do *gateway*-bastião corresponde comprometimento da *firewall*** e a carga de **processamento** está toda **numa única máquina**. Note que, na figura, é mostrado como se **pode** construir uma **DMZ completamente segregada** das restantes redes e máquinas, à custa da **colocação de outra placa de rede na *Gateway***, mas isso acarreta custos computacionais acrescidos.

3.3 Topologia Máquina Escondida

Screened Host Topology

Na topologia de máquina escondida, a ***gateway*** está ligada à rede por **uma única *interface* de rede** (por isso é uma máquina escondida). Todo o **tráfego de entrada** autorizado pelo *router* **passa primeiro pela *gateway***, que trata da sua filtragem, da alteração e encaminhamento para máquinas interiores. O **tráfego de saída é apenas filtrado pelo *router***.

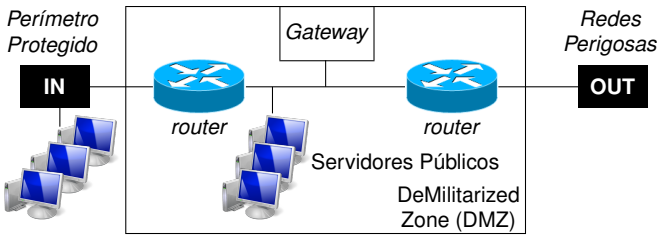


Esta topologia **permite balancear a carga** entre o *router* e a *gateway*. Neste caso, o *router* pode atuar como **filtro de pacotes**, enquanto que **a *gateway* pode operar como filtro de circuitos ou aplicacional para tráfego de entrada**. A **principal desvantagem** é que **não isola convenientemente os serviços públicos**, que ficam ligados à mesma rede que as restantes máquinas.

3.4 Topologia Sub-Rede Escondida

Screened Subnet Topology

Na topologia de sub-rede escondida, a **mais elaborada e potencialmente segura** de todas, a ***firewall* é constituída por dois *routers* e por um *gateway***, conforme se ilustra a seguir. Um dos *routers* **liga a DMZ à rede externa**, o outro **liga a DMZ à rede interna**. **A DMZ é**, é neste caso, **a sub-rede escondida**. É na DMZ que são colocados os servidores públicos, para minimizar os riscos do seu comprometimento.



A **grande vantagem** da topologia é a sua **flexibilidade**. **Todo o tráfego** interno, externo ou proveniente da DMZ **pode ser filtrado pela *gateway* ou por**, pelo menos, **um dos *routers***. O tráfego da DMZ **pode ser isolado do resto das redes**, protegendo-as com mais eficácia de ameaças externas.

A principal **desvantagem** é que **o tráfego vindo da rede protegida passa pela DMZ**, e impera algum cuidado para evitar que isso constitua um problema.

3.5 Zona Desmilitarizada

DeMilitarized Zone (DMZ)

Como visto antes, a **DMZ é a rede inerente à *firewall***. No caso da última topologia ilustrada, a DMZ encontra-se limitada pelos dois *routers* que servem como filtros internos e externos à rede que se quer proteger. Contudo, **é também comum atribuir a denominação de DMZ a qualquer rede da organização onde se colocam servidores que podem ser contactados do exterior** e que não possuam sistemas ou aplicações confiáveis (e.g., servidor *HyperText Transfer Protocol* (HTTP) ou *Simple Mail Transfer Protocol* (SMTP)). Normalmente **parte-se do pressuposto que estas máquinas podem ser comprometidas** e que são, portanto, **consideradas sacrificáveis** (i.e., **a máquina não deve possuir informação que não possa ser reposta na íntegra posterior** e rapidamente em caso de comprometimento).

Como as máquinas sacrificáveis podem comprometer outras máquinas da rede interna, importa limitar o seu papel na rede onde estão inseridas. Por exemplo, pode optar-se por se **isolar máquinas sacrificáveis em DMZs separadas**, usando *Local Area Networks* (LANs) ou *Virtual* LANs (VLANs) diferentes; ou então **limitar o tráfego única e exclusivamente àquele que é esperado** para as aplicações que correm nos servidores. Um bom exemplo de uma rede que deve ser sempre considerada uma DMZ é uma rede de acesso para equipamentos móveis, que podem ser acedidas por clientes não autorizados (portanto, esta rede é perigosa).

3.6 Modelos de Operação de uma *Firewall*

Firewall Operational Models

Em termos **de modo de funcionamento de *firewalls***, distinguem-se tipicamente os **três modelos** seguintes:

1. **Filtro de pacotes**, que será discutido em mais pormenor nas secções seguintes.
2. **Filtro de circuitos**, em que **a *gateway* estabelece as ligações TCP em vez de qualquer máquina da rede protegida**, depois de verificar que este último tem as permissões necessárias para o fazer e que a ligação remota é estabelecida.
3. **Filtro aplicacional**, em que a *gateway* tem vários *proxies* a correr, e medeia todas as comunicações entre um cliente e um servidor, **fazendo-se passar pelo servidor** para qualquer computador dentro da rede protegida.

