4.0 - Firewall

prof. Fábio Engel

fabioe@utfpr.edu.br

Conteúdo

- **Firewall**
  - Basicamente, um firewall é um software que trabalha bloqueando ou liberando as conexões de rede, de acordo com as regras preestabelecidas baseadas no endereço de origem, no endereço de destino e na porta de destino (serviço) da conexão, conforme exemplificado na tabela abaixo:

Regra	IP de origem	IP de destino	Porta de destino	Ação
1	172.20.16.60	8.8.8.8	UDP:53	Permite
2	172.20.16.60	Todos	Todas	Bloqueia

Firewall

- A maioria dos softwares de firewall hoje, senão todos, apresenta outras características além do firewall em si. Mas, para entendermos melhor os conceitos, vamos nos ater ao básico neste momento.
- O firewall sempre lê as regras de cima para baixo, executando a ação descrita na primeira regra em que o pacote se encaixar. Por exemplo, a regra 2 mostrada na tabela abaixo nunca será executada.

Regra	**IP de origem**	**IP de destino**	**Porta de destino**	**Ação**
1	172.20.120.4	Todos	Todas	Bloqueia
2	172.20.120.4	8.8.8.8	TCP:80	Permite

- Neste caso, o correto é colocar a regra 2 antes da 1.

Iptables

- O IPTABLES é uma ferramenta de edição da tabela de filtragem de pacotes, ou seja, com ele você é capaz de analisar o cabeçalho (header) e tomar decisões sobre os destinos destes pacotes.

Iptables

- O iptables é um firewall com estado, ou seja, um firewall stateful. Os anteriores eram stateless. O modo de filtragem "Stateless" tende a tratar cada pacote roteado pelo firewall como pacotes individuais, sendo mais simples de implementar e por terem uma resolução mais rápida que um do tipo stateful, podem ser usados para obterem um desempenho melhor em determinadas situações onde existem regras de nível de rede bem simples. O tipo de filtragem stateful (IPTABLES) cria um poderoso sistema de firewall que se "lembra" das conexões entrantes, evitando ataques do tipo Stealth Scans, que trazem flags especiais para técnicas de port scanning, como o uso da flag ACK para enganar tais firewalls.

Iptables

- O Linux utiliza-se de um recurso independente em termos de kernel para controlar e monitorar todo o tipo de fluxo de dados dentro de sua estrutura operacional.
- Para que o kernel possa controlar seu próprio fluxo interno foi lhe agregado uma ferramenta batizada de **Netfilter**.

Iptables

- O **Netfilter** é um conjunto de situações (*chains*) de fluxos de dados agregados inicialmente ao kernel do Linux e dividido em tabelas.
- Sob uma ótica mais prática, podemos ver o Netfilter como um grande banco de dados que contém em sua estrutura 3 tabelas padrões:
  - Filter - armazena todas as regras aplicadas a um Firewall filtro de pacotes;
  - Nat - regras direcionadas a um firewall Nat.
  - Mangle - funções mais complexas de tratamento de pacotes.

Tabela Filter

- A **Tabela Filter** contém 3 modelos de situações de fluxo:
  - INPUT - Tudo que entra no host;
  - FORWARD - Tudo o que chega ao host mas deve ser **redirecionado** a um host secundário ou outra interface de rede;
  - OUTPUT - Tudo o que sai do host;

Tabela Nat

- A **Tabela Nat** implementa as funções de NAT (*Network Address Translation* ao host firewall). Suas situações são:
  - PREROUTING - Utilizada quando há necessidade de se fazer alterações em pacotes antes que os mesmos sejam roteados;
  - OUTPUT - Trata os pacotes emitidos pelo host Firewall;
  - POSTROUTING - Utilizado quando há necessidade de se fazer alterações em pacotes após o tratamento de roteamento;

