



Hardening em Linux

Sandro Melo

A RNP – Rede Nacional de Ensino e Pesquisa – é qualificada como uma Organização Social (OS), sendo ligada ao Ministério da Ciência, Tecnologia e Inovação (MCTI) e responsável pelo Programa Interministerial RNP, que conta com a participação dos ministérios da Educação (MEC), da Saúde (MS) e da Cultura (MinC). Pioneira no acesso à Internet no Brasil, a RNP planeja e mantém a rede Ipê, a rede óptica nacional acadêmica de alto desempenho. Com Pontos de Presença nas 27 unidades da federação, a rede tem mais de 800 instituições conectadas. São aproximadamente 3,5 milhões de usuários usufruindo de uma infraestrutura de redes avançadas para comunicação, computação e experimentação, que contribui para a integração entre o sistema de Ciência e Tecnologia, Educação Superior, Saúde e Cultura.



Ministério da
Cultura

Ministério da
Saúde

Ministério da
Educação

Ministério da
Ciência, Tecnologia
e Inovação



Hardening em Linux

Sandro Melo



Hardening em Linux

Sandro Melo

Rio de Janeiro
Escola Superior de Redes
2014

Copyright © 2014 – Rede Nacional de Ensino e Pesquisa – RNP
Rua Lauro Müller, 116 sala 1103
22290-906 Rio de Janeiro, RJ

Diretor Geral
Nelson Simões

Diretor de Serviços e Soluções
José Luiz Ribeiro Filho

Escola Superior de Redes

Coordenação
Luiz Coelho

Edição
Lincoln da Mata

Revisão técnica
Edson Kowask e Wendel Soares

Equipe ESR (em ordem alfabética)
Adriana Pierro, Celia Maciel, Cristiane Oliveira, Derlinéa Miranda, Elimária Barbosa, Evelyn Feitosa, Felipe Nascimento, Lourdes Soncin, Luciana Batista, Luiz Carlos Lobato , Renato Duarte e Yve Marcial.

Capa, projeto visual e diagramação
Tecnodesign

Versão
1.0.0

Este material didático foi elaborado com fins educacionais. Solicitamos que qualquer erro encontrado ou dúvida com relação ao material ou seu uso seja enviado para a equipe de elaboração de conteúdo da Escola Superior de Redes, no e-mail info@esr.rnp.br. A Rede Nacional de Ensino e Pesquisa e os autores não assumem qualquer responsabilidade por eventuais danos ou perdas, a pessoas ou bens, originados do uso deste material.
As marcas registradas mencionadas neste material pertencem aos respectivos titulares.

Distribuição
Escola Superior de Redes
Rua Lauro Müller, 116 – sala 1103
22290-906 Rio de Janeiro, RJ
<http://esr.rnp.br>
info@esr.rnp.br

Dados Internacionais de Catalogação na Publicação (CIP)

M528h Melo, Sandro.
Hardening em Linux / Sandro Melo. – Rio de Janeiro: RNP/ESR, 2014.
278 p. : il. ; 27,5 cm.

Bibliografia: p.263.
ISBN 978-85-63630-27-8

1. Segurança de Computadores (Administração). 2. Linux (Software). 3. Segurança de software. I. Título.

CDD 005.268

Sumário

Escola Superior de Redes

A metodologia da ESR ix

Sobre o curso x

A quem se destina x

Convenções utilizadas neste livro x

Permissões de uso xi

Sobre o autor xii

1. Hardening em sistema Linux – pós-instalação

Exercício de nivelamento 1 – Introdução de hardening 2

O que é hardening 2

Exercício de fixação 1 – Vulnerabilidades 5

Pós-instalação 6

Exercício de fixação 2 – Pacotes instalados 7

Atualizações de segurança via Debsecan 8

Arquivos com permissão de Suid bit 10

Recomendações e boas práticas 10

Remoção de Suid bit 10

Segurança no sistema de arquivos 12

Exercício de fixação 2 – Particionamento 12

Execução do procedimento 12

Segurança no terminal 18

Desabilitar o uso de ‘CTRL+ALT+DEL’ 18

Exercício de fixação 2 – ‘CTRL+ALT+DEL’ 19



Limitar o uso dos terminais texto	19
Bloquear o terminal com a variável TMOUT	20
Gerenciamento de privilégios	21
Bloquear o login do root nos terminais texto	21
Determinar datas de expiração para contas de usuários	22
Remover shells válidas de usuários que não precisam delas	23
Instalação de pacotes específicos	24
Segurança do Grub	25
Habilitar senha nas versões antigas do Grub (versões 1 e 2)	26

Roteiro de Atividades 1 **27**

Atividade 1.1 – Hardening Linux – Apticron	27
Atividade 1.2 – Hardening Linux – Debsecan	27
Atividade 1.3 – Hardening Linux – Debsecan via cron	28
Atividade 1.4 – Hardening Linux – atualização	28
Atividade 1.5 – Hardening Linux – pacotes desnecessários	28
Atividade 1.6 – Hardening Linux – vlock	29
Atividade 1.7 – Hardening Linux – regras de montagem	29
Atividade 1.8 – Hardening Linux – Chkrootkit rkhunter	29
Atividade 1.9 – Hardening Linux – Baseline	30

2. Hardening em sistema Linux – controles de segurança para contas de usuários

Introdução	31
Exercício de nivelamento 1 – Controles de segurança para usuários	32
Controles de autenticação com a utilização do PAM	32
Sinalizadores de controle	33
Caminho do módulo	33
Exercício de fixação 1 – PAM	36
Procura por senhas fracas	37
Utilização do John the Ripper	37
Utilização de quota	38
Exercício de fixação 2 – Quota	40
Utilização de ACL – Posix 1e	41
Habilitação do suporte ao recurso de ACL	43
Comando de administração de ACL	43

Comando “sudo” (Super User Do – fazer como super usuário) **44**

Prevenção ‘Escape Shell’ **46**

Duas formas de desabilitar “Escape Shell” **47**

Roteiro de Atividades 2 49

Atividade 1 – PAM **49**

Atividade 2 – Procura por senhas fracas **50**

Atividade 3 – Quotas **50**

Atividade 4 – ACL **51**

Atividade 5 – Comando ‘sudo’ **53**

3. Hardening em sistema Linux – Registro de eventos (log) e HIDS

Exercício de nivelamento 1 – Registro de eventos (log) e HIDS **57**

Trilha de comandos **58**

Trilha de comando **58**

Registro de eventos – Logs do sistema **60**

Conformidade com as recomendações **60**

Boas práticas para arquivos de log **61**

Logs do sistema **62**

Syslog-NG **63**

Recursos (facility) **63**

Nível de registro (prioridade) **63**

Servidor Syslog centralizado **71**

Comando ‘logger’ **71**

Estrutura de Servidor e Cliente **72**

Configuração do servidor de logs **72**

Configuração do cliente **74**

Configuração do STUNNEL **75**

Definição da política de rotacionamento de logs **76**

Servidor NTP **78**

Contabilização de processos **79**

Auditoria com HIDS **80**

Roteiro de Atividades 3 83

Atividade 3.1 – Hardening Linux – Registro de eventos (logs) **83**

Atividade 3.2 – Auditoria HIDS **84**

4. Serviços de Redes – Parte 1

Checklist nos serviços do sistema	87
Exercício de fixação 1 – Checklist de Rede	92
Gerenciamento de serviços de redes em Inetd e Xinetd	94
Segurança em serviços de redes	96
Fingerprint de serviço	97
Exercício de fixação 1 – Levantamento de informação e Força Bruta	102
Hardening do serviço SSH	102
Mitigação de ataque de Força Bruta	106
Roteiro de Atividades 4	109
Atividade 4.1 – # service fail2ban restart	109

5. Serviços de Redes – Parte 2

Hardening	111
Remoção de módulos não utilizados	115
Ocultando a versão do servidor	116
Hardening do serviço DNS/BIND	121
PS-Watcher: Monitoração de serviços ativos	127

Roteiro de Atividades 5	129
--------------------------------	-----

Atividade 5.1 – Sistemas de detecção de intrusões (IDS) em redes WLAN	129
---	-----

6. Hardening em sistema Linux – Proxy Web

Introdução	131
Exercício de nivelamento 1 – Proxy web	131
Implementação de um Proxy Web com Squid – Instalação e Configuração do Proxy Cache	133
Criação de ACLs	139
Exercício de Fixação – Proxy web	141
ACL Time – Regras temporizadas	141

Roteiro de Atividades 6	147
--------------------------------	-----

Atividade 6.1 – Hardening Linux – Proxy web – Squid	147
---	-----

7. Firewall – Parte 1

Introdução	149
Arquitetura de firewall	150

Fundamentos para o IPTables	155
Trabalhando com IPTables	156
Tabelas	158
Listando (-L) as regras em tabelas e chains	160
Proposta de regras	164
Roteiro de Atividades 7	181
Atividade 7.1 – Hardening Linux – Firewall Linux	181

8. Segurança de Perímetro – Parte 2

Exercício de nivelamento 1 – Firewalls	183
Regras de controle	184
Construindo um script de firewall	192
Trabalhando com tradução de endereçamento IP	200
Trabalhando com redirecionamento de portas	202
Usando a Tabela Mangle	204
Analizando o conteúdo dos datagramas	207
Port Knocking – Conceito e prática	208
Roteiro de Atividades 8	211
Atividade 8.1 – Hardening Linux – Firewall Linux	211

9. Tuning de Kernel

Tuning de Kernel	215
Tuning TCP	216
Tuning ICMP	222
Definido resposta a Ping	222
Tuning IP	225
Habilitando repasse de Pacotes (IP_FORWARD)	225
Segurança de Kernel – Segurança na última fronteira	231
Definição de área do usuário	232
Definição de camada de Kernel	232
Utilização do módulo Yama	233
Tomoyo	234
Configuração e administração do Tomoyo	235

Roteiro de Atividades 9 241

Atividade 9.1 – Hardening Linux – Tuning de Kernel 241

10. Auditoria na camada de Kernel

Exercício de nivelamento 1 – Auditoria na camada de Kernel 243

Auditoria na camada de Kernel 243

Roteiro de Atividades 10 263

Atividade 10.1 – Hardening em Sistema Linux 263

Atividade 10.2 – Hardening Linux – Registro de Eventos (logs) e Auditoria 263

Escola Superior de Redes

A Escola Superior de Redes (ESR) é a unidade da Rede Nacional de Ensino e Pesquisa (RNP) responsável pela disseminação do conhecimento em Tecnologias da Informação e Comunicação (TIC). A ESR nasce com a proposta de ser a formadora e disseminadora de competências em TIC para o corpo técnico-administrativo das universidades federais, escolas técnicas e unidades federais de pesquisa. Sua missão fundamental é realizar a capacitação técnica do corpo funcional das organizações usuárias da RNP, para o exercício de competências aplicáveis ao uso eficaz e eficiente das TIC.

A ESR oferece dezenas de cursos distribuídos nas áreas temáticas: Administração e Projeto de Redes, Administração de Sistemas, Segurança, Mídias de Suporte à Colaboração Digital e Governança de TI.

A ESR também participa de diversos projetos de interesse público, como a elaboração e execução de planos de capacitação para formação de multiplicadores para projetos educacionais como: formação no uso da conferência web para a Universidade Aberta do Brasil (UAB), formação do suporte técnico de laboratórios do Proinfo e criação de um conjunto de cartilhas sobre redes sem fio para o programa Um Computador por Aluno (UCA).

A metodologia da ESR

A filosofia pedagógica e a metodologia que orientam os cursos da ESR são baseadas na aprendizagem como construção do conhecimento por meio da resolução de problemas típicos da realidade do profissional em formação. Os resultados obtidos nos cursos de natureza teórico-prática são otimizados, pois o instrutor, auxiliado pelo material didático, atua não apenas como expositor de conceitos e informações, mas principalmente como orientador do aluno na execução de atividades contextualizadas nas situações do cotidiano profissional.

A aprendizagem é entendida como a resposta do aluno ao desafio de situações-problema semelhantes às encontradas na prática profissional, que são superadas por meio de análise, síntese, julgamento, pensamento crítico e construção de hipóteses para a resolução do problema, em abordagem orientada ao desenvolvimento de competências.

Dessa forma, o instrutor tem participaçãoativa e dialógica como orientador do aluno para as atividades em laboratório. Até mesmo a apresentação da teoria no início da sessão de aprendizagem não é considerada uma simples exposição de conceitos e informações. O instrutor busca incentivar a participação dos alunos continuamente.

As sessões de aprendizagem onde se dão a apresentação dos conteúdos e a realização das atividades práticas têm formato presencial e essencialmente prático, utilizando técnicas de estudo dirigido individual, trabalho em equipe e práticas orientadas para o contexto de atuação do futuro especialista que se pretende formar.

As sessões de aprendizagem desenvolvem-se em três etapas, com predominância de tempo para as atividades práticas, conforme descrição a seguir:

Primeira etapa: apresentação da teoria e esclarecimento de dúvidas (de 60 a 90 minutos). O instrutor apresenta, de maneira sintética, os conceitos teóricos correspondentes ao tema da sessão de aprendizagem, com auxílio de slides em formato PowerPoint. O instrutor levanta questões sobre o conteúdo dos slides em vez de apenas apresentá-los, convidando a turma à reflexão e participação. Isso evita que as apresentações sejam monótonas e que o aluno se coloque em posição de passividade, o que reduziria a aprendizagem.

Segunda etapa: atividades práticas de aprendizagem (de 120 a 150 minutos).

Esta etapa é a essência dos cursos da ESR. A maioria das atividades dos cursos é assíncrona e realizada em duplas de alunos, que acompanham o ritmo do roteiro de atividades proposto no livro de apoio. Instrutor e monitor circulam entre as duplas para solucionar dúvidas e oferecer explicações complementares.

Terceira etapa: discussão das atividades realizadas (30 minutos).

O instrutor comenta cada atividade, apresentando uma das soluções possíveis para resolvê-la, devendo ater-se às aquelas que geram maior dificuldade e polêmica. Os alunos são convidados a comentar as soluções encontradas e o instrutor retoma tópicos que tenham gerado dúvidas, estimulando a participação dos alunos. O instrutor sempre estimula os alunos a encontrarem soluções alternativas às sugeridas por ele e pelos colegas e, caso existam, a comentá-las.

Sobre o curso

O curso consiste na análise de ameaças, minimização de riscos e execução de atividades corretivas das vulnerabilidades identificadas em servidores na plataforma Linux.

Apresenta os conceitos teóricos e através de atividades práticas a fim de reforçar e garantir a segurança de servidores Linux para as mais diversas aplicações. Serão apresentadas diversas técnicas, procedimentos e ferramentas para que os servidores Linux tenham a segurança de seus serviços elevada. Os assuntos tratados possibilitarão melhoria da segurança de servidores considerando o modelo DAC como também do modelo MAC, pois serão realizados procedimentos de segurança com ferramentas no contexto da userland e também recursos de segurança de Kernel.

A quem se destina

Este curso é destinado à administradores de redes e de servidores Linux interessados em proteger suas redes e realizar ajustes detalhados (tunning) em seus servidores web, através da implementação de módulos de segurança e Web Application Firewall (WAF).

Convenções utilizadas neste livro

As seguintes convenções tipográficas são usadas neste livro:

Itálico

Indica nomes de arquivos e referências bibliográficas relacionadas ao longo do texto.

Largura constante

Indica comandos e suas opções, variáveis e atributos, conteúdo de arquivos e resultado da saída de comandos. Comandos que serão digitados pelo usuário são grifados em negrito e possuem o prefixo do ambiente em uso (no Linux é normalmente # ou \$, enquanto no Windows é C:\).

Conteúdo de slide

Indica o conteúdo dos slides referentes ao curso apresentados em sala de aula.

Símbolo

Indica referência complementar disponível em site ou página na internet.

Símbolo

Indica um documento como referência complementar.

Símbolo

Indica um vídeo como referência complementar.

Símbolo

Indica um arquivo de áudio como referência complementar.

Símbolo

Indica um aviso ou precaução a ser considerada.

Símbolo

Indica questionamentos que estimulam a reflexão ou apresenta conteúdo de apoio ao entendimento do tema em questão.

Símbolo

Indica notas e informações complementares como dicas, sugestões de leitura adicional ou mesmo uma observação.

Permissões de uso

Todos os direitos reservados à RNP.

Agradecemos sempre citar esta fonte quando incluir parte deste livro em outra obra.

Exemplo de citação: TORRES, Pedro et al. *Administração de Sistemas Linux: Redes e Segurança*.

Rio de Janeiro: Escola Superior de Redes, RNP, 2013.

Comentários e perguntas

Para enviar comentários e perguntas sobre esta publicação:

Escola Superior de Redes RNP

Endereço: Av. Lauro Müller 116 sala 1103 – Botafogo

Rio de Janeiro – RJ – 22290-906

E-mail: info@esr.rnp.br

Sobre o autor

Sandro Melo É doutorando no TIDD-PUC/SP, Mestre em Engenharia da Computação no IPT/USP, Mestre em Engenharia de Redes pelo IPT/USP; Pós-graduado em Administração de Redes Linux pela UFLA-MG; Pós-graduado em Análise de Sistemas e Graduação em Processamento de Dados pela Universidade Mackenzie, atua na área de TI desde de 1997, realizando neste período vários projetos de implantação de serviços de rede e segurança. Atualmente atua como Coordenador do Curso de Redes de Computadores na BANDTEC responsável pela cadeira de Sistemas Operacionais, Computação Distribuída e Centralizada e Segurança e Auditoria. É um evangelista do Software Livre, sendo embaixador Fedora, Proctor BSDA e LPI, também atua como professor convidado responsável por cátedras inerentes a Segurança e Computação Forense em sistema Linux, já tendo a oportunidade de atuar na UFLA/ARL (MG), IBTA (Campinas), UNISALES (ES), Faculdade Pitágoras (Guarapari-ES, Teixeira de Freitas-BA), Universidade Potiguar (RN), ITA (SP), IEASAM (PA), Uniron (RO), FAAR(RO), FACID (PI) entre outras. Sendo idealizador e coordenador dos cursos de Pós Graduação Segurança e Computação Forense nas entidades: Universidade Maurício de Nassau (AL) , Faculdade FACID (PI), Faculdade CET(PI), Faculdade Atual (RR), Iquali/EEEMBA (BA); Instrutor autorizado FreeBSDBrasil e convidado da Academia Clavis.

Edson Kowask Bezerra é profissional da área de segurança da informação e governança há mais de quinze anos, atuando como auditor líder, pesquisador, gerente de projeto e gerente técnico, em inúmeros projetos de gestão de riscos, gestão de segurança da informação, continuidade de negócios, PCI, auditoria e recuperação de desastres em empresas de grande porte do setor de telecomunicações, financeiro, energia, indústria e governo. Com vasta experiência nos temas de segurança e governança, tem atuado também como palestrante nos principais eventos do Brasil e ainda como instrutor de treinamentos focados em segurança e governança. É professor e coordenador de cursos de pós-graduação na área de segurança da informação, gestão integrada, de inovação e tecnologias web. Hoje atua como Coordenador Acadêmico de Segurança e Governança de TI da Escola Superior de Redes.

Wendel Soares é formado em Segurança da Informação pela Faculdade Rogacionista e especialista em redes de computadores. Foi por oito anos administrador de redes do Ministério da Defesa Exército Brasileiro, trabalhando com redes de segurança de TI. Colaborador da Escola Superior de Redes desde 2008, tendo lecionado cursos de Linux e Segurança de Redes. Atualmente trabalha como Especialista de redes para uma Multinacional Americana.

1

Hardening em sistema Linux – pós-instalação

objetivos

Rever conceitos relacionados à segurança; Compreender conceitos importantes sobre hardening em Linux; Avaliar um Baseline; Realizar procedimentos de instalação diferenciados.

conceitos

Conceitos de Segurança; Conceitos de hardening; Baseline; Proposta de procedimentos após a instalação.

O coração do sistema Linux é o seu kernel e também seu Sistema Operacional. Combinados, eles formam a base onde estão todas as suas aplicações que rodam no computador. Comparativamente falando, o Sistema Operacional Linux e seu kernel são razoavelmente seguros. Um grande número de opções de segurança estão inclusas no kernel e uma grande variedade de ferramentas e configurações de segurança open-source já estão inclusas nas mais variadas distribuições. Adicionalmente, o Linux oferece excepcional controle sobre quem, como e quais recursos e aplicações os usuários podem ter acesso. Então, onde está o risco?

Normalmente, “o problema está nos detalhes”. A segurança do sistema depende de uma enorme quantidade de elementos de configuração em nível, tanto de Sistema Operacional quanto de aplicações. O próprio Sistema Operacional e seu kernel são bastante complexos, e sua correta configuração não é um processo trivial. Sistemas Operacionais baseados no kernel Linux possuem infinitas configurações, e cada ajuste pode causar sérios problemas de segurança. Vulnerabilidades e falhas de segurança não são fáceis de encontrar.

Para estar apto para numerar esses pontos, precisamos adquirir sólidos conhecimentos sobre os requisitos básicos de segurança do nosso Sistema Operacional e seu núcleo.

Hardening em Linux é um processo de proteção do Sistema Operacional, seu kernel e suas aplicações contra ameaças conhecidas ou não através da aplicação de técnicas específicas de controle, minimização de riscos e execução de atividades corretivas.

Exercício de nivelamento 1



Introdução de hardening

Como é possível perceber a necessidade de segurança nos serviços e nas máquinas servidoras de uma organização?

Execute um fork attack como usuário comum no shell da máquina virtual e avalie a partir do resultado o quanto uma instalação padrão de Linux é segura.

Fork Attack: :(){ :|:& };:

O que é hardening



É a proteção do sistema por meio da redução de suas possíveis vulnerabilidades.