3.7 *Firewalls* Filtro de Pacotes

Packet Filtering Firewalls

O **primeiro (e mais básico) tipo** de *firewalls* implementa um modelo de intervenção **baseado na filtragem de pacotes**. Neste modelo, as **decisões são tomadas pacote-a-pacote** (IP). Este modelo de operação pode

ainda ser **dividido em dois**, que distinguem as *firewalls* que guardam e atuam sobres os pacotes sabendo **o estado de uma ligação** contra as que apenas filtram os pacotes sem conhecimento dessas ligações. As primeiras são designadas por **filtros de pacotes dinâmicos**, enquanto que as segundas são *filtros de pacotes estáticos*.

No modelo **estático**, não é guardada qualquer tipo de informação acerca do estado de ligações. Portanto, estes equipamentos ou *software* **funcionam ao nível da camada de rede** do modelo *Open Systems Interconnection* (OSI) podendo ir até à **camada de transporte**, mas só para efeitos de **comparação de portas fonte e destino**. Estas *firewalls* aplicam filtros baseados em **regras de controlo de acesso definidas para os seguintes parâmetros**:

- Endereços (**IP**) de **origem** e **destino**;
- **Tipo de protocolo** da camada de sessão (TCP, UDP, etc);
- **Portos** de origem e destino.

Os sistemas operativos **Cisco para *routers***, por exemplo, suportam funcionalidades **deste tipo de *firewalls* por defeito**, com cadeias de regras que podem ser aplicadas **por *interface***.

As **vantagens** dos filtros de pacotes estáticos é que são tipicamente **muito fáceis de implementar e gerir**, comportando também soluções de **baixo custo**, **computacionalmente eficientes e transparentes** para o utilizador.

As **desvantagens** é que **não podem ser usadas para prevenir ataques que explorem vulnerabilidades ou funções especificas a aplicações** de rede, porque não podem examinar dados das camadas superiores, apenas cabeçalhos. Também **não suportam esquemas de autenticação avançados**, porque não têm qualquer funcionalidade nas camadas de aplicação. Adicionalmente, **é fácil escrever regras que vão contra a política de segurança de uma organização**, já que **o conjunto de variáveis** em que a regras são baseadas **é relativamente pequeno**.

Na tabela seguinte exemplifica-se uma **cadeia de regras** para controlo de acesso baseado em filtros de pacotes estáticos. As regras são **normalmente verificadas sequencialmente (do menor para o maior)**, sendo **despoletada a primeira que é preenchida pelo pacote em análise, e negligenciadas todas as seguintes**.

Rule	Direction	Source Address	Destination Address	Protocol	Source Port	Destination Port	Action
1	OUT	*	10.56.199.*	*	*	*	DROP
2	OUT	10.56.*	10.122.*	TCP	*	23(Telnet)	PASS
3	IN	10.122.*	10.56.199.*	TCP	23(Telnet)	*	PASS
4	IN and OUT	*	10.56.199.*	TCP	*	25(Mail)	PASS
5	IN	*	*	TCP	*	513(rLogin)	DROP
6	In	201.32.4.76	*	*	*	*	DROP
7	OUT	*	*	TCP	*	20(FTP)	PASS
8	IN	*	10.56.199.*	TCP	*	20(FTP)	DROP

3.8 Filtro de Pacotes Dinâmico
Dynamic Packet Filtering

Muitas *firewalls*, mesmo as menos elaboradas, implementam um mecanismo normalmente designado por *dynamic packet filtering*, em que **ligações ou sessões são identificadas, monitorizadas e usadas para efeitos de controlo de acesso**. **Se um pacote ainda não faz parte de uma sessão**, **aplicam-se as regras normais** de filtragem por pacotes. Caso o pacote seja legitimo e venha a estabelecer uma ligação, **a *firewall* mantém um registo dessa ligação** (endereços e portas fonte e destino, e estado atual da ligação). A partir daí, **aceita todos os pacotes que façam parte da ligação**. Esta técnica pode ser **usada para detetar pacotes fora de contexto** (e.g., imagine que a *firewall* recebe um segmento FIN ou ACK, vindo de fora, e de uma ligação que nunca foi iniciada). A *firewall* incluida com os núcleos Linux suporta esta funcionalidade.

3.9 Tradução de Endereços
Network Address Translation (NAT)

Um dos serviços que uma *firewall* pode oferecer é o **de tradução de endereços** (mais conhecido pelo acrónimo da expressão inglesa *Network Address Translation* (NAT)). A razão pela qual se fala na funcionalidade de NAT quando se estudam *firewalls* está relacionado com o segundo objetivo indicado em baixo. Os objectivos do NAT são:

1. **Simplificar a gestão de endereços** das redes internas ligadas à *Internet* através da *gateway*;
2. **Impedir um endereçamento das máquinas internas originado nas redes externas**.