Tabela Nat

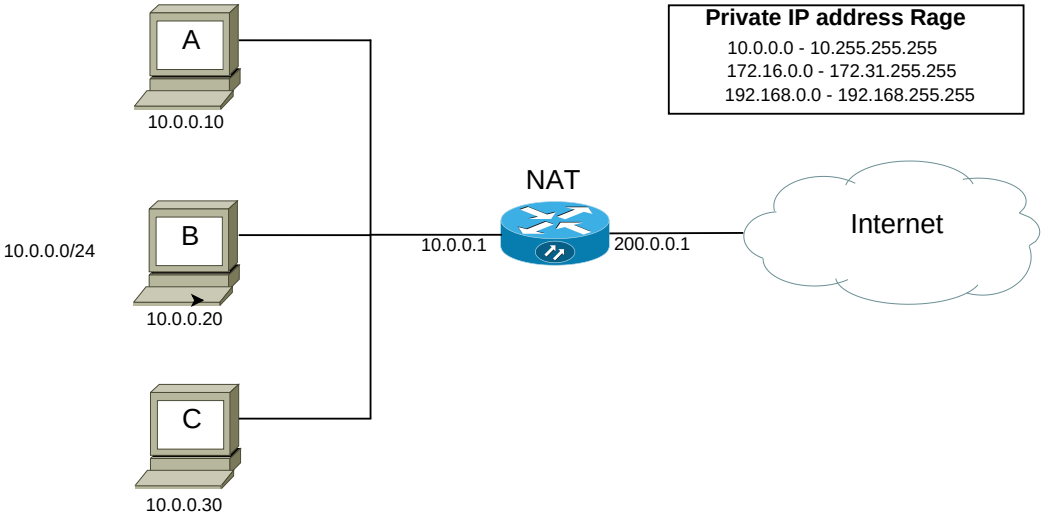

Private IP address Rage
10.0.0.0 - 10.255.255.255
172.16.0.0 - 172.31.255.255
192.168.0.0 - 192.168.255.255
A
10.0.0.10
B
10.0.0.20
C
10.0.0.30
10.0.0.0/24
NAT
10.0.0.1
200.0.0.1
Internet

Tabela Mangle

- A **Tabela Mangle** implementa alterações especiais em pacotes em um nível mais complexo. Por exemplo, alteração da prioridade de entrada e saída de um pacote baseado no tipo de serviço (ToS - *Type of Service*) o qual o pacote se destinava. Suas situações são:
  - PREROUTING - Modifica pacotes dando-lhes um tratamento especial antes que os mesmos sejam roteados.
  - OUTPUT - Altera pacotes de forma "especial" gerados localmente antes que os mesmos sejam roteados.

Netfilter

- O kernel do Linux possui funções de Firewall graças as tabelas que se agregam ao Netfilter, que por sua vez está originalmente agregado ao kernel.
- Tais tabelas (do Netfilter) nos possibilitam controlar todas as situações (*chains*) de um host.
- Porém, para que possamos vir a moldar o Netfilter conforme nossas necessidades, levando em consideração que o mesmo deve estar compilado com o Kernel, precisamos de uma ferramenta que nos sirva de Front-End nesta tarefa.

Netfilter

- Um Front-End lhe possibilitará o controle das situações (*chains*) contidas nas tabelas agregando-lhes regras de tráfego.
- Entenda por regras as pré-definições aplicadas a fim de disciplinar todo um tráfego de dados de uma rede/host.
- O Linux disponibilizou algumas ferramentas de manipulação nativas nestas versões:

Versão do kernel	Ferramenta
Kernel 2.0	IPFWADM
Kernel 2.2	IPCHAINS
Kernel 2.4/2.6	IPTABLES

Iptables

- O iptables, conforme sugerido anteriormente, trata-se, na verdade, de uma ferramenta de Front-End para lhe permitir manipular as tabelas do Netfilter, embora o mesmo seja constantemente confundido com um Firewall por si só.
- Ele é uma versão mais robusta, completa e tão estável quanto seus antecessores IPFWADM e IPCHAINS.
- Com ele é possível: implementação de filtros de pacotes utilizando as tabelas do Netfilter, desenvolvimento de QoS sobre o tráfego, suporte a SNAT e DNAT, redirecionamento de endereçamento e portas, mascaramento de conexões, detecção de fragmentos, monitoramento de tráfego, ToS, bloqueios a ataques Spoofing, Syn-Flood, DOS, scanners, pings da morte entre muitos outros.

Iptables

- A princípio o Iptables é composto pelos seguintes aplicativos:
  - iptables - Aplicativo principal do pacote iptables para protocolos ipv4.
  - ip6tables - Aplicativo principal do pacote iptables para protocolos ipv6.
  - iptables-save - Aplicativo que salva todas as regras inseridas na sessão ativa e ainda em memória em um determinado arquivo informado pelo administrador do firewall.
  - iptables-restore - Aplicativo que restaura todas as regras salvas pelo software iptables-save.

Iptables

- Por estar incorporado diretamente ao kernel, a configuração do Iptables não se dá por via de arquivos de configuração, ao contrário, sua manipulação é realizada por síntese digitada em shell.
- Quando inserimos uma regra na shell, ela estará valendo tão somente para aquela sessão em memória, sendo que uma vez resetado ou desligado o computador Firewall, tais regras serão perdidas e não mais poderão serem resgatadas.

Iptables

- Para que as configurações realizadas sejam salvas permanentemente uma função/programa é utilizado, o iptables-save.
- Para salvar as configurações atuais, utilize a síntese a seguir:
  - **iptables-save > /etc/firewall.conf**
- O comando anterior salvará as regras atuais do sistema (ainda em RAM) para o arquivo /etc/firewall.conf.

Iptables

- É possível substituir o iptables-save por qualquer outro shell script que contenha as regras a serem inseridas no sistemas. Tal arquivo pode ser posto para iniciar juntamente com o sistema de forma automática acrescentando uma chamada ao mesmo ao fim do arquivo /etc/rc.local:
  - **iptables-restore < /etc/firewall.conf**

Iptables - Sintaxe

- A sintaxe do iptables é:
  - **iptables [tabela] [comando] [ação] [alvo]**

Tabelas

- **Tabelas** - São as mesmas que compõem o Netfilter.
- Utilizamos esta opção para associar uma regra a uma tabela específica:
  - **iptables -t filter**
  - **iptables -t mangle**
  - **iptables -t nat**
- A tabela filter é a padrão, logo se adicionarmos uma regra sem utilizarmos a flag -t [tabela], o nome aplicará situações contidas na tabelas filter a tal regra.