Devemos:

- ▣ Configurar o sistema.
- ▣ Instalar pacotes destinados a algum procedimento de segurança.
- ▣ Modificar o permissionamento para melhorar e reforçar a segurança.



De acordo com a Wikipedia, “hardening é um processo de mapeamento das ameaças, mitigação dos riscos e execução das atividades corretivas – com foco na infraestrutura e com o objetivo principal de torná-la preparada para enfrentar tentativas de ataque”.

Já segundo a ITSecurity.com, “hardening é o processo de otimizar as configurações de segurança de um sistema, um termo comumente aplicado a Sistemas Operacionais”.

A palavra “hardening” tem origem no idioma inglês e significa “endurecimento”, mas no contexto da Segurança Computacional também é o processo de proteger um sistema através da redução de suas possíveis vulnerabilidades, por meio de configurações que implementam controles específicos. Esse processo implica realizar várias configurações, instalação de pacotes destinados a algum procedimento de segurança e modificação de permissionamento com o objetivo de melhorar e reforçar a segurança do ambiente. Antes de iniciar a explanação sobre o tema, é interessante perguntar: o que é hardening?

Ao considerar a principal tradução de hardening, que significa “endurecer-se”, é exatamente isso o que desejamos fazer com um Sistema Operacional antes de colocar um servidor em produção. Essa ideia fica mais clara se for considerada a tradução como “fortalecimento”, que do ponto de vista empírico pode ser explicado como um conjunto de configurações e melhoramento, ajustes finos que vão gerar controles para que o sistema torne-se mais seguro.

Administradores sem muita experiência em segurança preparam seus servidores com uma instalação básica e depois que suas aplicações já estão funcionando, acabam deixando da maneira que ficou, pois a possibilidade de fazer com que aplicação pare de funcionar realizando um procedimento de segurança é grande. Eis que muitas vezes surge a seguinte frase: “Ah, já está funcionando sem ninguém mexer!”



Saiba mais

O sistema Linux torna-se bastante seguro quando é devidamente trabalhado. Dessa forma, é necessário rever suas configurações-padrão. Adicionar controles específicos pode tornar qualquer servidor mais seguro e preparado para a internet.

Os Sistemas Operacionais modernos trazem muitos “controles” para melhorar a segurança e gerenciar melhor o uso dos recursos, mas quando se trata de Sistema Operacional moderno, desenhado para prover serviços de redes, é muito comum que esses controles estejam desabilitados. Uma prova disso é que a maioria das distribuições são vulneráveis a um ataque de fork bomb, ou seja, um ataque simples que consiste em exaurir os recursos, abrir um processo após o outro até não termos mais recursos e o servidor travar. Um ataque de fork bomb no Linux é facilmente efetuado executando a seguinte instrução em uma shell:

```
$ :(){ :|:& };:
```

Esse é um exemplo de função que é executada de forma recursiva. É um meio conhecido e frequentemente usado por administradores de sistemas para testar as limitações dos processos de usuário, embora os limites de execução de processos de usuário em um sistema Linux possa ser configurado via /etc/security/limits.conf e PAM. É comum não ter nenhuma definição estabelecida em um sistema Linux. Dessa forma, um usuário comum pode causar uma Denial of Service (DoS) através de um fork bomb.

Uma vez que um fork bomb é executado em um sistema, pode não ser possível mais retomar o funcionamento normal sem reiniciar o sistema como a única solução. Isso acontece porque a única solução é destruir todas as instâncias do sistema, mas a ação é tão rápida que raramente haverá uma oportunidade para interromper o ataque.

Entendendo o Fork Bomb via /bin/bash:

- “:()”: é a definição da função chamada “:”, uma função que não aceita argumentos. A sintaxe para a função vai Bash é:

```
:(){  
    :|:&  
};:
```
- “:|:”: em seguida, dentro do corpo da função, é feita uma chamada usando a técnica de programação chamada “recursão”, motivando a gradual alocação de recurso do Sistema Operacional, tendo por consequência um travamento do sistema;
- “&”: coloca a chamada de função em segundo plano para que cada processo-filho criado não morra e fique consumindo recursos do sistema;
- “,”: finaliza a definição da função;
- “:”: faz a chamada para execução novamente.

Um exemplo mais prático poderia ser:

```
fork_bomb() {  
  
    fork_bomb | fork_bomb &  
  
}; fork_bomb
```

Um ataque de fork bomb é motivo suficiente para se pensar em um hardening. Entretanto, no momento que se inicia a implementação das técnicas de hardening, é necessário pensar em três fatores: segurança, risco e flexibilidade, como ilustrado na figura a seguir:

Se não existe segurança 100% então o que existe?



Figura 1.1
Segurança,
flexibilidade e risco.

O desafio é saber dosar muito bem esses três fatores para definir um conjunto de controles que possam proporcionar ao sistema equilíbrio entre produtividade e segurança. Muitas perguntas podem ocorrer durante o processo, como: "Qual o nível de segurança desejado? Quanto o sistema vai ficar em conformidade com a políticas de segurança definidas? Quão auditável deverá ficar o sistema?"

Outro grande desafio é que os fatores "segurança" e "flexibilidade" ou mesmo "segurança" e "risco" são inversamente proporcionais e consequentemente impactam diretamente no risco. É um fato que não é possível ter segurança 100%, mas, por outro lado, quanto mais segurança, menores serão o risco e a flexibilidade.

Uma vez que não é possível ter 100% de segurança, deve-se elaborar o maior número possível de controles para tornar um Sistema Operacional mais seguro. Dessa forma, o hardening permite mitigar possíveis vulnerabilidades.

Para pensar

É fato: ao diminuir, por meio de um processo de hardening, as possibilidades de vulnerabilidades que possam trazer ameaças, serão reduzidos também os riscos ao sistema.



Qual saída?

▪ Com Metodologia...

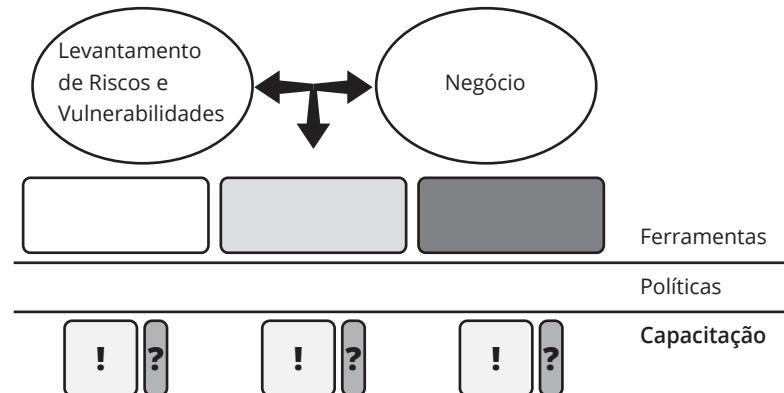


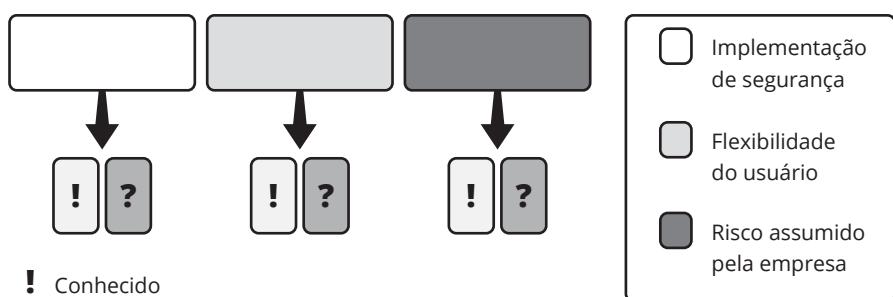
Figura 1.2
Dificultadores: não
é possível ter 100%
de segurança.

É notório que, quando são implementados controles, o sistema passa a prover um ambiente mais rígido, possibilitando pouca flexibilidade. Por outro lado, se em um sistema damos grande flexibilidade ao usuário, é fato que a segurança será diminuída e os riscos vão aumentar. Por exemplo: um cenário de Sistema Operacional onde não exista qualquer diretriz de segurança e se permita que qualquer tipo de programa possa ser executado por qualquer usuário. Somando-se a isso ainda existem complicadores, que podem ser divididos em dois grupos:

- **Vulnerabilidade conhecida:** servidor com um sistema legado que naquele momento não pode ser desligado, por exemplo;
- **Vulnerabilidade desconhecida:** uma vulnerabilidade do Sistema Operacional que ainda não foi reportada pelo fabricante ou uma correção não publicada (ilustrado na figura 1.2).

Qual saída?

- Tem “um” complicador...



Exercício de fixação 1

Vulnerabilidades

Cite duas vulnerabilidades conhecidas nos equipamentos servidores da sua organização.

Não existe regra direta para a dosagem desses três fatores ilustrados nas figuras 1.2. e 1.3, devido ao fato de que depende de cada situação e de acordo com o tipo de necessidade. Por isso, toda implementação de um servidor deve ter seu planejamento de segurança bem definido antes de sua instalação, ou seja, a elaboração do baseline que formalizará o hardening é fundamental.

Deve-se lembrar que o conjunto de técnicas e ferramentas para implementação de um hardening, normalmente utilizadas em um Sistema Operacional Linux, sejam interessantes do ponto de vista de garantia dos serviços e realmente agreguem à segurança. Não será incomum o fato de que um conjunto de ações pré-estabelecidos em uma baseline não se apliquem em todas as situações. Cada caso é um caso. É preciso analisar e descobrir que controles serão mais adequados para a necessidade em questão e também qualificar o quanto de segurança foi estabelecido no respectivo contexto.

A segurança na camada do Sistema Operacional é iniciada no hardening, mas não termina nele. E deve ser aderente, ou seja, estar em conformidade com as políticas das empresas.

Diante disso, é conveniente lembrar que ferramentas são importantes, mas não são o fim e sim o meio para a execução dos procedimentos do processo, que, somados a outras ações, como a capacitação das pessoas envolvidas, é fator determinante de sucesso, pois a imperícia é um inimigo de qualquer trabalho. Explicando de um ponto de vista prático, esse processo envolve:

- ▣ A remoção ou desativação de serviços desnecessários;
- ▣ Remoção de contas de usuários-padrão;
- ▣ Desinstalação de pacotes desnecessários;
- ▣ Definição do processo de atualização;
- ▣ Definição de controles para auditoria;
- ▣ Definir controles para limites do uso dos recursos pelas aplicações e/ou usuários;
- ▣ Instalação de pacotes de ferramentas de segurança e auditoria.

Pós-instalação

Após o planejamento da instalação e sua execução, o sysadmin vai iniciar a execução do hardening, que também foi previamente planejado no Baseline. É fato que um bom processo de hardening tem como princípio “menor recurso e menor privilégio”.

Um bom exemplo da aplicação desse princípio é verificar a lista de pacotes instalados, pois serão identificados possíveis pacotes que não necessários para a finalidade do servidor ou aplicações que demandarão ter o permissionamento revisto.

Os comandos a seguir podem ser utilizados:

No Debian:

```
# dpkg -l | awk '{print $2,$3}' | sed '1,7d'  
  
gravando o resultado em um arquivo  
# dpkg -l | awk '{print $2,$3}' | sed '1,7d' > /root/pacotes
```

No Red Hat:

```
# rpm -qa  
  
gravando o resultado em um arquivo  
# rpm -qa > /root/pacotes
```

A CIS Security também cita a remoção de programas que não estão em uso e ressalta que tais programas podem ser usados por um exploit (código que explora a vulnerabilidade de um programa para ganhar um acesso não autorizado ao sistema) para um ataque tanto local quanto remoto, dependendo do programa.

Exemplos de programas que podem ser desnecessários, dependendo do tipo de servidor:

- ▣ **lynx**: cliente http/ftp que possibilita transferência de malware;
- ▣ **wget**: cliente http/ftp que possibilita transferência de malware;

Boas práticas recomendadas na ISO/IEC 27002:2008: no que diz respeito a programas instalados, a norma ISO/IEC 27002:2008 recomenda que seja removido todo utilitário ou software desnecessário do sistema. Dessa forma, executar esse procedimento durante o hardening possibilitará a conformidade com o que é recomendado na ISO/IEC 27002:2008.

- **netcat (nc)**: “canivete suíço” que possibilita transferência de malware ou até mesmo criar backdoors;
- **hping**: montador de pacotes que possibilita criar backdoors via rawsocket.

! Todas as práticas propostas neste livro deverão ser executadas nas máquinas virtuais elaboradas para essa finalidade.

Por outro lado, existe também a questão dos pacotes básicos que devem ser instalados em um servidor, como, por exemplo, um bom sistema de registro de eventos (logs), um serviço de sincronização de relógio (NTP) e ferramentas que auxiliem o administrador a uma resposta inicial de incidente.

Outro ponto importante: que os comandos foram testados em duas máquinas virtuais (uma Debian e outra Ubuntu). No caso das exceções, no momento oportuno será informado. Isso também é importante para lembrar que, embora as sugestões recomendadas sejam aplicadas de qualquer distribuição Linux, deverá o sysadmin avaliar como fazer em sua distribuição caso ela seja diferente da utilizada como base para este livro. O fato de ser Debian 7 não limita ou mesmo tornam exclusivos os conceitos e recursos a esse cenário, pois administradores mais experientes deverão saber interpretar as diferenças entre as distribuições, além do fato de os procedimentos serem baseados no modo Command Line (CLI), o que torna a reprodução praticamente universal, pois fica vinculado apenas à shell usada, que foi /bin/bash.

Exercício de fixação 2

Pacotes instalados

Alguns desses programas apresentados estão instalados na sua organização? Por quê?

Avalie a proposta da realização de instalação com o mínimo de pacotes possíveis.
Cite as vantagens.



Saiba mais

A ação de avaliar os programas instalados em um respectivo Sistema Operacional deve estar elencada em qualquer baseline. Um bom exemplo que valida essa boa prática é o Sistema Operacional Windows, que também vem com programas cliente como o tftp, que são comumente usados por malware para transferências de arquivos.

Muitos outros aplicativos podem entrar nessa lista, como: nmap, tcpdump, telnet (client), ftp (client) e shred, pois, devido aos recursos que proporcionam, devem ser devidamente avaliados.

Caso a remoção do pacote seja a melhor decisão, esta pode ser realizada da seguinte forma:

```
# apt-get remove --purge wget
```

Uma alternativa é rever o permissionamento dos binaries com o comando *chmod*, deixando-os apenas com direitos para o root e em casos extremos definir regras via sudo para utilização desse aplicativos, considerando sempre as questões de segurança. O sudo é um recurso que pode ser útil, embora deva ser usado sempre com cautela e será ensinado no Capítulo 2.

A gestão de atualização de pacotes é uma tarefa necessária, pois é vital à segurança. Diante disso, é preciso ter controles definidos para habilitar recursos no Sistema Operacional que

facilitem essa tarefa importante. Em distribuições Like Debian com Ubuntu, um aplicativo interessante é o apticron, que informa por e-mail pacotes que devem ser atualizados. O apticron é um shell script que usa as informações do apt-listchanges. Para instalar o apticron:

```
# apt-get install -y apticron
```

Durante a instalação é inserida uma entrada no /etc/cron.d, similar a esta:

```
# cat /etc/cron.d/apticron  
15 * * * * root if test -x /usr/sbin/apticron; then /usr/sbin/  
apticron --cron; else true; fi
```

As configurações do apticron ficam no /etc/apticron. Normalmente basta inserir o e-mail do administrador. Segue um exemplo de configuração:

```
# cat /etc/apticron/apticron.conf | grep -v ^#  
EMAIL="sandro.melo@bandtec.com.br"  
DIFF_ONLY="1"  
LISTCHANGES_PROFILE="apticron"  
NOTIFY_NEW="0" CUSTOM SUBJECT="RNP Security: Aviso de atualização de  
pacotes
```

Atualizações de segurança via Debsecan

A maioria das distribuições Linux possibilita a atualização via repositório de pacotes disponíveis na internet. No caso de distribuições “Like Debian”, como o Ubuntu, é possível fazer uma avaliação com o comando *debsecan* (Debsecan tool – Debian Security Analyzer) sobre registro de vulnerabilidade do [CVE](http://www.mitre.org) (www.mitre.org).

CVE

Common Vulnerabilities and Exposures. Define-se como um padrão no tratamento e divulgação de informações sobre vulnerabilidades reportadas. É o padrão usado na área de segurança da informação para enumerar as vulnerabilidades. Esses identificadores fornecem pontos de referência na troca de dados para que os produtos de segurança da informação e serviços possam falar uns com os outros, ou seja, produtos e serviços certificados com o CVE podem proporcionar uma melhor cobertura, integração e segurança reforçada em seu ambiente.

Essa ferramenta verifica a base de pacotes instalados, correlacionando com informações de vulnerabilidades publicadas.

Devido ao fato de o debsecan sempre trazer a informação completa dos pacotes com notificação de vulnerabilidades, é interessante primeiro consultar os pacotes que possuem correção pronta, aplicá-la e gerar uma lista branca com os pacotes já corrigidos (whitelist), para que seja possível ter uma administração do que já foi efetivamente atualizado.

Identificação da disponibilidade dos pacotes do sistema que demandam atualização:

```
# debsecan --suite lenny --only-fixed --format packages  
libasn1-8-heimdal  
libcurl3  
libcurl3-gnutls  
libcurl3-nss  
libgssapi3-heimdal  
libhcrypto4-heimdal  
libheimbase1-heimdal  
libheimntlm0-heimdal
```

```
libhx509-5-heimdal  
libkrb5-26-heimdal  
libroken18-heimdal  
libt1-5  
libwind0-heimdal  
libxml2  
python-libxml2
```

Onde:

- **--suite**: define o release;
- **--only-fixed -format** -: para definir a forma em que as informações serão apresentadas.

Nesse cenário, é recomendado que todos os pacotes listados, após serem atualizados, sejam inseridos na whitelist. Assim sendo, segue o procedimento de atualização via apt-get:

```
#debsecan --suite wheezy --only-fixed --format simple | awk '{ print  
$1 }' > list_update
```

Após gerar a lista, use novamente o debsecan para atualizar os respectivos pacotes:

```
# apt-get install -y $(# debsecan --suite wheezy --only-fixed  
--format packages)
```

Após o término, atualize a whitelist:

```
# debsecan ---add-whitelist $(cat list_update)
```

Verifique a atualização da whitelist:

```
# debsecan --show-whitelist
```

Existe opção de gerar uma entrada no crontab para que o debsecan seja executado via cron. Basta executar o comando:

```
# debsecan-create-cron
```

Uma entrada similar a essa será criada no diretório "/etc/cront.d":

```
# cron entry for debsecan  
  
MAILTO=root  
  
19 * * * * daemon test -x /usr/bin/debsecan && /usr/bin/debsecan  
--cron  
  
# (Note: debsecan delays actual processing past 2:00 AM, and runs  
only  
  
# once per day.)
```

Uma forma interessante de uso do debsecan é forçar o envio de e-mail apenas com informações das atualizações necessárias disponíveis.

```
debsecan --suite wheezy --only-fixed --format report --mailto root  
--update-history
```

Arquivos com permissão de Suid bit

A permissão de **Suid bit** possibilita que um determinado binário possa ser executado por outros ou por membros do grupo com os mesmos direitos do “usuário efetivo”, ou seja, usuário dono. Isso cria a possibilidade se o usuário dono de um determinado binário for o usuário root, com a atribuição do direito especial de suidbit a esse binário, o binário poderá ser executado por outro usuário comum do sistema com o mesmo poder do root.

Muitos binários no sistema já vêm com a permissão de Suid bit, devido ao fato de que alguns desses binários cujo dono é o usuário root necessitam em algum contexto serem usados por um usuário comum. Exemplos clássicos desses comandos são o *su*, o *ping*, e o *passwd*, entre muitos outros. Mas uma coisa deve ser avaliada: será que todos esses binários que já estão com a permissão de Suid bit serão usados por algum usuário comum do sistema? Será que os usuários comuns do sistema precisam de todos esses comandos à disposição? Será que no servidor em questão usuários comuns terão shell válida? Demanda-se que o administrador fique atento para esses detalhes. Como o sistema possui muitos binários com Suid bit, problemas como a combinação de shells com Suid bit podem passar despercebidos. Dessa forma, o conceito de “menor privilégio e menor recurso” deve sempre ser um balizador para decisões. Então, o que pode ser feito? Retirar todas as permissões de Suid bit do sistema e somente deixar as permissões para os binários que sejam fundamentais para operações de um determinado usuário. No entanto, não existe regra geral: haverá casos diferentes, pois os binários que julgam-se importantes para um servidor firewall, por exemplo, podem não ser para um servidor de e-mail.

Recomendações e boas práticas

É uma boa prática que o acesso à informação e às funções dos sistemas de aplicações por usuário e pessoal de suporte seja restrito de acordo com o definido na política de controle de acesso. Dessa forma, a remoção da permissão de Suid bit dos diversos binários que a possuem deve ser mais um procedimento de hardening.

A CIS SECURITY também recomenda a remoção da permissão de Suid bit dos diversos binários que a possuem.

Remoção de Suid bit

Para a remoção de todas as permissões de Suid bit dos binários, é necessário fazer uma pesquisa no sistema e gerar lista com todos para melhor análise. Nesta exemplificação será criado um diretório chamado “teste”, dentro do /root para que seja gravada a lista de arquivos com Suid bit, que será gerada pelo comando *find*.

```
# cd /root  
  
# mkdir teste  
  
# find / -perm -04000 > /root/teste/lista.suid
```

Suid bit

Permissão que só trabalha com arquivos executáveis serve para o propósito de um usuário comum poder executar um determinado binário com o poder do “dono” efetivo. Essa permissão é representada pela letra “s”.



