



# Firewall Iptables

Software Livre na Cultura  
TchêLinux

Elgio Schlemer  
Ulbra Gravatai

<http://gravatai.ulbra.tche.br/~elgio>

18/Outubro/2009

# Sumário

- Introdução a Pilha TCP/IP
- Conceitos e tipos de Firewalls
- Iptables
  - suas tabelas
  - Sintaxe das regras
  - Poder *statefull* do iptables



Primeira parte

Introdução à Pilha TCP/IP

# Pilha TCP/IP

Aplicação

Transporte

Rede

Enlace

- Modelo de 4 camadas
  - nível físico é hardware
- Enlace
  - endereçamento de hardware
  - Ethernet: uso de mac address
  - Cabeçalho Ethernet tem apenas 3 campos: destino, origem, tipo
  - Não relevante para Firewall

# Nível de Rede

Aplicação

Transporte

Rede

Enlace

- Responsável pelo roteamento
  - como chegar ao destino
  - através do repasse Enlace de uma placa de rede para outra
  - analisa o endereço de destino
- Protocolos:
  - IPV4
    - números IPs de 32 bits
  - IPV6
    - números IPs de 128 bits
  - ICMP
    - controle de erros e diagnóstico

# Nível de Transporte

Aplicação

Transporte

Rede

Enlace

- Responsável pela confiabilidade
  - garantia de entrega, ordenamento, retransmissão.
- Identificação de programas
  - através do número da porta
- Protocolos:
  - UDP
    - não possui nenhuma garantia
  - TCP
    - possui todas as garantias

# Nível de Aplicação

Aplicação

Transporte

Rede

Enlace

- programas de usuário
- geram ou pegam dados da camada de transporte
  - mediante associação à uma porta
- Protocolos:
  - HTTP
    - tipicamente usa a porta 80
  - SSH
    - porta padrão a 22

# Protocolos de aplicação

- Cada aplicação pode possuir seu protocolo
  - Exemplo HTTP:
    - HOST: indica qual o domínio que se quer
    - GET: indica qual o arquivo, documento que se quer
    - POST: indica para onde os dados devem ser enviados
    - demonstração: uma conversa HTTP capturada
  - Exemplo SMTP:
    - hello, mail from, rcpt to, etc
- Cada aplicação tem seu cabeçalho
  - Mas que são meros bytes de dados para as camadas inferiores (transporte)

# Protocolos de transporte

- consideram tudo que receber da camada superior como dados.
- Usam números de portas para identificar qual processo gerou os dados.
- Dois protocolos populares:
  - *User Datagram Protocol* (UDP)
  - *Transmission Control Protocol* (TCP)
- Extremamente importantes para configurar firewalls

# Protocolo UDP

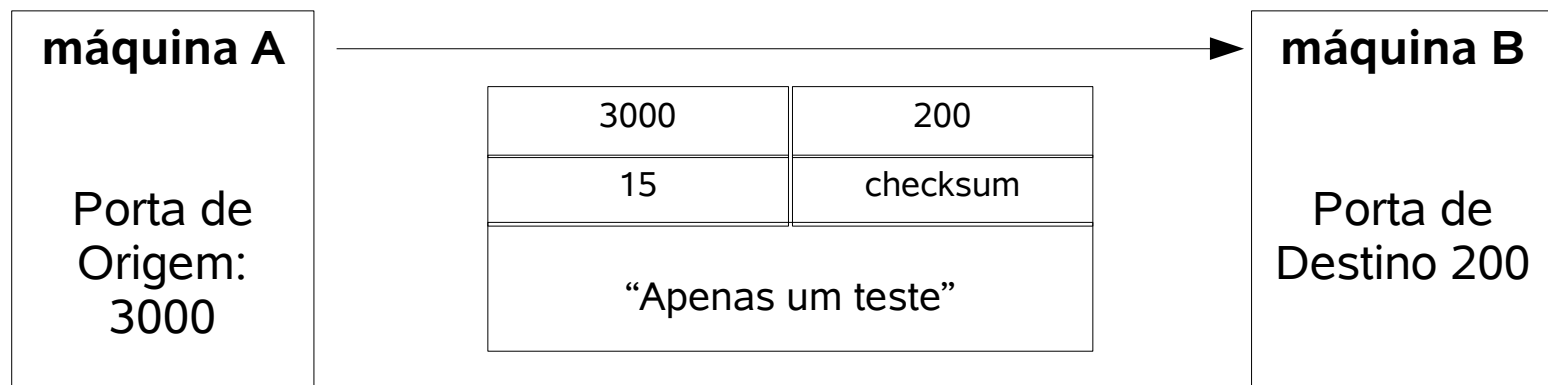
- Feito para ser simples e eficiente
- não trata nenhum problema de transmissão
  - desordenamento
  - perda de pacotes
  - repetição de pacotes
- Protocolo não confiável
  - se foi foi, se não foi, já era
  - Se necessário a confiabilidade deve ser implementada pelo programador na aplicação

# Protocolo UDP (2)

- Cabeçalho simples (apenas 8 bytes)
- Sem conexão e sem estado de sessão
  - quer enviar, apenas envia
- vantagens que o tornam interessantes
  - leve com pouco *overhead* (8 bytes de custo do cabeçalho)
  - permite transmissões *multicast* e *broadcast*
- Aplicações de sucesso: DNS, VoIP, Ghost

# Cabeçalho UDP

16 bits	16 bits
Porta de Origem	Porta de Destino
Tamanho	Soma de verificação
Dados	



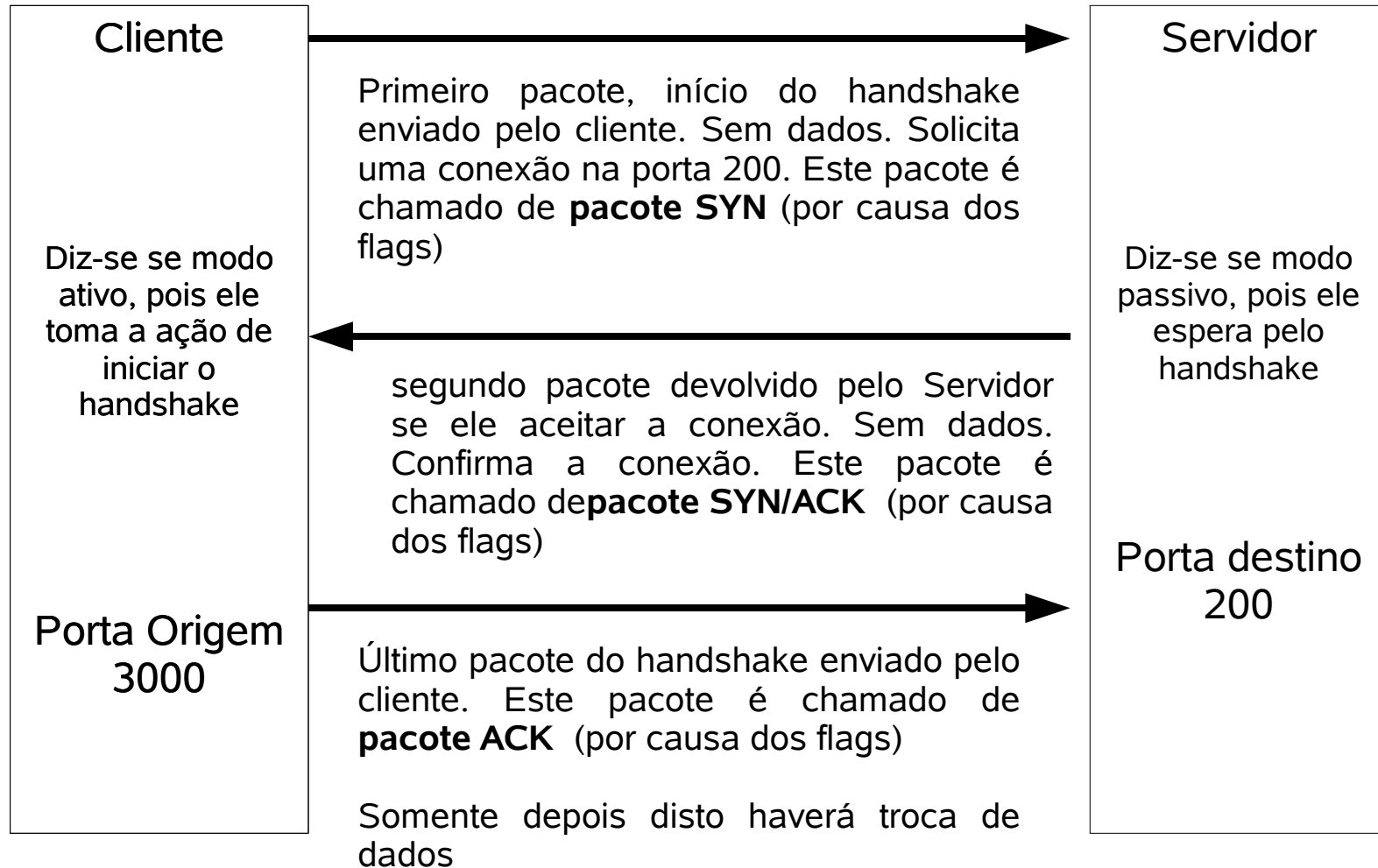
# Protocolo TCP

- Orientado a conexão
  - cliente e servidor decidem se conectar antes de enviar qualquer dado
  - como em uma ligação telefônica (*handshake*)
- garantia de tudo
  - retransmissão de pacotes perdidos
  - ordenamento de pacotes
  - tratamento de pacotes repetidos