Comando

- Comando:
  - **-A** - Adiciona (anexa) uma nova entrada ao fim da lista de regras;
    Ex: **iptables -A INPUT**
    # Adiciona uma nova regra ao final da lista referente a INPUT chain.
  - **-D** - Deleta uma regra especificada da lista;
    Ex: **iptables -D INPUT**
    # Apaga a regra inserida anteriormente apenas trocando o comando -A pelo -D

    # O Comando -D também lhe permite apagar uma certa regra por seu número da lista de ocorrência no Iptables;
    Ex: **iptables -D FORWARD 2**
    # Apaga a regra de número 2 referente a FORWARD chain.

Comando

- Comando:
  - **-L** - Lista as regras existente na lista;
    Ex: **iptables -L FORWARD**
    Chain FORWARD (policy ACCEPT)
    target prot opt source destination
    DROP icmp – anywhere anywhere icmp echo-request
    ACCEPT tcp – anywhere anywhere limit: avg 1/sec burst 5
    DROP all – anywhere anywhere unclean
    # Na listagem anterior, referentes as regras a FORWARD chain, foram relatadas 3 ocorrências.

Comando

- Comando:
  - **-P** - Altera a política padrão das chains. Inicialmente, todas as chains de uma tabelas estão setadas como ACCEPT, ou seja, aceitam todo e qualquer tipo de tráfego. Para modificar esta política utilizamos a flag -P;
    Ex: **iptables -P FORWARD DROP**
    # Modifica a política padrão da chain FORWARD e ao invés de direciona-lo para o alvo ACCEPT o leva a DROP. Um pacote que é conduzido ao alvo drop é descartado pelo sistema. Veja como fica a política da FORWARD chain após o comando digitado anteriormente:
    Ex: **iptables - L FORWARD**
    Chain FORWARD (policy DROP)
    target prot opt source destination
    DROP icmp – anywhere anywhere icmp echo-request
    ACCEPT tcp – anywhere anywhere limit: avg 1/sec burst 5
    DROP all – anywhere anywhere unclean

Comando

- Comando:
  - **-F** - Este comando é capaz de remover todas as entradas adicionadas a lista de regras do Iptables sem alterar a política padrão (-P);
    Ex: **iptables -F**
    # Remove todas as regras
    Ex: **iptables -F OUTPUT**
    # Remove todas as regras referentes a OUTPUT chain

Comando

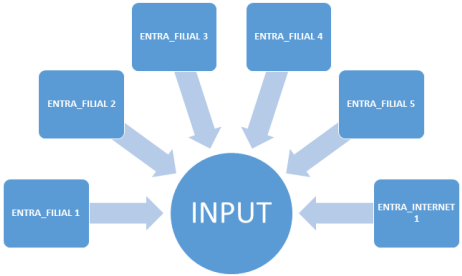
- Comando:
  - **-I** - Insere uma nova regra ao início da lista de regras (ao contrário do comando -A que insere ao final);
    Ex: **iptables -I OUTPUT**
  - **-R** - Substitui uma regra já adicionada por outra;
    Ex: **iptables -R FORWARD 2 -s 10.0.40.3 -d 10.0.30.0/8 -j DROP**
    # Substitui a segunda regra referente a FORWARD chain pela seguinte: -s 10.0.40.3 -d 10.0.30.0/8 -j DROP

Comando

- Comando:
  - **-N** - Permite inserir/criar uma nova chain a tabela especificada;
    Ex: **iptables -t filter -N internet**
    # Cria uma nova chains "internet" sobre a tabela filter
- A necessidade de se criar uma nova chain surge apenas para tornar a organização de seu firewall mais simples.

Comando

- Digamos que seu firewall tem que gerenciar o tráfego de sua LAN com mais 5 WANs e ainda Internet.
- As regras inseridas seriam tantas que surgiria a necessidade de se pegar chains como INPUT, por exemplo e dividi-las de forma a facilitar sua administração. Poderíamos criar por exemplo as seguintes chains:



Comando

- Note que todas as chains criadas são na verdade uma "cópia" da INPUT chain, ou seja, tratam dos pacotes que entram em seu host firewall.
- Uma vez feito isso, para modificar regras de entrada de um pacote vindo da Internet, bastaria que listássemos as regras da chain ENTRA_INTERNET:
  - Ex: **iptables -L ENTRA_INTERNET**

Comando

- Podemos agora analisar somente o tráfego entrante referente à rede Internet tal como das demais filiais ao invés de listar todas as regras de entrada em uma só chain (INPUT).
- A medida que o fluxo aumenta, naturalmente é sentida a necessidade de seccionar o tráfego por via das chains.
- Ao criar uma nova chain utilize um nome sugestivo. O Iptables permite criar chains cujo nome contenham até 31 caracteres sem espaços, tanto em letras minúsculas quanto maiúsculas.

