MATERIAL DIDÁTICO - ADMINISTRAÇÃO DE REDES

Curso Completo para Administradores de Rede

ESTRUTURA DO CURSO

Este material foi desenvolvido especialmente para profissionais que desejam aprofundar seus conhecimentos em administração de serviços de rede em ambientes Linux.

MÓDULOS DO CURSO

MÓDULO 1 - PROXY E SQUID

■ Arquivo: modulo1_proxy_squid.md

Conteúdo:

• Conceitos fundamentais de Proxy

• Instalação e configuração do Squid

• ACLs (Access Control Lists)

• Proxy transparente

• Autenticação de usuários

• Bloqueio de sites

• Monitoramento com SARG

• Cache e otimização

• Troubleshooting

Tempo estimado: 8-10 horas

MÓDULO 2 - POSTFIX (SERVIDOR DE E-MAIL)

■ Arquivo: modulo2_postfix_email.md

Conteúdo:

• Conceitos de MTA, MDA, MUA

• Protocolos: SMTP, POP3, IMAP

• Instalação do Postfix

• Configuração básica e avançada

• Aliases e domínios virtuais

• Autenticação SASL

• Relay SMTP (Gmail)

• Segurança: SPF, DKIM, DMARC

• Anti-spam

• Troubleshooting

Tempo estimado: 10-12 horas

MÓDULO 3 - VPN (VIRTUAL PRIVATE NETWORK)

■ Arquivo: modulo3_vpn.md

Conteúdo:

• Conceitos de VPN

• Tipos: Remote Access, Site-to-Site

• OpenVPN

• Instalação e PKI

• Configuração servidor/cliente

• Certificados

• WireGuard

• Instalação

• Configuração moderna

• Performance

• IPSec/L2TP

• Troubleshooting

• Monitoramento

Tempo estimado: 12-15 horas

MÓDULO 4 - SERVIDORES WEB (APACHE E NGINX)

■ Arquivo: modulo4_servidores_web.md

Conteúdo:

• Conceitos de servidores web

• Apache

• Instalação

• Virtual Hosts

• Módulos

• .htaccess

• Nginx

• Instalação

• Server Blocks

• Proxy reverso

• SSL/TLS com Let's Encrypt

• Certificados manuais

• Headers de segurança

• Performance e otimização

• Monitoramento

Tempo estimado: 10-12 horas

MÓDULO 5 - QoS, ACLs E NAT

■ Arquivo: modulo5_qos_acls_nat.md

Conteúdo:

• QoS (Quality of Service)

• Conceitos

• Traffic Control (TC)

• HTB (Hierarchical Token Bucket)

• Priorização de tráfego

• Wondershaper

• ACLs (Access Control Lists)

• iptables

• Regras de firewall

• Proteção contra ataques

• NAT (Network Address Translation)

• Masquerade (SNAT)

• Port Forwarding (DNAT)

• NAT 1:1

• Scripts integrados

• Monitoramento

Tempo estimado: 12-15 horas

OBJETIVOS DE APRENDIZAGEM

Ao concluir este curso, você será capaz de:

■ Configurar e administrar servidores Proxy (Squid)

■ Implementar servidores de e-mail corporativos (Postfix)

■ Criar e gerenciar VPNs seguras (OpenVPN e WireGuard)

■ Administrar servidores web (Apache e Nginx)

■ Implementar QoS para priorizar tráfego

■ Criar ACLs e políticas de segurança

■ Configurar NAT e Port Forwarding

■ Realizar troubleshooting avançado

■ Monitorar e otimizar serviços de rede

PRÉ-REQUISITOS

Conhecimentos Necessários:

• Linux básico (comandos, estrutura de diretórios)

• Redes básicas (TCP/IP, DNS, DHCP)

• Noções de segurança

• Editor de texto (nano, vim)

Ambiente Recomendado:

• Ubuntu Server 20.04+ ou Debian 11+

• Mínimo 2GB RAM

• 20GB disco

• Acesso root/sudo

• Conexão com Internet

Software:

• Máquina virtual (VirtualBox, VMware) ou servidor dedicado

• Cliente SSH (PuTTY, Terminal)

• Navegador web para testes

METODOLOGIA DE ESTUDO

Recomendações:

Sequencial: Siga os módulos em ordem

Prática: Execute todos os comandos em ambiente de teste

Exercícios: Complete os exercícios ao final de cada módulo

Anotações: Documente suas configurações

Laboratório: Monte um lab com VMs

Estrutura de Cada Módulo:

• Teoria e conceitos

• Exemplos práticos

• Comandos essenciais

• Troubleshooting

• Exercícios

• Resumo

LABORATÓRIO SUGERIDO

Topologia Básica:

```
Internet
```

```
    |
[Servidor Gateway/Firewall]
    |
    ■■■ [Servidor Web - Apache/Nginx]
    |
    ■■■ [Servidor E-mail - Postfix]
    |
    ■■■ [Servidor VPN - OpenVPN/WireGuard]
    |
    ■■■ [Servidor Proxy - Squid]
```

VMs Recomendadas:

• **VM1:** Gateway (Firewall, NAT, QoS)

• **VM2:** Web Server

• **VM3:** Mail Server

• **VM4:** VPN Server

• **VM5:** Proxy Server

• **VM6:** Cliente (testes)

DISTRIBUIÇÃO DE CARGA HORÁRIA

CERTIFICAÇÃO (SUGESTÃO)

Após completar todo o material:

■ Complete todos os exercícios

■ Monte laboratório completo

■ Documente suas configurações

■ Crie casos de uso reais

■ Pratique troubleshooting

RECURSOS ADICIONAIS

Documentação Oficial:

• Squid: http://www.squid-cache.org/

• Postfix: http://www.postfix.org/

• OpenVPN: https://openvpn.net/

• WireGuard: https://www.wireguard.com/

• Apache: https://httpd.apache.org/

• Nginx: https://nginx.org/

Ferramentas Úteis:

• iptables: https://netfilter.org/

• TC (Traffic Control): https://man7.org/linux/man-pages/man8/tc.8.html

• Let's Encrypt: https://letsencrypt.org/

FLUXO DE APRENDIZADO

```
Início
  ↓
Módulo 1 (Proxy/Squid) → Exercícios
  ↓
Módulo 2 (E-mail/Postfix) → Exercícios
  ↓
Módulo 3 (VPN) → Exercícios
  ↓
Módulo 4 (Web Servers) → Exercícios
  ↓
Módulo 5 (QoS/ACL/NAT) → Exercícios
  ↓
Integração de Todos os Módulos
  ↓
Projeto Final (Laboratório Completo)
  ↓
Conclusão
```

AVISOS IMPORTANTES

Ambiente de Teste: Sempre pratique em ambiente controlado

Backup: Faça backup antes de mudanças críticas

Documentação: Documente todas as configurações

Segurança: Nunca use senhas fracas em produção

Atualizações: Mantenha sistemas atualizados

DICAS DE SUCESSO

■ Pratique cada comando antes de avançar

■ Use máquinas virtuais para experimentar

■ Consulte logs quando algo falhar

■ Teste em ambiente isolado primeiro

■ Mantenha anotações organizadas

■ Participe de comunidades online

■ Leia documentação oficial

SUPORTE

Este é um material de estudo autodidata. Para suporte adicional:

• Fóruns oficiais das ferramentas

• Stack Overflow

• Reddit: r/sysadmin, r/linux

• Comunidades Linux Brasil

CHECKLIST DE CONCLUSÃO

Ao final do curso, você deve ser capaz de marcar:

• [ ] Configurar Squid Proxy com autenticação

• [ ] Bloquear sites e categorias de conteúdo

• [ ] Instalar e configurar Postfix

• [ ] Implementar SPF, DKIM e DMARC

• [ ] Criar servidor OpenVPN funcional

• [ ] Configurar WireGuard

• [ ] Hospedar múltiplos sites em Apache

• [ ] Configurar Nginx como proxy reverso

• [ ] Obter certificados SSL Let's Encrypt

• [ ] Implementar QoS com priorização

• [ ] Criar ACLs com iptables

• [ ] Configurar NAT e Port Forwarding

• [ ] Realizar troubleshooting em todos os serviços

PRÓXIMOS PASSOS

Após dominar este conteúdo:

Estude tópicos avançados:

• Kubernetes

• Docker

• Ansible/Puppet

• Monitoramento (Zabbix, Nagios)

Certificações:

• LPIC-2

• RHCE

• CompTIA Linux+

Especializações:

• Segurança (Firewall, IDS/IPS)

• Cloud (AWS, Azure, GCP)

• DevOps

LICENÇA E USO

VERSÃO

Versão: 1.0

Data: 2024

Autor: Material Didático para Administradores de Rede

Última Atualização: Fevereiro 2024

FEEDBACK

Encontrou algum erro ou tem sugestões?

Contribua com melhorias para este material!

BOA SORTE NOS ESTUDOS! ■

Lembre-se: A prática leva à perfeição!

MÓDULO 1: PROXY E SQUID

ÍNDICE

---

1. CONCEITOS FUNDAMENTAIS DE PROXY

O que é um Proxy?

Um servidor proxy atua como intermediário entre clientes e servidores de destino.