3.10 NAT – IP Masquerading
Network Address Translation (NAT) – IP Masquerading

O IP *masquerading*, **também conhecido como NAT dinâmico** (*Dynamic* NAT) visa **esconder toda a rede privada por detrás de endereços públicos da sua *gateway***. Quando **um pacote passa pela *gateway***, esta **muda o endereço IP de origem para um dos endere-**

ços públicos de que dispõe. E.g.,
`Pacote IP(192.168.0.61; 193.136.66.209; 2010; 80)`
$\rightarrow$
`Pacote IP(2.82.28.200; 193.136.66.209; 1025; 80).`
Já agora, note que a porta TCP ou UDP fonte pode também ser alterada neste processo. **A *gateway* guarda a tradução que fez** para que saiba direcionar **respostas vindas de fora para o seu verdadeiro destino dentro da rede interna**. Estas traduções são **guardadas numa tabela de sistema** e **descartadas quando não são mais precisas**. No caso do **TCP, a tradução é descartada quando a ligação termina**. No caso do **UDP, a *gateway* espera um período de tempo** após o qual elimina a tradução se não houver respostas.

Na maior parte dos casos, uma *firewall* está configurada para **deixar que uma máquina da rede interna estabeleça ligações para o exterior**, mas **não deixa que sejam estabelecidas ligações para o interior**. A regra por defeito é simples. **Se a máquina da rede interna inicia a ligação**, a *firewall* verifica se já existe alguma tradução na tabela de traduções e, se não houver, faz uma entrada nova. Quando recebe pacotes de volta (vindos do exterior), deixa-os passar, fazendo a tradução inversa. Caso contrário, descarta-os. Devido a esta forma de atuar, **muitos protocolos ou aplicações *Peer-to-Peer* (P2P) tentam fazer com que seja o programa na rede protegida a iniciar a ligação**.

3.11 NAT – Port Fowarding
NAT – Port Fowarding

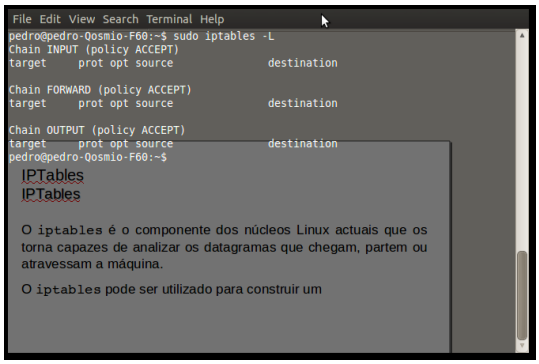
Como foi sugerido antes, a utilização de NAT previne que máquinas internas sejam diretamente endereçadas do exterior. Isto é **útil para a maior parte das máquinas internas**, mas **não para servidores públicos que queremos tornar disponíveis**. O *port fowarding*, é um mecanismo de NAT que **permite que sejam colocados servidores, acessíveis do exterior** via *gateway*, dentro de uma rede privada.

Basicamente, este mecanismo consiste em configurar **combinações** `[IP público da gateway, porto especifico` $\rightarrow$ `IP interno, porto específico]` **que são sempre direcionados para o mesmo endereço IP e porta dentro da rede protegida**. Um exemplo muito comum é **redirecionamento de ligações na porta 80**. E.g., cada vez que a *gateway* recebe um pedido vindo de fora para o seu endereço público na porta 80, reencaminha-o para o servidor HTTP da rede interna, para a mesma porta. Desta forma, **o endereço de rede interno do servidor não transpira cá para fora**, e **a *firewall* pode mediar todas as comunicações**.

3.12 Caso de Estudo: Firewall do Linux
Study Case: Linux Firewall

O `iptables` é **um programa** do espaço aplicacional do utilizador (*user space program*) que **permite a configuração das tabelas disponibilizadas pela *firewall* do núcleo Linux**, bem como das **regras e cadeias de regras** que armazenam. A *firewall* Linux é implementada por **diferentes módulos de filtragem de pacotes para diferentes protocolos**. A `iptables` é relativa ao IPv4, enquanto que a `ip6tables` configura as regras IPv6, a `arptables` configura as regras *Address Resolution Protocol* (ARP), e a `ebtables` configura as tabelas para tramas *Ethernet*. Note que **a `iptables` não é a *firewall***, embora seja **possível ligar ou desligar** as regras de filtragem de pacotes IP usando os comandos:
`$ su` (palavra-passe de administração)
`$ systemctl iptables start`
`$ systemctl iptables stop`

A figura seguinte mostra o *output* produzido pela emissão do comando `$ iptables -L` no terminal Linux. Este comando pode ser usado para se ver