# Protocolo TCP (2)

- Cabeçalho maior
  - mínimo de 5 words (20 bytes)
  - máximo de 15 Words (60 bytes)
  - por ser de tamanho variável, o tamanho do cabeçalho é informado
- Aplicações de sucesso:
  - HTTP, SMTP, POP, SSH
  - Todas que precisam de confiabilidade
- desvantagens:
  - oneroso (depende!)
  - não permite *multicast* e nem *broadcast*

# Handshake TCP



# Cabeçalho TCP

0	4	8	12	16	20	24	28
Porta Origem				Porta Destino			
Número de Seqüência							
Número de confirmação							
HLEN	Reservado	Flags		Tamanho da Janela			
Checksum				Ponteiro Urgente			
(...) DADOS (...)							

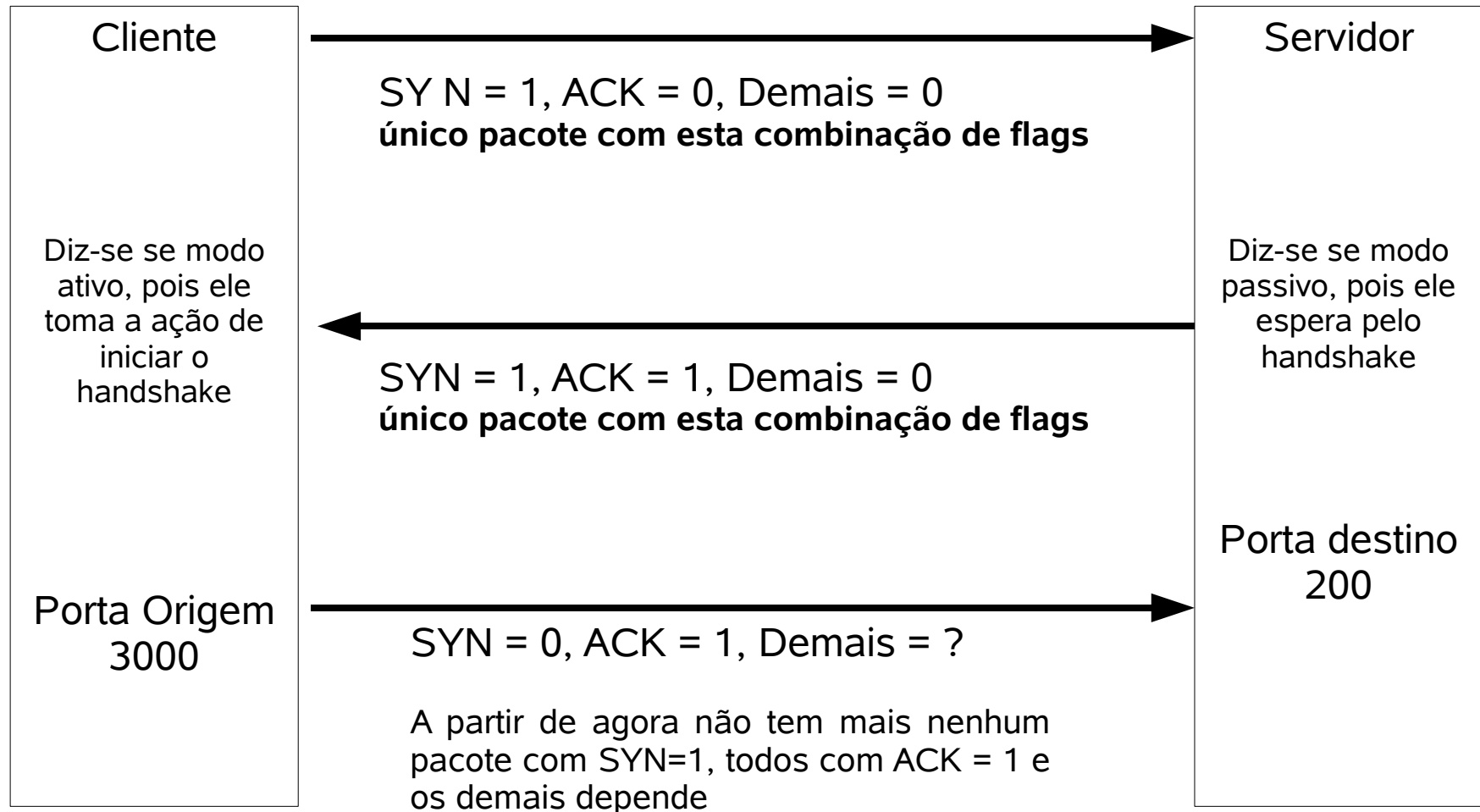
# Flags TCP

- Indicam o tipo de pacote (principalmente os de handshake)
- cada flag é apenas 0 (desligado) ou 1 (ligado)
- SYN:
  - indica sincronização de números sequenciais.
  - usado apenas no primeiro e segundo pacote do handshake
- ACK:
  - indica haver uma confirmação
  - usado a partir do segundo pacote do handhshake

# Flags TCP (2)

- RST:
  - reseta uma conexão (“bater o telefone”)
- FIN:
  - indica pacotes de finalização “educada” da conexão
  - Reset é uma finalização brusca, unilateral
- PSH:
  - para forçar o descarregamento do buffer
- URG:
  - indica que o pacote possui dados urgentes

# Handshake com flags



# Estados de uma conexão TCP

- Início do *handshake*:
  - pacote que tem SYN=1 e ACK=0
- segundo pacote do *handshake*:
  - pacote que tem SYN=1 e ACK=1
- Demais não tem mais o flag de SYN ligado
- Após o *handshake* diz-se que a conexão TCP está estabelecida.

# Protocolo IPv4

- Responsável pelo roteamento
- regras de roteamento analisam o IP de destino
- Ip possui 32 bits
  - exemplo:
    - 11000000 10101000 00000000 00001010
    - ou 192.168.0.10
- Máscaras de rede
  - Definem o ip da rede e de broadcast

# Máscaras de rede

- A máscara diz quantos bits iniciais são usados para rede e quantos para host
- Exemplo: 10.1.0.12/24
  - 24 bits para rede, logo apenas 8 para host
  - com 8 bits tem-se 256 ips
  - Ip da rede 10.1.0.0
  - Ip broadcast 10.1.0.255
- /24 é mais conhecida como 255.255.255.0, pois:
  - 11111111 11111111 11111111 00000000
    - 24 bits de rede em 1 e os 8 bits de host em 0
    - Ao se ler isto de forma decimal tem-se 255.255.255.0
- Usar IP/máscara é a única forma de se aplicar a mesma regra a mais de um número IP no iptables

# Ips Privados

- não podem ser usados na Internet
- frequentemente chamados de “ips não roteáveis” ou “ips inválidos”
  - mas eles são roteáveis na intranet!
  - e são válidos pois podem ser atribuídos
- Definidos pela RFC como:
  - 10.0.0.0/8: de 10.0.0.0 até 10.255.255.255
  - 172.16.0.0/12: de 172.16.0.0 até 172.31.255.255
  - 192.168.0.0/16: de 192.168.0.0 até 192.168.255.255
- Os Ips “roteáveis” na Internet são chamados de ips públicos

# Protocolo ICMP

- Responsável por sinalizar erros de rede
- possui vários tipos
- cada tipo pode ter um código
- tipo + código definem qual erro ocorreu
- Tipos interessantes
  - tipo 0: *echo reply*, resposta de um ping
  - tipo 8: *echo request*, solicitação de ping
  - tipo 3: destino inacessível

# ICMP tipo 3

- O mais retornado, pois indica que o pacote não chegou ao destino.
- Porque?
  - código diz o motivo
- Códigos para o tipo 3
  - Código 0: rede inacessível
  - Código 2: protocolo inacessível
  - Código 3: porta inacessível
    - Quando se envia um pacote UDP para uma porta não alocada se recebe este ICMP
    - Para TCP também, muito embora um reset é o recomendado pela RFC.