Tipos de Proxy:

• **Proxy Forward:** Cliente configura explicitamente o proxy

• **Proxy Transparente:** Intercepta tráfego sem configuração no cliente

• **Proxy Reverso:** Protege servidores backend dos clientes externos

Benefícios

Cache de Conteúdo

Controle de Acesso

Segurança

Auditoria

Economia de Banda

2. SQUID - INSTALAÇÃO

Ubuntu/Debian

```
sudo apt update
sudo apt install squid -y
```

```
sudo systemctl status squid
```

Estrutura de Diretórios

```
/etc/squid/squid.conf       # Configuração principal
/var/log/squid/access.log   # Log de acessos
/var/spool/squid/           # Cache
```

3. CONFIGURAÇÃO BÁSICA

squid.conf - Exemplo Simples

```
http_port 3128
visible_hostname proxy.local

cache_dir ufs /var/spool/squid 1000 16 256
cache_mem 256 MB

acl localnet src 192.168.1.0/24
acl SSL_ports port 443
acl Safe_ports port 80 443

http_access deny !Safe_ports
http_access allow localnet
http_access deny all
```

4. COMANDOS ESSENCIAIS

```
# Gerenciamento
systemctl start/stop/restart squid
squid -k reconfigure           # Recarregar config
squid -k parse                 # Testar config

# Cache
squid -z                       # Criar estrutura
rm -rf /var/spool/squid/*     # Limpar cache

# Logs
tail -f /var/log/squid/access.log
```

5. ACLs (LISTAS DE CONTROLE)

```
# Por IP
acl rede_ti src 192.168.1.0/24

# Por domínio
acl bloqueados dstdomain facebook.com

# Por horário
acl comercial time MTWHF 08:00-18:00

# Aplicar regras
http_access deny bloqueados
http_access allow rede_ti comercial
```

6. AUTENTICAÇÃO

```
# Criar usuários
apt install apache2-utils
htpasswd -c /etc/squid/passwords user1
```

```
# squid.conf
auth_param basic program /usr/lib/squid/basic_ncsa_auth /etc/squid/passwords
acl authenticated proxy_auth REQUIRED
http_access allow authenticated
```

7. PROXY TRANSPARENTE

```
# squid.conf
http_port 3129 intercept
```

```
# iptables
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j REDIRECT --to-port 3129
sysctl -w net.ipv4.ip_forward=1
```

8. MONITORAMENTO - SARG

```
apt install sarg
nano /etc/sarg/sarg.conf
```

```
access_log /var/log/squid/access.log
output_dir /var/www/html/squid-reports


sarg                          # Gerar relatório
```

9. TROUBLESHOOTING

```
# Verificar erros
squid -k parse
tail -50 /var/log/squid/cache.log

# Permissões
chown -R proxy:proxy /var/spool/squid

# Recriar cache
systemctl stop squid
rm -rf /var/spool/squid/*
squid -z
systemctl start squid
```

10. EXERCÍCIOS

Básico: Instale e configure proxy para rede 192.168.0.0/24

Bloqueio: Bloqueie 5 sites usando ACL

Autenticação: Configure 3 usuários

Relatórios: Instale SARG e gere relatório

Próximo: Módulo 2 - Postfix (E-mail)

MÓDULO 2: POSTFIX - SERVIDOR DE E-MAIL

ÍNDICE

---

1. CONCEITOS DE E-MAIL

Componentes de um Sistema de E-mail

MTA (Mail Transfer Agent): Postfix, Sendmail, Exim

• Envia e recebe e-mails entre servidores

MDA (Mail Delivery Agent): Dovecot, Procmail

• Entrega e-mails nas caixas dos usuários

MUA (Mail User Agent): Thunderbird, Outlook

• Cliente de e-mail do usuário

Protocolos

SMTP (Simple Mail Transfer Protocol): Porta 25, 587

• Envio de e-mails

POP3 (Post Office Protocol): Porta 110, 995 (SSL)

• Download de e-mails (remove do servidor)

IMAP (Internet Message Access Protocol): Porta 143, 993 (SSL)

• Sincronização de e-mails (mantém no servidor)

Fluxo de E-mail

```
Cliente → MUA → MTA (Postfix) → DNS (MX Record) →
MTA Destino → MDA → Caixa Postal → MUA Cliente
```

2. INSTALAÇÃO DO POSTFIX

Ubuntu/Debian

```
# Atualizar sistema
sudo apt update

# Instalar Postfix
sudo apt install postfix -y

# Durante instalação, escolher: "Internet Site"
# Nome do sistema: mail.seudominio.com.br

# Verificar status
sudo systemctl status postfix

# Verificar versão
postconf mail_version
```

CentOS/RHEL

```
sudo yum install postfix -y
sudo systemctl start postfix
sudo systemctl enable postfix
```

Estrutura de Diretórios

```
/etc/postfix/
■■■ main.cf              # Configuração principal
■■■ master.cf            # Configuração de serviços
■■■ virtual              # Domínios virtuais
■■■ aliases              # Aliases de e-mail
■■■ sasl_passwd          # Senhas SASL

/var/log/mail.log        # Log principal
/var/spool/postfix/      # Fila de e-mails
/var/mail/               # Caixas de e-mail
```

3. CONFIGURAÇÃO BÁSICA - main.cf

Parâmetros Essenciais

```
# Editar configuração
sudo nano /etc/postfix/main.cf



# IDENTIFICAÇÃO DO SERVIDOR
myhostname = mail.empresa.com.br
mydomain = empresa.com.br
myorigin = $mydomain

# INTERFACES DE REDE
inet_interfaces = all
inet_protocols = ipv4

# DESTINO LOCAL
mydestination = $myhostname, localhost.$mydomain, localhost, $mydomain

# REDES CONFIÁVEIS
mynetworks = 127.0.0.0/8, 192.168.1.0/24

# RELAY (repasse de e-mails)
relayhost =
relay_domains = $mydestination

# TAMANHO MÁXIMO DE MENSAGEM (10MB)
message_size_limit = 10240000

# TAMANHO DA CAIXA POSTAL (500MB)
mailbox_size_limit = 524288000

# ALIASES
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases

# FORMATO DE CAIXA POSTAL
home_mailbox = Maildir/

# BANNER SMTP (segurança)
smtpd_banner = $myhostname ESMTP
```

Aplicar Configurações

```
# Recarregar Postfix
sudo systemctl reload postfix

# OU
sudo postfix reload

# Verificar configuração
```

```
sudo postfix check
```

---

4. COMANDOS ESSENCIAIS

Gerenciamento do Serviço

```
# Iniciar
sudo systemctl start postfix

# Parar
sudo systemctl stop postfix

# Reiniciar
sudo systemctl restart postfix

# Recarregar (sem parar)
sudo systemctl reload postfix

# Status
sudo systemctl status postfix

# Habilitar no boot
sudo systemctl enable postfix
```

Comandos de Administração

```
# Ver configuração atual
postconf

# Ver parâmetro específico
postconf myhostname

# Alterar parâmetro
sudo postconf -e 'myhostname=mail.novodominio.com'

# Verificar sintaxe
sudo postfix check

# Ver fila de e-mails
postqueue -p
# OU
mailq

# Processar fila imediatamente
sudo postqueue -f

# Remover e-mail da fila
```

```
sudo postsuper -d <ID>

# Remover TODOS da fila
sudo postsuper -d ALL

# Remover e-mails adiados
sudo postsuper -d ALL deferred
```

Logs e Monitoramento

```
# Ver log em tempo real
sudo tail -f /var/log/mail.log

# Filtrar por remetente
sudo grep "from=<user@domain.com>" /var/log/mail.log

# Filtrar por destinatário
sudo grep "to=<user@domain.com>" /var/log/mail.log

# Ver estatísticas
sudo pflogsumm /var/log/mail.log

# Instalar pflogsumm
sudo apt install pflogsumm -y
```

Teste de Envio

```
# Enviar e-mail de teste (método 1)
echo "Corpo do email" | mail -s "Assunto" destino@exemplo.com

# Enviar e-mail de teste (método 2)
sendmail usuario@dominio.com << EOF
Subject: Teste
From: admin@empresa.com.br

Este é um email de teste.
EOF

# Telnet manual (diagnóstico)
telnet localhost 25
EHLO localhost
MAIL FROM: teste@empresa.com.br
RCPT TO: destino@exemplo.com
DATA
Subject: Teste via Telnet
Corpo do email.
.
QUIT
```

---

5. ALIASES E REDIRECIONAMENTO

Arquivo /etc/aliases

```
# Editar aliases
sudo nano /etc/aliases
```

```
# Exemplos de aliases
postmaster: root
root: admin@empresa.com.br
webmaster: ti@empresa.com.br
abuse: seguranca@empresa.com.br

# Múltiplos destinos
suporte: joao@empresa.com.br, maria@empresa.com.br

# Redirecionamento externo
vendas: vendas@empresa.com.br, contato@parceiro.com
```

```
# Atualizar banco de dados de aliases
sudo newaliases

# Testar alias
sudo postmap -q vendas hash:/etc/aliases
```

---

6. DOMÍNIOS VIRTUAIS

Configuração de Virtual Domains

```
# Editar main.cf
sudo nano /etc/postfix/main.cf
```

```
# Adicionar virtual domains
virtual_alias_domains = dominio1.com, dominio2.com.br
virtual_alias_maps = hash:/etc/postfix/virtual
```

```
# Criar arquivo virtual
sudo nano /etc/postfix/virtual
```

```
# Redirecionamentos virtuais
# dominio1.com
contato@dominio1.com        joao@empresa.com.br
vendas@dominio1.com         maria@empresa.com.br

# dominio2.com.br
info@dominio2.com.br        suporte@empresa.com.br
admin@dominio2.com.br       root@empresa.com.br

# Catch-all (capturar todos)
@dominio1.com               admin@empresa.com.br



# Compilar arquivo virtual
sudo postmap /etc/postfix/virtual

# Recarregar Postfix
sudo postfix reload

# Testar
sudo postmap -q contato@dominio1.com hash:/etc/postfix/virtual
```