Comando

- Criaremos agora uma nova chain denominada entnet, que tratará do tráfego entrante da rede Internet e fará o mesmo trabalho da chain INPUT, porém, direcionado apenas a este específico de tráfego (o da Internet):
  - **iptables -t filter -N entnet**
    iptables –A INPUT -j entnet
    # Primeiramente criamos a chain entnet e em segundo criamos um salto da chain input para a chain entnet. Isso fará com que sua regra não só funcione de forma similar a INPUT chain como também dirá a seu Iptables para analisar a chain entnet logo após a chain INPUT.

Comando

- Comando:
  - **-E** - Renomeia uma nova chain (criada por você);
    Ex: **iptables -e entnet ENTNET**
    # Renomeamos então a chain entnet para ENTNET.
  - **-X** - Apaga uma chain criada pelo administrador do firewall;
    Ex: **iptables -X ENTNET**
    # Apagamos então a chain ENTNET criada anteriormente.

Ação

- Ação:
  - **-p** - Especifica o protocolo aplicado a regra. Pode ser qualquer um numérico ou o próprio nome do protocolo (tcp, icmp, udp, etc...);
    Ex: **-p icmp**

Ação

- Ação:
  - **-i** - Especifica a interface de entrada a ser utilizada; Para listar suas interfaces de rede utilize o comando *ifconfig*. A regra -i não deve ser aplicada na OUTPUT chain pois refere-se apenas a interface de entrada (INPUT e FORWARD são as duas chains aplicáveis a flag -i)
    Ex: **-i eth0**

    Podemos também especificar todas as interfaces *eth* de nosso host da seguinte forma:
    Ex: **-i eth+**

Ação

- Ação:
  - **-o** - Especifica a interface de saída a ser utilizada e se aplica da mesma forma que a regra -i, porém, somente as chains OUTPUT e FORWARD se aplicam a regra;
    Ex: **-o eth0**

Ação

- Ação:
  - **-s** - Especifica a origem (*source*) do pacote ao qual a regra deve ser aplicada. A origem pode ser um host ou uma rede. Nesta opção geralmente utilizamos o IP seguido de sua sub-rede;
    Ex: **-s 10.0.10.0/255.0.0.0**

    A máscara de rede pode ser omitida deste comando, neste caso o iptables optará (de forma autônoma) pela máscara 255.255.255.0. O -s também possibilita a utilização de nomes ao invés do IP;
    Ex: **-s www.gnu.org**

Ação

- Ação:
  - **-d** - Especifica o destino do pacote (*destination*) ao qual a regra deve ser aplicada. Sua utilização se dá da mesma maneira que a ação -s;
  - **!** - Significa exclusão e é utilizado quando se deseja aplicar uma exceção a uma regra. É utilizado juntamente com as ações -s, -d, -p, -i, -o e etc.;
    Ex: **! -s 10.0.0.3**
    # Refere-se a todos os endereços possíveis com exceção do 10.0.0.3

    Ex: **! -p icmp**
    # Refere-se a todos os protocolos possíveis com exceção do icmp.

Ação

- Ação:
  - **-j** - Define o alvo (*target*) do pacote caso o mesmo se encaixe a uma regra. As principais são ACCEPT, DROP, REJECT e LOG;

Ação

- Ação:
  - −−**sport** - Porta de origem (source port), com esta regra é possível aplicar filtros com base na porta de origem do pacote. Só pode ser aplicada a portas referentes aos protocolos UDP e TCP;
    Ex: **−p tcp −−sport 80**
    #Refere-se a porta 80 do protocolo TCP
  - −−**dport** - Porta de destino (*destination port*), especifica a porta de destino do pacote e funciona de forma similar a regra −−sport.

Alvo

- Alvo: Quando um pacote se adequa a uma regra previamente criada ele deve ser direcionado a um alvo e quem o especifica é a própria regra. Os alvos (*targets*) aplicáveis são:
  - **ACCEPT** - Corresponde a aceitar, ou seja, permite a entrada/passagem do pacote em questão.
  - **DROP** - Corresponde a descartar. Um pacote que é conduzido a este alvo (*target*) é descartado imediatamente. O Target DROP não informa ao dispositivo emissor do pacote o que houve. No caso de um ping (solicitação icmp), o mesmo não retornará mensagens ao host de origem, isso dará a entender que o host que sofreu a solicitação (o ping) não existe, pois não enviará nenhum retorno ao mesmo.

Alvo

- Alvo:
  - **REJECT** - Corresponde a rejeitar; um pacote conduzido para este alvo (Target) é automaticamente descartado, a diferença do REJECT para o DROP é que o mesmo retorna uma mensagem de erro ao host emissor do pacote informando o que houve.
  - **LOG** - Cria uma entrada de log no arquivo /var/log/messages sobre a utilização dos demais alvos (*Targets*), justamente por isso deve ser utilizado antes dos demais alvos.
  - **RETURN** - Retorna o processamento do chain anterior sem processar o resto do chain atual.