# Exemplo

- sniffando uma comunicação UDP
- Sniffando uma conexão TCP
  - navegador acessando uma página
  - mostrar os flags
  - mostrar os cabeçalhos IP
- Os flags e o estado de uma conexão são muito importantes para um firewall

Fim primeira parte

Introdução ao TCP

Perguntas?



Segunda parte

Conceitos e tipos de Firewalls

# Tipos de firewalls

- Quanto a atuação nas camadas da pilha TCP/IP
- Quanto a abrangência da proteção
- Quanto ao poder
- Quanto a ação padrão

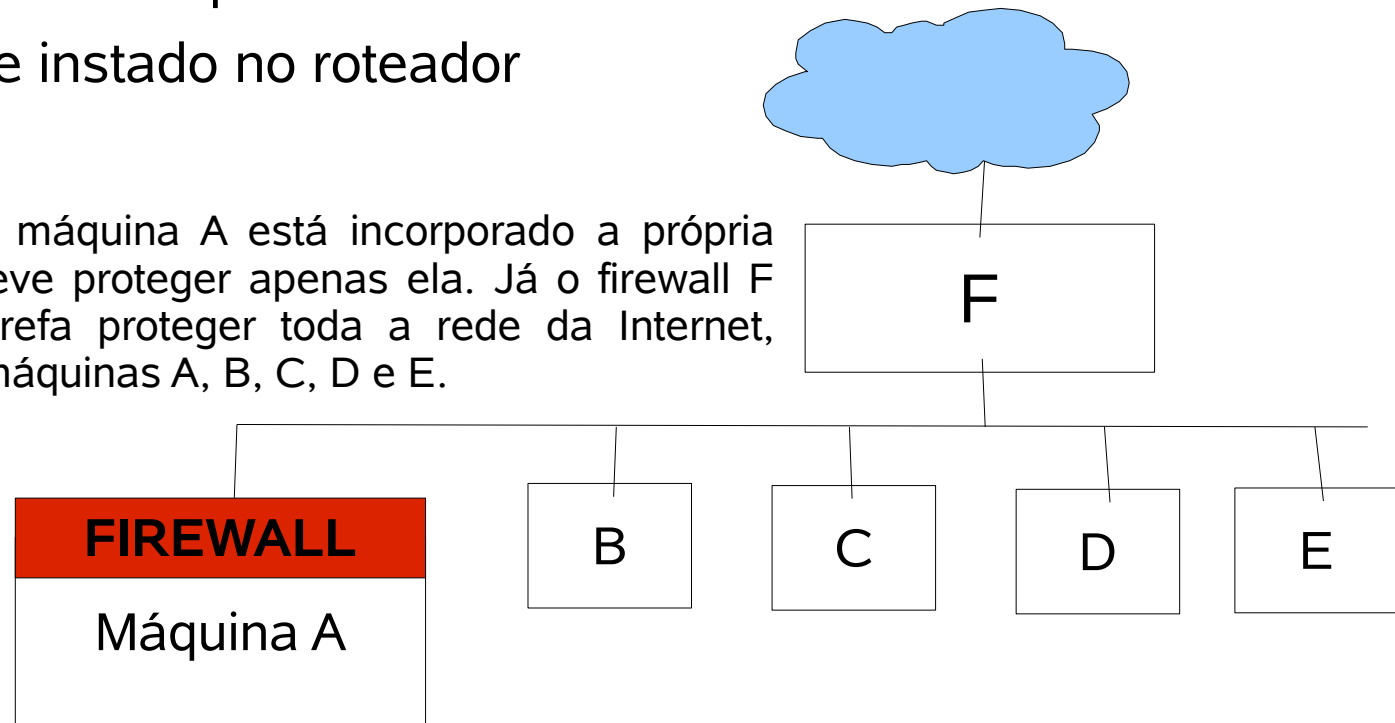
# Quando a pilha TCP

- de aplicação:
  - permitem analisar os dados e entender os protocolos
  - Proxy se enquadra neste tipo de firewall
    - Porque ele “entende” as requisições HTTP
    - permite bloquear URL por palavra chave
- filtro de pacotes
  - permitem apenas analisar cabeçalhos da pilha
  - Transporte: Portas, flags TCP...
  - Rede: números IP origem e destino, flags...
  - Enlace; números de mac address
    - não é muito útil no dia a dia

# Quando a abrangência

- de host (também chamado de pessoal)
  - protege apenas uma máquina, aquela na qual está instalado
- de rede
  - protege várias máquinas
  - geralmente instalado no roteador

O firewall da máquina A está incorporado a própria máquina e deve proteger apenas ela. Já o firewall F tem como tarefa proteger toda a rede da Internet, incluindo as máquinas A, B, C, D e E.



Fonte: Artigo para revista Viva o Linux ainda não publicado. Elgio

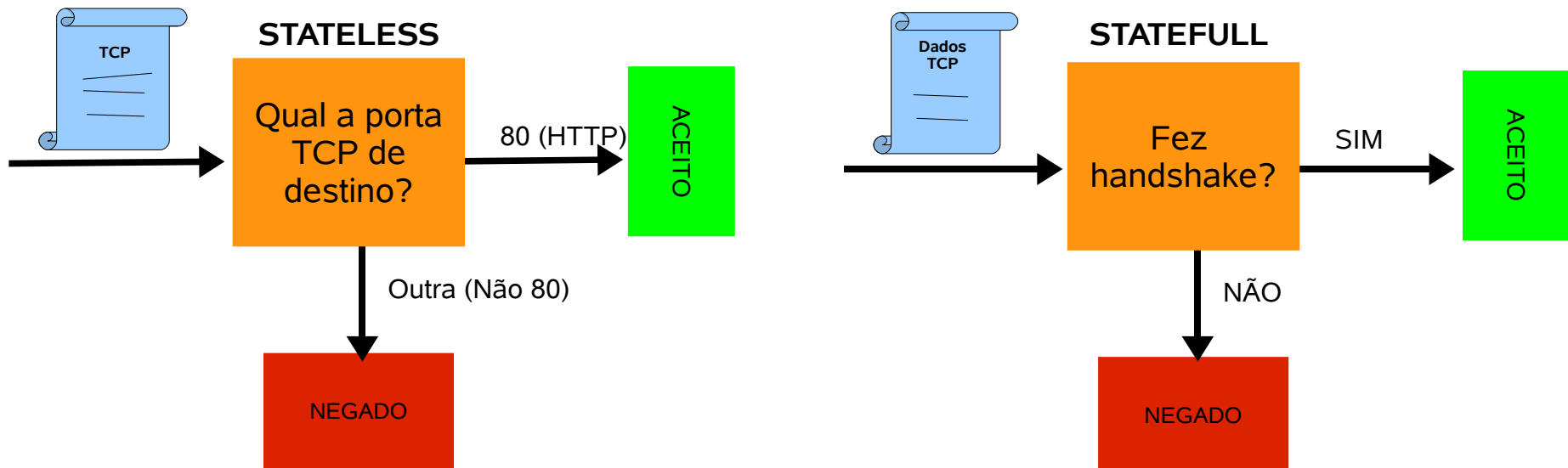
# Quando ao poder

- Stateless ou Statefull
- Stateless (livre de estado)
  - também chamado de estático.
  - não se “lembra” do passado
  - regras baseadas apenas no pacote atual e o que se pode extrair dele
  - ocupam menos recursos de hardware
  - praticamente todos os filtros incorporados à roteadores comerciais são *stateless*

# Quando ao poder (2)

- Statefull:
  - também chamado de dinâmico
  - possui “memória”, permite lembrar-se do passado
  - ação pode ser baseada ao que já passou
  - ocupam mais recursos de hardware
  - Exemplos de regras statefull:
    - cliente fez o handshake? Se sim, aceita conexões
    - quantos SYN's este cliente enviou? Muitos então passa a recusar novos.
    - Este IP bateu na porta 5000 a menos de 1 segundo atrás? Sim, então libera a porta 22 para ele (usado para implementar *port knocking*)

# Stateless x Statefull



**Stateless ou estático:** para aceitar o pacote basta o firewall analisar o conteúdo do atual pacote, olhando se a porta de destino dele é ou não porta 80. Se for, aceita, se não for, nega.

**Statefull ou dinâmico:** para aceitar o pacote é necessário que tenha ocorrido o handshake TCP. Se ocorreu o firewall deve se lembrar, vendo em suas tabelas os pacotes anteriores. É chamado de dinâmico porque suas regras mudam de acordo com os pacotes que passam (fez handshake? Insere uma regra aceitando os dados)

Fonte: artigo para a Revista Viva o Linux ainda não publicado. Elgio Schlemer

# O que um filtro pode fazer?

- Aceitar um pacote
  - permitir que ele siga o seu caminho
- Rejeitar um pacote
  - impedir que ele prossiga, descarta-o
- Realizar logs
  - registrar em arquivo de logs as características de um pacote.
- Alterar um pacote
  - trocar algumas informações dos cabeçalhos
  - nem todos permitem isto!