---

7. AUTENTICAÇÃO SMTP (SASL)

Instalação e Configuração

```
# Instalar pacotes necessários
sudo apt install sasl2-bin libsasl2-modules -y

# Editar main.cf
sudo nano /etc/postfix/main.cf



# CONFIGURAÇÃO SASL
smtpd_sasl_auth_enable = yes
smtpd_sasl_type = dovecot
smtpd_sasl_path = private/auth
smtpd_sasl_security_options = noanonymous
smtpd_sasl_local_domain = $myhostname

# RESTRIÇÕES DE RELAY
smtpd_recipient_restrictions =
    permit_mynetworks,
    permit_sasl_authenticated,
    reject_unauth_destination

# TLS/SSL (porta 587)
smtpd_tls_security_level = may
smtpd_tls_cert_file = /etc/ssl/certs/ssl-cert-snakeoil.pem
```

```
smtpd_tls_key_file = /etc/ssl/private/ssl-cert-snakeoil.key


# Editar master.cf para porta 587 (submission)
sudo nano /etc/postfix/master.cf


# Descomentar/adicionar
submission inet n       -       y       -       -       smtpd
  -o syslog_name=postfix/submission
  -o smtpd_tls_security_level=encrypt
  -o smtpd_sasl_auth_enable=yes
  -o smtpd_recipient_restrictions=permit_sasl_authenticated,reject
```

---

8. RELAY SMTP (Envio via Gmail/Outro)

Configurar Postfix para usar Gmail

```
# Editar main.cf
sudo nano /etc/postfix/main.cf


# RELAY via Gmail
relayhost = [smtp.gmail.com]:587

# SASL para autenticação
smtp_sasl_auth_enable = yes
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
smtp_sasl_security_options = noanonymous
smtp_tls_security_level = encrypt
smtp_tls_CAfile = /etc/ssl/certs/ca-certificates.crt


# Criar arquivo de senhas
sudo nano /etc/postfix/sasl_passwd


[smtp.gmail.com]:587 seuemail@gmail.com:suasenha-app


# Proteger e compilar
sudo chmod 600 /etc/postfix/sasl_passwd
sudo postmap /etc/postfix/sasl_passwd

# Recarregar
sudo postfix reload
```

Nota: No Gmail, use "Senha de App" em vez da senha normal.

---

9. SEGURANÇA E ANTI-SPAM

Restrições Básicas

```
# main.cf - Restrições de segurança

# Rejeitar remetentes inválidos
smtpd_sender_restrictions =
    reject_unknown_sender_domain,
    reject_non_fqdn_sender

# Rejeitar destinatários inválidos
smtpd_recipient_restrictions =
    permit_mynetworks,
    permit_sasl_authenticated,
    reject_unauth_destination,
    reject_invalid_hostname,
    reject_non_fqdn_recipient,
    reject_unknown_recipient_domain

# Rejeitar conexões suspeitas
smtpd_helo_restrictions =
    permit_mynetworks,
    reject_invalid_helo_hostname,
    reject_non_fqdn_helo_hostname

# Limites de taxa
smtpd_client_connection_count_limit = 10
smtpd_client_connection_rate_limit = 30
smtpd_error_sleep_time = 20s
```

SPF, DKIM e DMARC

SPF (Sender Policy Framework)

```
# Adicionar registro DNS TXT
v=spf1 mx a ip4:SEU.IP.AQUI ~all
```

DKIM (DomainKeys Identified Mail)

```
# Instalar OpenDKIM
sudo apt install opendkim opendkim-tools -y

# Gerar chaves
sudo mkdir -p /etc/opendkim/keys/empresa.com.br
cd /etc/opendkim/keys/empresa.com.br
```

```
sudo opendkim-genkey -s mail -d empresa.com.br

# Adicionar ao DNS
sudo cat mail.txt
```

DMARC

```
# Registro DNS TXT
_dmarc.empresa.com.br. IN TXT "v=DMARC1; p=quarantine; rua=mailto:dmarc@empresa.com.br"
```

---

10. TROUBLESHOOTING

Problemas Comuns

1. E-mails não enviando

```
# Verificar logs
sudo tail -100 /var/log/mail.log

# Verificar fila
mailq

# Testar conectividade
telnet smtp.destino.com 25

# Forçar processamento
sudo postqueue -f
```

2. Porta 25 bloqueada

```
# Testar porta
telnet smtp.gmail.com 25

# Usar porta alternativa (587)
# Configurar em main.cf
relayhost = [smtp.gmail.com]:587
```

3. E-mails caindo em SPAM

```
# Verificar SPF
dig TXT empresa.com.br

# Verificar reversão DNS
dig -x SEU.IP.SERVIDOR

# Testar score de spam
# Enviar para: check-auth@verifier.port25.com
```

4. Permissões

```
# Corrigir permissões
sudo chown -R postfix:postfix /var/spool/postfix
sudo chmod -R 755 /var/spool/postfix
```

---

11. MONITORAMENTO

Ferramentas

```
# PostfixAdmin (interface web)
sudo apt install postfixadmin -y

# Mailgraph (gráficos)
sudo apt install mailgraph -y

# Estatísticas manuais
sudo pflogsumm -d today /var/log/mail.log

# Top remetentes
sudo grep "from=" /var/log/mail.log | awk '{print $7}' | sort | uniq -c | sort -rn | head
-10

# Top destinatários
sudo grep "to=" /var/log/mail.log | awk '{print $7}' | sort | uniq -c | sort -rn | head -1
0
```

---

12. BACKUP

```
# Backup de configuração
sudo tar -czf /backup/postfix-$(date +%Y%m%d).tar.gz /etc/postfix/

# Backup de e-mails
sudo tar -czf /backup/mails-$(date +%Y%m%d).tar.gz /var/mail/

# Backup da fila
sudo tar -czf /backup/queue-$(date +%Y%m%d).tar.gz /var/spool/postfix/
```

---

13. EXERCÍCIOS PRÁTICOS

Básico: Instale Postfix e envie e-mail de teste

Aliases: Configure 3 aliases diferentes

Virtual: Configure 2 domínios virtuais

Relay: Configure envio via Gmail

Segurança: Implemente SPF e restrições básicas

RESUMO

Conceitos:

• MTA, MDA, MUA

• SMTP, POP3, IMAP

• Relay e autenticação

Comandos:

• postconf - Configuração

• postqueue - Gerenciar fila

• postsuper - Manutenção

• mailq - Ver fila

Arquivos:

• /etc/postfix/main.cf

• /etc/postfix/master.cf

• /etc/aliases

• /var/log/mail.log

Próximo: Módulo 3 - VPN

MÓDULO 3: VPN (VIRTUAL PRIVATE NETWORK)

ÍNDICE

---

1. CONCEITOS DE VPN

O que é VPN?

VPN (Virtual Private Network) cria uma conexão segura e criptografada sobre uma rede insegura (Internet), permitindo acesso remoto seguro.

Tipos de VPN

1. Remote Access VPN (Acesso Remoto)

• Usuários remotos → Rede corporativa

• Exemplo: Home office acessando servidor da empresa

2. Site-to-Site VPN (Site a Site)

• Matriz ↔ Filial

• Conecta redes inteiras

3. Client-to-Site VPN

• Cliente individual → Servidor VPN

Protocolos VPN

Benefícios

Segurança: Criptografia end-to-end

Privacidade: Oculta tráfego e IP

Acesso Remoto: Trabalho de qualquer lugar

Bypass Geo-restrição: Acesso a conteúdo regional

---

2. OPENVPN - INSTALAÇÃO

Servidor Ubuntu/Debian

```
# Atualizar sistema
sudo apt update && sudo apt upgrade -y

# Instalar OpenVPN e Easy-RSA
sudo apt install openvpn easy-rsa -y

# Criar diretório PKI
make-cadir ~/openvpn-ca
cd ~/openvpn-ca
```

Configurar PKI (Infraestrutura de Chaves)

```
# Editar variáveis
nano vars
```

```
# Configurações personalizadas
set_var EASYRSA_REQ_COUNTRY    "BR"
set_var EASYRSA_REQ_PROVINCE   "SP"
set_var EASYRSA_REQ_CITY       "Sao Paulo"
set_var EASYRSA_REQ_ORG        "MinhaEmpresa"
set_var EASYRSA_REQ_EMAIL      "admin@empresa.com.br"
set_var EASYRSA_REQ_OU         "TI"
```

```
# Inicializar PKI
./easyrsa init-pki

# Criar CA (Certificate Authority)
./easyrsa build-ca nopass

# Gerar chave do servidor
./easyrsa gen-req server nopass

# Assinar certificado do servidor
./easyrsa sign-req server server

# Gerar parâmetros Diffie-Hellman
./easyrsa gen-dh
```

```
# Gerar chave TLS
openvpn --genkey secret ta.key

# Copiar arquivos para OpenVPN
sudo cp pki/ca.crt /etc/openvpn/server/
sudo cp pki/issued/server.crt /etc/openvpn/server/
sudo cp pki/private/server.key /etc/openvpn/server/
sudo cp pki/dh.pem /etc/openvpn/server/
sudo cp ta.key /etc/openvpn/server/
```