O iptables **pode ser utilizado para configurar uma *firewall* pessoal bastante simples** ou ***gateways* capazes de controlar fluxos de comunicação entre várias redes**. O destino dos datagramas que chegam à máquina **depende do conjunto de regras** implementadas **nas várias tabelas** (`filter`, `nat` e `mangle`). O modelo de operação da *firewall* Linux **é do tipo filtro de datagramas**. Contudo, **é possível reencaminhar tráfego para outras aplicações locais**, o que fornece **meios para construir *firewalls* aplicacionais ou de circuitos**. A análise de datagramas segue o conceito de *ipchains* (cadeias de regras). **Um datagrama percorre todas as regras até que uma lhe seja aplicável**. As seguintes não são verificadas depois de uma ser preenchida. Para adicionar a primeira regra na tabela `INPUT` ilustrada antes, escrevia-se na linha de comandos:

```
$ iptables -A INPUT -p tcp -destination-port 22
-j ACCEPT
```

A *firewall* do Linux, bem como a ferramenta `iptables` serão alvo de estudo numa aula prática laboratorial. Nela ir-se-ão verificar que existem **cinco cadeias-padrão** disponibilizadas pelo Linux, que podem ser ajustadas no âmbito de **cinco tabelas** diferentes. Atualmente, são suportadas nativamente **três tipos de decisão** (*targets*) para um pacote IP, conforme de estrutura na tabela seguinte:
Os ***targets*** `REJECT` **e** `NFQUEUE`, que fazem parte de **extensões** à *firewall* também são bastante úteis neste contexto. Enquanto que o primeiro *target* permite **enviar o**

Tabelas	Descrição
filter	Para aplicação de filtros de pacotes.
nat	Para operações relativas ao NAT.
magle	Para alterações nos pacotes IP.
raw	Para utilizações especializadas.
security	Usada para controlo de acesso mandatório.

Cadeias	Descrição
`INPUT`	ver aula prática
`OUTPUT`	ver aula prática
`FOWARDING`	ver aula prática
`PREROUTING`	ver aula prática
`POSTROUTING`	ver aula prática

Targets	**Descrição**
`ACCEPT`	Aceita o pacote.
`DROP`	Descarta o pacote sem dar qualquer motivo.
`RETURN`	Devolve o pacote em análise à cadeia que o possa ter eventualmente enviado para aquela onde está atualmente.

pacote para uma aplicação externa (e *userspace*) para processamento adicional, o segundo permite fazer `DROP` dos pacotes, e simultaneamente **enviar um aviso para a máquina de onde esses pacotes vieram**.

As maiores **vantagens** das funcionalidades de *firewall* disponibilizadas pelo Linux são **constituírem uma solução eficaz** (mesmo quando comparado com soluções comerciais), **extensível**, **estável**, **confiável** e **escalável**. O facto de serem **desenvolvidas, testadas e melhoradas por uma grande quantidade de utilizadores** também abona em favor da solução. A **maior desvantagem** está relacionada com o facto de **a sua utilização não ser direta e requerer algum esforço** para perceber como é que, e.g., o `iptables` interage com o núcleo e como explorar todas as suas potencialidades. Não há muitas ferramentas gráficas para administração da solução.

4 Definição e Perfil de uma Intrusão

Definition and Profile of an Intrusion

A última parte deste capítulo discute sistemas de deteção de intrusões (*Intrusion Detection Systems* – IDSs). Antes disso, importa discutir o que é uma intrusão e o seu perfil.

4.1 Definição de uma Intrusão

Definition of an Intrusion

> Uma intrusão consiste **no conjunto de ações que visam comprometer a integridade**, a **confidencialidade** ou a **disponibilidade de um recurso**. **Resulta da ação de um ou mais ataques** a sistemas que gerem esse recurso.

Dada a definição, pode-se concluir que **um IDS pode detetar intrusões**, **ou meras tentativas de intrusão** procurando, para isso, **indícios do ataque ou de que este está para ocorrer**.

4.2 Perfil de uma Intrusão

Intrusion Profile

O desenvolvimento de um IDS presume o estudo do **perfil de uma intrusão**:

1. Um atacante **começa quase sempre por tentar identificar o sistema a atacar** numa ação topológica. O atacante procura também, por vezes, **identificar sistemas de proteção (*firewalls*) ou IDSs**.
2. A seguir, o atacante **tenta um reconhecimento interno**, procurando **identificar vulnerabilidades específicas da máquina alvo ou de uma máquina intermédia** que pode usar para comprometer o verdadeiro alvo. Há grande **possibilidade de haver tráfego de rede anormal** a circular nesta parte do reconhecimento.
3. A terceira parte consiste em **atacar o sistema onde detetou vulnerabilidades**. Estes ataques também podem implicar tráfego anormal, ou ações aparentemente legítimas, como a exploração de uma conta com uma senha comprometida.
4. A quarta fase consiste em **tirar proveito da intrusão** bem sucedida para transformar **a máquina comprometida como base de operações para esconder a verdadeira localização física do atacante**.
5. A quinta fase consiste em **usar a intrusão para causar dano aos sistemas atingidos**, quer pela destruição, roubo ou alteração de informação importante ou negação de prestação de serviços.