# O que um filtro NÃO pode fazer?

- verificar se os dados tem vírus
  - Observação: alguns “firewalls” pessoais para Windows fazem isto. Mas observe que eles não são firewalls, mas sim uma solução completa de Firewall + Antivirus + IDS.
- filtrar por conteúdo, strings
  - Observação: um filtro de pacotes não deveria fazer isto, pois se fizer se enquadraria como filtro de aplicação. O Iptables se confunde ao permitir isto através do módulo layer7. Duvidoso, muito embora algumas coisas legais, como bloquear alguns P2P são bem vindas
- garantir a segurança
  - ele é parte da segurança, mas nada garante a segurança
  - Observação: instalar um firewall pode dar a falsa sensação de segurança plena e incentivar o administrador a relaxar a guarda!

# Organização das regras

- cada regra possui
  - características que fazem um pacote casar com ela
    - Exemplo: “se o Ip de destino for...”, “se a porta de origem TCP não for...”
  - Ação a ser realizada com aquele pacote
    - Exemplo: “... então descarta”, “então aceita”
- A ordem da aplicação das regras é muito importante
  - pois o firewall testa da primeira até a última
  - pára o teste quando encontra uma que funciona (isto é, não testa com as regras seguintes)

# Ordenamento das regras

- Exemplo equivocado de regras inseridas em algum firewall:
  - Regra 1: se porta destino for 22, nega
  - Regra 2: se IP de origem for 10.1.0.1 e porta destino 22, aceita
- Regra 2 jamais será aplicada!
- Se a intenção do autor era fechar a porta 22 para todos, menos para 10.1.0.1, devia ser:
  - Regra 1: se IP de origem for 10.1.0.1 e porta destino 22, aceita
  - Regra 2: se porta destino for 22, nega
- Ou simplesmente (se o filtro suportar):
  - Regra 1: se porta destino for 22 e IP origem não for 10.1.0.1, nega

# E se não casar com nenhuma?

- O que ocorre quando um pacote não se enquadra em nenhuma regra?
- Exemplo: filtro com apenas 2 regras
  - Regra 1: se porta destino for 22, nega
  - Regra 2: se porta destino for 23, nega
- Chega um pacote destinado a porta 80:
  - não se enquadra na regra 1 e nem na regra 2
  - O que o filtro decide?
- Filtro permissivo e proibitivo

# Classificação quanto a ação padrão

- permissivo
  - todos os pacotes para os quais não tem uma regra específica serão aceitos
  - logo, se um pacote não casa com regra alguma, será aceito.
- proibitivo
  - todos os pacotes para os quais não tem uma regra específica serão rejeitados
  - logo, se um pacote não casa com regra alguma, será descartado.
- qual é o melhor?
- Como se comporta o iptables?
  - você decide!

Fim segunda parte

Conceitos Firewall

Perguntas?



Terceira parte

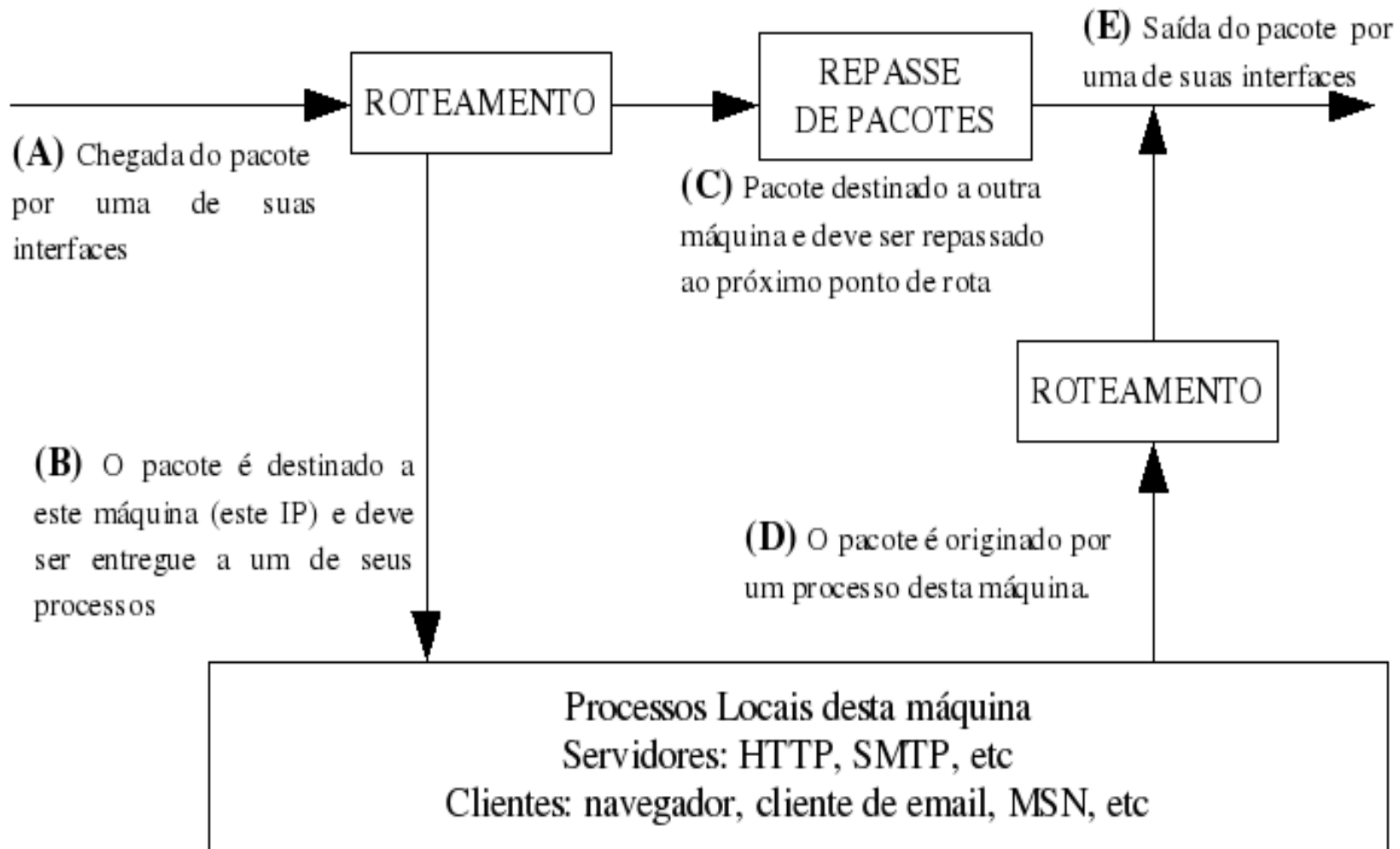
Iptables

Introdução, estrutura e organização

# Apresentando o netfilter

- na verdade o firewall do Linux se chama netfilter
- iptables é apenas um interpretador de comandos para inserir regras no netfilter
- netfilter é kernel
- iptables é aplicação
- netfilter permite aplicar regras em alguns pontos do “ciclo de vida” de um pacote no kernel