Boas práticas recomendadas na ABNT NBR ISO/IEC 27002:2008: por recomendação relacionada ao item 11.6.1, o acesso à informação e às funções dos sistemas de aplicações por usuário e pessoal de suporte deve ser restrito de acordo com o definido na política de controle de acesso. Dessa forma, impondo limites e controles sobre Suid bit o usuário estará em conformidade com a norma.



O número 4000 representa a permissão de Suid bit. Embora seja passado o parâmetro -04000, a parte importante são os quatro últimos dígitos. Os três zeros são as permissões-padrão do sistema (0 para usuário, 0 para grupo e 0 para outros), e o 4 representa a permissão Suid bit. Podemos ter no lugar do 4 o número 2, que é o direito especial denominado Sgid bit, ou 1, que representa stick bit.

Vamos fazer uma listagem detalhada em um desses binários para ver como a permissão de Suid bit é representada:

```
# ls -l /bin/su
```

Nas permissões do usuário, é representado um s no lugar do x para indicar Suid bit. Isso representa que a permissão de Suid bit está ativada, mas a permissão de execução (x) é intrínseca ao Suid bit.

Com a lista gerada com todos os binários, podemos avaliar quais binários realmente necessitam continuar com a permissão de Suid bit. Considere um exemplo de um **Bastion Host**, que é o firewall de uma rede. Podemos deixar dois binários com o Suid bit definidos para os binários `passwd` e `su`. Deixar o `passwd` para que um usuário comum possa mudar a sua senha de acesso caso seja um administrador e acesse essa máquina remotamente assumindo que em um contexto como esse o root não faz login diretamente, tanto local ou remotamente. Já o comando `su` vai ser de vital importância, pois caso o usuário root não tenha acesso direto, o administrador então terá de efetuar o login com seu usuário comum e poderá usar o `su` para se tornar o usuário root.

Como retirar todas as permissões de Suid bit dos binários:

```
# chmod -s -Rv /
```

Onde:

- **-s**: retira a permissão de Suid bit;
- **-R**: é recursivo, do / (barra) para baixo;
- **-v**: é o modo verbose (mostra o que está sendo feito pelo comando).

```
# ls -l /bin/su
```

Logo após remover o Suid bit de todo o sistema, basta definir a permissão de Suid bit somente para os dois binários que julgarmos realmente necessário.

```
# chmod +s /usr/bin/passwd
```

```
# chmod +s /bin/su
```

```
# ls -l /usr/bin/passwd
```

```
# ls -l /bin/su
```

Pode ser que para um determinado servidor só o `passwd` e o `su` não sejam suficientes. Provavelmente outros binários precisem estar com o Suid bit ativado. Cabe ao administrador analisar o que será necessário para cada tipo de serviço. Uma solução alternativa ao Suid bit é a adoção do comando `sudo`, que será tratado no capítulo 2.



Segurança no sistema de arquivos

Na instalação de um sistema Linux, as boas práticas nas instalações aconselham a partitionar o disco e colocar os principais diretórios em partições separadas. Isso pode proporcionar maior segurança, pois cada partição tem sua tabela separada e pode ter regras de montagem melhor elaboradas.

Logo, se toda instalação está concentrada em uma única partição, já se inicia um servidor com grande limitação. Mas se temos um exemplo onde o diretório "/" (barra) está em uma partição e o "/home" em outra, no caso de algum problema no sistema ou até em uma reinstalação, pode-se atuar na partição "/home" sem interferir, em um primeiro momento, nas informações vinculadas ao diretório "/home". As definições de montagem que devem ser avaliadas para esse contexto são:

- **Nodev**: tira o suporte a arquivos de dispositivos;
- **Noexec**: desabilita o suporte à execução dos arquivos;
- **Nosuid**: deixa desabilitado o suporte ao direito especial de suidbit;
- **Noatime**: desativa o registro de tempo/data de acesso dos arquivos (denominado atime);
- **Ro**: define que será somente leitura.

Exercício de fixação 2

Particionamento

Como é feita a segurança dos particionamentos das máquinas servidoras na sua organização?

Entretanto, é relevante destacar que a segurança de um particionamento não se resume a isso. Todas essas partições são montadas nos diretórios, e um erro que muitos administradores de sistemas Linux cometem é não ler a documentação para conhecer melhor os recursos e descobrir que o comando *mount* possibilita a utilização de algumas opções muito interessantes, que permitem melhorar muito a segurança nas partições. Caso todos os diretórios estivessem na mesma partição – no "/" (barra) –, não seria possível ter essas opções e usufruir desses recursos.

Mostramos neste capítulo que é possível retirar a permissão de Suid bit dos binários, mas só isso não resolveria. "Pode-se cortar o mal pela raiz" com as opções do *mount*, dizendo que na partição em questão os binários com permissão de Suid bit não terão efeito, ou seja, esse é um recurso muito interessante para montagem: desabilitar a execução de binários com suidbit em uma partição definida, aumentando a segurança do sistema através da limitação de um recurso poderoso.

Execução do procedimento

A primeira opção do *mount* que será mostrada é o *nosuid*, que faz com que binários com permissão de Suid bit não tenham efeito na partição na qual estão definidos. Para um exemplo prático, adicione um usuário:

```
# adduser teste
```

Faça uma cópia das shells do seu sistema para o diretório “home” desse usuário e atribua às shells a permissão de Suid bit.

```
# cp /bin/*sh* /home/teste  
  
# chmod 4755 /home/teste/*sh*
```

Efetue login em outro terminal com um usuário comum que foi criado anteriormente e tente executar uma dessas shells.

```
$ cd /home/teste  
  
$ ./sh  
  
# id
```

Na saída do comando *id*, pode-se ver que esse usuário conseguiu permissões de root.

É recomendável, para fins de melhor conhecimento do sistema, que esse teste seja feito em todas as shells disponíveis. Assim, é possível estimar melhor o poder de uma ameaça em um contexto similar a esse.

Para resolver isso, basta remontar a partição onde está montado o “/home”, mas com a opção de nosuid.

```
# mount -o remount,rw,nosuid /home  
  
# mount
```

O comando *mount* sem parâmetros nos mostra quais as partições estão montadas e suas respectivas opções. Com a partição remontada com nosuid, faça o teste novamente e veja que as shells continuam com o Suid bit ativado. Entretanto, quando forem executadas, não vão mais possibilitar o acesso no nível de root.

Outra opção interessante de montagem que pode ser usada é o noexec, que impossibilita a execução de qualquer binário ou arquivo executável dentro da partição na qual essa opção está ativada. Essa opção pode ser aplicada a todos os diretórios que contenham binários necessários às funções do sistema. Um clássico exemplo é aplicá-la a diretórios como “/home” e “/tmp”. Crackers podem se aproveitar do diretório “/tmp”, onde por padrão qualquer usuário pode escrever para introduzir backdoors ou qualquer outro programa malicioso para tentar ter acesso completo a um sistema comprometido.

Para melhor entendimento, pode-se fazer o mesmo o que feito com o nosuid. Basta remontar a partição com a opção noexec e tentar executar umas das shells que foram copiadas para o “/home”.

```
# mount -o remount,rw,noexec /home  
  
# mount
```



```
# ./sh
```

Essa simulação mostra que não é possível executar a shell. Na tentativa de executá-la, recebemos a mensagem de “permissão negada”, ou seja, nessa partição, nada de execução.

A terceira opção que podemos usar é a noatime, que não é especificamente uma opção para aumentar a segurança, mas sim para melhorar a performance, pois faz com que o kernel do Linux execute uma rotina a menos quando o noatime está definido e destinado à atualização do tempo de acesso de arquivo. Veja um exemplo para entender como funciona:

O comando *stat*, usado para verificar informações de metadados de sistema de arquivos de um arquivo:

```
# stat /etc/passwd
```

A saída desse comando nos retorna informações importantes, nas quais destacam-se as informações do Access (atime), do Modify (mtime) e do Change (ctime).

São criados dois arquivos, um no diretório “/root” e outro no “/tmp”, assumindo que o “/root” e o “/tmp” estão em partições diferentes.

```
# touch /root/teste1
```

Depois da criação dos dois arquivos, são consultadas as informações de metadados, mas com foco nas informações do Access, do Modify e do Change.

```
# stat /root/teste1
```

```
# stat /tmp/teste2
```

Agora que as informações já foram coletas, basta fazer os testes e ver o que será mudado. O próximo procedimento consiste em visualizar o conteúdo dos arquivos com o comando *cat*, supondo que eles tenham algum conteúdo, e logo em seguida vamos verificar novamente o status dos arquivos.

```
# cat /root/teste1
```

```
# cat /tmp/teste2
```

```
# stat /root/teste1
```

```
# stat /tmp/teste2
```

Olhando com atenção, será verificado que o tempo do Access (atime) dos arquivos estão diferentes, ou seja, quando foi visualizado o conteúdo dos arquivos, evidentemente foi feito um acesso a ele, mas somente de leitura do seu conteúdo. Por consequência, o seu tempo de acesso (Access) foi modificado.

O experimento ilustra um teste de modificação de permissão com o comando *chmod* e a consulta das informações de metadados com o comando *stat*, para verificar como o sistema guarda esse tipo de informação.

```
# chmod 777 /root/teste1
```

```
# chmod 777 /tmp/teste2
```

```
# stat /root/teste1  
# stat /tmp/teste2
```

É percebido que o tempo Access não foi modificado, mas o Change (ctime) foi alterado. Assim sendo, quando for mudada uma informação de metadados, como, por exemplo, uma permissão de um arquivo, o Change será mudado, registrando a última data e a hora da mudança.

Outro experimento para avaliar o que é modificado ao inserir um conteúdo nesses arquivos, ou seja, que tipo de metadados são modificados considerando as informações do experimento anterior.

```
# echo abc > /root/teste1  
# echo abc > /tmp/teste2  
# stat /root/teste1  
# stat /tmp/teste2
```

É observado que o Change foi alterado novamente, mas não alterado sozinho: o Modify (mtime) também foi, pois agora não ocorreu só uma mudança de metadados (no caso o tamanho), mas uma alteração no conteúdo do arquivo.

Com esses experimentos foi possível exemplificar a diferença entre o Access, o Modify e o Change. Agora já conseguimos fazer a alteração na partição e ver o que será melhorado.

Para essa exemplificação, a partição vinculada ao diretório “/tmp”, que será usada para demonstrar que basta remontar a partição com a opção noatime para desligar esse controle do sistema.

```
# mount -o remount,rw,noatime /tmp  
# mount
```

Com a partição remontada com a opção noatime, deve-se listar o conteúdo dos arquivos e em seguida verificar o que foi modificado em relação às informações de metadados dos arquivos.

```
# cat /root/teste1  
# cat /tmp/teste2  
# stat /root/teste1  
# stat /tmp/teste2
```

Será observado que o Access (atime) do arquivo *teste2* dentro do “/tmp” não será modificado, justamente por causa da opção noatime, que foi parametrizada na remontagem da partição.

Isso pode ser muito útil para diretórios de logs, onde o acesso é feito constantemente pelo sistema, uma vez que arquivos de logs são atualizados constantemente. Dessa forma, com “noatime” parametrizado em uma partição, o kernel deixa de executar uma rotina de atualização dos metadados de tempo de acesso, o que ajuda na performance do sistema. Outro ponto relevante é que o tempo de Access (atime) de um arquivo em uma partição com



`noatime` sempre será o primeiro tempo registrado, ou seja, a data de acesso no momento da criação do arquivo. Nesse contexto, mesmo com a mudança do tempo de modificação (`mtime`), o administrador terá o tempo de Access (`atime`) como data efetiva da criação, e o tempo de modificação (`mtime`) como tempo da última modificação.

A tabela 1.1 é um bom exemplo da maneira com a qual um administrador pode planejar a aplicação dessas opções especiais de montagem separando de forma estratégica diretórios importantes com regras de montagens diferentes.

Ponto de Montagem	<code>nosuid</code>	<code>nodev</code>	<code>noexec</code>	<code>noatime</code>
/boot	X	-	-	-
/	-	-	-	-
/home	X	X	X	-
/usr	X	X	-	-
/tmp	X	X	X	-
/var	X	X	X	-
/var/log	X	X	X	X
/var/spool/squid	X	X	X	X
/var/personal	X	X	X	X



Saiba mais

Essas não são as únicas opções que o comando `mount` pode oferecer. Para ver todas as opções possíveis, consulte o man do comando `mount`.

Tabela 1.1
Opções para as partções do arquivo `/etc/fstab`.

Segue um exemplo de como essa tabela ficaria no `/etc/fstab`.

# vi /etc/fstab					
/dev/hda1	/boot	ext3	defaults,nosuid	0	2
/dev/hda3	/	ext3	defaults	0	1
/dev/hda4	/home	ext3	defaults,nosuid,noexec	0	2
/dev/hda5	/usr	ext3	defaults,nosuid	0	2
/dev/hda6	/tmp	ext3	defaults,nosuid,noexec	0	2
/dev/hda7	/var	ext3	defaults,nosuid,noexec	0	2
/dev/hda8	/var/log	ext3	defaults,nosuid,noexec,noatime	0	2
/dev/hda9	/var/spool/squid	ext3	defaults,nosuid,noexec,noatime	0	2
/dev/hda10	/var/personal	ext3	defaults,nosuid,noexec,noatime	0	2
/dev/hda2	none	swap	sw	0	2
/dev/hdb	/media/cdrom0	iso9660	ro,user,noauto	0	0
/dev/fd0	/media/floppy0	auto	rw,user,noauto	0	0

Tabela 1.2
Como a tabela anterior ficaria no `/etc/fstab`.



Para que as regras de montagem fiquem definitivas no sistema, devem ser feitos ajustes no “/etc/fstab”. Para validar, é interessante que se reinicie o sistema, para que todas as modificações entrem em vigor e tenhamos a certeza de que nenhum erro foi cometido durante a parametrização do “/etc/fstab”. Isso, é claro, se as regras de montagem não são efetuadas durante a instalação.

Com essas opções ativadas, é importante lembrar que podemos ter um pequeno problema quando formos instalar um novo pacote com o apt-get ou dpkg. Esses utilitários, em muitos casos, executam e gravam informações nos diretórios “/var” e “/tmp” e, caso estejam parametrizados com a opção noexec, nada poderá ser instalado corretamente, gerando erros na gestão de pacotes. Então, podemos habilitar as regras de montagem no momento em que formos colocar o servidor em produção. Uma alternativa é criar um script bem simples com os comandos para remontar essas partições e poder instalar os pacotes.

```
# vi /root/noexec#!/bin/bash

case $1 in
    start)
        mount -o remount,rw,noexec /var
        mount -o remount,rw,noexec /tmp

        mount

        echo "Partições SEM permissão de execução"
;;
    stop)
        mount -o remount,rw,exec /var
        mount -o remount,rw,exec /tmp

        mount

        echo "Partições COM permissão de execução"
;;
*) echo "erro use $0 {start|stop}"
exit 0
;;

```



```
esac
```

```
exit 1
```

Para ativar a regra de noexec na partição:

```
# ./noexec start
```

O script vai deixar as partições sem permissão de execução, o que será útil quando formos instalar os outros pacotes. Para voltar à partição a noexec, basta digitar:

```
# ./noexec stop
```

O script vai permitir novamente que possa ser executado algo dentro dessas partições, possibilitando a instalação de novos pacotes. Para melhorar, pode-se copiar esse script para um diretório do PATH do root. Por exemplo, o "/sbin". Dessa maneira, será possível executar o script de qualquer diretório do sistema.

Segurança no terminal

Em muitos momentos, um administrador pode inicialmente concentrar seus esforços para mitigar a possibilidade de ameaças remotas. Mas se o esforço for exclusivo para esse ponto de vista, deve-se assumir que o todo não é mitigado, pois servidores que não estão conectados diretamente à internet também estão correm riscos.

Existem vários tipos diferentes de vulnerabilidades que possibilitam ameaças, um exemplo é a possibilidade de um **cracker** conseguir passar pela segurança de perímetro definida por firewalls através de um ataque a uma vulnerabilidade de um navegador (ataque do tipo side client) e, através da estação de trabalho comprometida, chegar até outros servidores.

Cracker

Pessoa que faz a quebra (ou cracking) de um sistema de segurança, de forma ilegal ou sem ética.

Outra preocupação importante é com a ameaça interna. Por exemplo, quando um funcionário mal-intencionado tem acesso local (físico) a um servidor. Isso pode acontecer durante a ausência do administrador que, ao sair de sua sala, se esquece e deixa o terminal do servidor com o login ativo como usuário de root.

Para esse tipo de situação, pode-se aplicar alguns procedimentos para atribuir um grau de "segurança física" ao servidor, sejam eles destinados a serviços que estarão disponíveis na internet ou dedicados à rede local. Mas em momento algum parametrizações no sistema podem substituir um aparato de segurança física, como uma sala de CPD/Datacenter com controle biométrico, apenas vão agregar mais segurança ao todo.

Desabilitar o uso de 'CTRL+ALT+DEL'

Deixar o "CTRL+ALT+DEL" desabilitado no sistema Linux pode ser uma boa ideia, pois não permitirá que alguém pressione essa sequência de teclas e faça com que o servidor reinicie. Isso é bom principalmente quando o servidor Linux está no mesmo armário que um servidor com o sistema Windows. Assim, evitamos que em algum momento alguém pressione "CTRL+ALT+DEL" no teclado de um servidor Linux e cause uma reinicialização desnecessária do sistema.

Em algumas distribuições Linux que ainda usam o /etc/inittab, essa parametrização pode ser feita editando o /etc/inittab e comentando a linha a seguir:

```
# ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
```



Pode-se também mapear as teclas para executar um outro comando qualquer, por exemplo:

```
# ca:12345:ctrlaltdel:/bin/echo "Esse recurso foi desabilitado"
```

Depois das alterações, deve-se ativar as mudanças realizadas no arquivo */etc/inittab*. Isso é feito com o comando:

```
# init q
```

No entanto, existem exceções: um bom exemplo são as versões mais recentes do Ubuntu, que embora seja uma distribuição Like Debian, possui configuração diferente para esse procedimento. É necessário editar o arquivo `# vi /etc/init/control-alt-delete.conf`. Comente a linha a seguir:

```
#exec shutdown -r now "Control-Alt-Delete pressed"
```

Para essa mudança ser efetivada, execute este comando:

```
# initctl reload-configuration
```

Exercício de fixação 2 ‘CTRL+ALT+DEL’

Qual a principal vantagem de desabilitarmos o uso do “CTRL+ALT+DEL”?

Limitar o uso dos terminais texto

Dependendo do caso, na instalação de um servidor pode ser interessante deixar o login habilitado em todos os terminais texto, por exemplo, para poder parametrizar ou impedir o login nos terminais 4, 5 e 6.

Em distribuições Linux que usam o */etc/inittab*, esse procedimento é realizado comentando as linhas do respectivos terminais que desejamos desabilitar.

```
# vi /etc/inittab  
  
#4:23:respawn:/sbin/getty 38400 tty4  
#5:23:respawn:/sbin/getty 38400 tty5  
#6:23:respawn:/sbin/getty 38400 tty6
```

Depois das alterações, precisamos atualizar novamente o arquivo */etc/inittab*.

```
# init q
```

Para as outras distribuições que não possuem o */etc/inittab*, possivelmente o arquivo que permite esse tipo de parametrização é o */etc/default/console-setup*, onde deve-se comentar as seguintes linhas:

```
ACTIVE_CONSOLES="/dev/tty[1-6]"
```

Troque por esta, que vai liberar apenas 1 tty modo texto. Não é preciso configurar para o modo gráfico, pois automaticamente será feita a utilização da próxima. Nesse caso, o modo X pode ficar na tty2 ou na tty7, mesmo que estejam desabilitadas.

```
ACTIVE_CONSOLES="/dev/tty[1]"
```

Feito isso, é hora de editar as tty para impedir a utilização. O arquivo está em `/etc/init/tty1.conf` até `tty6.conf`, lembrando que não é possível editar todas, pois uma delas terá de estar habilitada para ser usada.

Bloquear o terminal com a variável TMOUT

Quando usamos a shell bash, temos uma variável de ambiente entre tantas outras que normalmente não vêm parametrizadas por padrão. Essa variável é a TMOUT, que controla em quanto tempo o terminal fica aberto sem atividade. Ela pode ser parametrizada com o comando `exporte` manualmente ou via terminal. Para fins de teste, defina um tempo, lembrando que essa variável é parametrizada em segundos.

```
# TMOUT=15
```

Definido o valor 15, ela foi definida para fechar o terminal após 15 segundos de ociosidade. Logo, se o terminal não for utilizado durante 15 segundos, ele será fechado e o usuário, obrigado a efetuar o processo de login novamente.

Para definir o valor padrão da variável TMOUT, pode-se defini-la dentro do arquivo global de variáveis do sistema, o arquivo `/etc/profile` ou no `.bashrc` do "home" usuário. Caso a parametrização da variável TMOUT seja feita no TMOUT, será exportada automaticamente toda vez que o sistema for iniciado – ou se for parametrizada no `.bashrc` do "home" do usuário, a variável será exportada automaticamente quando o usuário executar o login, respectivamente. Dessa maneira, com a variável TMOUT parametrizada, o sistema voltará para a tela de login se nenhum comando for executado no intervalo pré-estabelecido.

Exemplificando o `/etc/profile`:

```
# echo "export TMOUT=300" >> /etc/profile
```

Exemplificando o `/root/.bashrc`:

```
# echo "export TMOUT=300" >> /root/.bashrc
```

Caso a opção seja pelo `.bashrc`, é interessante também parametrizar o `.bashrc` do `/etc/skel`, para que os usuários que forem criados no futuro também tenham a variável TMOUT definida.