---

3. CONFIGURAÇÃO DO SERVIDOR OPENVPN

Arquivo de Configuração

```
# Criar configuração
sudo nano /etc/openvpn/server/server.conf



# PORTA E PROTOCOLO
port 1194
proto udp

# DISPOSITIVO DE REDE
dev tun

# CERTIFICADOS E CHAVES
ca ca.crt
cert server.crt
key server.key
dh dh.pem
tls-auth ta.key 0

# REDE VPN
server 10.8.0.0 255.255.255.0

# CONFIGURAÇÕES DE REDE
push "redirect-gateway def1 bypass-dhcp"
push "dhcp-option DNS 8.8.8.8"
push "dhcp-option DNS 8.8.4.4"

# ROTAS (opcional - acesso à rede local)
push "route 192.168.1.0 255.255.255.0"

# SEGURANÇA
cipher AES-256-GCM
auth SHA256
tls-version-min 1.2

# PERSISTÊNCIA
```

```
keepalive 10 120
persist-key
persist-tun

# COMPRESSÃO
compress lz4-v2
push "compress lz4-v2"

# USUÁRIO E GRUPO
user nobody
group nogroup

# LOGS
status /var/log/openvpn/openvpn-status.log
log-append /var/log/openvpn/openvpn.log
verb 3

# MÚLTIPLOS CLIENTES
client-to-client
duplicate-cn
```

IP Forwarding

```
# Habilitar IP forwarding
sudo nano /etc/sysctl.conf



# Descomentar ou adicionar
net.ipv4.ip_forward=1



# Aplicar
sudo sysctl -p
```

Firewall (iptables)

```
# Obter interface de rede
ip route | grep default

# Configurar NAT (substitua eth0 pela sua interface)
sudo iptables -t nat -A POSTROUTING -s 10.8.0.0/24 -o eth0 -j MASQUERADE

# Permitir tráfego VPN
sudo iptables -A INPUT -i tun0 -j ACCEPT
sudo iptables -A FORWARD -i tun0 -j ACCEPT
sudo iptables -A FORWARD -i tun0 -o eth0 -m state --state RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -i eth0 -o tun0 -m state --state RELATED,ESTABLISHED -j ACCEPT

# Salvar regras
sudo apt install iptables-persistent -y
sudo netfilter-persistent save
```

UFW (Firewall simplificado)

```
# Permitir OpenVPN
sudo ufw allow 1194/udp

# Configurar NAT no UFW
sudo nano /etc/ufw/before.rules



# Adicionar no início do arquivo
*nat
:POSTROUTING ACCEPT [0:0]
-A POSTROUTING -s 10.8.0.0/24 -o eth0 -j MASQUERADE
COMMIT



# Habilitar forwarding no UFW
sudo nano /etc/default/ufw



DEFAULT_FORWARD_POLICY="ACCEPT"



# Recarregar UFW
sudo ufw disable
sudo ufw enable
```

---

4. INICIAR SERVIDOR OPENVPN

```
# Criar diretório de logs
sudo mkdir -p /var/log/openvpn

# Iniciar OpenVPN
sudo systemctl start openvpn-server@server

# Habilitar no boot
sudo systemctl enable openvpn-server@server

# Verificar status
sudo systemctl status openvpn-server@server

# Ver logs
sudo journalctl -u openvpn-server@server -f
```

5. CRIAR CLIENTE OPENVPN

Gerar Certificado de Cliente

```
cd ~/openvpn-ca

# Gerar requisição do cliente
./easyrsa gen-req client1 nopass

# Assinar certificado
./easyrsa sign-req client client1

# Copiar arquivos
mkdir -p ~/client-configs/keys
cp pki/ca.crt ~/client-configs/keys/
cp pki/issued/client1.crt ~/client-configs/keys/
cp pki/private/client1.key ~/client-configs/keys/
cp ta.key ~/client-configs/keys/
```

Configuração do Cliente

```
# Criar configuração base
nano ~/client-configs/client.ovpn



client
dev tun
proto udp

# IP DO SERVIDOR (SUBSTITUIR)
remote SEU.IP.SERVIDOR.AQUI 1194

resolv-retry infinite
nobind
persist-key
persist-tun

remote-cert-tls server
cipher AES-256-GCM
auth SHA256
key-direction 1

compress lz4-v2
verb 3

# CERTIFICADOS INLINE
<ca>
# CONTEÚDO DO ca.crt
```

```
</ca>

<cert>
# CONTEÚDO DO client1.crt
</cert>

<key>
# CONTEÚDO DO client1.key
</key>

<tls-auth>
# CONTEÚDO DO ta.key
</tls-auth>
```

Script para Gerar Config Cliente

```
nano ~/client-configs/make_config.sh
```

```
#!/bin/bash

# Argumentos
CLIENT=$1
SERVER_IP=$2

# Verificar argumentos
if [ -z "$CLIENT" ] || [ -z "$SERVER_IP" ]; then
    echo "Uso: $0 <nome-cliente> <ip-servidor>"
    exit 1
fi

# Diretórios
KEY_DIR=~/client-configs/keys
OUTPUT_DIR=~/client-configs/files
BASE_CONFIG=~/client-configs/client.ovpn

# Criar diretório de saída
mkdir -p $OUTPUT_DIR

# Criar configuração
cat $BASE_CONFIG \
    <(echo -e '<ca>') \
    $KEY_DIR/ca.crt \
    <(echo -e '</ca>\n<cert>') \
    $KEY_DIR/${CLIENT}.crt \
    <(echo -e '</cert>\n<key>') \
    $KEY_DIR/${CLIENT}.key \
    <(echo -e '</key>\n<tls-auth>') \
    $KEY_DIR/ta.key \
    <(echo -e '</tls-auth>') \
    | sed "s/remote .*/remote $SERVER_IP 1194/" \
    > $OUTPUT_DIR/${CLIENT}.ovpn

echo "Arquivo criado: $OUTPUT_DIR/${CLIENT}.ovpn"
```

```
# Dar permissão
chmod +x ~/client-configs/make_config.sh

# Executar
./make_config.sh client1 200.100.50.25
```

---

6. WIREGUARD - VPN MODERNA

Vantagens do WireGuard

• **Ultra-rápido:** Código enxuto (4.000 linhas vs 100.000 do OpenVPN)

• **Criptografia moderna:** ChaCha20, Poly1305

• **Simples:** Configuração mínima

• **Integrado:** Kernel Linux 5.6+

Instalação do WireGuard

```
# Ubuntu 20.04+
sudo apt update
sudo apt install wireguard -y

# Verificar módulo do kernel
sudo modprobe wireguard
lsmod | grep wireguard
```

Configuração do Servidor

```
# Gerar chaves
cd /etc/wireguard
umask 077
wg genkey | tee server_private.key | wg pubkey > server_public.key
wg genkey | tee client_private.key | wg pubkey > client_public.key

# Criar configuração
sudo nano /etc/wireguard/wg0.conf



[Interface]
Address = 10.200.0.1/24
ListenPort = 51820
PrivateKey = <CONTEUDO_server_private.key>

# IP Forwarding
PostUp = sysctl -w net.ipv4.ip_forward=1
```

```
PostUp = iptables -A FORWARD -i %i -j ACCEPT
PostUp = iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
PostDown = iptables -D FORWARD -i %i -j ACCEPT
PostDown = iptables -t nat -D POSTROUTING -o eth0 -j MASQUERADE

[Peer]
# Cliente 1
PublicKey = <CONTEUDO_client_public.key>
AllowedIPs = 10.200.0.2/32
```

Configuração do Cliente

```
[Interface]
Address = 10.200.0.2/24
PrivateKey = <CONTEUDO_client_private.key>
DNS = 8.8.8.8

[Peer]
PublicKey = <CONTEUDO_server_public.key>
Endpoint = SEU.IP.SERVIDOR:51820
AllowedIPs = 0.0.0.0/0
PersistentKeepalive = 25
```

Gerenciar WireGuard

```
# Iniciar VPN
sudo wg-quick up wg0

# Parar VPN
sudo wg-quick down wg0

# Status
sudo wg show

# Habilitar no boot
sudo systemctl enable wg-quick@wg0

# Adicionar cliente sem reiniciar
sudo wg set wg0 peer <PUBLIC_KEY> allowed-ips 10.200.0.3/32
```

---

7. COMANDOS DE ADMINISTRAÇÃO

OpenVPN

```
# Status do servidor
sudo systemctl status openvpn-server@server

# Ver clientes conectados
sudo cat /var/log/openvpn/openvpn-status.log

# Logs em tempo real
sudo tail -f /var/log/openvpn/openvpn.log

# Reiniciar servidor
sudo systemctl restart openvpn-server@server

# Revogar certificado de cliente
cd ~/openvpn-ca
./easyrsa revoke client1
./easyrsa gen-crl
sudo cp pki/crl.pem /etc/openvpn/server/

# Adicionar CRL no server.conf
# crl-verify crl.pem
```

WireGuard

```
# Ver configuração ativa
sudo wg show wg0

# Ver peers conectados
sudo wg show wg0 peers

# Ver estatísticas
sudo wg show wg0 transfer

# Recarregar configuração
sudo wg-quick down wg0 && sudo wg-quick up wg0

# Testar conectividade
ping -c 4 10.200.0.1
```

---

8. TROUBLESHOOTING

OpenVPN

```
# Problema: Servidor não inicia
sudo journalctl -xe -u openvpn-server@server
sudo openvpn --config /etc/openvpn/server/server.conf

# Problema: Cliente não conecta
```

```
# Verificar firewall
sudo ufw status
sudo iptables -L -n -v

# Testar porta
nc -zv SEU.IP 1194

# Problema: Sem internet após conectar
# Verificar IP forwarding
cat /proc/sys/net/ipv4/ip_forward  # Deve ser 1

# Verificar NAT
sudo iptables -t nat -L -n -v
```