# Ciclo de vida de um pacote

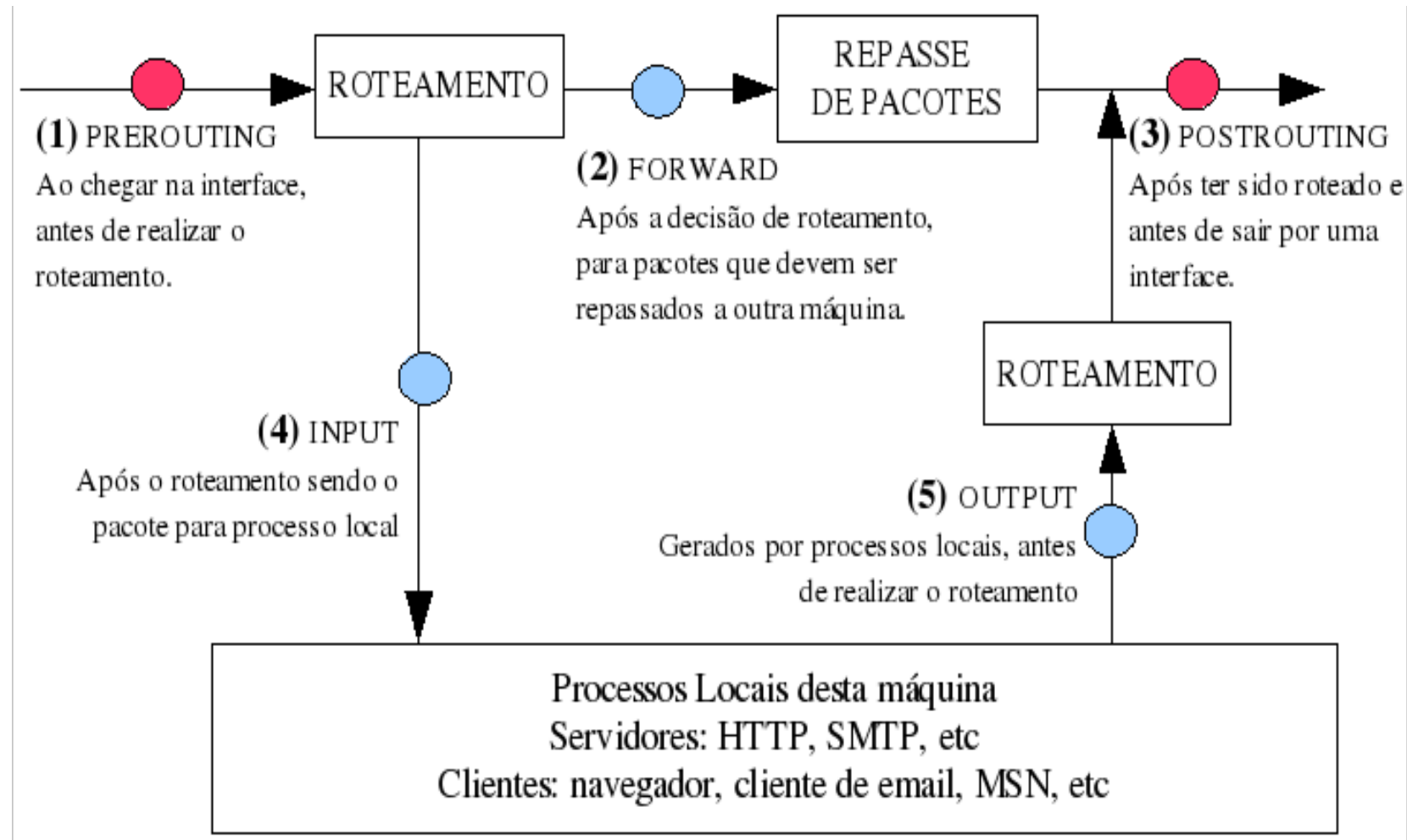


Fonte: artigo “Estrutura do Iptables”, Viva o Linux, Elgio Schlemer, 2007

# Ganchos do netfilter

- Dentro do ciclo de vida de um pacote
  - o netfilter inseriu “ganchos”
  - ganchos são pontos onde o pacote pode ser analisado
- Entender bem estes ganchos é a resposta definitiva para saber “onde uma regra vai” no iptables.

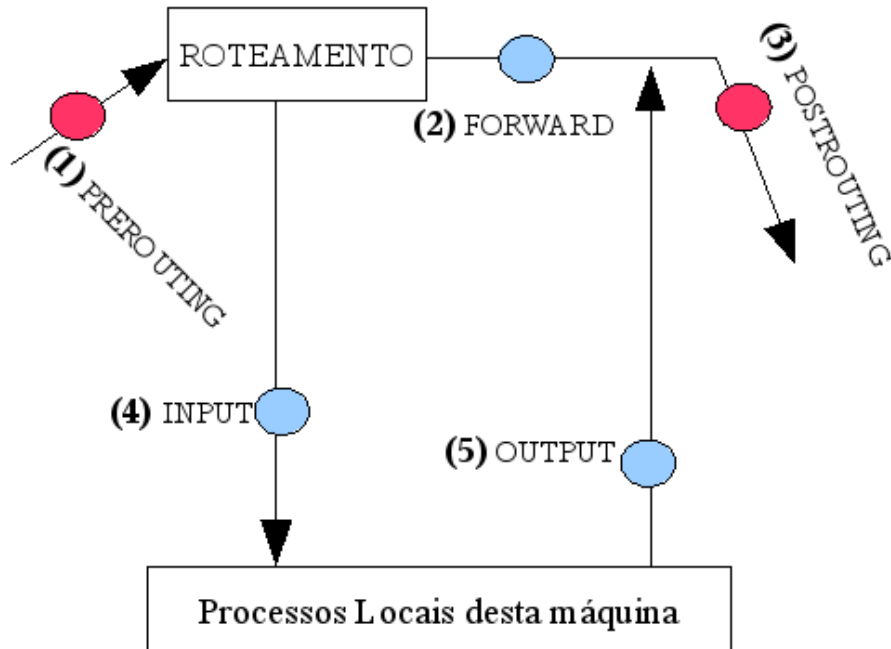
# Ganchos do netfilter



Fonte: artigo “Estrutura do Iptables”, Viva o Linux, Elgio Schlemer, 2007

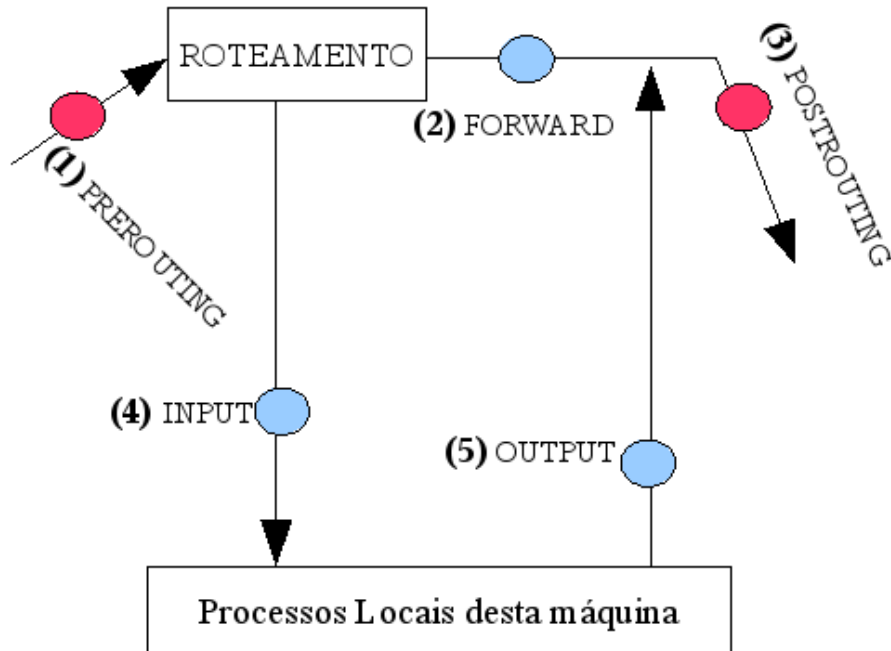
# Ganchos do netfilter (2)

- Um pacote destinado a um processo local:
  - pode ser analisado no PREROUTING, logo que entra na placa
  - pode ser analisado no INPUT
  - não passa pelo FORWARD!!



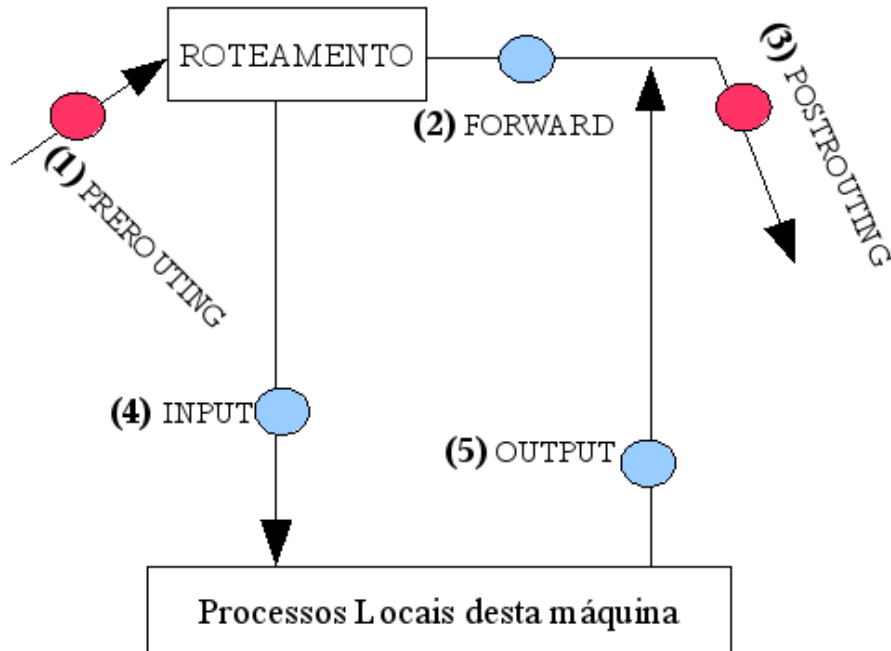
# Ganchos do netfilter (3)

- Um pacote gerado por um processo local:



- pode ser analisado no gancho OUTPUT
- pode ser analisado no POSTROUTING
- não passa pelo FORWARD!!