Exemplificando o `/etc/skel/.bashrc`:

```
# echo "export TMOUT=300" >> /etc/skel/.bashrc
```

Bloquear o terminal com o programa Vlock

A instalação do pacote `vlock` para complementar a segurança de terminais é interessante, pois mesmo com o conceito de temporização definido na variável `tmout` para fechamento automático, em alguns momentos o administrador pode querer fechar os terminais manualmente. Nesses casos, o `vlock` pode ser uma ferramenta interessante.

```
# apt-get install vlock
```

Boas práticas recomendadas na ABNT NBR ISO/IEC 27002:2008: a limitação de tempo imposta pela ativação da variável TMOUT possibilita conformidade com o item 11.5.5, embora não seja explicitado o acesso a terminal.

Para travar todos os terminais, digite:

```
# vlock -a
```

O vlock executado com o parâmetro -a bloqueia o uso de todos os terminais. Os terminais só serão liberados mediante autenticação com a senha do usuário que os travou. Assim sendo, deve-se assumir que pequenos detalhes em configurações, como uma variável de ambiente da shell (tmout) e um simples programa, podem proporcionar grande ajuda para manter um servidor seguro, inclusive com controles de segurança para todos os terminais protegidos. Mesmo assumindo que a segurança local está também vinculada a controles de segurança física.

Gerenciamento de privilégios

Deve-se ter em mente que o usuário root é o usuário mais conhecido em sistemas *nix (o termo *nix aborda os sistemas baseados em Unix). Quando algum cracker ou funcionário mal-intencionado (insider) tentar obter acesso ao sistema, seu foco em muitos momentos vai ser conseguir a senha do usuário root – se ele tiver sorte e consegui-la, terá acesso ao sistema. Como evitar isso?

Devemos adotar alguns procedimentos para evitar que o usuário root tenha acesso direto ao sistema. A alternativa é obrigar os administradores a utilizar seus usuários comuns para efetuar o login e depois usarem o *sudo* ou o *su* para entrar em uma sessão de root.

Mas é interessante não só limitar o usuário root: é recomendável impor limites também para os usuários comuns, permitindo que somente o(s) usuário(s) do(s) administrador(es) possa(m) ter um pouco mais de privilégios. Assim sendo, não cria-se uma janela de oportunidade para que um cracker possa se aproveitar de um usuário comum facilmente.

Bloquear o login do root nos terminais texto

Não é recomendável deixar o login como root habilitado para os terminais texto. O ideal é bloquear o login do root em todos os terminais texto, focando o procedimento de login para ser iniciado com um usuário comum e, quando for necessária a realização de alguma tarefa administrativa, usar a conta do usuário root usando o comando *su*.

Criação do usuário comum:

```
# adduser tuxrnp
```

Uma das maneiras pelas quais podemos bloquear o login do root é editar o arquivo */etc/securetty* e comentar as linhas referentes aos terminais que queremos desabilitar para o root. Comentar as seguintes linhas:

```
# vi /etc/securetty
```

```
#tty1  
#tty2  
#tty3  
#tty4  
#tty5  
#tty6
```

```
#tty7  
#tty8  
#tty9  
#tty10  
#tty11  
#tty12
```

Agora que as linhas dos terminais estão comentadas, podemos tentar efetuar o processo de login em outro terminal com o usuário root. Será percebido que o usuário root não tem mais permissão para efetuar um login direto via um terminal local.

Determinar datas de expiração para contas de usuários

Se o servidor for alojar grande quantidade de usuários, pode-se especificar datas de expiração para essas contas. Dessa maneira, se um usuário usa a sua conta todos os dias, quando sua senha expirar ele poderá modificá-la e renovar a conta. Mas, se por algum motivo uma conta de um usuário não está sendo usada, ela será desativada no tempo escondido, dando-nos a segurança de que essa conta não será usada com más intenções.

Utiliza-se o comando *chage* para modificar os parâmetros do arquivo */etc/shadow*, onde são feitos esses controles.

Primeiro, é interessante visualizar a configuração padrão de uma conta de um usuário, por exemplo, do usuário teste.

```
# chage -l teste
```

O uso do comando *chage* possibilita fazer algumas modificações:

```
# chage -M 30 -W 5 -I 2 teste  
# chage -l teste
```

Onde:

- **-M:** é o tempo máximo de validade da conta;
- **-W:** é o tempo de aviso;
- **-I:** é o tempo antes de a conta ser desativada.

Com essa configuração, está sendo parametrizado que a conta do usuário é válida por 30 dias. Quando estiverem faltando 5 dias para a conta expirar, o usuário começará a ser avisado quando realizar o procedimento de login. Depois desses 30 dias, o usuário terá 2 dias para poder modificar a senha e para que a conta seja renovada. Enquanto ele não fizer isso, não conseguirá efetuar um login no sistema, mas se nesses 2 dias o usuário não modificar a senha, a conta será desativada e somente o usuário root poderá reativá-la.

Em suma, essa é uma forma de controlar senhas e contas de usuários que têm contas com shell válida. Normalmente, esse tipo de controle é aplicável sem muito esforço em um contexto onde o servidor é dedicado a um determinado serviço e existem atividades de login exclusivamente administrativas.



Saiba mais

Essa é uma das maneiras pelas quais se pode bloquear o acesso do root pelos terminais, mas existem outras possibilidades, que serão abordadas em outros tópicos.

Remover shells válidas de usuários que não precisam delas

Usuários que não usam shell não têm a necessidade de tê-la, ou seja, todo aquele usuário – que é de sistema ou é cliente de serviço –, com exceção dos usuários que utilizam serviços como TELNET ou SSH, e também que realizam login no sistema através do terminal local (tty).

Para remover as shells válidas de todos os usuários que não precisam de uma shell válida, é recomendável primeiro criar um ou mais usuários estratégicos que pertencem ao grupo de administradores. Os usuários pertencentes a esse grupo terão permissão para realizar o login e posteriormente usar o su para se tornarem root, uma vez que isso esteja devidamente parametrizado.

Nessa simulação, primeiro devemos criar o grupo “administradores”. Logo em seguida, criaremos um usuário administrador e o adicionaremos ao grupo.

```
# groupadd administradores  
# adduser toor  
# gpasswd -a toor administradores
```

Visualize o arquivo */etc/passwd* e identifique se todos os usuários de sistema também possuem shells válidas.

```
# cat /etc/passwd | less
```

Em um segundo momento, pode-se criar um Script shell que automatiza a remoção de todas as shells válidas de todos os usuários, exceto daqueles que forem especificados como parte do grupo de administradores e o usuário “root”:

```
# cd /root  
# vi invalidos.sh#!/bin/bash  
  
for USER in $(cat /etc/passwd| cut -f 1 -d ":" | grep -v root | grep -v toor | grep -v tuxrnp)  
  
do  
  
    usermod -s /bin/false $USER  
done  
  
# chmod +x /root/inválidos.sh  
#./inválidos.sh
```

Resumindo esse script: ele executa um for que pegará somente a primeira coluna do arquivo */etc/passwd*, que são os nomes dos usuários que não estão especificados no grep -v. Isso é armazenado dentro da variável USER. Logo depois, com o comando *usermod*, esse script passa todas as shells dos usuários que estão na variável USER para /bin/false, ou seja, uma shell inválida.

Depois da execução do script, visualize novamente o arquivo `/etc/passwd` e veja como ficou.

```
# cat /etc/passwd | less
```

Repare que todas as shells dos usuários passaram a ser `/bin/false`, menos as dos usuários root, toor e tuxrnp.

Deve-se adotar a política ao criar um usuário novo, para que ele seja criado com uma shell inválida. Caso precise de uma shell válida, depois basta mudar isso no arquivo `/etc/passwd`. Para fazer isso, é preciso editar os seguintes arquivos:

No Debian:

```
# vi /etc/adduser.conf
```

Nesse arquivo, podemos mudar a variável DSHELL para uma shell inválida.

```
D SHELL=/bin/false
```

No Red Hat:

```
# vi /etc/default/useradd
```

Dessa maneira, os usuários serão criados por padrão com a shell `/bin/false`, uma shell inválida.

Para finalizar essa tarefa, deve-se visualizar o arquivo `/etc/shells`, um arquivo texto que contém os nomes de caminho completos das shells de login válidos. Esse arquivo é consultado pelo comando `chsh` (utilizado para mudar a shell de um usuário) e fica disponível para consulta por outros programas.

```
# cat /etc/shells
```

É fortemente recomendável ficar atento, porque há programas que consultam esse arquivo para descobrir se um usuário é um usuário normal. Por exemplo, daemons (processos que são executados em background) de ftp tradicionalmente desaprovam acesso aos usuários com interpretadores de comando não incluídos nesse arquivo, todavia isso tem de ser avaliado, pois não é uma regra absoluta para todas as implementações de servidores FTP disponíveis.

Instalação de pacotes específicos

Embora um sysadmin deva ter em mente que prepara um servidor utilizando as boas práticas de segurança, com o objetivo de que este não seja vítima de algum tipo de incidente de segurança, ele também deve assumir que, se não existe segurança 100%, existe sempre um risco assumido ainda que não seja mensurável. Diante desse contexto, é recomendável que durante o processo de hardening seja feita a instalação de aplicações que poderão auxiliá-lo em execução de Auditorias futuras ou em uma Resposta a Incidente de Segurança.

```
# apt-get install -y rkhunter chkrootkit unhide debsecan mtr-tiny  
apticron htop vim vlock whowatch
```

Algumas ferramentas:

- ▣ **rkhunter**: serve para fazer a identificação de rootkit e apoio à Resposta Incidente;
- ▣ **chkrootkit**: identifica o rootkit e dá apoio à Resposta Incidente;
- ▣ **unhide**: ferramenta para identificação de rootkit e apoio à Resposta Incidente;

- ▣ **lsof**: útil para a gestão de processos;
- ▣ **htop**: arrojada ferramenta para a gestão de processos;
- ▣ **vlock**: ferramenta de travamento de terminal texto;
- ▣ **debsecan**: para o suporte à gestão de atualizações de segurança;
- ▣ **apticron**: útil para o suporte à gestão de atualizações de segurança;
- ▣ **whowatch**: ferramenta interessante para a gestão de uso de terminais;
- ▣ **vim**: editor de texto.

Um ponto crítico para servidores Linux são os “códigos maliciosos” do tipo rootkit, uma categoria de malware usada por invasores. Um Host IDS, que é o tema tratado no capítulo 3, em muitos casos poderá ser uma forma de identificação da presença de um rootkit no sistema, mas o uso de ferramentas como chkrootkit e rkhunter é recomendável, pois possibilitam auditar o sistema em busca de possíveis rootkits, que são necessários em um servidor Linux.

Durante a instalação de pacotes, deve-se ativar o exec nas partições /var e /tmp. Essa tarefa pode ser feita com o uso do script noexec. Em alguns casos, pode ser interessante somente ativar os controles adicionais de montagem no momento em que o servidor estará sendo promovido a produção, para que durante o processo de instalação, homologação e testes, esses controles não sejam um problema.

É recomendável também a atualização de todo o sistema, não somente dos pacotes que estão com notificação de vulnerabilidade, pois antes de instalar e configurar os serviços para os quais se destina o servidor, é o melhor momento para atualização completa do sistema.

```
# apt-get install update && apt-get install -y upgrade
```

Segurança do Grub

Habilitar senha Grub é uma tarefa recomendada, que pode agregar mais valor à segurança física, mas é óbvio que esse tipo de procedimento não substitui controles tradicionais de segurança física.

Nesse momento, o Grub está em sua terceira versão, mas ainda é comum ter distribuições Linux que usam a segunda versão.

Na versão 2, a parametrização se concentra em único arquivo de configuração, que é */boot/grub/grub.cfg*. A forma de configurar uma senha e outros parâmetros no GRUB mudou na versão 3 de tal forma que a configuração/parametrização se divide em vários arquivos do diretório “*/etc/default/grub*”.

Mudar ou incluir a variável *timeout=0* no arquivo */etc/default/grub*, para não mostrar menu no momento da inicialização.

Gerar hash da senha SHA512 por meio do comando:

```
#grub-mkpasswd-pbkdf2
Enter password:
Reenter password:
Your PBKDF2 is grub.pbkdf2.sha512.10000.706A070CD168B759801D2790C6D4
8D5C3842B9165CF08600918CD9A496B6BFF9CD9BB8F7C99DEC431DF3AD0D466709
ECE041FC00C5C1B58F00A879E0322959B7.6FC5058001DFFC1CD6B35F9A5DA66ED6C
```

```
8745E4999E064E712C9BF302E8F2547CD0B591C33A340F229FD79D2252E23CFC4141  
0C9A3300537E54C9CE6F7008100
```

Adicionando o código de proteção a seguir no final do arquivo */etc/grub.d/00_header*. Código:

```
cat << EOF  
  
set superusers="testuser"  
  
password_pbkdf2 testuser grub.pbkdf2.sha512.10000.706A070CD168B75980  
1D2790C6D48D5C3842B9165CF08600918CD9A496B6BFF9CD9BB8F7C99DEC431DF  
3AD0D466709ECE041FC00C5C1B58F00A879E0322959B7.6FC5058001DFFC1CD6B35F  
9A5DA66ED6C8745E4999E064E712C9BF302E8F2547CD0B591C33A340F229FD79D22  
52E23CFC41410C9A3300537E54C9CE6F7008100  
  
EOF
```

O testuser é o usuário que está sendo definido na configuração do Grub, combinando a senha gerada pelo comando *grub-mkpasswd-pbkdf2*.

Para efetivar as configurações, deve-se executar:

```
# update-grub
```

Dessa forma, o arquivo */boot/grub/grub.cfg* será recompilado e será adicionado à proteção.

Proteja o arquivo */boot/grub/grub.cfg*:

```
chmod 600 /boot/grub/grub.cfg  
chattr +i /boot/grub/grub.cfg
```

Proteja o arquivo */etc/grub.d/00_header*:

```
chmod 600 /etc/grub.d/00_header  
chattr +i /etc/grub.d/00_header
```

Habilitar senha nas versões antigas do Grub (versões 1 e 2)

O arquivo de Configuração do GRUB, onde é possível parametrizar senha, é o arquivo */boot/grub/menu.lst*, utilizando o parâmetro “password --md5 <senha>”. A senha é criada com o auxílio do comando *md5crypt* via shell do grub, ou via o comando *grub-md5-crypt* na shell do sistema. Além da parametrização da senha, é recomendável também alterar a variável *timeout=0*, para não mostrar menu do grub na inicialização, exemplo:

```
timeout=0  
  
password --md5 <senha md5>
```

Já na versão GRUB2, a senha é criada com o auxílio do comando *grub-mkpasswd-pbkdf2*, e a parametrização da senha no arquivo *00_header*.

Proteger o arquivo */boot/grub/menu.lst*.

```
chmod 600 /boot/grub/menu.lst  
chattr +i /boot/grub/menu.lst
```



Roteiro de Atividades 1

Na prática de reconhecimento das ferramentas aptinfo, debsecan e vlock, será necessário que o aluno use a máquina virtual preparada como Virtuabox. Dessa forma, será possível aplicar todos os conhecimentos (técnicas e ferramentas) de hardening propostos até o momento.

Atividade 1.1 – Hardening Linux – Apticron

Verifique se o apticron está instalado, revise a sua configuração e teste. Descreva os comandos executados e os resultados.

Atividade 1.2 – Hardening Linux – Debsecan

Verifique se o debsecan está instalado, revise a sua configuração e teste. Atualize todos os pacotes que já possuem correções disponíveis e atualize a whitelist. Descreva os comandos executados e os resultados.

Atividade 1.3 – Hardening Linux – Debsecan via cron

Configure o debsecan para funcionar via cron, enviando somente informações e novas atualizações pendentes. Descreva os comandos executados e os resultados.

Atividade 1.4 – Hardening Linux – atualização

Realize a atualização completa do seu sistema Linux. Descreva os comandos executados e os resultados.

Atividade 1.5 – Hardening Linux – pacotes desnecessários

Verifique o permissionamento das ferramentas consideradas críticas, deixando somente execução permitida para o root (exemplo: nmap, netcat, shred, tcpdump, telnet etc.). Descreva os comandos executados e os resultados.



Atividade 1.6 – Hardening Linux – vlock

Avalie o funcionamento da ferramenta vlock e a configuração dos terminais. Descreva os comandos executados e os resultados.

Atividade 1.7 – Hardening Linux – regras de montagem

Visualize as regras de montagens ativas. Descreva os comandos executados e os resultados.

Atividade 1.8 – Hardening Linux – Chkrootkit rkHunter

Execute o comando `chkrootkit` e o comando `rkhunter` para avaliar o sistema, sendo que, no caso do `rkhunter`, utilize a opção de geração de hash para melhor para posterior auditoria.

Atividade 1.9 – Hardening Linux – Baseline

Avalie o baseline proposto (planilha xls fornecida como material de apoio) e apresente suas observações. Como deve ser configurado um baseline para a sua organização?



2

Hardening em sistema Linux – controles de segurança para contas de usuários

objetivos

Compreender e usar módulos PAM; realizar auditoria de senhas de usuários; criar definição de uso do sistema de arquivos; delegar tarefas administrativas via *sudo*; criar permissionamentos mais elaborados e amplos com ACL.

conceitos

Utilização do módulo PAM; segurança de terminal; auditoria de senhas; quota de sistema de arquivos; comandos via *sudo*; permissionamento via ACL.

Introdução

Módulos PAM.

- Sinalizadores de Controle.
- Procura por senhas fracas.
- Utilização do John the Ripper.
- Utilização de Quota.
- Utilização de ACL – Posix 1e.
- Comando *sudo* (Superuser Do).
- Prevenção “Escape Shell”.

A criação de usuários e grupos em sistemas Linux é importante para definir que recursos podem ser acessados por quais usuários e/ou grupos e ainda permite ao Sistema Operacional gerenciar a execução dos processos de cada usuário de forma adequada.

Manter o controle do que seus usuários estão fazendo é uma parte fundamental para o gerenciamento de usuários. O Sistema Operacional Linux possui vários módulos que ampliam a segurança e a gerência sobre seus utilizadores. Administradores sem experiência em segurança normalmente preparam seus servidores com uma instalação básica e, depois que suas aplicações estão disponíveis, nenhum procedimento é feito para manter a integridade do sistema e estabelecer sua trilha de auditoria.

Exercício de nivelamento 1



Controles de segurança para usuários

1. Como você avalia a necessidade de uso de controles específicos para recursos de autenticação em sua organização?

2. Na política de segurança de sua empresa já existem definições para a criação de controles no uso de terminais ou mesmo quanto ao tratamento do tempo de ociosidade de um terminal?

3. Como você avalia na sua organização o controle do uso do recurso de armazenamento? Existem critérios de quotas definidos?

4. Em sua empresa, existe qual critério de auditoria de senhas para que seja possível identificar senhas fracas?

Controles de autenticação com a utilização do PAM

Módulos de Autenticação Plugáveis (PAM) são uma interface de módulo que corresponde ao tipo de autorização baseada em um módulo. Um módulo PAM pode usar apenas uma ou todas as quatro interfaces possíveis. Essas interfaces são: account, auth, password e session.

Cada uma delas será especificada no arquivo de configuração de um serviço, se for apropriada para o módulo e o administrador desejar usar essa interface:

- **account:** essa interface verifica se uma conta tem autorização para usar o sistema, o que pode significar verificar se existe, se está vencida ou se tem acesso permitido em determinado horário, vinculada a um grupo ou através de determinado serviço;
- **auth:** interface que serve para autenticar usuários. Isso pode ocorrer através de senhas, banco de dados ou outro mecanismo. Além disso, esses módulos também têm permissão para definir credenciais, como membros de grupo ou mesmos baseados em tokens Kerberos;
- **password:** a interface password é usada para verificar e definir a autenticação de senha ou o critério de definição de senha;
- **session:** é responsável pela configuração e gerenciamento de sessões de usuário. Pode incluir tarefas de organização, como montar diretórios, criar arquivos etc.

Com esse recurso, é possível auxiliar os métodos de autenticação tradicionais, possibilitando que executem novas funções. Um bom exemplo é o programa de login, que se baseia no `/etc/passwd` e no `/etc/shadow` do sistema.



Saiba mais

Além de ser parte da maioria das distribuições Linux, esse recurso também é encontrado em outros sistemas, como Solaris.



Mas será que esses dois arquivos possibilitariam a criação de um controle definindo o horário em que um determinado usuário pode realizar um login? Ou o número de terminais simultâneos que um usuário pode fazer o login?

A resposta para essas perguntas é: não, mas o PAM pode fazer isso com os seus módulos que são plugados ao programa de login. O PAM não faz só isso: ele possui muitos módulos com funções bem diferentes, e não é somente ao programa de login que se aplica; o PAM já tem suporte a muitos programas. Dessa forma, o uso do PAM em um processo de Hardening deve sempre ser considerado.

Para certifica-se se um determinado programa tem suporte a PAM, é possível consultar se este foi compilado com suporte a libpam.so.

```
# ldd /bin/su  
# ldd /bin/login  
# ldd /usr/sbin/sshd
```

Como é possível saber se um determinado programa tem suporte ao PAM? Outra forma é acessar o diretório */etc/pam.d*. Dentro desse diretório, deve existir um arquivo com o nome do programa. No entanto, essa não é uma regra absoluta.

Por exemplo, vejamos se existe um arquivo chamado *login* dentro do */etc/pam.d*.

```
# ls -l /etc/pam.d
```

Veja como exemplo o arquivo *login*. Existem muitos outros também. Alguns serão abordados mais à frente.