WireGuard

```
# Problema: Interface não sobe
sudo dmesg | grep wireguard
sudo wg-quick up wg0

# Problema: Peer não conecta
# Verificar chaves públicas
sudo wg show wg0

# Verificar firewall
sudo ufw allow 51820/udp

# Problema: Sem roteamento
sudo ip route show
```

---

9. MONITORAMENTO

OpenVPN

```
# Script de monitoramento
cat > /usr/local/bin/openvpn-monitor.sh << 'EOF'
#!/bin/bash
echo "=== OpenVPN Status ==="
systemctl status openvpn-server@server | grep Active

echo "=== Connected Clients ==="
cat /var/log/openvpn/openvpn-status.log | grep "^CLIENT_LIST" | awk '{print $2, $3}'

echo "=== Traffic ==="
cat /var/log/openvpn/openvpn-status.log | grep "^CLIENT_LIST" | awk '{print $2, $5, $6}'
EOF
```

```
chmod +x /usr/local/bin/openvpn-monitor.sh
```

WireGuard

```
# Status completo
watch -n 2 'sudo wg show all'

# Script personalizado
cat > /usr/local/bin/wg-monitor.sh << 'EOF'
#!/bin/bash
echo "=== WireGuard Status ==="
sudo wg show wg0 | grep -E "interface|peer|transfer"
EOF

chmod +x /usr/local/bin/wg-monitor.sh
```

---

10. SEGURANÇA

Boas Práticas

```
# OpenVPN - Hardening
tls-version-min 1.3
cipher AES-256-GCM
auth SHA512
tls-cipher TLS-ECDHE-RSA-WITH-AES-256-GCM-SHA384

# Limitar tentativas
max-clients 10
client-cert-not-required off
```

Firewall Adicional

```
# Limitar conexões por IP
sudo iptables -A INPUT -p udp --dport 1194 -m state --state NEW -m recent --set
sudo iptables -A INPUT -p udp --dport 1194 -m state --state NEW -m recent --update --secon
ds 60 --hitcount 4 -j DROP
```

---

11. EXERCÍCIOS

OpenVPN Básico: Configure servidor e conecte 1 cliente

Multi-cliente: Crie 3 clientes diferentes

WireGuard: Configure WireGuard e compare performance

Site-to-Site: Configure VPN entre duas redes

Monitoramento: Implemente script de monitoramento

RESUMO

Protocolos:

• OpenVPN: Robusto, multi-plataforma

• WireGuard: Rápido, moderno, simples

Comandos OpenVPN:

• systemctl start openvpn-server@server

• easyrsa build-ca/sign-req

Comandos WireGuard:

• wg-quick up/down wg0

• wg show

Arquivos:

• /etc/openvpn/server/server.conf

• /etc/wireguard/wg0.conf

Próximo: Módulo 4 - Servidores Web (Apache/Nginx)

MÓDULO 4: SERVIDORES WEB (APACHE E NGINX)

ÍNDICE

---

1. CONCEITOS DE SERVIDORES WEB

O que é um Servidor Web?

Servidor web é um software que recebe requisições HTTP/HTTPS e retorna conteúdo (HTML, imagens, arquivos) aos clientes (navegadores).

Principais Servidores Web

Apache vs Nginx

Apache (httpd):

• Tradicional, maduro

• .htaccess dinâmico

• Módulos extensivos

• Melhor para conteúdo dinâmico

Nginx:

• Alto desempenho

• Baixo consumo de memória

• Excelente para arquivos estáticos

• Proxy reverso eficiente

---

2. APACHE - INSTALAÇÃO

Ubuntu/Debian

```
# Instalar Apache
sudo apt update
sudo apt install apache2 -y

# Verificar status
sudo systemctl status apache2

# Verificar versão
apache2 -v

# Testar configuração
sudo apache2ctl configtest
```

CentOS/RHEL

```
sudo yum install httpd -y
sudo systemctl start httpd
sudo systemctl enable httpd
```

Estrutura de Diretórios

```
/etc/apache2/               # Configurações (Debian)
■■■ apache2.conf             # Config principal
■■■ ports.conf               # Portas
■■■ sites-available/         # Sites disponíveis
■■■ sites-enabled/           # Sites ativos
■■■ mods-available/          # Módulos disponíveis
■■■ mods-enabled/            # Módulos ativos

/var/www/html/              # Documentos web
/var/log/apache2/           # Logs
■■■ access.log
■■■ error.log
```

CentOS/RHEL:

```
/etc/httpd/conf/httpd.conf  # Config principal
/var/www/html/              # Documentos
/var/log/httpd/             # Logs
```

3. CONFIGURAÇÃO BÁSICA DO APACHE

Arquivo Principal (apache2.conf)

```
sudo nano /etc/apache2/apache2.conf
```

```
# DIRETÓRIO RAIZ
ServerRoot "/etc/apache2"

# TIMEOUT
Timeout 300

# KEEPALIVE
KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 5

# MPM (Multi-Processing Module) - Event
<IfModule mpm_event_module>
    StartServers             2
    MinSpareThreads         25
    MaxSpareThreads         75
    ThreadLimit             64
    ThreadsPerChild         25
    MaxRequestWorkers      150
    MaxConnectionsPerChild   0
</IfModule>

# DIRETÓRIO PADRÃO
<Directory /var/www/>
    Options Indexes FollowSymLinks
    AllowOverride All
    Require all granted
</Directory>

# SEGURANÇA
ServerTokens Prod
ServerSignature Off
TraceEnable Off

# LOGS
ErrorLog ${APACHE_LOG_DIR}/error.log
LogLevel warn
```

Configuração de Portas

```
sudo nano /etc/apache2/ports.conf
```

```
Listen 80

<IfModule ssl_module>
    Listen 443
</IfModule>

<IfModule mod_gnutls.c>
    Listen 443
</IfModule>
```

---

4. VIRTUAL HOSTS (APACHE)

Criar Virtual Host

```
# Criar diretório do site
sudo mkdir -p /var/www/site1.com.br/public_html

# Criar página de teste
sudo nano /var/www/site1.com.br/public_html/index.html
```

```
<!DOCTYPE html>
<html>
<head>
    <title>Site 1</title>
</head>
<body>
    <h1>Bem-vindo ao Site 1</h1>
    <p>Virtual Host funcionando!</p>
</body>
</html>
```

```
# Ajustar permissões
sudo chown -R www-data:www-data /var/www/site1.com.br
sudo chmod -R 755 /var/www/site1.com.br
```

Configuração do Virtual Host

```
sudo nano /etc/apache2/sites-available/site1.com.br.conf
```

```
<VirtualHost *:80>
```

```
    ServerName site1.com.br
    ServerAlias www.site1.com.br
    ServerAdmin admin@site1.com.br

    DocumentRoot /var/www/site1.com.br/public_html

    <Directory /var/www/site1.com.br/public_html>
        Options -Indexes +FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/site1-error.log
    CustomLog ${APACHE_LOG_DIR}/site1-access.log combined
</VirtualHost>
```

Ativar Site

```
# Habilitar site
sudo a2ensite site1.com.br.conf

# Desabilitar site padrão (opcional)
sudo a2dissite 000-default.conf

# Testar configuração
sudo apache2ctl configtest

# Recarregar Apache
sudo systemctl reload apache2

# Testar
curl -H "Host: site1.com.br" http://localhost
```

---

5. MÓDULOS DO APACHE

Gerenciar Módulos

```
# Listar módulos habilitados
apache2ctl -M

# Habilitar módulo
sudo a2enmod rewrite
sudo a2enmod ssl
sudo a2enmod headers

# Desabilitar módulo
sudo a2dismod status
```

```
# Módulos importantes
sudo a2enmod rewrite      # URL rewrite
sudo a2enmod ssl          # HTTPS
sudo a2enmod headers      # Headers HTTP
sudo a2enmod proxy        # Proxy
sudo a2enmod proxy_http   # Proxy HTTP
sudo a2enmod deflate      # Compressão

# Recarregar após mudanças
sudo systemctl reload apache2
```

.htaccess - Reescrita de URL

```
sudo nano /var/www/site1.com.br/public_html/.htaccess
```

```
# Habilitar Rewrite
RewriteEngine On

# Redirecionar www para não-www
RewriteCond %{HTTP_HOST} ^www\.(.*)$ [NC]
RewriteRule ^(.*)$ http://%1/$1 [R=301,L]

# Forçar HTTPS
RewriteCond %{HTTPS} off
RewriteRule ^(.*)$ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]

# URL amigável - remover .html
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME}.html -f
RewriteRule ^(.*)$ $1.html [L]

# Bloquear acesso a arquivos
<FilesMatch "\.(htaccess|htpasswd|ini|log|sh|sql)$">
    Require all denied
</FilesMatch>
```

---

6. NGINX - INSTALAÇÃO

Ubuntu/Debian

```
# Instalar Nginx
sudo apt update
sudo apt install nginx -y

# Verificar status
```

```
sudo systemctl status nginx

# Verificar versão
nginx -v

# Testar configuração
sudo nginx -t
```