# Ganchos do netfilter (4)



- Um pacote roteado por este kernel:
  - pode ser analisado no PREROUTING, logo que entra na placa.
  - pode ser analisado no gancho FORWARD
  - Pode ser analisado no POSTROUTING, antes de sair da placa

# Iptables e suas tabelas

- Em alguns filtros se cria listas de regras
  - e dá-se nome a elas
- No iptables já existem três tabelas de regras fixas com nomes fixos: filter, nat e mangle
- Cada tabela serve para realizar determinada ação.
- cada tabela tem regras que atuam neste ou naquele gancho
  - ganchos PREROUTING, INPUT, OUTPUT, FORWARD e POSTROUTING

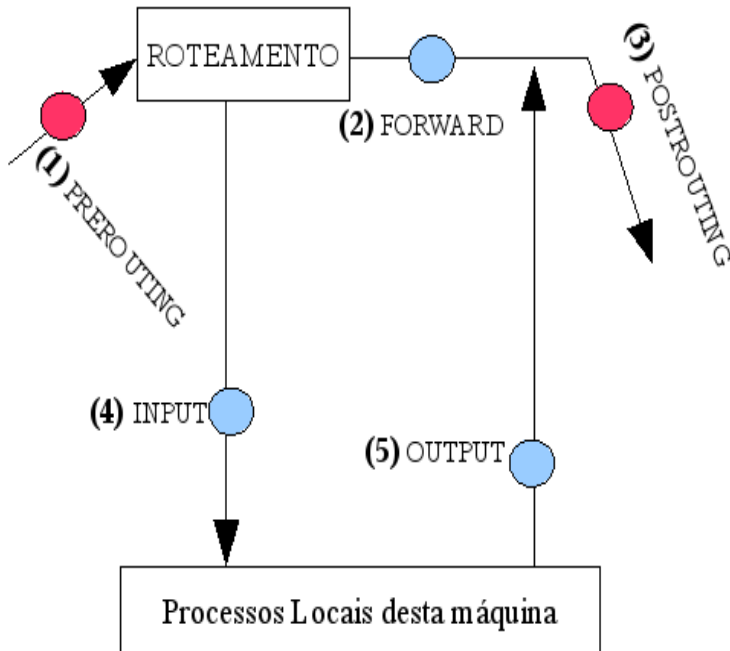
# A tabela filter

- esta é a única que pode ser chamada de firewall mesmo, filtro de pacotes
- é nela que devem ser colocadas todas as regras de filtragem
- permite atuar nos ganchos INPUT, OUTPUT e FORWARD
- **não atua** nos ganchos PREROUTING e POSTROUTING

# Em qual gancho?

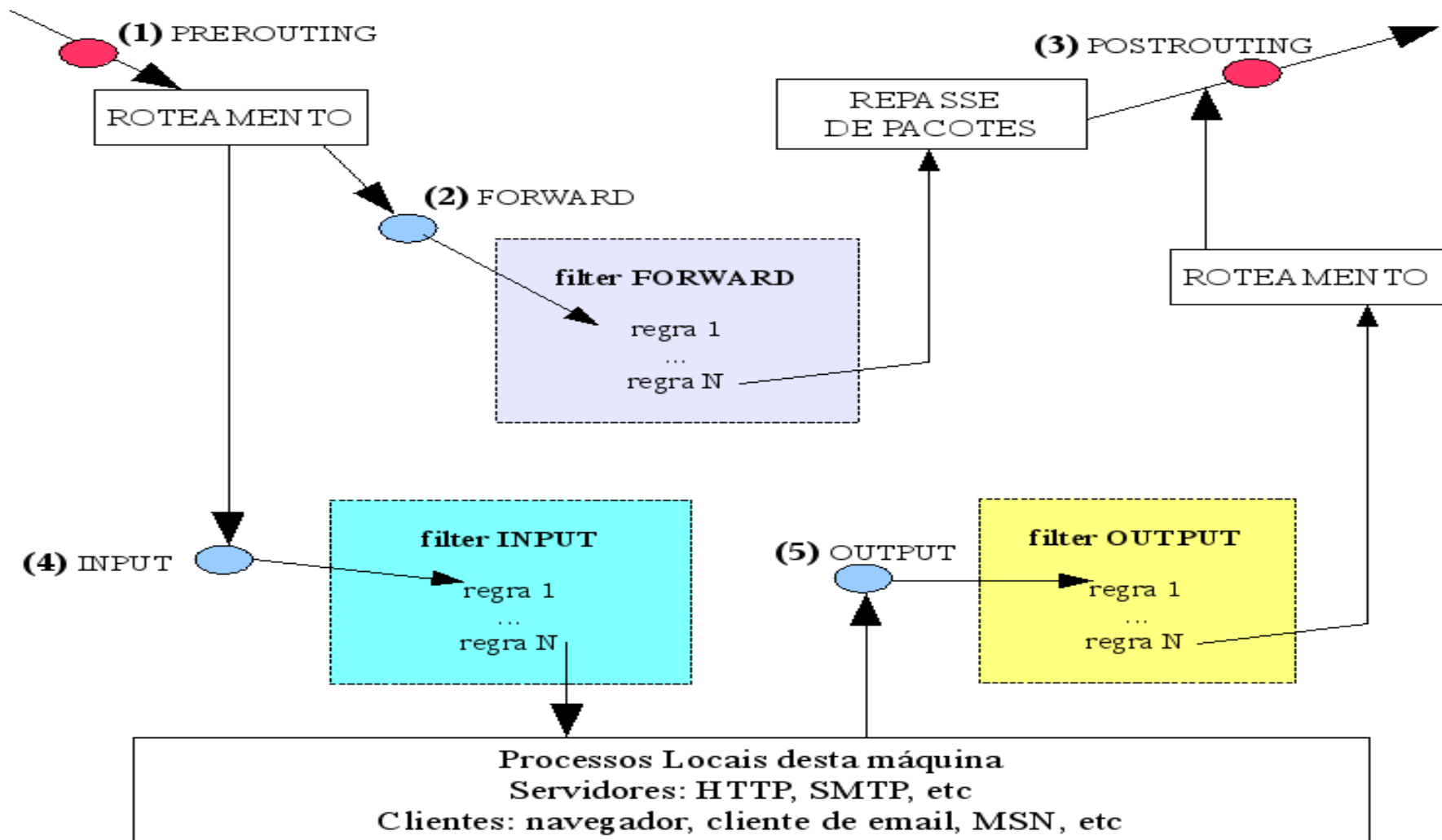
- vai no INPUT, OUTPUT ou FORWARD?
  - dúvida comum aos iniciantes
- OUTPUT: pega apenas pacotes **gerados** por processos locais
  - quero impedir que um usuário com um navegador na máquina acesse o IP 10.1.0.1: insiro no OUTPUT
- INPUT: pega apenas pacotes **destinados** ao IP desta máquina, aos processos locais
  - quero impedir que usuários acessem a minha porta 22: insiro no gancho INPUT
- logo:
  - INPUT e OUTPUT apenas quando o *firewall* é de host, protegendo a minha única máquina

# Em qual gancho? (2)



- FORWARD pega pacotes que “passam” pela máquina”
  - não é gerado por ela e nem destinada a ela
  - Quero que um pacote vindo da Internet destinado a uma máquina da minha rede na porta 22 seja bloqueado: FORWARD
- logo, FORWARD é quando o Linux está atuando com um filtro de rede.
- Veja que uma máquina pode ser ambos, rede pois protege várias máquinas, mas host pois protege a si mesma:
  - Quero que apenas os IPs 10.1.0.0/24 acessem a porta 22 do firewall para gerenciá-lo: INPUT

# Tabela filter atuando nos ganchos



# Poderees da tabela filter

- filter é o firewall propriamente dito
- ela só tem o poder de aceitar ou não um pacote
  - não pode modificar ele, por exemplo
    - nat e mangle é que tem este poder
- ou de registrar em logs
  - deve ter uma regra própria para registrar em logs

# Sintaxe do Iptables

- Uma regra iptables possui uma sintaxe
  - são parâmetros de linha de comando e alguns podem ser omitidos
- Principais parâmetros:
  - -t indica em qual tabela. Exemplo: iptables -t filter
    - se omitido, sempre será considerada a filter:
  - -A GANCHO: indica inserir esta regra ao final das regras para este gancho
    - exemplo: iptables -t filter -A FORWARD ...
  - -I GANCHO: indica inserir uma regra no início das regras para este gancho

# Principais parâmetros

- `-s IP/MASC`: indica qual IP/máscara de origem esta regra se aplica
  - se omitido, será considerado para qualquer ip
  - exemplo: `iptables -t filter -A FORWARD -s 10.1.0.1`
- `-d IP/MASC`: indica para qual IP/Máscara de destino esta regra se aplica.
  - se omitido, será considerado qualquer
  - exemplo: `iptables -t filter -A FORWARD -s 10.1.0.1 -d 172.16.0.0/24 ...`