Sinalizadores de controle

Para cada interface, o arquivo de configuração especifica um sinalizador de controle, que determina o que o PAM fará em seguida, com base no resultado da verificação realizada. Existem quatro sinalizadores de controle: optional, required, requisite e sufficient:

- ▣ **optional**: os módulos opcionais não afetam o sucesso nem a falha da autenticação, a menos que não haja outros módulos para determinada interface;
- ▣ **required**: para que o usuário possa continuar, deverá ser retornado um resultado de sucesso. A notificação de usuário não ocorre até todos os módulos para uma interface estarem satisfeitos;
- ▣ **requisite**: o usuário só poderá continuar se o módulo resultar com sucesso. A notificação do usuário acontece imediatamente em caso de falha no primeiro módulo requisite ou required de uma interface;
- ▣ **sufficient**: um resultado de sucesso combinado sem falhas do módulo required ou requisite viabiliza uma boa autenticação, pressupondo que nenhum outro módulo virá a seguir. A falha de um módulo sufficient é ignorada.

Caminho do módulo

O caminho do módulo informa ao PAM a localização do módulo. Normalmente, é um nome de caminho completo, que contém o nome e a extensão do módulo, como */lib/security/pam_unix.so*. Se não for especificado um caminho, o PAM assume como padrão */lib/security* para poder localizar o módulo.

Pam_time

É uma boa prática em sistema Like Unix não deixar o login como root habilitado para conexão direta. O ideal é a execução do “login” com um usuário comum inicialmente, e depois obter acesso como root quando precisar executar alguma tarefa administrativa. Usando um conceito de autenticação em duas camadas, o que gera uma contramedida contra ataques de força bruta, destinado ao usuário root.

Foi mostrado que é possível travar o login de root em todos os terminais comentando as linhas dos terminais no arquivo */etc/securetty*. Outra forma interessante para criar um controle para a conta do usuário root é “descomentar” a linha que refere-se a pam_time.so para bloquear o login de root utilizando o conceito baseado horário de acesso.

Edite o arquivo */etc/pam.d/login* e “descomente” a seguinte linha:

```
# vi /etc/pam.d/login  
account    requisite      pam_time.so
```

Essa linha está tratando da conta dos usuários (account) com o módulo pam_time.so. Até aqui essa linha não está fazendo nada, pois a maioria dos módulos do PAM depende de um arquivo de configuração adicional.

O módulo pam_time.so trabalha em conjunto com o arquivo */etc/security/time.conf* pelo nome do módulo e do arquivo de configuração.

Para definir o horário de acesso, deve-se editar o arquivo */etc/security/time.conf* e acrescentar a seguinte linha no final do arquivo:

```
# vi /etc/security/time.conf  
login;tty*;root;!A10000-2359
```

Onde:

- ▣ **login:** indica o serviço;
- ▣ **tty*:** indica os terminais onde a política será aplicada;
- ▣ **root:** determina que a política será aplicada para o usuário root.
- ▣ **!A10000-2359:** indica o horário permitido. O caractere “!” estabelece que aquele horário não é permitido.

Sintaxe: services;ttys;users;times

Com essa configuração, está definido que o root não pode efetuar login em nenhum terminal, em nenhum dia ou horário. Dessa maneira, cria-se um controle que bloqueia a usuário root para que este não tenha acesso direto ao “login” do sistema.

Para validar o controle, basta tentar realizar um login em outro terminal como root, o que não deverá ocorrer. Todavia, essa proposta imagina que haverá um usuário definido que terá acesso ao recurso login e, a partir dele, será permitido usar o comando *su* ou *sudo* para usar a conta do root.

Pam_limit

A seguir será mostrado como definir uma política para não permitir que um usuário faça login em mais de um console, consecutivamente com a utilização da biblioteca pam_limits.so.

Primeiro é preciso habilitar o módulo que vai possibilitar a utilização desse recurso. Para isso, é necessário editar o */etc/pam.d/login* e inserir a linha a seguir:

```
# vi /etc/pam.d/login  
session required pam_limits.so
```

Com o módulo habilitado, temos de editar o arquivo de configuração adicional desse módulo. Na sequência, deve-se ir até o diretório */etc/security*, editar o arquivo *limits.conf* e inserir a seguinte linha:

```
# vi /etc/security/limits.conf  
usuário      hard   maxlogins      1
```

Com isso, é criado um controle que limita o usuário para que utilize somente um terminal. Para testar, basta executar o “login” em um terminal com o usuário e tentar efetuar login em outro simultaneamente.

Outra utilização da biblioteca *pam_limits.so* é a parametrização para limitar os processos dos usuários ou um grupo, o que é importante para a execução de um sistema estável.

Exemplificação para coibir um forkbomb:

```
rnpadmin hard nproc 300  
@systelcom hard nproc 150  
@ sysprinters soft nproc 100  
@sysop hard nproc 200
```

Nessa parametrização, ficou definido que membros de um grupo de estudantes poderão abrir 150 processos; membros do grupo sysprinters, 100; e sysop, 200 processos. Já o usuário rnpadmin poderá criar até 300 processos. Todavia, esse tipo de configuração deve ser bem avaliado, para que seja dimensionado o número de processo ideal.

Pam_wheel

Nessa exemplificação, será usado como exemplo um usuário denominado toor e um grupo denominado “administradores”, feitos para criar mais uma camada de controle com o objetivo de aumentar ainda mais a segurança na autenticação:

```
# adduser toor  
# groupadd administradores  
# usermod -G toor administradores
```

A proposta é somar ao procedimento de limitação de “logins”, já mencionado, uma política que não permita o uso do comando *su* por qualquer usuário, possibilitando o uso do *su* somente para os usuários do grupo administradores.

Essa configuração vai ser pouco diferente das outras, pois não precisa de um arquivo de configuração adicional. Basta limitar a utilização do programa *su*, com a edição do arquivo *su*, que está dentro do diretório */etc/pam.d*.

```
# vi /etc/pam.d/su  
auth required pam_wheel.so group=administradores
```

Essa configuração trata a autenticação do usuário (auth) com o módulo pam_wheel.so, utilizando o grupo administradores. Uma vez parametrizado, somente usuários que pertencem ao grupo administradores é que poderão usar o comando *su*.

Para testar, execute o “login” com o usuário *toor* em um terminal e com o *tux* em outro. Então é necessário tentar fazer o *su* com os dois usuários. Caso as configurações tenham sido feitas de forma correta, será observado que somente o usuário *toor* conseguirá, pois só ele pertence ao grupo administradores.

Por questões de segurança, toda a atividade do comando *su* deve ser monitorada. Os procedimentos a seguir servem para registrar em arquivo de log o uso do comando *su* pelos usuários.

Modifique o arquivo */etc/login.defs* e “descomente” a seguinte linha:

```
SULOG_FILE /var/log/sulog
```

Logo em seguida, é necessário criar o arquivo *sulog* em */var/log*.

```
# touch /var/log/sulog
```

Para validação, deve-se fazer um teste e usar em paralelo o comando *tail* para melhor visualização do log, podendo-se desviar a saída padrão do arquivo para ver o conteúdo do arquivo de log em tempo real, para um determinado terminal. No exemplo foi escolhido o terminal 12.

```
# tail -f /var/log/sulog > /dev/tty12 &
```

Na sequência, devemos efetuar login com o usuário *toor* e usar o comando *su* para alternar para o usuário *root*.

```
$ su
```

Feito isso, visualize os registros do log no terminal 12.

Com alguns ajustes nos arquivos de configuração do PAM, é possível limitar bastante os privilégios comuns e até mesmo limitar o próprio *root*, o que em muitas situações pode ser útil. Sendo também possível fazer essas e outras configurações utilizando o PAM, o que ajuda um sysadmin a ter grande controle e mostra que o uso do recurso PAM é uma grande ferramenta para gerenciar privilégios.

Exercício de fixação 1 PAM

1. Cite duas vulnerabilidades conhecidas nos servidores da sua organização, relacionadas à autenticação, que o recurso do PAM pode mitigar.

2. Existem definições restritivas do uso do comando *su* nos servidores de sua organização?

Procura por senhas fracas

A senha de login de um usuário é algo pessoal: nem mesmo os administradores devem saber as senhas dos usuários. Normalmente, quando as contas de usuário são cadastradas em um sistema, ou o cadastro é feito com uma senha inicial que deverá ser trocada no primeiro login efetuado pelos usuários ou, em alguns casos, o administrador solicita ao usuário que digite a senha. Entretanto, se não existem controles que definem o critério para ter uma senha forte, cria-se a oportunidade para o usuário digitar uma senha fraca, que pode ser totalmente composta por números e com poucos dígitos, palavras existentes em dicionários de bruteforce e até mesmo uma data simbólica. Essas são senhas muito fáceis de serem descobertas, às vezes nem sendo necessário um programa de bruteforce (Força Bruta) elaborado para serem quebradas.

Para realizar esse trabalho de auditoria em busca de senhas fracas, pode ser interessante usar uma ferramenta da mesma categoria e funcionalidade utilizada por um cracker. No mundo Unix, a ferramenta recomendada para essa tarefa é o John the Ripper.

Utilização do John the Ripper

O John the Ripper é uma ferramenta de bruteforce clássica do mundo Unix, que tenta descobrir as senhas do arquivo */etc/shadow* usando uma WordList (lista com senhas a serem tentadas) padrão que pode ser incrementada ou usada randomicamente.

É fato que, quando a senha do usuário for mal elaborada e estiver contida no dicionário, ou mesmo se a senha for igual ao login, será facilmente descoberta por essa ferramenta.

É importante destacar que uma ferramenta como o John the Ripper não é o tipo de ferramenta que se deve deixar instalada em um servidor. Isso seria o mesmo que fechar a porta de casa, mas deixar um pé-de-cabra embaixo do tapete. É recomendável sempre colocar ferramentas desse tipo em outra máquina, ou seja, a máquina do próprio administrador. Para auditar, basta copiar o */etc/shadow* do servidor para a máquina do administrador e testá-lo.

Para realizar um experimento para compreender o funcionamento do John the Ripper, crie um usuário chamado “teste”, definindo a senha como 123456. Instale o John the Ripper.

```
# adduser teste  
# apt-get install john
```

Com o John the Ripper instalado, entre no diretório */usr/share/john* e visualize o arquivo de WordList chamado *password.lst*.

```
# cd /usr/share/john  
# cat password.lst
```

É interessante, sempre que realizar uma auditoria em busca de senhas fracas, incrementar essa WordList, tornando a busca ainda mais refinada. Outra possibilidade é incrementar a WordList com outras disponíveis na internet.

Copie o */etc/shadow* do servidor para o */root* da máquina do administrador para colocar o John the Ripper em ação. Pode-se copiar o arquivo via scp (utilitário do ssh para fazer cópias remotas).

```
# scp root@IP_do_Servidor:/etc/shadow /root
```

Com o */etc/shadow*, execute o teste do John the Ripper.

```
# john /root/shadow
```

Será percebido que senhas fracas são quebradas facilmente. Todas essas senhas que o John the Ripper descobre são armazenadas em um arquivo chamado *john.pot*, no diretório */usr/share/john*.

Então, se for necessário fazer a verificação novamente, deve-se apagar esse arquivo.

```
# rm -f john.pot
```

```
# john /root/shadow
```

Com esse experimento é que percebemos que senhas fracas em um servidor são um grande problema, mas com uma simples ferramenta pode-se descobrir as senhas fracas. Devemos fazer esse tipo verificação de tempos em tempos, caso os usuários mudem as senhas com frequência. Se deixarmos os usuários com senhas fracas, teremos uma grande porta de entrada para um cracker.

Utilização de quota

O recurso de quota é uma ferramenta importante para definir controle com o foco do uso do recurso. Nesse caso, o sistema de arquivos. Dessa forma, usar quota é mais um valor de segurança a partir do Sistema de arquivos, sendo um deles o de controlar a utilização dos sistemas de arquivos entre todos os usuários, criando controles que impeçam um único usuário de ter poder de escrita em seu diretório pessoal ou exceder os limites físicos de espaço em um sistema de arquivos, comprometendo a sua utilização pelos outros usuários. Além disso, tal procedimento ajudará bastante a ter um backup de tamanho controlado.

Para começar a configuração do recurso de quota, devemos identificar quais os parâmetros de montagem do diretório, neste exemplo, o diretório */home*.

```
# mount
```

O recurso quota deve ser especificado para partições e não para diretórios, o que demanda a edição do arquivo */etc/fstab* (sistemas de arquivos montados) para configurar a partição que deve ter suporte, caso isso não tenha ocorrido durante a instalação. Veja o exemplo:

```
# vi /etc/fstab
/dev/hda6 /home ext3 defaults,usrquota,grpquota 0 2
```

No quarto campo da linha, temos as opções que parametrizam a forma como a partição será montada. Devemos usar o parâmetro *usrquota* para configurar quotas para usuários e *grpquota* para configuração de quotas para grupos de usuários.

Após a parametrização do */etc/fstab* é necessário entrar no ponto de montagem da partição especificada, no nosso caso, */home*, e criar os dois arquivos de controle de quota:

- **aquota.user**: gerencia quotas para os usuários;
- **aquota.group**: gerencia quotas para os grupos.

Conforme a exemplificação, o sistema Linux atualmente trabalha com o QUOTA 2, que é uma evolução do antigo sistema de quotas disponível no Linux.



Saiba mais

Quando a instalação do John the Ripper é realizada a partir de um pacote pré-compilado, o arquivo *john.pot* fica no diretório *.john*, na home do usuário.



Boas práticas recomendadas na ABNT NBR ISO/IEC 27002:2008: o uso do John the Ripper torna-se uma opção interessante no que diz respeito ao processo de auditoria de senhas. Para buscar conformidade com os itens 11.5.3 e 11.3.1, é orientado que as senhas sejam controladas através de um processo de gerenciamento formal e que os usuários sejam solicitados a seguir boas práticas de segurança da informação na seleção e no uso de senhas.



```
# apt-get install quota  
# cd /home  
# quotacheck -vagumF  
# /etc/init.d/quota stop  
# /etc/init.d/quota start
```

Os arquivos de controle de quota devem ter suas permissões modificadas de maneira que só o root tenha permissão de leitura e gravação sobre eles, ou seja, o valor 600, conforme o exemplo nos comandos:

```
# chmod 600 quota.user  
# chmod 600 quota.group
```

Após isso, devemos remontar o sistema de arquivos para que as configurações de quota para a partição entrem em vigor. Todavia, como na maioria das vezes o sistema de arquivos estará ocupado com processo vinculado à partição, é recomendável que seja reiniciado o sistema. Para tanto, devemos digitar o comando “reboot”:

```
# reboot
```

Para verificar o sistema após inicializar, devemos executar o comando *repquota* para consultar o status de quota para aquela partição ou partições.

```
# repquota -v -a -s
```

Para exemplificar o uso de conta, será demonstrado o uso do recurso com o usuário chamado “tuxrnp”, que será criado para esse propósito. Lembramos que é possível definir quota de duas maneiras, por tamanho e por quantidade de arquivos. Neste primeiro exemplo será configurada a quota por tamanho.

Criação do usuário:

```
# adduser tuxrnp1
```

Definição de quanto do sistema de arquivos o usuário “tuxrnp” poderá usar. O comando para configurar quotas é o *edquota*.

```
# edquota -u tuxrnp1
```

Serão definidas quotas para o usuário, estabelecendo o limite de 30 MB de espaço em disco e limite máximo de 40 MB.

```
Disk quotas for user tuxrnp (uid 1009):
```

Filesystem	blocks	soft	hard	inodes	soft	hard
/dev/hda6	30000	40000	0	0	0	0

Onde:

- **blocks:** número de blocos utilizados pelo usuário – 1M equivale mais ou menos a 1.000 blocos;
- **soft limit:** limite em blocos que o usuário poderá utilizar (o limite máximo é definido em hard limit);

Saiba mais

A quota do usuário será de 30 MB, mas ele terá um “bônus” de 10 MB por um determinado tempo (chamado de *grace period*), totalizando 40 MB.

- **hard limit:** limite em blocos que o usuário poderá usar – valor que nunca é ultrapassado.

Consulte o status geral dos usuários já definidos, inclusive para tuxrnp.

```
# repquota -v -a -s
```

Consulta de informações de quota do usuário tuxrnp.

```
# quota -u tuxrnp1
```

O grace period não é fixo, e pode ser mudado conforme a nossa necessidade.

```
# edquota -t
```

Exercício de fixação 2

Quota

1. Cite duas vulnerabilidades conhecidas nos equipamentos servidores da sua organização.

Para fazer um teste, efetue login em outro terminal, como usuário tuxrnp. Será exemplificado um script que pode ser chamado de “lota disco”, que aumentará o tamanho de um arquivo até estourar o limite pré-definido de quota.

```
$ echo teste > a; while true; do cat a >> b; cat b >> a; done
```

Depois da execução do script com o limite da quota estourado, basta voltar ao terminal do root e examinar o status da quota.

```
# repquota -v -a
```

Para uma segunda exemplificação, só que agora usando outro método, o da quota por quantidade, será criado outro usuário.

```
# adduser rnpteste
# adduser tuxrnp2
```

Serão definidas quotas para o usuário tuxrnp2: limite de 50 arquivos, com o máximo chegando a 60 arquivos.

```
# edquota -u tuxrnp2
```

Disk quotas for user tux2 (uid 1009):

Filesystem	blocks	soft	hard	inodes	soft	hard
/dev/hda6	0	0	0	0	50	60

-  A quota do usuário será de 50 arquivos, mas ele terá um “bônus” de 10 arquivos, independente do tamanho, por um determinado tempo.

Onde:

- **inodes:** número de inodes (arquivos utilizados pelo usuário);
- **soft limit:** número de inodes (arquivos) que o usuário poderá criar;



Saiba mais

Grace period é um período de tolerância em que o usuário, após ter estourado sua quota, receberá avisos de que ultrapassou seu soft limit, e que deve tomar alguma providência para voltar a respeitar seu limite de quota. Entretanto, é válido lembrar que esse período de tolerância não se aplica ao hard limit, pois, uma vez que o valor de hard limit seja superado, o sistema não mais permitirá gravação de arquivos, independente do valor do grace period. O grace period padrão é de 7 dias.



- **hard limit:** número máximo de inodes que o usuário poderá criar.

Consulte o status desse usuário:

```
# repquota -v -a -s
```

Consulte a quota do usuário tux2:

```
# quota -u rnpteste
# quota -u tuxrnp2
```

Para validar a configuração, basta realizar um teste com um script. O ideal é efetuar login em outro terminal como usuário tuxrnp2 e rodar o script, que pode ser chamado de script gerador-de-arquivos, para gerar entradas de arquivos até estourar o limite definido pela quota.

```
$ i=1; while true ; do touch $i; let i++; done
```

Depois que a quota estourar, basta voltar ao terminal do root e examinar o status da quota.

```
# repquota -v -a
```

Podemos verificar os detalhes mais avançados sobre o uso das quotas nas partições.

```
# quotastats
```

Caso seja necessário desativar o recurso de quota de uma partição, basta usar o comando **quotoff**:

```
# quotoff -v /home
```

Onde:

- **-v:** exibe mensagens para cada sistema de arquivos onde as quotas de disco são ativadas.

Para fazer uma verificação na quota da partição, para ver se está tudo bem:

```
# quotacheck -vcug /home
```

Ative a quota da partição novamente:

```
# quotaon -v /home
```

Utilização de ACL – Posix 1e

O recurso de ACL atende algumas das diretrizes de segurança propostas no **Posix 1e**, criando a possibilidade de tratativas de direitos mais amplas para um arquivo ou diretório em um sistema Like Unix. Sem o uso de ACL, existem limitações complicadas como, por exemplo, a simples tarefa de definir segurança de grupo para um arquivo, pois através dos direitos é possível somente ter um grupo, conforme ilustrado:

```
- rwx r--r-- 1 root admin 4096 2013-03-08 05:00 arquivo.conf
```

No exemplo, vemos que rwx define o direito do dono que é o usuário root; r-- define o direito do grupo que é admin, e r- define o direito de outros usuários sobre o arquivo.

Esse conceito de permissionamento acaba, em ambientes mais complexos, sendo um limitador, pois só é possível definir um único grupo.

Nesse cenário, caso seja necessário atribuir direitos a outro grupo ou mudar o grupo atual do arquivo, o que traz uma limitação para o sysadmin. Embora ainda exista a possibilidade

de manipular as permissões considerando o direito de “outros”, em muitos casos isso não vai resolver o problema da limitação de ser possível atribuir critérios de permissionamento considerando: “o dono, o grupo e outros”. Diante disso, o que fazer? A resposta é ativar o recurso de acl no sistema de arquivo, o que trará grande flexibilidade para definição de permissionamento.

A implementação do recurso de ACL em sistema Like Unix já existe há algum tempo no kernel do Linux. Desde a série 2.4, de 1999, já era possível, através de aplicação de patches, mas já na versão 2.5.46 (desenvolvimento) passou a ser um recurso totalmente integrado ao kernel. Dessa forma, a série 2.6 já sairá com pleno suporte a ACL.