Estrutura de Diretórios

```
/etc/nginx/
■■■ nginx.conf              # Config principal
■■■ sites-available/        # Sites disponíveis
■■■ sites-enabled/          # Sites ativos
■■■ conf.d/                 # Configurações adicionais
■■■ snippets/               # Trechos reutilizáveis

/var/www/html/              # Documentos web
/var/log/nginx/             # Logs
■■■ access.log
■■■ error.log
```

---

7. CONFIGURAÇÃO BÁSICA DO NGINX

nginx.conf

```
sudo nano /etc/nginx/nginx.conf
```

```
user www-data;
worker_processes auto;
pid /run/nginx.pid;

events {
    worker_connections 768;
    use epoll;
    multi_accept on;
}

http {
    # BÁSICO
    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;
    types_hash_max_size 2048;
```

```
    # SEGURANÇA
    server_tokens off;
    client_max_body_size 20M;

    # MIME TYPES
    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    # LOGS
    access_log /var/log/nginx/access.log;
    error_log /var/log/nginx/error.log;

    # COMPRESSÃO GZIP
    gzip on;
    gzip_vary on;
    gzip_proxied any;
    gzip_comp_level 6;
    gzip_types text/plain text/css text/xml text/javascript
               application/json application/javascript application/xml+rss;

    # VIRTUAL HOSTS
    include /etc/nginx/conf.d/*.conf;
    include /etc/nginx/sites-enabled/*;
}
```

---

8. SERVER BLOCKS (NGINX)

Criar Server Block

```
# Criar diretório
sudo mkdir -p /var/www/site1.com.br/html

# Criar página
sudo nano /var/www/site1.com.br/html/index.html



<!DOCTYPE html>
<html>
<head>
    <title>Site 1 - Nginx</title>
</head>
<body>
    <h1>Nginx Server Block Funcionando!</h1>
</body>
</html>



# Permissões
sudo chown -R www-data:www-data /var/www/site1.com.br
```

```
sudo chmod -R 755 /var/www/site1.com.br
```

Configuração do Server Block

```
sudo nano /etc/nginx/sites-available/site1.com.br
```

```
server {
    listen 80;
    listen [::]:80;

    server_name site1.com.br www.site1.com.br;

    root /var/www/site1.com.br/html;
    index index.html index.htm index.php;

    # LOGS
    access_log /var/log/nginx/site1-access.log;
    error_log /var/log/nginx/site1-error.log;

    # LOCALIZAÇÃO
    location / {
        try_files $uri $uri/ =404;
    }

    # PHP (se necessário)
    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php8.1-fpm.sock;
    }

    # NEGAR ACESSO A .htaccess
    location ~ /\.ht {
        deny all;
    }

    # CACHE ESTÁTICO
    location ~* \.(jpg|jpeg|png|gif|ico|css|js)$ {
        expires 365d;
        add_header Cache-Control "public, immutable";
    }
}
```

Ativar Server Block

```
# Criar link simbólico
sudo ln -s /etc/nginx/sites-available/site1.com.br /etc/nginx/sites-enabled/

# Testar configuração
sudo nginx -t

# Recarregar Nginx
```

```
sudo systemctl reload nginx

# Testar
curl http://localhost
```

---

9. SSL/TLS - HTTPS

Certificado Let's Encrypt (Gratuito)

```
# Instalar Certbot
sudo apt install certbot python3-certbot-apache -y  # Apache
sudo apt install certbot python3-certbot-nginx -y   # Nginx

# Obter certificado - Apache
sudo certbot --apache -d site1.com.br -d www.site1.com.br

# Obter certificado - Nginx
sudo certbot --nginx -d site1.com.br -d www.site1.com.br

# Renovação automática (já configurada)
sudo certbot renew --dry-run

# Listar certificados
sudo certbot certificates
```

SSL Manual (Certificado Próprio)

```
# Gerar certificado autoassinado
sudo mkdir /etc/ssl/private
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
    -keyout /etc/ssl/private/site1.key \
    -out /etc/ssl/certs/site1.crt
```

Virtual Host Apache com SSL

```
<VirtualHost *:443>
    ServerName site1.com.br
    ServerAlias www.site1.com.br
    DocumentRoot /var/www/site1.com.br/public_html

    SSLEngine on
    SSLCertificateFile /etc/ssl/certs/site1.crt
    SSLCertificateKeyFile /etc/ssl/private/site1.key
```

```
    # SEGURANÇA SSL
    SSLProtocol all -SSLv3 -TLSv1 -TLSv1.1
    SSLCipherSuite ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256
    SSLHonorCipherOrder off

    # HSTS
    Header always set Strict-Transport-Security "max-age=31536000"

    ErrorLog ${APACHE_LOG_DIR}/site1-ssl-error.log
    CustomLog ${APACHE_LOG_DIR}/site1-ssl-access.log combined
</VirtualHost>
```

Server Block Nginx com SSL

```
server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;

    server_name site1.com.br www.site1.com.br;

    # CERTIFICADOS SSL
    ssl_certificate /etc/ssl/certs/site1.crt;
    ssl_certificate_key /etc/ssl/private/site1.key;

    # CONFIGURAÇÃO SSL
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384;
    ssl_prefer_server_ciphers off;

    # SSL SESSION
    ssl_session_cache shared:SSL:10m;
    ssl_session_timeout 10m;

    # HSTS
    add_header Strict-Transport-Security "max-age=31536000" always;

    root /var/www/site1.com.br/html;
    index index.html index.php;

    location / {
        try_files $uri $uri/ =404;
    }
}

# Redirecionar HTTP para HTTPS
server {
    listen 80;
    server_name site1.com.br www.site1.com.br;
    return 301 https://$server_name$request_uri;
}
```

---

10. PROXY REVERSO (NGINX)

Nginx como Proxy para Apache

```
server {
    listen 80;
    server_name site1.com.br;

    location / {
        proxy_pass http://127.0.0.1:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    # Cache estático no Nginx
    location ~* \.(jpg|jpeg|png|gif|ico|css|js)$ {
        root /var/www/site1.com.br/html;
        expires 365d;
    }
}
```

```
# Alterar Apache para porta 8080
sudo nano /etc/apache2/ports.conf
# Mudar: Listen 8080

sudo nano /etc/apache2/sites-available/site1.com.br.conf
# Mudar: <VirtualHost *:8080>

sudo systemctl restart apache2
sudo systemctl reload nginx
```

---

11. COMANDOS DE ADMINISTRAÇÃO

Apache

```
# Gerenciamento
sudo systemctl start apache2
sudo systemctl stop apache2
sudo systemctl restart apache2
sudo systemctl reload apache2
sudo systemctl status apache2

# Configuração
```

```
sudo apache2ctl configtest     # Testar config
sudo apache2ctl -M             # Listar módulos
sudo apache2ctl -S             # Listar virtual hosts

# Sites
sudo a2ensite site.conf        # Habilitar site
sudo a2dissite site.conf       # Desabilitar site

# Módulos
sudo a2enmod rewrite           # Habilitar módulo
sudo a2dismod status           # Desabilitar módulo

# Logs
sudo tail -f /var/log/apache2/access.log
sudo tail -f /var/log/apache2/error.log
```

Nginx

```
# Gerenciamento
sudo systemctl start nginx
sudo systemctl stop nginx
sudo systemctl restart nginx
sudo systemctl reload nginx
sudo systemctl status nginx

# Configuração
sudo nginx -t                  # Testar config
sudo nginx -T                  # Mostrar config completa
sudo nginx -s reload           # Reload graceful

# Logs
sudo tail -f /var/log/nginx/access.log
sudo tail -f /var/log/nginx/error.log

# Processos
ps aux | grep nginx
```

---

12. SEGURANÇA

Headers de Segurança

Apache:

```
# Habilitar mod_headers
sudo a2enmod headers

# Adicionar no VirtualHost
Header always set X-Frame-Options "SAMEORIGIN"
```

```
Header always set X-Content-Type-Options "nosniff"
Header always set X-XSS-Protection "1; mode=block"
Header always set Referrer-Policy "no-referrer-when-downgrade"
Header always set Content-Security-Policy "default-src 'self'"
```

Nginx:

```
add_header X-Frame-Options "SAMEORIGIN" always;
add_header X-Content-Type-Options "nosniff" always;
add_header X-XSS-Protection "1; mode=block" always;
add_header Referrer-Policy "no-referrer-when-downgrade" always;
add_header Content-Security-Policy "default-src 'self'" always;
```

Proteção contra DDoS

Nginx - Rate Limiting:

```
http {
    limit_req_zone $binary_remote_addr zone=mylimit:10m rate=10r/s;

    server {
        location / {
            limit_req zone=mylimit burst=20 nodelay;
        }
    }
}
```

---

13. MONITORAMENTO

Apache Status

```
# Habilitar mod_status
sudo a2enmod status

# Configurar
sudo nano /etc/apache2/mods-enabled/status.conf



<Location /server-status>
    SetHandler server-status
    Require local
    Require ip 192.168.1.0/24
</Location>



# Acessar
```

```
curl http://localhost/server-status
```

Nginx Status

```
server {
    listen 80;
    server_name localhost;

    location /nginx_status {
        stub_status on;
        access_log off;
        allow 127.0.0.1;
        allow 192.168.1.0/24;
        deny all;
    }
}


# Acessar
curl http://localhost/nginx_status
```

---

14. EXERCÍCIOS

Apache: Instale e configure 2 virtual hosts

Nginx: Configure 2 server blocks

SSL: Obtenha certificado Let's Encrypt

Proxy: Configure Nginx como proxy para Apache

Segurança: Implemente headers de segurança

RESUMO

Apache:

• Tradicional, robusto

• .htaccess dinâmico

• Virtual Hosts

Nginx:

• Alto desempenho

• Proxy reverso

• Server Blocks

Comandos:

• a2ensite/a2dissite (Apache)

• nginx -t (Nginx)

• certbot (SSL)

Arquivos:

• /etc/apache2/sites-available/

• /etc/nginx/sites-available/

Próximo: Módulo 5 - QoS e ACLs

MÓDULO 5: QoS, ACLs e NAT

ÍNDICE

---

1. CONCEITOS DE QoS

O que é QoS (Quality of Service)?