# Principais parâmetros (2)

- -i indica estar entrando por uma interface de rede
  - se omitido, será considerado todas as interfaces
  - exemplo: `iptables -t filter -A FORWARD -i eth0 -s 10.1.0.1 -d 172.16.0.0/24 ...`
  - -i não tem qualquer sentido para o gancho OUTPUT, pois o pacote não entrou em interface alguma:
    - `iptables -t filter -A OUTPUT -i eth0 ...` (erro de sintaxe)

# Principais parâmetros (3)

- -o indica estar saindo por uma interface de rede
  - se omitido, será considerado todas as interfaces
  - exemplo: `iptables -t filter -A FORWARD -i eth0 -s 10.1.0.1 -o eth1 -d 172.16.0.0/24 ...`
  - -o não tem qualquer sentido para o gancho INPUT, pois o pacote não irá sair por nenhuma interface (será consumido por um processo interno)
    - `iptables -t filter -A INPUT -o eth0 ...` (erro de sintaxe)

# Principais parâmetros (4)

- -j indica a ação a ser tomada para este pacote
- algumas ações:
  - ACCEPT: o pacote deve ser aceito, nenhuma outra regra desta tabela será avaliada.
  - DROP: o pacote será descartado, nenhuma outra regra desta tabela será avaliada.
- exemplo:
  - `iptables -t filter -A FORWARD -i eth0 -s 10.1.0.1 -o eth1 -d 172.16.0.0/24 -j ACCEPT`
  - todo pacote entrando pela eth0, vindo do IP 10.1.0.1, saindo pela eth1 para algum IP da rede 172.16.0.0/24 será aceito

# Outras ações da filter

- REJECT: o pacote será descartado, nenhuma outra regra desta tabela será avaliada.
  - mas um retorno será dado ao usuário
  - um ICMP tipo 3, código 3 para o caso de UDP e TCP
    - para TCP isto pode ser mudado para um RST, pois ferramentas como nmap identificam firewall assim:
      - recebeu RST: porta inativa
      - recebeu ICMP: firewall bloqueou
      - não recebeu nada: firewall bloqueou com DROP
- DROP ou REJECT?
  - REJECT diz ao atacante “oi, eu sou o firewall”
  - DROP nada diz, mas o cliente insiste

# Outras ações da filter (2)

- LOG:
  - o pacote não será descartado
  - também não será aceito
  - apenas registrado em log
  - única ação que continua os testes após a aplicação da regra
  - Se desejo DROPAR e logar, devo:
    - primeiro logar e depois DROPAR  
`iptables -A FORWARD -s 10.1.0.1 -j LOG`
    - depois DROPAR  
`iptables -A FORWARD -s 10.1.0.1 -j DROP`
  - os logs são enviados para o syslog como sendo kernel
    - tipicamente ficam no `/var/messages`

# Parâmetros para protocolos

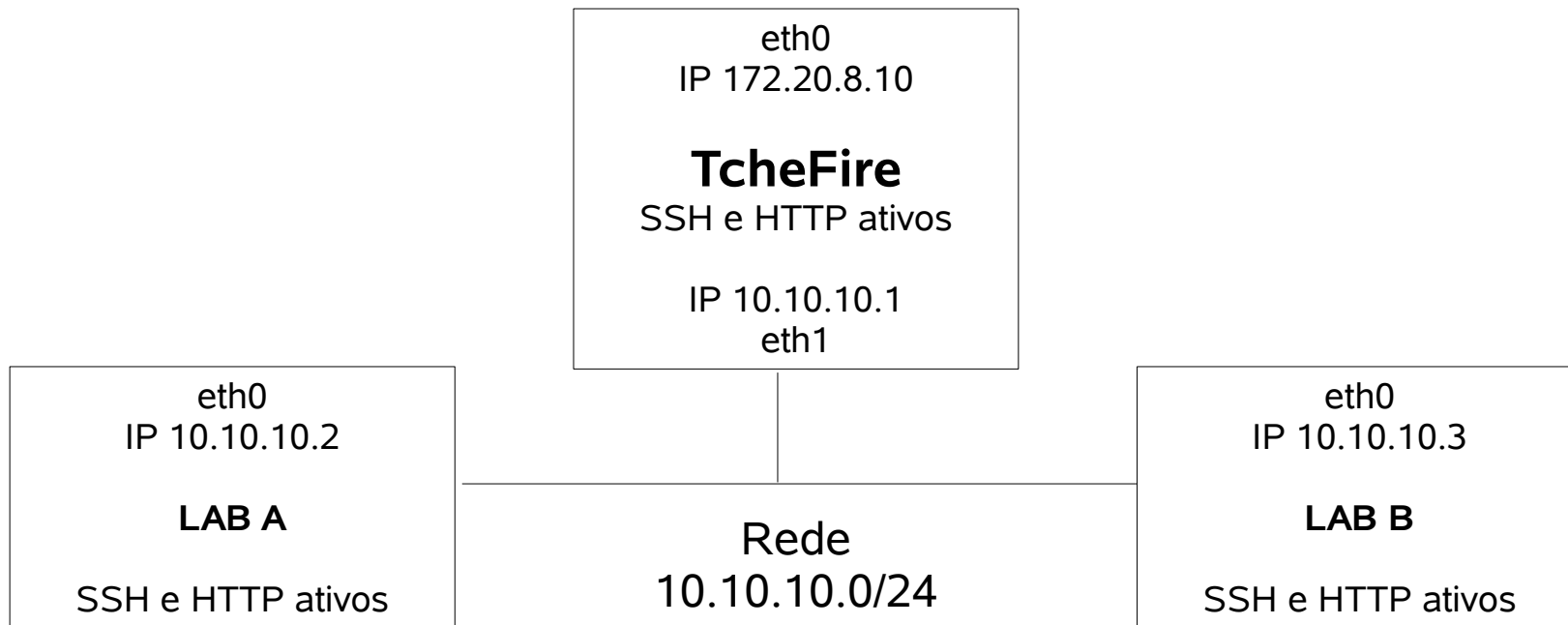
- -p: indica algum protocolo de transporte, como TCP ou UDP (ou mesmo ICMP, embora não seja de transporte)
- Apenas com um -p é possível usar portas
  - --dport: porta destino
  - --sport: porta origem
- Exemplo:
  - iptables -I INPUT -p tcp --dport 22 -j DROP

# Parâmetros para protocolos (2)

- No caso de ICMP, que não tem porta, pode ser usar `--icmp-type` passando nome ou tipo
  - `iptables -I INPUT -p icmp --icmp-type 8 -j DROP`  
(bloqueia ping)
  - `iptables -I INPUT -p icmp --icmp-type 3/3 -j ACCEPT` )  
aceita icmp tipo 3, código 3, porta inacessível)
- **ERRO MUITO COMUM:** não quero que pinguem minha rede:
  - `iptables -A FORWARD -p icmp -j DROP`
    - agora todos os ICMPs estão dropados!

# Demonstração

- Uma rede virtual criada com VMWARE
  - DROP x REJECT
  - Bloqueando SSH no FIREWALL
  - Bloqueando SSH nos LABS
  - Bloqueando HTTP no LABA vindo do LABB



Fim da terceira parte

Perguntas?



Quarta parte

Iptables

o firewall *statefull* do Linux  
a poderosa tabela nat

# Sobre os módulos

- Iptables é extensível através de módulos
- muitos módulos, alguns até exóticos, foram criados:
  - módulo random (???)
  - módulo limit
  - módulo recent
  - módulo state
- Poder statefull está nos módulos

# Poderes do módulo state

- permite regras que casem com o “estado” de uma conexão.
- parâmetro `-m state` invoca o módulo
- Exemplo:
  - `iptables -I INPUT -m state --state ESTABLISHED -j ACCEPT`
  - permite que entrem pacotes para os quais já se tem uma conexão estabelecida.
  - apenas TCP?