É simples confirmar que o kernel de uma determinada distribuição está com suporte a ACL: basta consultar as informações do arquivo `/boot/config<kernel version>`.

```
# uname -a  
  
Linux hardening 3.2.0-29-generic #46-Ubuntu SMP Fri Jul 27 17:03:23  
UTC 2012 x86_64 x86_64 x86_64 GNU/Linux  
  
# uname -r  
  
3.2.0-29-generic  
  
# grep -i acl /boot/config-3.2.0-29-generic  
  
CONFIG_EXT2_FS_POSIX_ACL=y  
  
CONFIG_EXT3_FS_POSIX_ACL=y  
  
CONFIG_EXT4_FS_POSIX_ACL=y  
  
CONFIG_REISERFS_FS_POSIX_ACL=y  
  
CONFIG_JFS_POSIX_ACL=y  
  
CONFIG_XFS_POSIX_ACL=y  
  
CONFIG_BTRFS_FS_POSIX_ACL=y  
  
CONFIG_FS_POSIX_ACL=y  
  
CONFIG_GENERIC_ACL=y  
  
CONFIG_TMPFS_POSIX_ACL=y  
  
CONFIG_NFS_V3_ACL=y  
  
CONFIG_NFSD_V2_ACL=y  
  
CONFIG_NFSD_V3_ACL=y  
  
CONFIG_NFS_ACL_SUPPORT=m  
  
# CONFIG_CIFS_ACL is not set  
  
CONFIG_9P_FS_POSIX_ACL=y
```

Embora tenhamos o recurso no kernel, são necessárias as ferramentas na userland para administração do recurso de ACL. Para instalar as ferramentas de administração de ACL em distribuição like debian, como Ubuntu, podemos usar o comando `aptitude` ou `apt`:

```
# apt-get update  
# apt-get install acl
```



Habilitação do suporte ao recurso de ACL

Para usar o ACL, devemos parametrizar a partição desejada no `/etc/fstab` ou remontar a partição com suporte a ACL. No entanto, remontar habilita o recurso em tempo de execução. Parametrizar via `/etc/fstab` garante que toda vez que o servidor for inicializado, o recurso será ativado. Dessa forma, é necessário editar o arquivo `/etc/fstab` e adicionar o parâmetro "acl" nas opções do sistema de arquivo desejado:

```
# vim /etc/fstab  
  
/dev/sda6 /var/log ext4 default,noexec,nosuid,noatime,acl 0 2
```

Conforme já informado, essa parametrização só terá efeito no próximo boot. Para ativar o suporte no momento da execução, execute o comando `mount` para remontar a partição:

```
# mount -o remount,acl /var/log  
  
# ls -l /var/log
```

Na lista do `ls -l /var/log`, além dos nove dígitos de direitos, também haverá um sinal de "+" indicando o suporte a ACL.

Comando de administração de ACL

`Setfacl`: permite parametrizar os direitos de ACLs em arquivos e diretórios:

- ▣ `-m, --modify`: modifica os ACL de um arquivo ou diretório.
- ▣ `-x, --remove`: suprime as entradas ACLs.
- ▣ `-b, --remove-all`: suprime todas as entradas ACLs.
- ▣ `-L, --logical`: acompanhamento dos links simbólicos.
- ▣ `-R, --recursiva`: aplicação dos ACLs de maneira recursiva.

`Getfacl`: permite consultar informações de ACL de um arquivo ou diretório:

- ▣ `-R`: permite ver os ACLs de maneira recursiva.
- ▣ `-L`: para seguir links simbólicos.

Exemplo de configuração de ACL para logs, combinando com direitos via `chmod`, onde será definido que o dono terá pleno direito, o grupo terá direito de acesso a subdiretórios e leitura de arquivo, e outro usuário não terá direito. Tudo isso sendo parametrizado usando o recurso de permissionamento clássico, com as ferramentas `chmod` e `chown`:

```
# chmod -Rv 750 /var/log  
  
# chown root.sysadmin /var/log
```

Mas também será demandado direito de leitura a membros do grupo `sysop` a todos os logs de `/var/log` e direito de leitura a membros do grupo `webmaster` ao diretório `/var/log/httpd`.

```
# vim /etc/fstab  
  
/dev/sda6 /var/log ext4 default,noexec,nosuid,noatime,acl 0 2  
  
# mount -o remount,acl /var/log  
  
# setfacl -R -m g:sysop:rx /var/log  
  
# setfacl -R -m g:webmaster:rx /var/log/httpd
```

Para consultar informações sobre o permissionamento de ACL aplicado:

```
# getfacl -R /var/log  
# getfacl -R /var/log/httpd
```

Comando “sudo” (Super User Do – fazer como super usuário)

O comando *sudo* permite aos usuários executar comandos como outro usuário. Uma vantagem em usar o *sudo* é a de poder conceder aos usuários definidos acesso restrito, embora privilegiado, a programas como Super User durante a execução. Dessa forma o uso do comando *sudo* acaba sendo uma alternativa interessante ao uso da permissão especial de *suid* bit.

Entre vantagens do uso do *sudo*, podemos destacar:

- É muito comum em um ambiente onde existam múltiplos administradores com difícil controle do acesso root, pois a necessidade de várias pessoas executarem atividades que demandam recursos administrativos é um complicador. Isso pode motivar algo não recomendado, que é tornar a conta root uma conta compartilhada. Todavia, nem todos que necessitam executar recursos administrativos precisam ter acesso root;
- Frequentemente argumentado em lista de segurança que o uso do *sudo* pode ser um problema de segurança, pois a configuração errada do *sudo* fatalmente traz risco de conformidade com uma política de segurança. Um bom exemplo é o fato de o programa *sudo* ser configurado de modo que os membros de um grupo designado não precise de autenticação adicional para executar um determinado comando como *root*, o que resulta em maior produtividade. No entanto, deve ser avaliado pela equipe de segurança se deve-se ou não ser permitida essa “produtividade”. Na opinião do autor, é melhor optar pelo uso de autenticação em qualquer regra definida via *sudo*;
- É fato que o uso do *sudo* tem de ser bem avaliado e planejado, pois a execução de um comando via *sudo* é executá-lo com poder de *root*. Evidentemente, o *sudo* passa a ser mais um programa que demanda *suid* bit no sistema, devido à forma como funciona.

Instalação do *sudo*:

```
# apt-get install sudo
```

Depois que o *sudo* estiver instalado o próximo passo é editar o arquivo */etc/sudoers*, que é onde são parametrizados quais usuários podem ter acesso aos comandos definidos.

É recomendável editar o */etc/sudoers* com o comando *visudo*, que abre o arquivo utilizando o editor de texto padrão que pode ser parametrizado pela variável de ambiente *EDITOR*:

```
# export EDITOR=/usr/bin/vim  
# visudo
```

Exemplificação do uso do *sudo* com a criação de uma regra para o usuário *rnpentes* executar os comandos *ifconfig* e *Iptables*:

```
# visudo  
rnpente ALL= /sbin/ifconfig, /sbin/iptables
```

Nesse primeiro exemplo, foi definida a permissão de execução para que o usuário rnpteste possa executar os comandos *ifconfig* e *iptables* como se fosse root. Todavia, da forma que foi parametrizado, ainda assim será solicitada a senha do usuário na primeira vez que executar esses comandos via *sudo*. Lembrando que para executá-los devemos sempre preceder a digitação do comando *sudo* da seguinte maneira:

```
Su - rnpteste  
$ sudo ifconfig  
$ sudo iptables  
  
Exemplo 2:  
  
rnpteste ALL = NOPASSWD: /bin/reboot , /bin/halt
```

Nesse exemplo, há a permissão para que o usuário rnpteste possa executar os comandos *reboot* e *halt*, mas dessa vez sem que seja necessário que ele digite a senha. Isso é possível devido ao uso do parâmetro NOPASSWD.

Exemplo 3:

```
rnpteste ALL = /usr/bin/passwd [A-z]*, !/usr/bin/passwd root
```

Nesse terceiro exemplo foi parametrizada a execução do comando *passwd*, o que vai permitir que o usuário rnpteste mude a senha de qualquer usuário cujo login esteja no intervalo de A a Z, exceto o root.

Esses três exemplos iniciais ilustram como podem ser disponibilizados alguns comandos via *sudo*. Dessa forma, a maioria das distribuições Linux que trazem comandos com a permissão de Suid bit ativada podem, durante o processo de Hardening, terem essas permissões de suid bit removidas e as necessidades particulares tratadas pelo uso do *sudo*.

Organização de regras do sudo

Para tornar mais interessante, devem ser usados os parâmetros adicionais de configurações do *sudo*, como:

- ▣ **User_Alias:** especifica um alias para um único usuário, o que não é muito útil, ou um grupo de usuários que terão acesso a um conjunto de comandos. No entanto, esse grupo só terá valor dentro do contexto de execução do *sudo*. Um usuário pode aparecer em várias alias e ser vinculado a vários grupos de comandos.

Além de nome de usuários:

- ▣ User_List é definido através de um ou mais IDs de usuário;
- ▣ Nomes de sistema de grupo e ID (prefixo "%" e "#%", respectivamente);
- ▣ Grupos de usuário de rede – netgroups (prefixo "+");
- ▣ Usuários não Unix, nomes de grupo e IDs (prefixo "%:" e "%: #", respectivamente);
- ▣ E outros User_Aliases.

Cada item da lista pode ser pré-fixado com zero ou mais "!" operadores, o que possibilita negar o valor, ou seja, criar uma regra de exceção.

Exemplos:

```
User_Alias USERS_SYSOP =  eragon, Aragon, frodo
```



```
User_Alias USERS_SYSADMIN = harry,hermione,drago
```

```
User_Alias WEBMASTERS = bidu,monica,magali
```

- **Runas_Alias:** possibilita especificar que outro usuário sudo será capaz de funcionar no lugar do root. Por padrão, o *sudo* está vinculado ao usuário root, mas pode-se querer executar como outro usuário.

Exemplos:

```
Runas_Alias OPERATOR = root, operator
```

```
Runas_Alias DATABASE = oracle, Sybase
```

- **Host_Alias:** especifica os hosts em que os direitos se aplicam. Isso normalmente não é usado, ao menos que se deseje fazer trabalhos sysadmin para um grupo de outros servidores.

Exemplo:

```
Host_Alias MAC = 12.13.0.0/255.255.255.0
```

- **Cmnd_Alias:** especifica um alias para um comando específico ou um conjunto de comandos.

Exemplos:

```
Cmnd_Alias DUMPS = /usr/bin/mt, /usr/sbin/dump, /usr/sbin/rdump, /usr/sbin/restore
```

```
Cmnd_Alias KILL = /usr/bin/kill, /usr/bin/pkill
```

```
Cmnd_Alias PRINTING = /usr/sbin/lpc, /usr/bin/lprm
```

```
Cmnd_Alias DOWNOFF = /usr/sbin/shutdown, /usr/sbin/halt, /usr/sbin/reboot,
```

```
Cmnd_Alias SHELLS = /usr/bin/sh, /usr/bin/csh, /usr/bin/ksh, /usr/local/bin/tcsh, /usr/bin/rsh, /usr/local/bin/zsh
```

Exemplificação de criação de regras combinados as declarações de User_Alias com Cmnd_Alias.:

```
USERS_SYSOP      ALL = !SU, !SHELLS, DUMPS, KILL, PRINTING, DOWNOFF
```

```
USERS_SYSADMIN   ALL = ALL
```

```
USERS_WEBMASTER  ALL = !SU, !SHELLS, INTERNET
```

Prevenção ‘Escape Shell’

Para verificação da possibilidade de “Escape Shell”, basta digitar dentro do ambiente de comandos, como ftp, more, less, vi e vim:

- “!sh”

Assim que retornar a shell, digitar o comando “id” para identificar que o usuário está ativo na shell:

- :!sh
- #id

O administrador deve sempre lembrar que *sudo* executa um programa, e esse programa está sendo executado com poder de root, inclusive a chamada interna para executar outros programas. Isso pode ser um problema de segurança, uma vez que não é incomum para um programa permitir que “Escape Shell”, ou seja, a execução de uma shell, pode possibilitar que um usuário tenha uma shell com poder de root. Então devemos ter muito cuidado com programas que possuem esse tipo de recurso. Exemplos clássicos: vim, vi, ftp, more e less entre outros.

Duas formas de desabilitar “Escape Shell”

- **Parâmetro “restrict”:** deve ser evitado dar aos usuários acesso a comandos que permitem que o usuário execute comandos arbitrários. No entanto, muitos editores têm um modo restrito, onde os “Escape Shell” estão desativados, embora *sudoedit* seja a melhor solução para execução de editor via *sudo*. Mas devido ao grande número de programas que oferecem “Escape Shell”, é muitas vezes impraticável utilizar determinados comandos;
- **Parâmetro “noexec”:** muitos sistemas like Unix que suportam bibliotecas compartilhadas têm a capacidade de substituir as funções da biblioteca padrão, apontando uma variável de ambiente (geralmente LD_PRELOAD) a uma biblioteca alternativa compartilhada. Em tais sistemas, a funcionalidade noexec sudo pode ser usada para impedir que um programa executado pelo *sudo* execute chamadas para outros programas. Note, no entanto, que esta só se aplica aos nativos **dinamicamente compilados**. Executáveis **compilados estaticamente** e binários rodando sob emulação não são afetados.

Dinamicamente e estaticamente compilados:

dynamics compiled and estaticaly compiled referem-se às duas formas de gerar um pacote instalável. No pacote compilado estaticamente, são incluídos todos os arquivos e bibliotecas de que o programa precisa para funcionar.

Ele pode rodar em qualquer distribuição Linux: basta descompactar o pacote e executar. No compilado dinamicamente, todos os arquivos são compilados separadamente, de forma que uma mesma biblioteca nunca é carregada duas vezes, maximizando o uso do espaço e fazendo com que o sistema fique muito mais leve e rápido, porém faz com que surjam problemas de dependências (o programa x precisa da biblioteca y para funcionar).



O bom senso no que tange à gerência de permissionamento deve ser usado, ou seja: usar o recurso mínimo e o mínimo direito. Então, se a definição de direitos com chmod for necessário, não utilize acl. Se o ajuste com acl atende, não use *sudo* e, se regras de *sudo* atendem, não use suid bit







Roteiro de Atividades 2

Atividade 1 – PAM

- ## 1. Crie os seguintes usuários:

- eregon
 - Aragon
 - Frodo
 - Bidu
 - Monica
 - Magali
 - Harry
 - Hermione
 - Drago

- ## 2. Crie os grupos:

- ❑ sysop
 - ❑ sysadmin
 - ❑ webmaster

3. Defina uma regra utilizando o recurso PAM, permitindo acesso via SSH somente durante o horário das 7h às 19h para os usuários:

- eragon
 - Aragon
 - Monica
 - Magali
 - Harry
 - Drago
-
-
-
-

Atividade 2 – Procura por senhas fracas

1. Utilize o John the Ripper para avaliar as senhas dos usuários do sistema.

Apresente suas observações.

Atividade 3 – Quotas

1. Crie quotas de 200 MB por grupo no diretório */home* para todos os usuários criados no exercício 1.

2. Crie quotas de 200 MB no diretório */tmp* para todos os usuários.



Atividade 4 – ACL

1. Edite o `/etc/fstab` e ative o suporte a ACL nas partições vinculadas aos diretórios `/var/log` e `/var/personal`.

2. Remonte a partição para ativar o suporte a ACL nas partições vinculadas aos diretórios `/var/log` e `/var/personal` em tempo de execução:

```
# mount -o remount,acl /var/log  
# mount -o remount,acl /var/log  
# setfacl -R -m g:webmaster:rx /var/log/httpd
```

3. Crie regras de ACL permitindo acesso e leitura aos arquivos de diretório em `/var/log` para o grupo `syslog`:

```
# setfacl -R -m g:sysop:rx /var/log
```

4. Crie regras de ACL permitindo acesso de leitura a arquivos de diretório em `/var/personal` para o grupo `syslog`:

```
# setfacl -R -m g:sysop:rx /var/personal
```

5. Crie regras de ACL permitindo acesso de leitura a arquivo de diretório em */var/log/httpd* para o grupo webmaster:

```
# setfacl -R -m g:sysop:rx /var/log/httpd
```

6. Consulte as informações sobre o permissionamento de ACL aplicado:

```
# getfacl -R /var/log  
# getfacl -R /var/personal
```

7. Salve as informações de ACL dos diretórios */var/log* e */var/personal* para o arquivo *acl.txt*.

```
# getfacl -R /var/log /var/personal > /root/acl.txt
```

8. Apague todas as ACL.

```
# getfacl -R -b /var/log  
# getfacl -R -b /var/personal
```



9. Consulte para verificar se foram removidas.

```
# getfacl -R /var/log /var/personal
```

10. Restaure as informações da ACL:

```
setfacl --restore=/root/acl.txt
```

11. Consulte para verificar se foram restauradas:

```
# getfacl -R /var/log /var/personal > /root/acl.txt
```

Atividade 5 – Comando ‘sudo’

- Este exercício tem como objetivo criar um cenário com erros comuns e avaliar o nível de ameaça que certas configurações no *sudo* podem trazer a um sistema Linux. Dessa forma, acesse o ambiente shell da VM do Debian como root e verifique o arquivo */etc/sudoers*, adicionando e validando a configuração sugerida:

```
_ User_Alias USERS_SYSOP =  eragon, aragon, frodo, rnpteste  
User_Alias USERS_SYSADMIN = harry,hermione,drago  
User_Alias WEBMASTERS = bidu,monica,magali  
Cmnd_Alias DUMPS = /usr/bin/mt, /usr/sbin/dump, /usr/sbin/rdump, /  
usr/sbin/restore  
Cmnd_Alias KILL = /usr/bin/kill, /usr/bin/pkill  
bill      ALL = ALL, !SU, !SHELLS  
Cmnd_Alias PRINTING = /usr/sbin/lpc, /usr/bin/lprm  
Cmnd_Alias DOWNOFF = /usr/sbin/shutdown, /usr/sbin/halt, /usr/sbin/  
reboot,
```



```
Cmnd_Alias SHELLS = /usr/bin/sh, /usr/bin/csh, /usr/bin/ksh, /usr/  
local/bin/tcsh, /usr/bin/rsh, /usr/local/bin/zsh  
  
USERS_SYSOP      ALL = !SU, !SHELLS, DUMPS, KILL, PRINTING, DOWNOFF  
  
USERS_SYSADMIN   ALL = ALL  
  
USERS_WEBMASTER  ALL = !SU, !SHELLS, INTERNET
```

2. Adicione as novas regras no `/etc/sudoers` e avalie o problema do “Escape Shell”.

a. Crie um novo Cmnd_Alias:

```
Cmnd_Alias EDIT= /usr/bin/vim /etc/hosts, /usr/bin/less /var/log/  
syslog, /usr/bin/more /var/log/authlog
```

b. Edite a regras para USERS_SYSOP:

```
USERS_SYSOP      ALL = !SU, !SHELLS, DUMPS, KILL, PRINTING, DOWNOFF,  
EDIT
```

3. Após avaliar o risco do “Escape Shell”, avalie o uso do parâmetro NOEXEC:

```
USERS_SYSOP      ALL = NOEXEC: !SU, !SHELLS, DUMPS, KILL, PRINTING,  
DOWNOFF, EDIT
```

4. Reedite o `/etc/sudoers`, trocando o uso do comando `vim` pelo `sudoedit`:

- Este exercício será baseado em uma VM com Linux Fedora. O objetivo é criar um cenário com erros comuns e avaliar o nível de ameaça que certas configurações no `sudo` podem trazer a um sistema Linux. Dessa forma, acesse o ambiente shell da VM e, como root, verifique o arquivo `/etc/sudoers` adicionando e validando cada configuração. Assim, temos o grande desafio “Sudo the flag” (um trocadilho com “Capture the flag”), que consiste em identificar as configurações que possibilitam escala de privilégio para usuário root. O desafio consiste em identificar cada uma das falhas (flag) e explicar como ela poderia ser uma ameaça. Quem encontrar mais falhas – ou seja, conquistar mais bandeiras – vence.

6. Sugerir correções efetivas para todos os problemas identificados no “Sudo the flag”.



3

Hardening em sistema Linux – Registro de eventos (log) e HIDS

objetivos

Customizar a trilha de comandos com timestamp; Customizar a configuração estruturada de logs; Configurar um servidor de logs remotos; Contabilizar o uso de recursos do sistema pelos processos; Configurar um sistema de detecção de modificações no sistema – Host Intrusion Detect System (HIDS).

conceitos

Trilha de comandos do terminal (history); Customizando registro de eventos de um sistema Linux; Configuração de um servidor de logs; Contabilização de processos; Detecção de alteração no sistema – HIDS.

Neste capítulo, veremos como customizar a trilha de comandos do history, conceitos de registro de eventos (logs), contabilização de processo e HIDS.

- Trilha de comandos.
- Registro de eventos – logs do sistema.
- Syslog-NG.
- Servidor Syslog Centralizado.
- Configuração do STUNNEL.
- Rotacionamento de logs.
- Contabilização de processos.
- Auditoria com HIDS.

Exercício de nivelamento 1 Registro de eventos (log) e HIDS

1. Como você percebe a necessidade de preservar a trilha de comandos (histórico) executados em um servidor Like Unix?

2. Como são tratados em sua empresa os registros de eventos (logs) dos serviços fornecidos? Existe um servidor concentrador de registros de eventos (logs)?

3. Na política de segurança da sua empresa, existem definições de controles sobre mudanças no sistema que possam caracterizar um possível ataque ou mesmo uma mudança?

Trilha de comandos

Todos os comandos digitados pelo usuário são mantidos em um histórico.



- Usuário pode consultar quando precisar.
 - Esse recurso permite que o usuário possa retornar aos comandos que já digitou simplesmente voltando com a seta do teclado.

Essa funcionalidade é o history:

- Cada usuário possui o seu arquivo com o seu histórico.