5 Sistemas de Deteção de Intrusões

Intrusion Detection Systems

Os sistemas computacionais são cada vez mais poderosos e complexos, e consequentemente, mais difíceis de configurar e inteiramente controlar. Como **a maioria dos utilizadores dos Sistemas Operativos (SO) de hoje não são especialistas**, não conseguem resolver completamente as vulnerabilidades que vêm com o SO ou com as aplicações que instalam. **O objetivo dos IDSs é detetar e**, e em certos casos, **contrariar intrusões**.

5.1 Definição de Sistema de Deteção de Intrusões

Definition of Intrusion Detection Systems

> Um IDS é **um conjunto de componentes de *software* ou *hardware* com a função de detetar, identificar e por vezes responder a atividades não au-**

torizadas ou anormais num dado contexto de comunicação ou sistema-alvo.

Os IDSs são **normalmente independentes** de qualquer mecanismo usado para garantir a segurança de um sistema. Começaram a ser **estudados no início da década de 80**, mas só na década de 90 é que passaram a ser realmente necessários, após a **adoção mundial da Internet**. São sistemas **muito sofisticados**, mas normalmente **difíceis de gerir** (requerem especialistas). Não existem normas para este tipo de sistemas, e neste capítulo iremos apenas tentar observar os IDSs do ponto de vista da sua classificação e tipo de análise que conduzem.

5.2 Métricas de Qualidade de um IDS
Quality Measures of an IDS

A qualidade de um IDS mede-se em relação às **intrusões que é capaz de detetar** e também em relação aos **alarmes que levanta erroneamente**. Diz-se que existe **um falso positivo quando um IDS conclui que existe uma intrusão quando, na verdade, não existe nenhuma**. Falsos positivos **em demasia desacreditam um IDS**.

Diz-se que houve um **falso negativo quando houve uma intrusão e o IDS não foi capaz de assinalar**. Falsos negativos **em demasia tornam o IDS inútil** no contexto para o qual é definido. A tabela seguinte mostra precisamente a designação dos alarmes em relação ao acerto na identificação de um alarme. A vermelho estão as situações indesejadas, a azul simbolizam-se os alarmes desejados:

IDS detetou? | Ataque?	SIM	NÃO
SIM	Verdadeiro positivo	Falso Positivo
NÃO	Falso Negativo	Verdadeiro Negativo

5.3 Potes de Mel
Honeypots

Há IDSs cujo principal objetivo é o de **iludir o atacante, levando-o a atacar recursos aparentemente interessantes, mas que não têm utilidade nenhuma** em termos de rede. Estes recursos são normalmente **conhecidos por Potes de Mel**. A ideia dos potes de mel é a de **desviar o atacante de recursos organizacionais sensíveis**. Estes potes de mel têm como objetivos secundários a **recolha de informação forense aplicável a deteção de ataques** (modo de intrusão, rasto deixado pelo atacante, origem dos ataques, etc.), ou **alimentam mecanismos de aprendizagem automática** de, e.g., assinaturas de tráfego.

5.4 Arquitetura dos Sistemas de Deteção de Intrusões
Architecture of Intrusion Detetion Systems

A **arquitetura funcional dos IDSs** é normalmente construida a partir de **quatro componentes**:

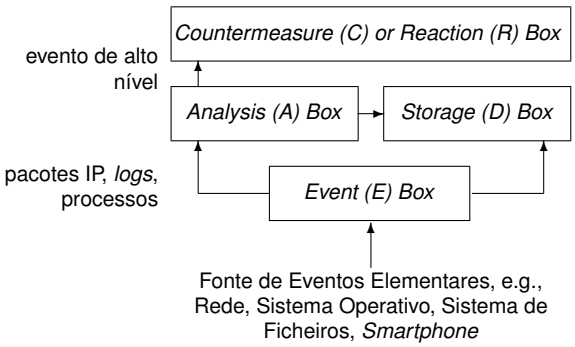
E boxes – Sensores de captura de eventos de baixo nível ou elementares, e.g., pacotes IP, *logs* do sistema, nomes de ficheiros.

A boxes – Módulos de análise, e.g., algoritmos de processamento estatístico e comparação com assinaturas em bases de dados.

D boxes – Módulos de armazenamento de eventos (provas forenses) e de resultados da análise.

C or R boxes – Componente com medidas de reação a ataques, e.g., envio de *e-mail* ao administrador, alteração de regras na *firewall* ou isolamento de ficheiros contaminados com vírus.

Num IDS, os componentes antes descritos organizam-se e interligam-se, normalmente, como se mostra na figura seguinte:



As várias componentes são discutidas brevemente nas secções seguintes.