Alvo

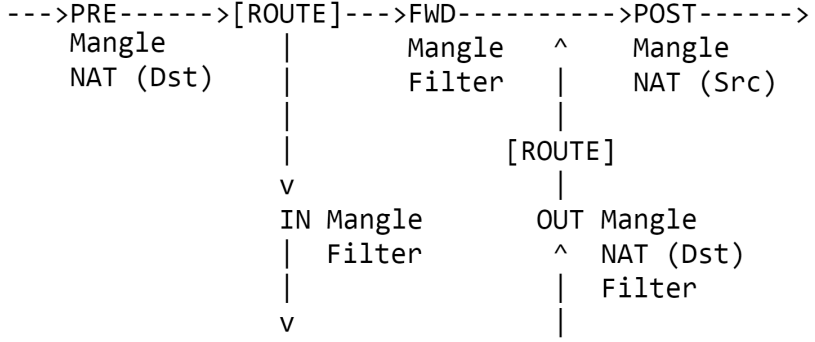
- Alvo:
  - **SNAT** - Altera o endereço de origem das máquinas clientes antes dos pacotes serem roteados. Pode, por exemplo, enviar um pacote do host A ao host B e informar ao host B que tal pacote fora enviado pelo host C.
  - **DNAT** - Altera o endereço de destino das máquinas clientes. Pode, por exemplo, receber um certo pacote destinado a porta 80 do host A e encaminha-lo por conta própria a porta 3128 do host B. Isso é o que chamamos de Proxy transparente, um encaminhamento dos pacotes dos clientes sem que os mesmos possuam a opção de escolher ou não tal roteamento.

Alvo

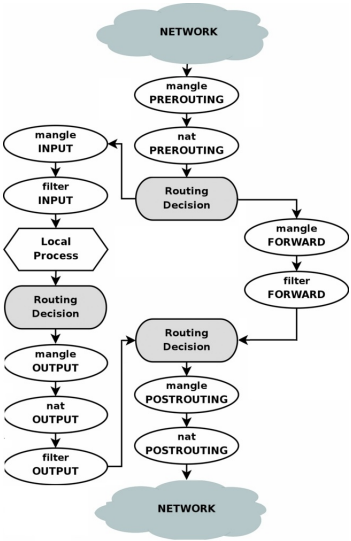
- Alvo:
  - **REDIRECT** - Realiza redirecionamento de portas em conjunto com a opção $--$*to-port*
  - **TOS** - Prioriza a entrada e saída de pacotes baseado em seu tipo de serviço, informação esta especificada no header do IPv4.

Atravessando as tabelas e chains

- Quando um pacote alcança um firewall, ele é tratado pelo driver do kernel. Então, ele passa por uma série de passos no kernel, antes de que seja encaminhado adequadamente para uma aplicação ou então encaminhado para outro host.



Atravessando as tabelas e chains



Praticando

- A seguir, utilizaremos alguns exemplos reais para a montagem passo-a-passo de regras que se adequam ao nosso objetivo de compreender a ferramenta iptables.

Praticando

- Caso 1:
  - Antes de começarmos é interessante que listemos as regras anexadas a base do Iptables, desta forma também podemos observar a política padrão de nossas chains (que provavelmente estarão setadas como ACCEPT por ser este o padrão inicial do iptables). Utilizaremos então o comando "-L" que nos fornece uma listagem dos registros de regras do iptables.

iptables -L

```
Chain INPUT (policy ACCEPT)
target prot opt source               destination
Chain FORWARD (policy ACCEPT)
target prot opt source               destination
Chain OUTPUT (policy ACCEPT)
target prot opt source               destination
```

Praticando

- Caso 1:
  - Você deve ter observado também que não especificamos com a flag -t a tabela a ser utilizada em tal listagem. Por ser padrão do Netfilter a tabela filter fora eleita automaticamente pelo sistema. Diante disso, a seguir listamos as regras referentes a tabela NAT

iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target prot opt source destination
Chain POSTROUTING (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination

Praticando

- Caso 1:
  - E mais a seguir as regras referentes a tabela Mangle

iptables -t mangle -L
Chain PREROUTING (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination

Praticando

- Caso 2:
  - Diante da listagem de regras do iptables, podemos observar que o Target (alvo) padrão de nossas chains está setado como ACCEPT. Vamos configuras o alvo padrão das chains referente a tabela filter como DROP;

    iptables -P INPUT DROP
    iptables -P FORWARD DROP
    iptables -P OUTPUT DROP

Praticando

- Caso 3:
  - Vamos agora liberar totalmente o tráfego de entrada de nossa interface de loopback (lo). Esta regra deve obrigatoriamente fazer parte de seus *script firewall* para permitir então que a comunicação entre processos seja possível. Lembre-se, esta regra não é opcional!

    iptables -A INPUT -i lo -j ACCEPT

Praticando

- Caso 4:
  - Proibiremos então que qualquer pacote oriundo de nossa LAN 10.0.30.0 possa direcionar-se ao site www.sexo.com.br. A síntese é então basicamente a seguinte:

    iptables -A FORWARD -s 10.0.30.0/8 -d www.sexo.com.br -j DROP

Praticando

- Caso 5:
  - Especificaremos agora que qualquer pacote oriundo do host www.cracker.com não pode entrar em nossa rede (10.0.30.0):

    iptables -A FORWARD -s www.cracker.com -d 10.0.30.0 -j DROP

Praticando

- Caso 6:
  - Faremos agora com que os pacotes provenientes do site www.suaempresa.com penetrem livremente em nossa rede:

    iptables -A FORWARD -s www.suaempresa.com -d 10.0.30.0 -j ACCEPT

Praticando

- Caso 7:
  - Agora, todos os pacotes oriundos de qualquer rede que penetrem em nosso Firewall pela interface de rede eth2 serão redirecionados para o computador 10.0.30.47. Observe que nesta regra utilizamos a tabela NAT:

    iptables -t nat -A PREROUTING -i eth2 -j DNAT −−to 10.0.30.47

    Note que agora utilizamos a tabela nat e sua chain PREROUTING, que nada mais faz do que aplicar a regra antes do roteamento do pacote.