QoS é um conjunto de técnicas para gerenciar e priorizar tráfego de rede, garantindo performance adequada para aplicações críticas.

Objetivos do QoS

Bandwidth Management: Controlar uso de banda

Latência: Reduzir atrasos

Jitter: Estabilizar variação de latência

Packet Loss: Minimizar perda de pacotes

Priorização: Aplicações críticas primeiro

Técnicas de QoS

Traffic Shaping: Controla taxa de envio

Traffic Policing: Descarta pacotes excedentes

Priority Queuing: Fila com prioridades

Class-Based Queuing: Classificação por tipo

Bandwidth Reservation: Reserva garantida

Classes de Tráfego

```
Classe 1 - Tempo Real (Voz/Vídeo)
```

```
■■■ VoIP: <150ms latência, <30ms jitter
■■■ Videoconferência: <200ms latência

Classe 2 - Interativo (Web, Email)
■■■ HTTP/HTTPS: Prioridade média
■■■ Email: Baixa latência

Classe 3 - Bulk (Downloads, Backups)
■■■ FTP, Torrents: Melhor esforço

Classe 4 - Background
■■■ Atualizações, sincronização
```

---

2. QoS NO LINUX - TC (Traffic Control)

Estrutura do TC

```
qdisc (Queueing Discipline)
■■■ class (Classes de tráfego)
■■■ filter (Classificadores)
```

Instalação

```
# Ubuntu/Debian
sudo apt install iproute2 -y

# Verificar
tc -Version
```

---

3. IMPLEMENTAÇÃO BÁSICA - HTB (Hierarchical Token Bucket)

Cenário: Limitar Banda por Serviço

Requisitos:

• Interface: eth0

• Banda total: 10 Mbps

• HTTP: 5 Mbps garantidos

• SSH: 2 Mbps garantidos

• Resto: 3 Mbps

Script de Configuração

```
#!/bin/bash

# INTERFACE
IF=eth0

# BANDA TOTAL
TOTAL=10mbit

# REMOVER CONFIGURAÇÃO ANTERIOR
tc qdisc del dev $IF root 2>/dev/null

# CRIAR QDISC RAIZ HTB
tc qdisc add dev $IF root handle 1: htb default 30

# CLASSE RAIZ (TOTAL)
tc class add dev $IF parent 1: classid 1:1 htb rate $TOTAL

# CLASSE HTTP (5 Mbps)
tc class add dev $IF parent 1:1 classid 1:10 htb rate 5mbit ceil 8mbit prio 1

# CLASSE SSH (2 Mbps)
tc class add dev $IF parent 1:1 classid 1:20 htb rate 2mbit ceil 4mbit prio 2

# CLASSE PADRÃO (3 Mbps)
tc class add dev $IF parent 1:1 classid 1:30 htb rate 3mbit ceil 6mbit prio 3

# FILTROS
# HTTP (porta 80 e 443)
tc filter add dev $IF protocol ip parent 1:0 prio 1 u32 \
   match ip dport 80 0xffff flowid 1:10

tc filter add dev $IF protocol ip parent 1:0 prio 1 u32 \
   match ip dport 443 0xffff flowid 1:10

# SSH (porta 22)
tc filter add dev $IF protocol ip parent 1:0 prio 2 u32 \
   match ip dport 22 0xffff flowid 1:20

echo "QoS configurado em $IF"



# Salvar script
sudo nano /usr/local/bin/qos-setup.sh
chmod +x /usr/local/bin/qos-setup.sh

# Executar
sudo /usr/local/bin/qos-setup.sh
```

Verificar QoS

```
# Ver qdisc
tc qdisc show dev eth0

# Ver classes
tc class show dev eth0

# Ver filtros
tc filter show dev eth0

# Estatísticas detalhadas
tc -s class show dev eth0
```

---

4. QoS AVANÇADO - PRIORIZAÇÃO VoIP

Cenário: Priorizar Tráfego VoIP

```
#!/bin/bash

IF=eth0
TOTAL=10mbit

# Limpar
tc qdisc del dev $IF root 2>/dev/null

# QDISC HTB
tc qdisc add dev $IF root handle 1: htb default 40

# CLASSE RAIZ
tc class add dev $IF parent 1: classid 1:1 htb rate $TOTAL

# CLASSE 1: VoIP (Prioridade Alta)
tc class add dev $IF parent 1:1 classid 1:10 htb \
   rate 1mbit ceil 3mbit prio 0

# CLASSE 2: Interativo (Web, Email)
tc class add dev $IF parent 1:1 classid 1:20 htb \
   rate 4mbit ceil 7mbit prio 1

# CLASSE 3: Bulk (Downloads)
tc class add dev $IF parent 1:1 classid 1:30 htb \
   rate 3mbit ceil 8mbit prio 2

# CLASSE 4: Padrão
tc class add dev $IF parent 1:1 classid 1:40 htb \
   rate 2mbit ceil 5mbit prio 3

# QDISC SFQ (Stochastic Fairness Queueing) para cada classe
```

```
tc qdisc add dev $IF parent 1:10 handle 10: sfq perturb 10
tc qdisc add dev $IF parent 1:20 handle 20: sfq perturb 10
tc qdisc add dev $IF parent 1:30 handle 30: sfq perturb 10
tc qdisc add dev $IF parent 1:40 handle 40: sfq perturb 10

# FILTROS

# VoIP - SIP (porta 5060) e RTP (10000-20000)
tc filter add dev $IF protocol ip parent 1:0 prio 0 u32 \
   match ip dport 5060 0xffff flowid 1:10

tc filter add dev $IF protocol ip parent 1:0 prio 0 u32 \
   match ip dport 10000 0x0000 flowid 1:10

# Marcar por TOS/DSCP (VoIP geralmente usa EF - DSCP 46)
tc filter add dev $IF protocol ip parent 1:0 prio 0 u32 \
   match ip tos 0xb8 0xff flowid 1:10

# HTTP/HTTPS
tc filter add dev $IF protocol ip parent 1:0 prio 1 u32 \
   match ip dport 80 0xffff flowid 1:20

tc filter add dev $IF protocol ip parent 1:0 prio 1 u32 \
   match ip dport 443 0xffff flowid 1:20

# FTP
tc filter add dev $IF protocol ip parent 1:0 prio 2 u32 \
   match ip dport 21 0xffff flowid 1:30

echo "QoS VoIP configurado!"
```

---

5. WONDERSHAPER - FERRAMENTA SIMPLIFICADA

Instalação

```
# Instalar
sudo apt install wondershaper -y

# Limitar interface a 5 Mbps download / 1 Mbps upload
sudo wondershaper eth0 5000 1000

# Remover limitação
sudo wondershaper clear eth0

# Persistir no boot
sudo nano /etc/systemd/system/wondershaper.service



[Unit]
```

```
Description=Wondershaper QoS
After=network.target

[Service]
Type=oneshot
ExecStart=/sbin/wondershaper eth0 5000 1000
ExecStop=/sbin/wondershaper clear eth0
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target



sudo systemctl enable wondershaper
sudo systemctl start wondershaper
```

---

6. ACLs (ACCESS CONTROL LISTS)

O que são ACLs?

ACLs são regras que controlam acesso a recursos baseado em:

• Endereços IP

• Portas

• Protocolos

• Usuários

ACLs com iptables

```
# REGRA BÁSICA
# Bloquear IP específico
sudo iptables -A INPUT -s 192.168.1.100 -j DROP

# Permitir apenas IPs da rede local
sudo iptables -A INPUT -s 192.168.1.0/24 -j ACCEPT
sudo iptables -A INPUT -j DROP

# Bloquear porta
sudo iptables -A INPUT -p tcp --dport 23 -j DROP  # Telnet

# Permitir SSH apenas de IP específico
sudo iptables -A INPUT -p tcp --dport 22 -s 192.168.1.10 -j ACCEPT
sudo iptables -A INPUT -p tcp --dport 22 -j DROP
```

ACL Completa - Firewall Corporativo

```
#!/bin/bash

# LIMPAR REGRAS
iptables -F
iptables -X
iptables -t nat -F

# POLÍTICA PADRÃO (NEGAR TUDO)
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT

# PERMITIR LOOPBACK
iptables -A INPUT -i lo -j ACCEPT

# PERMITIR CONEXÕES ESTABELECIDAS
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# SSH - Apenas da rede administrativa
iptables -A INPUT -p tcp --dport 22 -s 192.168.100.0/24 -j ACCEPT

# HTTP/HTTPS - Público
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -p tcp --dport 443 -j ACCEPT

# DNS
iptables -A INPUT -p udp --dport 53 -j ACCEPT
iptables -A INPUT -p tcp --dport 53 -j ACCEPT

# Email (SMTP, IMAP, POP3)
iptables -A INPUT -p tcp --dport 25 -j ACCEPT   # SMTP
iptables -A INPUT -p tcp --dport 587 -j ACCEPT  # Submission
iptables -A INPUT -p tcp --dport 143 -j ACCEPT  # IMAP
iptables -A INPUT -p tcp --dport 993 -j ACCEPT  # IMAPS
iptables -A INPUT -p tcp --dport 110 -j ACCEPT  # POP3
iptables -A INPUT -p tcp --dport 995 -j ACCEPT  # POP3S

# ICMP (Ping) - Limitado
iptables -A INPUT -p icmp --icmp-type echo-request -m limit --limit 1/s -j ACCEPT

# PROTEÇÃO CONTRA ATAQUES

# SYN Flood
iptables -A INPUT -p tcp --syn -m limit --limit 1/s --limit-burst 3 -j ACCEPT

# Port Scan
iptables -A INPUT -p tcp --tcp-flags ALL NONE -j DROP
iptables -A INPUT -p tcp --tcp-flags ALL ALL -j DROP

# Logar bloqueios
iptables -A INPUT -j LOG --log-prefix "FIREWALL-DROP: " --log-level 4
iptables -A INPUT -j DROP

echo "ACL configurada!"
```