5.5 *E Boxes*
E Boxes

Os **sensores de captura** de um IDS podem atuar nas mais **diversas fontes de informação**, quer ao **nível da rede**, quer ao **nível do sistema** onde estão a operar. Assim, as principais fontes de eventos destes sistemas são: unidades de dados de **tráfego de rede**; **utilizadores com sessão iniciada**; **número de processos** em execução; **carga de processamento**; **alterações de ficheiros**; etc.

Estes sensores **não precisam estar** necessariamente **todos no mesmo ponto** de uma rede de sistemas. **Podem estar distribuídos em pontos estratégicos** (e.g. *gateway*, máquinas da DMZ ou computadores da rede interna protegida), **e enviarem eventos a pedido ou periodicamente** para a máquina central IDS, onde é feito o processamento desses eventos.

5.6 *A Boxes*
A Boxes

Tudo o que é capturado nas *E-boxes* é enviado para os módulos de análise (**A-boxes**), que **combinam, uniformizam e correlacionam os dados obtidos**. Estas *A-Boxes* **produzem eventos de alto nível** que indicam se uma **intrusão foi detetada e a sua designação** (e.g., criação de sessões locais ou remotas, infeção por ciberpragas –vírus, tentativa de DoS, etc.).

5.7 *D and C Boxes*
D and C Boxes

Caso um evento seja classificado como indicador de uma intrusão, é armazenado pela *D box*, juntamente com o resultado devolvido pela *A Box*. Paralelamente, os eventos reveladores de uma intrusão são enviados para *C boxes*, que contêm medidas de reação a esses eventos, nomeadamente:

Executar ações de informação e relatório – Envio de alarmes e produção de relatórios.

Proteção – Reconfiguração de mecanismos ou sistemas de segurança (*firewalls*) ou descarte de datagramas.

Sobrevivência – Isolamento de sistemas atacados e ativação de sistemas alternativos (e.g., máquinas virtuais) e potes de mel.

Contra-ataque – Despoletar ataques contra as máquinas que iniciaram o ataque. Quase nunca é uma boa ideia.

6 Classificação de IDSs
Classification of IDSs

A classificação dos IDSs é um tópico importante no contexto da análise destes sistemas. Os IDSs podem ser classificados segundo inúmeras **caraterísticas operacionais**, a saber:

- Pelo **método de deteção**, para o qual se distinguem as categorias
  - IDS **baseado em assinaturas**;
  - IDS **baseado em anomalias**.
- Pela **fonte de eventos**, que os segrega em
  - IDS baseado em Máquinas Anfitriãs (***Host based*** IDS (HIDS));
  - IDS baseado na Rede (***Network based*** IDS (NIDS)).
- Pela **celeridade de deteção**, que segrega estes sistemas em
  - Capazes de operar em **tempo de execução**;
  - Capazes de operar em **tempo real virtual**;
  - Apenas operam em **tempo indeferido**.
- Pela **reatividade**, que definem sistemas
  - **Ativos** e;
  - **Passivos**.
- E pela **distributividade**, que determinam sistemas que atuam de forma
  - **Singular** ou isolada e;
  - **Cooperativa**.

Esta classificação é alvo de discussão nas secções seguintes.

6.1 IDS Baseado em Assinaturas
Signature Based IDS

Os IDSs cujo método de deteção é **baseado em assinaturas** (também conhecidos como IDS **baseado em conhecimento**) analisa a atividade do sistema à **procura de padrões de ataques conhecidos**, denominados assinaturas. A titulo de exemplo, pode-se dizer que **a assinatura de um vírus é a existência de *strings* conhecidas no código** do *software malicioso* ou de padrões nas tentativas de propagação conhecidos. A assinatura de um TCP SYN *flood*, por exemplo, é a ausência excessiva de segmentos TCP ACK. Um programa **antí-virus** é um bom exemplo de **um IDS baseado em assinaturas**.

As **desvantagens** deste tipo de análise são que (i) estes sistemas **só detetam ataques conhecidos**; (ii) **não escalam com muita facilidade** e (iii) necessitam de uma **base de dados de assinaturas sempre atualizada**.

6.2 IDSs Baseados em Comportamento
Behavior Based IDSs

Os IDS cujo método de deteção é **baseado em comportamento** (também conhecidos como IDS **baseado em anomalias**) procura **definir o comportamento normal de um sistema e desvios abruptos a essa normalidade**.

Um IDS deste tipo **requer** normalmente que o sistema passe por **um período de aprendizagem** em que o comportamento normal é aprendido através de, e.g., **técnicas de inteligência artificial ou análise e modelação estatística**.

A **grande vantagem** é que permite, teoricamente, **detetar ataques desconhecidos**. As principais **desvantagens** deste tipo de análise são (i) a **dificuldade em definir modelos robustos** de bom comportamento (até porque o comportamento pode ir mudando); (ii) o **ajuste do grau de sensibilidade** do sistema; (iii) a **falta de informação dada pelas *A-boxes*** aquando da deteção.