Podemos analisar algumas das configurações-padrão dos sistemas Linux relacionadas ao histórico (trilha) de comandos. Por exemplo, é possível usar o comando *history* para visualizar o histórico do usuário que estamos utilizando.

```
# history
```

Existem algumas variáveis de ambiente importantes, entre elas a HISTFILE, que armazena a localidade do arquivo de histórico do usuário – que pode ser visualizado para que tenhamos a certeza da localidade do arquivo de histórico (trilha) de comandos.

```
# set | grep HISTFILE
```

Neste exemplo, está sendo usado o usuário root. Dessa forma, é possível identificar o arquivo de histórico, que é o */root/.bash_history*.

```
# cat /root/.bash_history | less
```

Trilha de comando



Shell bash:

- Armazena os comandos digitados originalmente no arquivo *~/.bash_history*, após a sessão ser finalizada.
 - Todo usuário tem seu próprio arquivo *.bash_history*.

É interessante ajustar as variáveis para reter um número grande de comandos armazenados no arquivo *.bash_history*:

- Deve-se ajustar as variáveis HISTFILESIZE e HISTSIZE, que podem ser parametrizadas via arquivo *.bashrc*.
- Outra variável importante e recomendada é a HISTTIMEFORMAT, pois por meio desta é possível ativar um controle com "timestamp" para a trilha de comando.



Pode-se executar essas parametrizações manualmente via arquivo *.bashrc* do respectivo usuário já criado e depois inserir no */etc/skel/.bashrc*.

Em servidores onde existem mais de um administrador, é interessante termos a trilha de comandos separada. Para isso, basta fazer um ajuste fino na variável HISTFILE do usuário root, inserindo uma linha redefinindo o favor da variável no arquivo */root/.bashrc*, conforme ilustrado a seguir:

```
export HISTFILE="/root/.bash_history_$(who am i | awk '{ print $1 }';exit)"
```

Em muitos casos é interessante customizar ações de Hardening via shellscript, conforme exemplificado no script da próxima figura. Esse script pode ser criado no diretório */root* com o nome *ctrl_history.sh*.

```
#!/bin/bash

HISTFILESIZE=30000
HISTSIZE=20000
HISTTIMEFORMAT="- %d/%m/%Y %H:%M:%S - "
RULES='HISTTIMEFORMAT="- %d/%m/%Y %H:%M:%S - "'
BASHRC_GERAL="/etc/bashrc"
BASHRC_SKEL="/etc/skel/.bashrc"
BASHRC_ROOT="/root/.bash_history_$(who am i | awk '{ print $1 }';exit)"

func_READONLY_hist()
{
    _ARQ="$1"
    POLICE_RONLY="readonly -n $VAR_HIST"
    for VAR_HIST in HISTFILE HISTFILESIZE HISTSIZE HISTTIMEFORMAT
        do
            grep $POLICE_RONLY "$_ARQ" || echo $POLICE_RONLY >> "$_ARQ"
        done
}
func_files()
{
    for ARQ in $BASHRC_GERAL $BASHRC_SKEL $BASHRC_ROOT
        do
            if [ -f $ARQ ]

```



```

        then
            grep HISTTIMEFORMAT "$ARQ" || echo "$RULES" >> "$ARQ"
            func_READONLY_hist "$ARQ"
        fi
    done
}

func_user()
{
    for USER_HOME in $(cut -f 6 -d ":" /etc/passwd)
    do
        if [ -f "$USER_HOME"/.bashrc ]
        then
            grep HISTTIMEFORMAT "$USER_HOME"/.bashrc || echo "$RULES" >>
"$USER_HOME"/.bashrc
            . "$USER_HOME"/.bashrc
        fi
    done
}
func_files && func_user

```



Boas práticas recomendadas na ABNT NBR ISO/IEC 27002:2008: a proposta de trilha de comando de root segregada por administrador e com o valor de timestamp possibilita plena conformidade com a recomendação do item 10.10.4.

Registro de eventos – Logs do sistema

Necessidade de registrar atividades dos usuários e serviços do sistema:

- ▣ Muito importante para os administradores.
 - ▣ Administração se torna muito mais auditável e detalhada quando se tem controle sobre todos os logs que o Linux tem a oferecer.



Logs:

- ▣ Ajudam também a descobrir se no sistema houve algum acesso indevido, registros de tentativas que podem sugerir uma tentativa de invasão ou mesmo intrusões.

As distribuições Linux, por padrão, já trazem habilitada boa parte dos registros de eventos (logs) necessários a uma administração, ou seja, muitas das atividades que acontecem já são registradas pelo sistema. Entretanto, é fortemente recomendável refinar os controles de logs para que seja possível ter registro ainda mais detalhado e organizado.

Conformidade com as recomendações

O registro de eventos é uma necessidade de normas de segurança como NBR IEC 27002 e PCI DSS, entre outros. Normas e documentos de boas práticas de segurança dedicam vários tópicos à importância dos logs.



As boas práticas recomendadas na ABNT NBR ISO/IEC 27002:2008 informam que, tradicionalmente, um bom log tem de ter informações que possibilitem uma avaliação, auditoria ou mesmo uso durante a resposta a um incidente de segurança. Essa recomendação é muito bem destacada, do item 10.10.1 ao 10.10.5, que definem diretrizes orientando o registro, monitoramento e retenção de informações para auditorias sobre atividades no sistema, sejam elas autorizadas ou não. Assim, é prioridade adotar uma política de segurança na qual os registros devam atender a características como:

- Identificação dos usuários;
- Datas e horários de entrada (login e logout);
- Identidade do terminal, nome da máquina ou IP;
- Registro das tentativas de acesso aos aceitos e rejeitados;
- Registro das tentativas de acesso a outros recursos e dados aceitos e rejeitados;
- Alteração de arquivos;
- Uso de privilégios, aplicativos e utilitários do sistema.

Boas práticas para arquivos de log

Alguns critérios são sugeridos para implantação de uma política de arquivos de log, no que tange a centralização, arquivamento e acesso.

Logs centralizados:

- Mais fáceis de analisar.
 - Quando temos um servidor dedicado a essa tarefa, evitamos que dados importantes sejam perdidos por acidente, excluídos ou alterados de propósito, caso ocorra um incidente.

Provisões para backup de arquivos de log:

- Arquivos de log são dados importantes.
 - Precisam ter definido um procedimento de arquivamento (backup). É interessante definir um procedimento para backup periódico, que deve também estar alinhado ao termo de retenção já estabelecido.

Proteção dos arquivos de log:

- Arquivos de log do Sistema Operacional devem ter acesso restrito a administradores e operadores.
 - Considerando o tipo de informação que há em um determinado arquivo de log, uma vez que algumas vezes podem incluir senhas e outras informações sensíveis e confidenciais.

Período de conservação dos arquivos de log:

- Arquivar logs mantidos eternamente não é uma solução razoável.
 - Outro ponto importante é a forma como o log será mantido, pois um arquivo de log compactado ocupa muito menos espaço que um não compactado e, caso seja um arquivo do tipo ASCII, ainda sim pode ser facilmente auditado com a ajuda de ferramentas como zcat ou bzcat. Por isso, não há motivos para não realizarmos procedimento de “rotacionamento de logs” com o uso de compactação de dados.

Logs do sistema

Diretório `/var/log`:

- Tradicionalmente, nos sistemas Linux, é onde ficam registrados todos os logs do sistema.
 - Esses logs são controlados pelo serviço Syslog, o programa padrão do Linux para essa tarefa de logs, mas também comum em outros unix e dispositivos de rede como accesspoints, roteadores, firewall, IPS, IDS e outros.

Dentro do diretório `/var/log`, destacam-se alguns logs que não são escritos em formato texto (ASCII). Então, não são lidos por um editor de texto comum. Eles estão em formatos binários e somente os seus comandos relacionados podem mostrar o seu conteúdo. Um desses logs é o `/var/log/wtmp`, que contém a informação de todos os últimos registros e logins dos usuários. As informações desse arquivo só podem ser visualizadas pelo comando `last`.

```
# last  
  
# last root  
  
# last toor
```

Existe também o `/var/log/btmp`, que é bem semelhante ao `wtmp`, mas mostra as últimas tentativas de login que falharam, o que pode ser muito útil em uma auditoria ou mesmo em uma Resposta a Incidente, pois, dependendo das informações que ele forneça, podemos identificar um possível ataque de Força Bruta. O seu conteúdo pode ser visualizado com o comando `lastb`.

```
# lastb  
  
# lastb root  
  
# lastb toor
```

Outro arquivo é o `/var/log/lastlog`, que nos mostra quando cada usuário realizou o processo de login pela última vez e permite ter uma visão bem interessante. Pode-se visualizar o seu conteúdo com o comando `lastlog`.

```
# lastlog
```

O arquivo `/var/run/utmp` permite uma visão bem detalhada de quais usuários estão ativos naquele momento, em qual terminal, horário e mais algumas informações, até mesmo se for uma conexão remota, via ssh ou telnet. O seu conteúdo pode ser visualizado por dois comandos, o `w` e o `who`, que mostram informações bem semelhantes, ambos tratando o que foi registrado no `/var/run/utmp`.

```
# w  
  
# who
```



Boas práticas recomendadas na ABNT NBR ISO/IEC 27002:2008: as informações de log e contabilização de processo que são usadas pelos comandos: `w`, `who`, `last`, `lastb`, `lastlog` e `lastcomm` são de grande utilidade para auditoria no que tange a informações de logins dos usuários do sistema. Esses recursos, que já são padrão, já permitem a conformidade com o item 11.5 da norma.

Syslog-NG



Syslog-NG:

- ▣ Tem como objetivo dar alternativa ágil e flexível para organizar melhor os logs fornecidos por um sistema Linux.

Atualmente, existem muitas opções de logs no Linux; isso também vale para outros “sabores” de Unix. No caso do Linux, é muito comum observar que a maioria das instalações atuais que usavam o Daemon Syslog tradicional passaram a usar ou o RSYSLOG ou mesmo o próprio SYSLOG-NG.

Mas antes de montar uma estrutura para organizar os registros das atividades do sistema, é necessário entender melhor o protocolo Syslog.

A definição do formato dos registros do protocolo Syslog, que é dividida em 19 recursos, também pode ser chamada de “facility”. Entretanto, esses recursos podem ser divididos em nove níveis – os quais também são definidos como “prioridade” – de acordo com o grau crítico da ação que será registrada.

Recursos (facility)

- ▣ **auth**: mensagens de segurança/autORIZAÇÃO/autENTICAÇÃO que falharam, desaprovadas pelo sistema;
- ▣ **Authpriv**: mensagens de segurança/autENTICAÇÃO/autORIZAÇÃO;
- ▣ **cron**: atividades do cron e at;
- ▣ **daemon**: deamons do sistema (sshd, inetd, pppd etc.);
- ▣ **Kern**: mensagens do Kernel;
- ▣ **Ipr**: mensagens de impressão;
- ▣ **Mail**: subsistema de e-mail (sendmail, postfix, qmail etc.);
- ▣ **news**: mensagens de sistemas de newsgroups;
- ▣ **user**: mensagens genéricas no âmbito do usuário;
- ▣ **uucp**: informações do serviço UUCP;
- ▣ **syslog**: mensagens internas do syslog;
- ▣ **local0 a local7**: níveis definidos localmente, e geralmente usados para locais específicos com logs remotos de Access point.

Nível de registro (prioridade)

- ▣ **emerg**: indicam condições de emergências do sistema;
- ▣ **Alert**: problemas no sistema que necessitam de atenção imediata;
- ▣ **Crit**: problemas fatais e condições críticas do sistema;
- ▣ **Debug**: informações decodificadas de depuração dos processos em andamento;
- ▣ **err**: mensagens de condições de erro geradas pelo STDERR;
- ▣ **Info**: informações de programas em execução;
- ▣ **Notice**: mensagens de condições normais, mas relevantes;
- ▣ **Warning**: aviso padrão de execução de um processo;
- ▣ **None**: mensagens genéricas.

Por muito tempo o aplicativo padrão do Linux para gerenciamento de logs foi o Syslogd; todavia, muitas distribuições estão trazendo alternativas, como Syslog-ng e Rsyslog.

Será usado neste capítulo um deamon muito melhorado em relação ao tradicional Syslod. O alvo neste capítulo será o deamon do Syslog-NG, que tem muitos mais recursos que o seu antecessor.

Antes de instalar e vermos os arquivos de configuração do Syslog-NG, vamos entender como ele funciona.

A estrutura do Syslog-NG é como um quebra-cabeça: há várias peças separadas e temos de montá-las conforme desejamos para ter o nosso log personalizado.

O Syslog-NG possui três peças básicas: Source (Origem), Filter (Filtro) e Destination (Destino). Essas três peças unidas formarão o nosso log. Veja uma representação gráfica desse quebra-cabeça na figura a seguir.

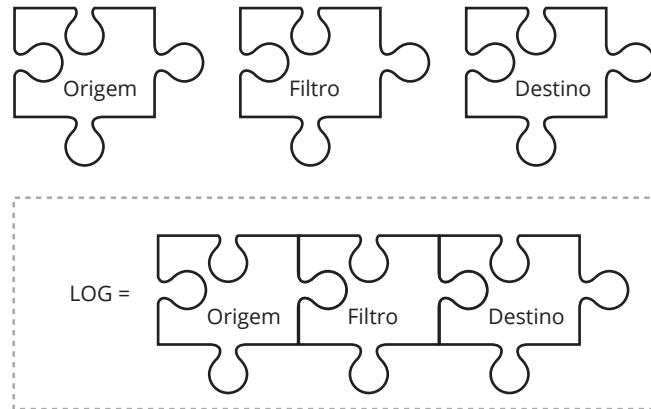


Figura 3.1
Representação
de módulos de
configuração.

Com esse conceito em mente, pode-se instalar o Syslog-NG.

```
# apt-get install syslog-ng
```

Depois de instalado o Syslog-NG, cria-se uma estrutura personalizada para os logs. Vamos utilizar o diretório `/var/personal`, que está em uma partição separada, para alocar essa nossa nova estrutura personalizada.

O arquivo de configuração base do Syslog-NG normalmente está localizado em `/etc/syslog-nginx.conf`, no qual é definida a forma como será organizada nossa estrutura de logs e também recursos do próprio serviço. Entretanto, para tirar proveito, é importante entender como funciona o protocolo Syslog.

Primeiros vamos renomear o `syslog-nginx.conf` original e criar um novo.

```
# cd /etc/syslog-nginx
# mv syslog-nginx.conf syslog-nginx.conf_original
```

Logo em seguida, temos a exemplificação de como criar uma nova com a estrutura a seguir, considerando as configurações globais.

```
# vi syslog-nginx.conf
#OPÇÕES GLOBAIS
```

```

options { long_hostnames(off);};

#OPÇÕES DE ORIGEM

source src { unix-dgram("/dev/log"); internal();};

# OPCÕES DE FILTROS

filter f_authpriv { facility(auth, authpriv); };

filter f_syslog { not facility(auth, authpriv); };

filter f_cron { facility(cron); };

filter f_daemon { facility(daemon); };

filter f_kern { facility(kern); };

filter f_lpr { facility(lpr); };

filter f_mail { facility(mail); };

filter f_user { facility(user); };

filter f_uucp { facility(uucp); };

filter f_news { facility(news); };

filter f_debug { not facility(auth, authpriv, news, mail); };

filter f_messages { level(info .. warn)

#vi syslog-ng.conf

and not facility(auth, authpriv, cron, daemon, mail, news); };

filter f_emergency { level(emerg); };

filter f_info { level(info); };

# vim /etc/syslog-ng/syslog-ng.conf

filter f_notice { level(notice); };

filter f_warn { level(warn); };

filter f_crit { level(crit); };

filter f_err { level(err); };

filter f_cnews { level(notice, err, crit) and facility(news); };

filter f_cother { level(debug, info, notice, warn) or
facility(daemon, mail); };

filter ppp { facility(local2); };

# OPCÕES DE DESTINO

destination authlog { file("/var/personal/auth.log" owner("root")
group("root") perm(0640)); };

```

```

destination syslog { file("/var/personal/syslog" owner("root")
group("root") perm(0640)); };

destination cron { file("/var/personal/cron.log" owner("root")
group("root") perm(0640)); };

destination daemon { file("/var/personal/daemon.log" owner("root")
group("root") perm(0640)); };

destination kern { file("/var/personal/kern.log" owner("root")
group("root") perm(0640)); };

destination lpr { file("/var/personal/lpr.log" owner("root")
group("root") perm(0640)); };

destination mail { file("/var/personal/mail.log" owner("root")
group("root") perm(0640)); };

destination user { file("/var/personal/user.log" owner("root")
group("root") perm(0640)); };

destination uucp { file("/var/personal/uucp.log" owner("root")
group("root") perm(0640)); };

destination mailinfo { file("/var/personal/mail.info" owner("root")
group("root") perm(0640)); };

destination mailwarn { file("/var/personal/mail.warn" owner("root")
group("root") perm(0640)); };

destination mailerr { file("/var/personal/mail.err" owner("root")
group("root") perm(0640)); };

destination newscrit { file("/var/personal/news/news.crit"
owner("root") group("root") perm(0640)); };

destination newserr { file("/var/personal/news/news.err"
owner("root") group("root") perm(0640)); };

destination newsnotice { file("/var/personal/news/news.notice"
owner("root") group("root") perm(0640)); };

destination debug { file("/var/personal/debug" owner("root")
group("root") perm(0640)); };

destination messages { file("/var/personal/messages" owner("root")
group("root") perm(0640)); };

destination console { usertty("root"); };

destination console_all { file("/dev/tty8"); };

destination xconsole { pipe("/dev/xconsole"); };

destination ppp { file("/var/personal/ppp.log" owner("root")
group("root") perm(0640)); };

# OPCÕES DE LOG (Montagem do log)

```

```

log { source(src); filter(f_authpriv); destination(authlog); };

log { source(src); filter(f_syslog); destination(syslog); };

log { source(src); filter(f_cron); destination(cron);};

log { source(src); filter(f_daemon); destination(daemon); };

log { source(src); filter(f_kern); destination(kern); };

log { source(src); filter(f_lpr); destination(lpr); };

log { source(src); filter(f_mail); destination(mail); };

log { source(src); filter(f_user); destination(user); };

log { source(src); filter(f_uucp); destination(uucp); };

log { source(src); filter(f_mail); filter(f_info);
destination(mailinfo); };

log { source(src); filter(f_mail); filter(f_warn);
destination(mailwarn); };

log { source(src); filter(f_mail); filter(f_err);
destination(mailerr); };

log { source(src); filter(f_news); filter(f_crit);
destination(newscrit); };

log { source(src); filter(f_news); filter(f_err);
destination(newserr); };

log { source(src); filter(f_news); filter(f_notice);
destination(newsnotice); };

log { source(src); filter(f_debug); destination(debug); };

log { source(src); filter(f_messages); destination(messages); };

log { source(src); filter(f_emergency); destination(console); };

log { source(src); filter(f_cnews); destination(xconsole); };

log { source(src); filter(f_cother); destination(xconsole); };

log { source(src); filter(ppp); destination(ppp); };

```

Feita essa configuração, e lembrando a figura do quebra-cabeça, vamos entender o arquivo do Syslog-NG.

Para analisá-lo por partes: primeiro, a origem.

```
source src { unix-dgram("/dev/log"); internal();};
```

As linhas começam com a palavra source (origem), e logo em seguida vem src, que pode ser qualquer palavra desde que depois possamos entender a que se refere.

Essa origem que aponta para `/dev/log` são logs locais, ou seja, a origem em questão são todos os logs gerados pelo nosso sistema. Essa é a nossa primeira peça do quebra-cabeça, a nossa origem, mas ainda não estamos fazendo nada com ela.



Depois da origem, nós temos algumas linhas relacionadas à parte de filtro.

Vamos a alguns exemplos.

```
filter f_authpriv { facility(auth, authpriv); };
```

As linhas de filtro começam com a palavra filter, e em seguida vem f_authpriv, para a qual, assim como nas linhas de origem, nós podemos definir o nome que quisermos, mas sempre é bom adotarmos um padrão para um melhor controle. Por exemplo, adotamos o padrão de que todos os filtros começem com f_ e o nome do filtro.

Essa linha está criando um filtro somente com as facilidades (facility), auth e authpriv em todos os níveis (level), ou seja, esse filtro engloba todas as mensagens de autenticação, sejam elas avisos, informações, emergências, críticas ou qualquer um dos outros níveis.

```
filter f_syslog { not facility(auth, authpriv); };
```

Esse exemplo é o contrário do primeiro, pois está criando um filtro chamado f_syslog, que engloba todas as facilidades e níveis, com exceção das facilidades auth e authpriv.

```
filter f_cron { facility(cron); };
```

```
filter f_kern { facility(kern); };
```

Exemplo de dois filtros: o f_cron, que registrará todas as mensagens relacionadas ao cron; e o f_kern, que registrará todas as mensagens relacionadas ao kernel.

```
filter f_messages { level(info .. warn)  
    and not facility(auth, authpriv, cron, daemon, mail, news); };
```

Esse já é um filtro mais elaborado. Com o nome de f_messages, engloba todas as facilidades, com exceção de auth, authpriv, cron, daemon, mail e news, nos níveis info até warn.

Os filtros são nossa segunda peça do quebra-cabeça. Fizemos todos os filtros que desejamos, mas até aqui não os utilizamos.

A parte seguinte do arquivo é a que trata os destinos (destination). Esses destinos são onde os logs serão gravados.

Vejamos o exemplo.

```
destination authlog { file("/var/personal/auth.log" owner("root")  
    group("root") perm(0640)); };
```

```
destination syslog { file("/var/personal/syslog" owner("root")  
    group("root") perm(0640)); };
```

```
destination cron { file("/var/personal/cron.log" owner("root")  
    group("root") perm(0640)); };
```

Seguindo o mesmo padrão das outras configurações, origem e filtro, os destinos começam com a palavra "destination", e depois vem o nome que escolhermos. Esses destinos apontam para dentro do diretório /var/personal (diretório que adotamos na nossa estrutura personalizada) e para os arquivos que serão criados, que são os arquivos de logs.