Praticando

- Caso 8:
  - Nosso próximo caso também utilizará a tabela NAT e fará com que qualquer pacote que deseje sair da rede local para outra rede possua seu endereço de origem alterado para 192.168.0.33, implementando assim o conceito de mascaramento de ip. Este mesmo pacote somente sofrerá modificações se sair pela interface de rede eth2. Utilizaremos a ação "-o", que se refere a interface de saída (OUTPUT), ao contrário da ação -i que se refere a interface de entrada (INPUT):

    iptables -t nat -A POSTROUTING -o eth2 -j SNAT ––to 192.168.0.33

    Note que agora utilizamos a tabela nat e sua chain PREROUTING, que nada mais faz do que aplicar a regra antes do roteamento do pacote.

Praticando

- Caso 9:
  - Utilizaremos a tabela filter para rejeitar (REJECT) pacotes entrantes pela interface de rede eth1:

    iptables -A FORWARD -i eth1 -j REJECT

Praticando

- Caso 10:
  - Faremos agora com que pacotes que entram por qualquer interface de rede com excessão da eth0 sejam descartados:

    iptables -A FORWARD ! -i eth0 -j DROP

Praticando

- Caso 11:
  - Vamos então deletar a segunda regra inserida sobre a chain FORWARD:

    iptables -D FORWARD 2

Praticando

- Caso 12:
  - Listaremos agora a segunda regra associada as output chain:

    iptables -L OUTPUT

Praticando

- Caso 13:
  - Vamos então descartar qualquer pacote oriundo do IP 10.0.80.32 destinado ao IP 10.0.30.4:

    iptables -A FORWARD -s 10.0.80.32 -d 10.0.30.4 -j DROP

Praticando

- Caso 14:
  - Pacotes TCP destinados à porta 80 de nosso host firewall deverão ser descartados:

    iptables -A INPUT -p tcp −−dport 80 -j DROP

Praticando

- Caso 14:
  - Faremos com que pacotes destinados a porta 25 de nosso host firewall sejam arquivadas em log (/var/log/messages):

    iptables -A INPUT -p tcp −−dport 25 -j LOG
  - Note que diante de tal regra temos a possibilidade de, por exemplo, arquivar tal tráfego em Log e logo depois descarta-lo conforme o exemplo a seguir:

    iptables -A INPUT -p tcp −−dport 25 -j LOG
    iptables -A INPUT -p tcp −−dport 25 -j DROP

Praticando

- Especificando **RETURN** como alvo:
  - O alvo RETURN diz ao iptables para interromper o processamento no chain atual e retornar o processamento ao chain anterior. Ele é útil quando criamos um chain que faz um determinado tratamento de pacotes, por exemplo bloquear conexões vindas da internet para portas baixas, exceto para um endereço IP específico. Como segue:
    1. **iptables -t filter -N internet**
    2. **iptables -t filter -A INPUT -i ppp0 -j internet**
    3. **iptables -t filter -A INPUT -j ACCEPT**
    4. **iptables -t filter -A internet -s www.debian.org -p tcp −−dport 80 -j RETURN**
    5. **iptables -t filter -A internet -p tcp −−dport 21 -j DROP**
    6. **iptables -t filter -A internet -p tcp −−dport 23 -j DROP**
    7. **iptables -t filter -A internet -p tcp −−dport 25 -j DROP**
    8. **iptables -t filter -A internet -p tcp −−dport 80 -j DROP**

Praticando

- Quando um pacote com o endereço www.debian.org tentando acessar a porta www (80) de nossa máquina através da internet (via interface ppp0), o chain número 1 confere, então o processamento continua no chain número 4, o chain número 4 confere então o processamento volta para a regra número 2, que diz para aceitar o pacote.
- Agora se um pacote vem com o endereço www.dominio.com.br tentando acessar a porta 80 (www) de nossa máquina através da internet (via interface ppp0), o chain número 1 confere, então o processamento continua no chain número 4, que não confere. O mesmo acontece com os chains 5, 6 e 7. O chain número 8 confere, então o acesso é bloqueado.

NAT

- Uma das funções de um Firewall Nat é o que conhecemos por SNAT, ou simplesmente "tradução de endereçamento de origem" (*source nat*).
- O alvo (*target*) SNAT lhe dá a possibilidade de alterar os endereços/portas de origem dos pacotes que atravessam seu host Firewall antes que os mesmos sejam roteados a seu destino final.