# ESTABLISHED todo poderoso

- iptables -I INPUT -m state --state ESTABLISHED -j ACCEPT
- O iptables impõe sentido de “estado” para UDP e até mesmo para ICMP.
- UDP:
  - Se um pacote saiu vindo da porta 200 indo para a 53 (dns) e foi permitido
  - ele “entende” como estabelecida e deixa que volte um pacote da porta 53 para a 200
  - como UDP não tem finalização, esta abertura é por time out.
- ICMP:
  - só tem sentido no caso do ping que pode ter “estado”
  - um icmp echo request sai, deixa entrar a resposta

# RELATED todo poderoso

- `iptables -I INPUT -m state --state RELATED -j ACCEPT`
- permite a entrada de conexões relacionadas com algo que já foi permitido antes.
- exemplo:
  - ICMP totalmente fechado (mas com o RELATED acima primeira regra)
  - Sai um pacote para a porta UDP 53
    - no destino, porta fechada. Volta um ICMP tipo 3, cod 3
    - o RELATED entende que este ICMP está relacionado com o pacote UDP que foi e assim deixar entrar

# Módulo limit

- `iptables -I INPUT -p tcp --syn -m limit --limit 10/min -j LOG`
- registra em log todos os pacotes SYN (início de conexão TCP) que entrarem para qualquer porta. Mas somente 10 por minuto
  - para evitar encher o arquivo de logs!
- **MUITO CUIDADO COM O LIMIT**
  - muitos fazem uso desastroso do limit

# Desastres com o limit

```
iptables -A INPUT -p tcp --syn -m limit --limit 10/min -j ACCEPT
```

```
iptables -A INPUT -p tcp --syn -j DROP
```

- a intenção era nobre: mais do que 10 conexões por minuto eu dropo para evitar sobrecarregar meu servidor
- Porém o que acontece com o que exceder os 10?
- serão dropados!
- agora um atacante só precisa ter condições de gerar 10 syns por minuto para tirar todos os teus serviços do ar
- Pessoalmente eu só percebo um uso racional do limit para controle de logs
- Demais usos devem ser muito bem pensados!

# A tabela nat

- Até agora tudo era tabela filter
  - como dito, filter é realmente o firewall
- tabela nat serve para editar um pacote
  - trocar ip ou trocar porta
- chama-se nat porque permite realizar tradução de endereços (nat: *network address translation*)
- a ação de uma regra na tabela nat permite trocar porta ou IP apenas, de origem ou de destino
- Atua nos ganchos PREROUTING, OUTPUT e POSTROUTING

# Ações da tabela nat

- -j DNAT
  - permite trocar um parâmetro de destino
  - ip ou porta de destino
  - importante: destino é a essência do roteamento
    - logo não tem sentido trocar o destino depois que o pacote já teve a decisão de roteamento!
    - por isto que DNAT só funciona no PREROUTING ou OUTPUT
    - DNAT gera erro de sintaxe no POSTROUTING
      - trocar destino depois que já foi roteado??
- -j SNAT
  - permite trocar um parâmetro de origem
  - ip de origem ou porta de origem
  - só pode ser usado no gancho POSTROUTING

# Exemplos da tabela nat

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to 10.10.10.3
```

- pacotes que entrarem nesta máquina e que forem para a porta 80 serão reencaminhados para a máquina 10.10.10.3
  - isto é, o iptables altera o cabeçalho IPv4 trocando o IP de destino para 10.10.10.3
  - automaticamente ele destroca quando o pacote voltar
  - pode-se trocar a porta também:

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to 10.10.10.3:8080
```

# Exemplos da tabela nat (2)

```
iptables -t nat -A POSTROUTING -p tcp --dport 80 -j SNAT --to 10.10.10.1
```

- pacotes que estiverem saindo desta máquina e que forem para a porta 80 serão enviados como se fossem gerados pelo IP 10.10.10.1
  - isto é, o iptables altera o cabeçalho IPv4 trocando o IP de origem para 10.10.10.3
  - automaticamente ele destroca quando o pacote voltar
  - pode-se trocar a porta de origem também:

```
iptables -t nat -A POSTROUTING -p tcp --dport 80 -j SNAT --to 10.10.10.1:10000
```

- porta de origem (seja qual for) será reescrita para a porta 10000 (seja lá qual seria a utilidade disto!)

# Casos de sucesso tabela nat

- redirecionar trafego da Internet para um servidor interno que tem ip privado
- realizar nat dinâmico, conhecido com mascaramento:

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

- MASQUERADE:
  - um apelido para -j SNAT <meu IP da eth0>

# Casos de sucesso tabela nat (2)

- Proxy transparente

```
iptables -t nat -A PREROUTING -i eth1 -s 10.10.10.0/24 -p tcp  
--dport 80 -j DNAT --to 10.10.10.1:3128
```

ou

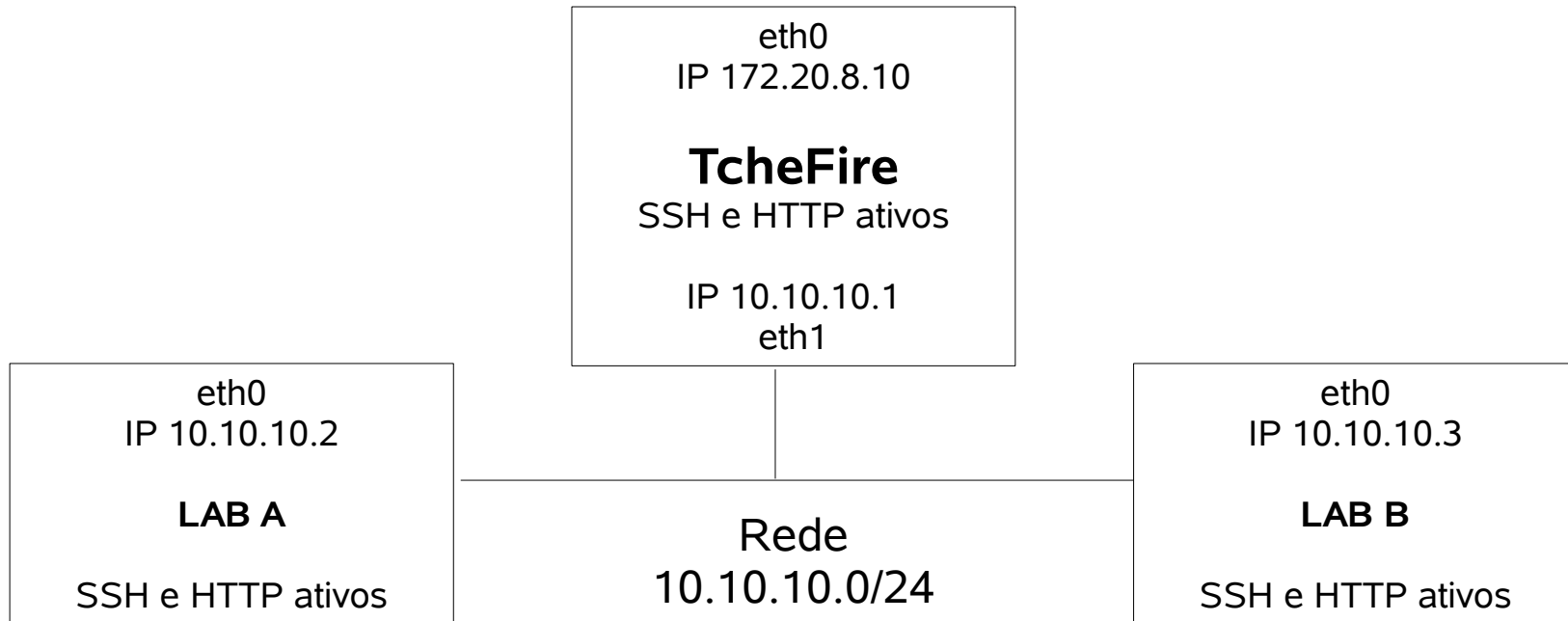
```
iptables -t nat -A PREROUTING -i eth1 -s 10.10.10.0/24 -p tcp  
--dport 80 -j REDIRECT --to 3128
```

- REDIRECT:

- um apelido para -j DNAT --to <MEU IP>:porta

# Demonstração

- Uma rede virtual criada com VMWARE
  - limit
  - state
    - ESTABLISHED
    - RELATED
  - tabela nat



# Conclusões

- Iptables super poderoso
  - até demais, segundo crítica do BSD
    - que dizem que firewall é firewall e não um faz tudo
  - módulos até mesmo bizarros e de desempenho duvidoso
- sempre é statefull mesmo que não se aplique regras
  - no BSD se usa a diretiva keep-state
- Com criatividade e conhecimento, uma máquina Linux ganha em muito de qualquer roteador/firewall comercial

# Referências

- Artigos do Viva o Linux
  - <http://www.vivaolinux.com.br/artigo/255.255.255.0-A-matematica-das-mascaras-de-rede/>
  - <http://www.vivaolinux.com.br/artigo/Estrutura-do-Iptables/>
  - <http://www.vivaolinux.com.br/artigo/Estrutura-do-IPTables-2-a-tabela-nat/>
  - <http://www.vivaolinux.com.br/artigo/Iptables-protege-contra-SYN-FLOOD/>
  - <http://www.vivaolinux.com.br/dica/Firewall-SIMPLES-e-eficiente-para-DESKTOP-em-5-linhas/>
- Man page do iptables
- <http://www.netfilter.org/documentation/HOWTO/netfilter-hacking-HOWTO-3.html>

# Perguntas?

elgios@gmail.com

<http://www.vivaolinux.com.br/perfil/verPerfil.php?login=elgio>

<http://people.tchelinux.org/people.php?id=74>