6.3 Celeridade da Resposta

Response Celerity

Os IDSs podem ser segregados em **3 tipos diferentes** no que diz respeito **à velocidade como produzem resultados**. Estes 3 tipos apresentam as seguintes características:

- O IDS actua em **tempo de execução**, se permite **detetar ataques exatamente quando estão a decorrer e evitar que tenham sucesso**. É uma realidade difícil de alcançar na maior parte dos casos.

- O IDS actua em **tempo real virtual**, se **deteta intrusões e ataques pouco tempo depois de acontecerem, mas ainda a tempo de evitar que tenham sucesso uma segunda vez**. Este é um tipo de deteção mais realista.

- O IDS actua em **tempo indiferido**, quando **permite detetar intrusões passadas** pela **análise de registos (*logs*)** do sistema. Este é o método normal usado por técnicas de auditoria.

6.4 Reatividade

Reactivity

Em termos de **reatividade** é possível segregar os IDS em sistemas passivos ou ativos:

- Os IDS **passivos** apenas **produzem relatórios e alarmes** para os administradores humanos dos sistemas ou rede.

- Os IDS **ativos** são **capazes de reagir de forma automática a ataques ou intrusões**, aplicando políticas de **defesa** ou **contra-ataque**.

  Defesa – **Isolamento** de componentes comprometidas e **ativação de substitutos e de potes de mel**. **Redução da largura de banda** para reduzir efeitos de ataques baseados em inundação. Bloqueio de IPs fonte na *Gateway* através da reconfiguração da *firewall*.

  Contra-ataque – Tão ilegal como o ataque (ataque a máquinas trampolim).

6.5 Colaboratividade

Collaborative

Atualmente existem já sistemas IDS que **cooperam de forma a melhorar a capacidade de deteção**. Diz-se que a análise que estes sistemas fazem é cooperativa, e apontam-se como principal **vantagem** para o seu uso o facto de se obter **uma visão mais abrangente do domínio de segurança**. Por exemplo, se a rede tiver vários IDSs a comunicar entre sí, **é possível detetar em que lado começou determinada infeção**. O grande **problema** é que **ainda não há uma norma de facto que uniformize a comunicação entre IDSs de diferentes fabricantes**. Os **IDSs que atuam sozinhos são ditos singulares**, ou que efetuam uma análise singular.

7 Casos de Estudo – Tripwire e Snort

Study Cases – Tripwire and Snort

7.1 *Tripwire*

Tripwire

O *Tripwire* é um HIDS, simples mas efetivo, que baseia o seu modo de operação na deteção de alterações de ficheiros em sistemas UNIX. A ideia fundamental do programa é a de que existem muitos ficheiros do sistema de ficheiros (e.g., *Kernel*, aplicações do sistema, bibliotecas estáticas, ficheiros de configuração, etc.) que **não são alterados frequentemente** e que **a sua alteração inesperada é sinal de intrusão**. O *Tripwire* monitoriza **diversos atributos-chave de ficheiros**, e.g.:

- Lista de permissões de acesso;
- *User* ID (UID) do dono e *Group* ID (GID) do ficheiro;
- data e instante de criação;
- data e instante da última modificação;
- conteúdo (usando funções de dispersão criptográficas, nomeadamente o **MD5**), etc.

O funcionamento do *Tripwire* é simples. Quando colocado em funcionamento, o programa **produz uma base de dados** contendo as **informações atuais dos ficheiros a monitorizar** (esta base de dados não deve poder ser alterada por atacantes, pelo que deve ser protegida recorrendo a artifícios de nível físico). **Durante a execução, o programa produz outra base de dados** (atual) e **compara-a com a que está guardada** para inferir intrusões. Com alguma frequência (definida pelo administrador), o *Tripwire* substitui a base de dados com uma mais atual, para manter o programa sensível à evolução do sistema.

7.2 *Snort*

Snort

O *Snort* é o **NIDS de código aberto mais popular**. Trata-se de um **sistema baseado em assinaturas**, produzidas por uma **comunidade muito ativa**. A **facilidade de produzir assinaturas** para o *Snort* é, de resto, **uma das melhores caraterísticas** / **vantagens** deste NIDS. O *Snort* assenta sobre **uma biblioteca de captura de tráfego de rede** chamada `libpcap`, e **corre em várias plataformas**. Após receber o tráfego dos sensores, um conjunto de pré-processadores normalizam a informação, e um conjunto de módulos adicionais permitem **reconstruir os fluxos de informação até ao nível da camada de aplicação** o que, por sua vez, **facilita a deteção de intrusões até este nível**. E.g., é possível bloquear certos *sites* ou detetar vírus ou *worms* com este IDS. A **natureza modular** do *Snort* fornece o substrato ideal para **expandir facilmente** as suas funcionalidades.