Salvar Regras

```
# Instalar iptables-persistent
sudo apt install iptables-persistent -y

# Salvar regras atuais
sudo netfilter-persistent save

# OU manualmente
sudo iptables-save > /etc/iptables/rules.v4
sudo ip6tables-save > /etc/iptables/rules.v6
```

---

7. NAT (NETWORK ADDRESS TRANSLATION)

Conceitos de NAT

NAT (Network Address Translation): Traduz endereços IP privados para públicos.

Tipos:

SNAT (Source NAT): Altera IP origem (saída para Internet)

DNAT (Destination NAT): Altera IP destino (port forwarding)

Masquerade: SNAT dinâmico

PAT (Port Address Translation): Tradução com portas

Topologia Típica

```
Internet
    |
[Gateway/Router]
    |
192.168.1.0/24 (Rede Interna)
```

---

8. CONFIGURAR NAT - MASQUERADE

Servidor como Gateway

```
#!/bin/bash

# HABILITAR IP FORWARDING
```

```
echo "net.ipv4.ip_forward=1" >> /etc/sysctl.conf
sysctl -p

# INTERFACES
WAN=eth0  # Interface externa (Internet)
LAN=eth1  # Interface interna (Rede local)

# LIMPAR NAT
iptables -t nat -F

# MASQUERADE (SNAT)
iptables -t nat -A POSTROUTING -o $WAN -j MASQUERADE

# FORWARD
iptables -A FORWARD -i $LAN -o $WAN -j ACCEPT
iptables -A FORWARD -i $WAN -o $LAN -m state --state RELATED,ESTABLISHED -j ACCEPT

# Salvar
netfilter-persistent save

echo "NAT configurado!"
echo "Rede interna pode acessar Internet via $WAN"
```

Configurar Clientes

```
# Nos clientes da rede interna
# Definir gateway
sudo ip route add default via 192.168.1.1  # IP do servidor NAT

# DNS
echo "nameserver 8.8.8.8" | sudo tee /etc/resolv.conf
```

---

9. PORT FORWARDING (DNAT)

Redirecionar Porta Externa para Servidor Interno

Cenário:

• Servidor web interno: 192.168.1.10

• Redirecionar porta 80 externa para 192.168.1.10:80

```
# DNAT - Redirecionar porta 80
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 \
   -j DNAT --to-destination 192.168.1.10:80

# FORWARD
iptables -A FORWARD -p tcp -d 192.168.1.10 --dport 80 -j ACCEPT
```

```
# Salvar
netfilter-persistent save
```

Port Forwarding Múltiplo

```
# Web (80, 443)
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 \
   -j DNAT --to-destination 192.168.1.10:80
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 443 \
   -j DNAT --to-destination 192.168.1.10:443

# SSH (2222 → 22)
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 2222 \
   -j DNAT --to-destination 192.168.1.20:22

# Email (25, 587, 143, 993)
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 25 \
   -j DNAT --to-destination 192.168.1.30:25

# FORWARDs correspondentes
iptables -A FORWARD -p tcp -d 192.168.1.10 --dport 80 -j ACCEPT
iptables -A FORWARD -p tcp -d 192.168.1.10 --dport 443 -j ACCEPT
iptables -A FORWARD -p tcp -d 192.168.1.20 --dport 22 -j ACCEPT
iptables -A FORWARD -p tcp -d 192.168.1.30 --dport 25 -j ACCEPT
```

---

10. NAT 1:1 (FULL NAT)

Mapeamento 1:1 de IPs

```
# IP Público: 200.100.50.10
# IP Privado: 192.168.1.50

# DNAT (entrada)
iptables -t nat -A PREROUTING -d 200.100.50.10 \
   -j DNAT --to-destination 192.168.1.50

# SNAT (saída)
iptables -t nat -A POSTROUTING -s 192.168.1.50 \
   -j SNAT --to-source 200.100.50.10
```

---

11. COMANDOS ÚTEIS

QoS (TC)

```
# Ver configuração
tc qdisc show dev eth0
tc class show dev eth0
tc filter show dev eth0

# Estatísticas
tc -s qdisc show dev eth0
tc -s class show dev eth0

# Remover QoS
tc qdisc del dev eth0 root
```

iptables

```
# Listar regras
iptables -L -n -v
iptables -t nat -L -n -v

# Listar com números
iptables -L -n --line-numbers

# Deletar regra específica
iptables -D INPUT 5  # Remove regra 5

# Limpar todas
iptables -F
iptables -t nat -F

# Contar pacotes/bytes
iptables -L -n -v | grep "22"  # Tráfego SSH

# Logar tráfego
iptables -A INPUT -j LOG --log-prefix "FIREWALL: "
tail -f /var/log/syslog | grep FIREWALL
```

---

12. MONITORAMENTO

Monitor de Banda - iftop

```
# Instalar
sudo apt install iftop -y
```

```
# Executar
sudo iftop -i eth0

# Por portas
sudo iftop -i eth0 -P
```

nethogs - Por Processo

```
sudo apt install nethogs -y
sudo nethogs eth0
```

vnstat - Estatísticas

```
sudo apt install vnstat -y
sudo vnstat -l -i eth0     # Tempo real
vnstat -d                  # Diário
vnstat -m                  # Mensal
```

---

13. TROUBLESHOOTING

QoS não funciona

```
# Verificar módulos do kernel
lsmod | grep sch_htb
lsmod | grep sch_sfq

# Carregar módulos
sudo modprobe sch_htb
sudo modprobe sch_sfq

# Testar largura de banda
iperf3 -s                  # Servidor
iperf3 -c IP_SERVIDOR      # Cliente
```

NAT não funciona

```
# Verificar IP forwarding
cat /proc/sys/net/ipv4/ip_forward  # Deve ser 1

# Verificar regras NAT
iptables -t nat -L -n -v
```

```
# Testar conectividade
ping -c 4 8.8.8.8          # Do cliente interno

# Verificar rotas
ip route show
```

---

14. SCRIPTS COMPLETOS

Script QoS + Firewall Integrado

```
#!/bin/bash

IF_WAN=eth0
IF_LAN=eth1
TOTAL_BW=10mbit

# === QoS ===
tc qdisc del dev $IF_WAN root 2>/dev/null
tc qdisc add dev $IF_WAN root handle 1: htb default 40

tc class add dev $IF_WAN parent 1: classid 1:1 htb rate $TOTAL_BW
tc class add dev $IF_WAN parent 1:1 classid 1:10 htb rate 2mbit ceil 5mbit prio 0
tc class add dev $IF_WAN parent 1:1 classid 1:20 htb rate 4mbit ceil 7mbit prio 1
tc class add dev $IF_WAN parent 1:1 classid 1:30 htb rate 2mbit ceil 4mbit prio 2
tc class add dev $IF_WAN parent 1:1 classid 1:40 htb rate 2mbit ceil 3mbit prio 3

# Filtros VoIP
tc filter add dev $IF_WAN protocol ip parent 1:0 prio 0 u32 \
   match ip dport 5060 0xffff flowid 1:10

# === FIREWALL ===
iptables -F
iptables -t nat -F

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

# Básico
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# Serviços
iptables -A INPUT -p tcp --dport 22 -s 192.168.100.0/24 -j ACCEPT
iptables -A INPUT -p tcp -m multiport --dports 80,443 -j ACCEPT

# === NAT ===
echo 1 > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A POSTROUTING -o $IF_WAN -j MASQUERADE
iptables -A FORWARD -i $IF_LAN -o $IF_WAN -j ACCEPT
```

```
iptables -A FORWARD -i $IF_WAN -o $IF_LAN -m state --state RELATED,ESTABLISHED -j ACCEPT

# Salvar
netfilter-persistent save

echo "Configuração completa aplicada!"
```

---

15. EXERCÍCIOS

QoS: Configure HTB com 3 classes de tráfego

ACL: Crie firewall permitindo apenas HTTP/HTTPS/SSH

NAT: Configure servidor como gateway

Port Forward: Redirecione porta 8080 para servidor interno

Integração: Script completo QoS + Firewall + NAT

RESUMO

QoS:

• TC (Traffic Control)

• HTB (Hierarchical Token Bucket)

• Classes e filtros

ACLs:

• iptables para controle de acesso

• Políticas de segurança

• Logging

NAT:

• Masquerade (SNAT)

• Port Forwarding (DNAT)

• NAT 1:1

Comandos:

• tc qdisc/class/filter

• iptables -t nat

• netfilter-persistent

FIM DO MATERIAL DIDÁTICO

Todos os 5 módulos completos!