NAT

- Considerações importantes:
  - Qualquer regra aplicada a **SNAT** utiliza-se somente da chain **POSTROUTING**. Logo, se é SNAT que você quer, é POSTROUTING que você usa.
  - Antes de iniciarmos a manipulação de qualquer regra que se utilize da tabela NAT, é importante que habilitemos a função de redirecionamento de pacotes (forward) em nosso kernel através do seguinte comando:
    - echo 1 > /proc/sys/net/ipv4/ip_forward

- Exemplo SNAT 1:
  - **iptables -t nat -A POSTROUTING -s 10.0.3.1 -o eth1 -j SNAT −−to 192.111.22.33**

    Qualquer pacote que possua como origem (-s 10.0.3.1) o host 10.0.3.1 e que saia por nossa eth1 (-o eth1) deverá possuir seu endereço de destino alterado (-j SNAT) para 192.111.22.33 (−−to 192.111.22.33)

- Exemplo SNAT 2:
  - **iptables -t nat -A POSTROUTING -s 10.0.3.0/8 -o eth0 -j SNAT −−to 192.111.22.33**

    Neste exemplo, qualquer pacote que possua como origem a rede 10.0.3.0/8, e que sair por nossa interface eth0 deverá ter seu endereço alterado para 192.111.22.33

NAT

- Outra função agregada a um Firewall Nat é o DNAT, ou tradução de endereço de destino (*destination nat*).
- O alvo (*Target*) DNAT lhe dá a possibilidade de alterar os endereços/portas de destino dos pacotes que atravessam seu host Firewall antes que os mesmos sejam roteados a seu destino final. Com isso o DNAT nos possibilita o desenvolvimento de proxys transparentes, balanceamento de carga, entre outros.
- As regras do **DNAT**, ao contrário do SNAT, utilizam-se tão somente da chain **PREROUTING**. Logo, se é DNAT que você vai fazer, é PREROUTING que você vai usar!

NAT

- Exemplo DNAT 1:
  - **iptables -t nat -A PREROUTING -s 10.0.3.1 -i eth1 -j DNAT −−to 192.111.22.33**

    Qualquer pacote que possua como origem o host 10.0.3.1 e que entre por nossa interface eth1 deve ter seu endereço de destino alterado para 192.111.22.33.

- Exemplo DNAT 2:
  - **iptables -t nat -A PREROUTING -i eth0 -j DNAT −−to 192.11.22.10-192.11.22.13**

    Qualquer pacote que entre por nossa interface de rede eth0 independente de quem o enviou, deve ser automaticamente redirecionando aos hosts 192.11.22.10, 192.11.22.11, 192.168.11.12 e 192.11.22.13.

Módulos

- **Módulos**
  - Um módulo nada mais e do que uma forma de se ampliar a funcionalidade da ferramenta iptables.
  - Para utilizar um módulo é necessário usar a opção -m <*módulo*> ou −−match <*módulo*>.

Módulos

- Exemplos de Módulos:

limit	Limita o número de vezes que uma regra será executada antes de passar para a próxima regra.
state	Observa o estado da conexão. Estes podem ser (NEW, ESTABLISHED, RELATED e INVALID).
mac	Permite que o iptables trabalhe com endereçamento físico.
multiport	Permite que sejam especificadas até 15 portas e uma regra de uma só vez.
string	Observa o conteúdo do pacote para somente então aplicar a regra.
owner	Observe o usuário que criou o pacote.

limite

- Regras sob o módulo *limit* especificam exatamente quantas vezes as mesmas devem ser executadas em um intervalo de tempo específico, e, se isso acontecer, ela automaticamente deve executar a regra seguinte.
- Observe o exemplo a seguir:
  - **iptables -A INPUT -p icmp –icmp-type echo-request -m limit ––limit 1/s -j ACCEPT**

limite

- O exemplo anterior tem como principal finalidade impedir o velho golpe denominado "Ping da morte". Sua síntese basicamente diz que pacotes respostas de ICMP (-p icmp) serão aceitos somente se recebidos em um intervalo de tempo de 1 segundo (-m limit −−limit 1/s -j ACCEPT).
- Caso algum pacote ultrapasse este limite imposto pela regra, a mesma deverá automaticamente executar a seguinte (próxima regra do firewall) que deverá ser algo como:
  - **iptables -A INPUT -p icmp -j DROP**

limite

- ––limit num/tempo - Permite especificar a taxa de conferências do limit. O parâmetro num especifica um número e tempo pode ser
  - s - Segundo
  - m - Minuto
  - h - Hora
  - d - Dia

state

- O módulo **state** atribui regras mediante a análise do estado da conexão de um pacote. Este estados são:
  - **NEW** - Indica que o pacote está criando uma nova conexão;
  - **ESTABLISHED** - Informa que o pacote pertence a uma conexão já existente, logo, trata-se de um pacote de resposta;
  - **RELATED** - Referente a pacotes que relacionam-se indiretamente com outro pacote, a exemplo das mensagens de erros de conexão;
  - **INVALID** - Referente a pacotes não identificados por algum motivo desconhecido (Erros de ICMP que não correspondam a 8 e outros). Geralmente aconselha-se que tais pacotes sejam descartados pelo firewall (DROP).
- É possível então utilizar-se de mais do que apenas um estado de conexão por regra. Neste caso os separamos com uma simples vírgula (sem deixar espaços). Por exemplo: NEW,INVALID.