Hoje em dia, e para técnicos habituados a interfaces de linha de comandos, **a instalação e colocação em funcionamento** do *Snort* é relativamente **simples**. Por exemplo, no Sistema Operativo Ubuntu, a instalação e funcionamento passa pela inserção das seguintes instruções no terminal:

```
$ sudo apt-get install snort
```

```
$ sudo snort -i eth0
```

As duas instruções anteriores instalam e colocam o NIDS em funcionamento para a *interface* `eth0` com o conjunto de regras por defeito. Noutros sistemas operativos da família Unix, os passos para o colocar em funcionamento são semelhantes.

O *Snort* vem com um conjunto de regras por defeito (algumas delas já desatualizadas), que podem ser encontradas em `/etc/snort/rules/`. A configuração do NIDS é feita no ficheiro `/etc/snort/snort.conf`. A **escrita de regras** para o *Snort* já comporta uma **operação mais sensível**, e requer um **conhecimento mais profundo ao nível dos protocolos e da segurança em redes**.

7.3 Regras *Snort*

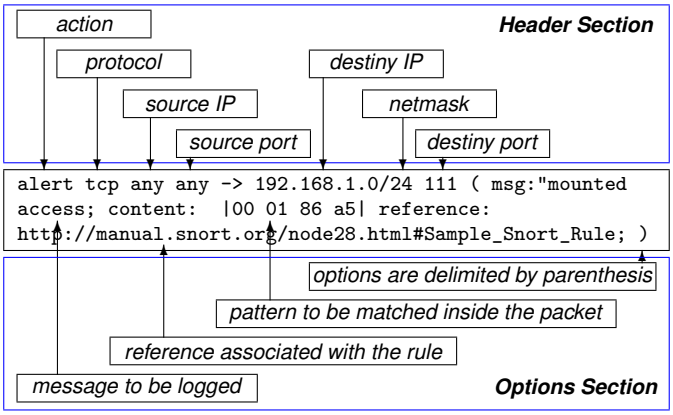
Snort Rules

O Snort usa uma linguagem simples de descrição de regras que é **flexível** e **bastante poderosa**[1]. A maior parte das regras podem ser escritas **numa única linha**, embora possam ser definidas em duas ou mais linhas se separadas por um *backslash* (\), ao estilo *bash*. Cada regra está dividida **em duas secções lógicas principais**: o **cabeçalho** e as suas **opções**:

- O cabeçalho contém a **ação** da regra, o **protocolo**, os **endereços IP fonte e destino** e as **máscaras de rede** que se aplicam, bem como informação acerca das **portas fonte e destino** do protocolo de transporte (e.g., TCP);
- A secção de **opções** contém a definição das **mensagens de alerta** que podem ser mostradas ou enviadas ao administrador aquando do seu despoletar, bem como a eventual discriminação de **partes de pacotes IP** que podem ser inspecionadas aquando da deteção (e.g., para procurar assinaturas de vírus). O Snort **tem tecnologia de inspeção profunda de pacotes** (*deep packet inspection*).

[1] O conteúdo desta secção e da que se lhe segue foi adaptado de `http://www.snort.org/snort-rules/`.

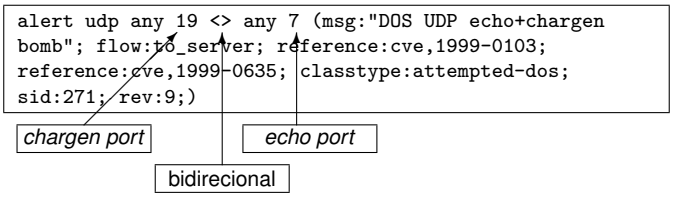
A figura seguinte mostra, através de um exemplo, as várias partes de uma regra *Snort*:



7.4 Regras *Snort* – Examples

Snort Rules – Examples

Na figura seguinte é incl uída a regra que permitiria detetar um ataque *Echo Chargen*. Esta regra foi retirada do ficheiro `dos.rules`.



Em baixo mostram-se as duas regras que podem ser implementadas no *Snort* de forma a detetar o *Sasser Worm*. Neste caso, torna-se evidente que o *Snort* **é capaz de fazer *deep packet inspection***, já que recorre ao *byte code* para detetar o *software* malicioso.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any
(msg:"W32/Sasser.worm.a [NAI])"; content:BC 3B
74 0B 50 8B 3D E8 46 A7 3D 09 85 B8 F8 CD 76 40
DE 7C 5B 5C D7 2A A8 E8 58 75 62 96 25 24|";
classtype:miscactivity;sid:1000003;rev:1;)
```

Byte code (em hexadecimal)

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any
(msg:"W32/Sasser.worm.b [NAI])"; content:58 BC
0C FF 59 57 32 31 BD EC 34 64 6E D6 E3 8D 65 04
68 58 62 79 DF D8 2C 25 6A B5 28 BA 13 74|";
classtype:misc-activity;sid:1000004;rev:1;)
```

Nota: o conteúdo exposto na aula e aqui contido não é (nem deve ser considerado) suficiente para total entendimento do conteúdo programático desta unidade curricular e deve ser complementado com algum empenho e investigação pessoal.