state

- Observe o seguinte exemplo de uma regra que faz uso de módulo state:
  - **iptables -A INPUT -m state −−state NEW -i eth0 -j ACCEPT**
- A regra acima permite qualquer nova conexão que parta da interface eth0.
  - **iptables -A INPUT -m state −−state INVALID -i eth0 -j DROP**.
- Já a regra acima bloqueia qualquer pacote cujo estado de conexão seja considerado inválido.
  - **iptables -A INPUT -m state −−state ESTABLISHED,RELATED -j ACCEPT**
- Acima aceitamos qualquer pacote sob os estados ESTABLISHED e RELATED, ou seja, a pacotes que já estabeleceram uma conexão e pacotes não identificados mas que possuem alguma ligação indireta com outros pacotes identificados.

mac

- O módulo **mac** permite que o Firewall filtre por endereços físicos. Acompanhe o exemplo:
  - **iptables -A INPUT -m mac −−mac-source 40:F0:B2:8F:00:01 -j DROP**
- Utilizamos o módulo mac (-m mac) seguido do parâmetro −−mac-source para indicar o endereço mac de origem do pacote, que neste caso é 40:F0:B2:8F:00:01 para então bloquearmos (-j DROP) qualquer pacote proveniente de tal endereço físico.

multiport

- Com o módulo **multiport** é possível que especifiquemos múltiplas portas a um determinado alvo sob o limite máximo de 15. Exemplo:
  - **iptables -A INPUT -p tcp -i eth0 -m multiport −−dport 80,25,53,110 -j DROP**
- Acima indicamos em uma só regra por via de multiport que nosso firewall descartará (-j DROP) qualquer pacote que entre pela interface eth0 (-i eth0) destinados às portas 80,25,53 e 110.
- O módulo multiport pode ser especificado tanto para portas de destino (−−dport) quanto para portas de origem (−−sport) tal como também sob o parâmetro único −−port, este fará o trabalho de ambos os parâmetros (−−sport e −−dport) dentro da regra, ou seja, se especificado na regra "−−port 80,53 -j DROP", esta recusará tanto pacotes provenientes as portas 80 e 53 quanto destinadas as mesmas.

string

- O módulo **string** é útil quando precisamos realizar um controle de tráfego baseado no conteúdo de um pacote. A sintaxe utiliza é:
  - -m string −−algo <*algoritmo*> −−string "<*palavra-chave*>"
  - Onde <*algoritmo*> é o algoritmo empregado para o rastreamento de palavras chaves que pode ser:
    - bm
    - kmp

string

- Exemplo para o módulo **string**:
  - **iptables -A INPUT -m string −−algo bm −−string “sexo” -j DROP**

owner

- O módulo **owner** é pouco utilizado e mencionado em literaturas do gênero. Este módulo é capaz de determinar precisamente algumas informações valiosas sobre o criados de um determinado pacote definido em regra, de modo que se tornará possível identificar o emissor real do pacote (no nível do usuário).
- Este módulo pode ser utilizado tão somente em combinação com a chain OUTPUT conforme veremos no exemplo a seguir, que bloqueará a saída de qualquer pacote UDP que seja criado por um usuário do grupo sob o GID 81:
  - iptables -A OUTPUT -m owner −−gid-owner 81 -p udp -j DROP

owner

- As opções de filtro no nível de usuários, grupos e processos sob owner são:

Opções	**Descrição**
−−uid-owner	Controla e executa a regra se o pacote fora criado por um usuário sob o **userid** especificado
−−gid-owner	Controla e executa a regra se o pacote fora criado por um usuário sob o **groupid** especificado
−−pid-owner	Controla e executa a regra se o pacote fora criado por um número de processo (*processid*) especificado
−−sid-owner	Controla e executa a regra se o pacote fora criado por um processo concebido por session group

owner

- Analise então o seguinte exemplo:
  - **iptables -A OUTPUT -m owner −−uid-owner 42 -p tcp -j ACCEPT**
  - **iptables -A OUTPUT -p tcp -j DROP**
- Note que acima a primeira regra de nosso exemplo permite (-j ACCEPT) apenas que o usuário sob ID 42 (-m owner −−uid-owner 42) envie pacotes para fora de nossa rede sob o protocolo tcp (-p tcp).
- Já a regra seguinte proíbe (-j DROP) qualquer saída de pacotes tcp (-p tcp -j DROP) independente do usuário que os enviou. Logo, concluímos que o único usuário da rede que poderá enviar seus pacotes tcp pelo Firewall será o que estiver sob UID (userid) 40.

Referências

- Linux Firewall Iptables. Urubatan Neto, 2004. Ed. Ciência Moderna.
- https://www.vivaolinux.com.br/artigo/Dominando-o-iptables-(parte-1)
- http://www.guiafoca.org/cgs/guia/avancado/ch-fw-iptables.html