Saiba mais

Para visualizar todas as facilidades, níveis e ordens de criticidade dos níveis, podemos consultar o manual do syslog.



Esses arquivos têm como usuário-dono (owner) o root, como grupo-dono (group) o root, e como permissão (perm) 640.

Pode-se pensar em definir o destino tradicionalmente como um arquivo, mas também é possível parametrizar o destino para um arquivo Fifo, uma aplicação ou um servidor remoto.

Bom, ainda não definimos qual será o conteúdo desses destinos, mas eles são nossa terceira peça do quebra-cabeça. Com as três peças definidas, podemos formar o nosso log. É agora que vamos unir a origem, os filtros e o destino, ou seja, vamos encaixar as peças.

```
log { source(src); filter(f_authpriv); destination(authlog); };
```

As linhas que realmente geram os logs se iniciam com a palavra "log". Em seguida, são encaixadas as peças: primeiro o source (origem, que no nosso caso é o src, que é a origem local); depois os filtros (estamos usando somente o f_authpriv); e, por fim, o destino, que é em qual arquivo essas informações serão gravadas (estamos usando o destino authlog, que é o arquivo */var/personal/auth.log*).

Resumindo: todas as informações com origem em nosso sistema que se encaixem no filtro f_authpriv (autenticação de usuários) serão gravadas no destino */var/personal/auth.log*.

```
log { source(src); filter(f_mail); filter(f_info);  
destination(mailinfo); };
```

Essa linha é semelhante à anterior, com uma única diferença: está aplicando dois filtros ao mesmo tempo.

Caso a instalação padrão da distribuição Linux em uso não seja o Syslog-ng, toda a estrutura de novos arquivos de logs seria criada, pois o Syslog-NG não funciona da mesma maneira que as outras opções de servidores de log. Assim sendo, todos esses destinos que definimos no arquivo de configuração do Syslog- NG ou precisam ser criados manualmente ou a diretriz "create_dirs()" deve ser utilizada para que a criação da estrutura de arquivos aconteça automaticamente.

Caso queira criar automaticamente, vamos a um exemplo de script para automatizar a tarefa:

```
# cd /root  
  
# vi estrutura.sh  
  
#!/bin/bash  
  
dir="/var/personal"  
  
arq="auth.log syslog cron.log daemon.log kern.log lpr.log mail.log  
user.log uucp.log mail.info mail.warn mail.err news.crit news.err  
news.notice debug messages ppp.log"
```

```

if [!-d "$dir"]; then
    mkdir "$dir"
fi

for est in $arq
do
    touch "$dir/$est"
    chmod 640 "$dir/$est"
    chattr +a "$dir/$est"
done
clear
echo "Estrutura Personalizada Criada"

# ./estrutura.sh

```

Observe que esse script utiliza o comando *chattr +a*. O comando *chattr* muda o atributo de um arquivo ou diretório, e o atributo *+a* faz com que nesses arquivos só sejam adicionadas informações, impossibilitando que parte delas ou até mesmo os arquivos inteiros sejam apagados. Isso é importante para seguir as boas práticas de segurança que a norma recomenda. Embora o usuário root ainda tenha a possibilidade de remover o atributo e manipular o arquivo, essa janela de “possibilidade” fica sendo exclusiva do root.

No entanto, só será necessário o uso de um script como esse se o servidor de log escolhido não fizer a criação automática. No caso do syslog-ng, basta inserir a função “*create_dirs()*”.

Outro ponto que esse rigor motiva é que também seja necessário manipular o atributo antes de qualquer processo de rotacionamento.

Depois de executar o script, vamos verificar se os arquivos foram criados.

```
# ls -l /var/personal
```

Com a estrutura criada, podemos reiniciar o serviço do Syslog-NG e testar se está funcionando.

```
# /etc/init.d/syslog-ng restart
```

Para testar, podemos reiniciar alguns serviços ou realizar o processo de login em outros terminais e checar se tudo está começando a ser registrado na nossa nova estrutura de logs.

Veja um exemplo:

```
# /etc/init.d/cron restart
```

```
# ls -l /var/personal/
```

```
# cat /var/personal/cron.log
```

Servidor Syslog centralizado

Criação de servidor concentrador de logs dos servidores:

- Ótima e necessária implementação, porque, entre outras razões, todos os sistemas diretamente expostos à internet encontram-se nas linhas de frente para a internet.

Durante um momento crítico:

- Se um desses servidores estiver comprometido, não será surpresa saber se os seus logs também estarão.
 - É provável que o invasor apague toda a evidência que puder. Dessa forma, manter cópias logs em um segundo sistema dificultará que eles encubram os rastros.

A razão relevante é que um servidor centralizado com logs permite que a equipe administrativa fique de olho a partir de um único local, em vez de precisar efetuar o processo de logon em todos os servidores frequentemente para analisar logs.

É importante lembrar que os serviços rede DNS e NTP deverão também estar configurados corretamente, pois são muito importantes na geração do logs, tanto para melhor identificação dos hosts e respectivos domínios vinculados, como também o horário correto nos servidores para que não exista dificuldade na correlação de logs de servidores diferentes.

Um servidor de log centralizado procurará resolver um nome de domínio totalmente qualificado (FQDN) das máquinas que estão enviando informações para ele, mas se não houver no mínimo um servidor DNS de Cache, como qualquer serviço de rede, se o daemon syslog for incapaz de resolver o endereço, ele continuará tentando até o tempo de "timeout", o que não é interessante.

Da mesma forma, se os horários dos sistemas não estiverem sincronizados, o servidor de log centralizado registrará horários em eventos que não correspondem aos horários da máquina que está enviando as mensagens. Isso tornará a classificação de eventos mais confusa do que o normal.

Para sincronizar os horários e manter a consistência dos logs da rede, devemos editar o arquivo `/etc/ntp.conf` e apontar para uma fonte de tempo centralizada e confiável. Além disso, ativar o `ntpd` para manter sincronizados os horários dos diferentes servidores, ou seja, essa operação tem de ser realizada em todos os servidores.

Comando ‘logger’

O comando `logger` possibilita enviar uma mensagem nos logs do sistema. Dessa forma, essa pode ser uma ferramenta interessante tanto para validar o funcionamento do servidor de log como também para o uso de scripts shell.

A mensagem de log é enviada ao daemon que está gerenciando o log, sendo possível especificar a prioridade, nível e um nome.

```
logger [opções] [mensagem]
```

Onde:

- ▣ “mensagem” é a mensagem que será enviada ao daemon syslog;
- ▣ “opções”:
 - ▣ **-d** usa um datagrama, em vez de uma conexão de fluxo para esse socket;
 - ▣ **-i** registra o PID do processo;
 - ▣ **-s** envia a mensagem para saída padrão (STDOUT) e para o deamon;
 - ▣ **-f [arquivo]** envia o conteúdo do arquivo especificado como mensagem ao deamon;
 - ▣ **-t [nome]** especifica o nome do processo responsável pelo log que será exibido antes do PID na mensagem do syslog;
 - ▣ **-p [prioridade]** especifica a prioridade da mensagem do syslog, especificada como facilidade.nível.

Exemplo:

```
logger -i -t hardening Definir controles de Hardening -i -f /tvar/
log/syslog -p security.warn
```

Estrutura de Servidor e Cliente

As configurações realizadas até agora são para registrar os logs localmente. Apesar de termos todos os logs organizados, não temos a certeza de segurança em relação a eles, pois quando um cracker consegue acesso a uma máquina, uma de suas tarefas será apagar os rastros que deixou. Isso implica apagar os logs ou modificá-los. É necessário tomar provisão em relação a isso adicionando um atributo aos arquivos, o que não é o suficiente.

Vamos trabalhar com o conceito de Servidor e Cliente. Configuremos uma máquina em nossa rede para ser o servidor de logs, e todos os outros servidores serão clientes desse servidor de logs. Os clientes registrarão os logs localmente, mas gravarão seu logs também no servidor de logs. Se um cracker conseguir limpar os rastros no servidor invadido, ele terá de conseguir o acesso ao servidor de logs para limpá-lo também. Isso se ele descobrir que existe um servidor de logs.

Configuração do servidor de logs

Podemos aproveitar toda a configuração que fizemos e acrescentar as seguintes linhas no syslog-ng.conf:

```
# vi /etc/syslog-ng.conf

# OPÇÕES DE ORIGEM
## Origem Máquinas Remotas
source servremotos { udp();};
```

Estamos criando uma nova origem (source), com nome de servremotos. Essa origem aponta para o protocolo udp, o protocolo com que o Syslog e o Syslog-NG trabalham.

Com essa linha, fazemos com que nosso servidor aceite os logs originados pelos outros servidores.

```

# OPCÕES DE FILTRO

## Filtro para o Servidor Remoto 1

filter f_servremoto1 { netmask(192.168.0.1); };

## Filtro para o Servidor Remoto 2

#filter f_servremoto2 { netmask(192.168.0.2); };

```

Estamos criando novos filtros para separar as informações vindas de cada servidor. Na opção `netmask`, estamos especificando os IPs dos servidores remotos.

```

# OPCÕES DE DESTINO

## Destino dos Logs do Servidor 1

destination servremoto1 { file("/var/personal/servremoto1.log"
owner("root") group("root") perm(0640)); };

## Destino dos Logs do Servidor 2

destination servremoto2 { file("/var/personal/servremoto2.log"
owner("root") group("root") perm(0640)); };

```

Essa é uma das grandes vantagens do Syslog-NG em relação ao Syslog. Podemos fazer com que cada servidor remoto tenha o seu arquivo de logs separado, o que ajuda muito na administração, pois podemos ter foco em qual servidor desejamos analisar. Já no Syslog, todos os logs das máquinas remotas ficam nos mesmos arquivos de logs locais do servidor de logs.

```

# OPCÕES DE LOG (Montagem do log)

## Log do Servidor Remoto 1

log { source(servremotos); filter(f_servremoto1); ;
destination(servremoto1); };

## Log Servidor Remoto 2

log { source(servremotos); filter(f_servremoto2); filter(f_authpriv)
destination(servremoto2); };

```

No final, estamos gerando os nossos logs dos servidores remotos. No caso do servidor remoto 1, não especificamos nenhum filtro adicional, então serão aplicados todos os filtros a ele. No servidor 2, só serão registradas informações relacionadas ao filtro `f_authpriv`.

Outro recurso muito interessante do Syslog-NG é o uso de macros, que, além de facilitar muito a criação de regras de logs, ainda possibilita montar organizações de logs melhor estruturadas de forma rápida e objetiva. Algumas macros úteis:

- **DAY**: retorna o número do dia da data corrente;
- **MONTH**: retorna o número do mês da data corrente;
- **YEAR**: retorna o número do ano da data corrente;
- **HOST**: retorna o nome ou IP do servidor;
- **FACILITY**: retorna o nome da facility, o que possibilita “filtrar por facility o log”;

- **PRIORITY:** retorna o nome da priority, o que possibilita “filtrar pelo nível do log”;
- **PROGRAM:** retorna o nome do programa que enviou a mensagem de log; no entanto, não é totalmente confiável, pois depende de como a mensagem é enviada.

A seguir, exemplos de uso, pensando em um contexto de um concentrado de log que vai receber logs de muitos servidores. É necessário separar os logs por servidor e por período, considerando ano, mês e dia, mas esperando o tipo de log (Facility) e o nível da mensagem (Priority):

```
destination servidores_remotos { file("/var/personal/SERVIDORES_
REMOTOS/$HOST/$YEAR/$MONTH/$DAY/$FACILITY.$PRIORITY" owner("root")
group("root") perm(0640)); };

log { source(servremotos) ;destination(servidores_remotos); };
```

Agora, exemplos de uso pensando em um contexto de um concentrado de log que vai receber logs de muitos servidores. É preciso separar os logs por servidor e por período, considerando ano, mês, dia e o tipo de programa que enviou a mensagem de log:

```
destination logs_programas { file("/var/personal/SERVIDORES_
REMOTOS/$HOST/$YEAR/$MONTH/$DAY/$PROGRAM.log" owner("root")
group("root") perm(0640)); };

log { source(servremotos) ;destination(logs_programas); };
```

Configuração do cliente

A configuração dos clientes é mais simples, pois podemos aproveitar a configuração inicial que fizemos e acrescentar as seguintes linhas:

```
# vi /etc/syslog-ng/syslog-ng.conf

# OPCÕES DE DESTINO

destination servlog { udp("192.168.0.10" port(514)); };
```

O nome do destino é servlog, que aponta para o IP do servidor de logs na porta 514/UDP, porta padrão do Syslog e do Syslog-NG.

```
# OPCÕES DE LOG (Montagem do log)

## Registro dos Logs no Servidor

log { source(src); destination(servlog); };
```

Tudo que é de origem local será encaminhado para o servidor de logs. Feito isso, temos de reiniciar o Syslog-NG em ambos os servidores.

```
# /etc/init.d/syslog-ng restart
```

Para testar, podemos reiniciar algum serviço no servidor 1 e no servidor 2 e verificar se o servidor de logs registrará as informações nos arquivos determinados.



Configuração do STUNNEL

Devido à simplicidade do protocolo Syslog, em alguma implementação pode ser interessante realizar comunicação entre o cliente e o servidor de log através de um canal criptografado. Para isso, primeiro devemos instalar o STUNNEL. Esse programa cria um canal criptografado entre o servidor syslog-ng e os clientes.

```
# apt-get install stunnel sslcert
```

Altere o valor da variável ENABLE:

```
# vi /etc/default/stunnel4
```

ENABLED=1Configuração do servidor, ou seja, as configurações do servidor de concentrador de logs:

```
# vi /etc/stunnel/stunnel.conf

cert = /etc/stunnel/syslogserver.pem

key = /etc/stunnel/syslogserver.pem

; Servicelevelconfiguration

[5140]

accept = 5140

connect = 514
```

Em seguida, é necessário configurar o cliente, ou seja, todos os servidores que encaminharão logs:

```
# vi /etc/stunnel/stunnel.conf

cert = /etc/stunnel/syslogclient.pem

key = /etc/stunnel/syslogclient.pem

client = yes

; Servicelevelconfiguration

[5140]

accept = 514

connect = 5140
```

É necessário criar os certificados do servidor e do cliente:

```
# makesslcert /usr/share/sslcert/ssleay.cnf /etc/stunnel/syslogserver.pem

# makesslcert /usr/share/sslcert/ssleay.cnf /etc/stunnel/syslogclient.pem
```

Após criar os certificados, será copiado o certificado do cliente, alterada sua permissão e iniciado o serviço do STUNNEL.

```
# scp /etc/stunnel/syslogclient.pem sysadmin@remoteclient:/etc/stunnel/
# ssh sysadmin@remoteclient chmod 400 /etc/stunnel/syslogclient.pem
# /etc/init.d/stunnel4 start
```



Definição da política de rotacionamento de logs

Todos os logs que criamos serão valiosos para analisarmos o nosso sistema, mas eles tendem a crescer muito rápido, o que pode ocasionar estouro de disco ou da partição onde está o diretório de logs. Pensando nisso, podemos criar uma política de logs e utilizar o recurso de rotacionamento de logs nativo do sistema, que é o LogRotate.

O arquivo de configuração do rotacionamento é o `/etc/logrotate.conf`, no Debian, e o `/etc/rotate.conf`, no Red Hat. Vamos visualizar algumas opções globais do LogRotate.

```
# vi /etc/logrotate.conf  
weekly
```

Essa opção faz com que os logs sejam rotacionados semanalmente.

```
rotate 4
```

Rotate 4 define que serão mantidos os quatro últimos rotacionamentos, para não fugir do controle.

```
mail root
```

Essa opção faz com que, em caso de erro de não existência dos logs, eles sejam enviados para o usuário root.

```
create
```

Com essa opção, estamos configurando para que sejam criados novos arquivos de log (vazios) após os antigos rodarem.

```
compress
```

Essa opção faz com que as cópias de logs sejam compactadas, mantendo sempre o último “rodado” descompactado.

Não vamos utilizar as configurações globais. Vamos montar uma política de rotacionamento para toda a estrutura de logs personalizada que criamos no `/var/personal`. Assim, serão verificados os logs diariamente e, caso o arquivo alcance o tamanho de 3 MB, ele será “rotacionado” e, após o processo o Syslog, será reinicializado para que continue trabalhando normalmente. Esse processo será feito no máximo cinco vezes, o que é representado por `rotate 5`. Quando chegar ao sexto rotacionamento, ele apaga o primeiro e renomeia os outros.

Vamos mostrar alguns logs como exemplo. A estrutura pode ser aplicada a todos os logs, basta adicioná-los.

Podemos adicionar essa estrutura no final do arquivo `logrotate.conf`.

```
/var/personal/*.err /var/personal/*.info /var/personal/*.notice {  
    daily  
    size 3M  
    sharedscripts  
    postrotate  
        /usr/bin/killall -1 syslogd
```

```
    endscript  
  
    rotate 5  
  
}
```

Exemplo de configuração de logrotate com compressão bzip2:

```
/var/log/*.log {  
  
    missingok  
  
    rotate 10  
  
    daily  
  
    compress  
  
    delaycompress  
  
    compresscmd /usr/bin/bzip2  
  
    compressoptions --best  
  
    compressesext .bz2  
  
    create 0644 root root  
  
}
```

Exemplo mais arrojado de configuração com compressão bzip2 e mantendo logs antigos separados em outro diretório:

```
/var/personal/ {  
  
    maillast sandro.melo@grupobem.com.br  
  
    missingok  
  
    size=100M  
  
    olddir /var/personal/logs_antigos  
  
    rotate 5  
  
    weekly  
  
    compress  
  
    delaycompress  
  
    compresscmd /usr/bin/bzip2  
  
    compressoptions --best  
  
    compressesext .bz2  
  
  
    create 0644 root root  
  
    sharedscripts
```

```
postrotate  
/opt/apache/bin/apachectl restart  
endscript  
}
```

Mesmo que o rotacionamento seja diário, podemos forçá-lo a rodar quando quisermos.

Basta executar o seguinte comando:

```
# logrotate /etc/logrotate.conf
```

Depois, visualize o diretório de logs e veja o que aconteceu.

```
# ls -l /var/personal
```

O controle das ações do logrotate é feito pelo utilitário de agendamento de tarefas do sistema, o cron. O cron possui uma configuração específica para tarefas que são rodadas durante certo período (diariamente, mensalmente etc.).

Confira o script de controle das ações do logrotate no cron:

```
# cat /etc/cron.daily/logrotate
```

Verifique em qual o horário o logrotate executará.

```
# cat /etc/crontab
```

Podemos ver que o rotacionamento acontecerá periodicamente, de acordo com as configurações. Dessa forma, os arquivos não crescerão eternamente.

Servidor NTP

O servidor NTP tem por função estabelecer uma sincronia entre o relógio do sistema e a referência mundial de tempo, o padrão Horário Universal Coordenado (UTC), através do protocolo NTP. Isso traz a vantagem de o usuário não precisar ficar alterando o relógio do seu sistema, pois ele estará somente buscando as configurações de um servidor. Ele trabalha com uma associação de hierarquia dos servidores, que são denominadas stratum.

O stratum 0 é a origem da hierarquia dos servidores de tempo, onde estão os equipamentos de fonte de horários reais precisos (receptores GPS e relógios atômicos). Logo abaixo, no stratum 1, estão os servidores públicos diretamente sincronizados com os equipamentos de precisão. No stratum 2, estão os servidores públicos utilizados por padrão pelos demais servidores e clientes ntp do mundo, para evitar sobrecarga no stratum 1.

Por questões de tráfego de rede, configuraremos no nosso exemplo um único servidor NTP. As demais máquinas atualizarão o seu horário a partir desse servidor, sendo clientes dele:

No servidor

1. Instalar o Servidor NTP.

```
# apt-get install ntp ntp-server
```

2. Vamos entrar agora no arquivo de configuração do servidor NTP:

```
# vi /etc/ntp.conf
```



Saiba mais

O sistema Linux proporciona um grande número de logs, o que possibilita ao administrador ter grande controle sobre o sistema. Sabendo administrar esses logs corretamente, eles se tornam uma arma poderosa para que seja possível prevenir erros e até mesmo responder melhor as tentativas de invasão.

Nós veremos somente algumas opções desse arquivo, pois a configuração do servidor é bem simples.

- ▣ **server 127.127.1.0** nesse parâmetro, é configurado o IP ou nome do servidor ao qual o ntp deve se sincronizar (por exemplo, <http://ntp.cais.rnp.br> é um exemplo de servidor de stratum 2 no Brasil);
 - ▣ **statsdir /var/log/ntpstats/** diretório onde vão ficar os logs de estatísticas do meu servidor NTP;
 - ▣ **driftfile /var/lib/ntp/ntp.drift** arquivo onde ficará configurado o valor estimado de erro de frequência entre o relógio do sistema e o servidor de sincronia de stratum anterior.
3. Que comando executo para monitorar a minha comunicação com os servidores de sincronização?

```
# ntpq -p 192.168.100.100
```

4. Onde posso encontrar referências aos servidores NTP públicos de stratum 1 e 2?

<http://ntp.isc.org/bin/view/Servers/WebHome>

5. Depois disso, vamos reiniciar o daemon do ntp-server para sincronizar e passar a ser um servidor NTP.

```
# /etc/init.d/ntp-server restart
```

6. Vamos verificar o log em outro terminal para ver o processo de sincronização.

```
# tail -f /var/log/syslog
```

Nos clientes

1. Instalar o cliente NTP.

```
# apt-get install ntpdate
```

2. Para as máquinas da rede que forem os clientes NTP, é possível fazer a sincronização do horário com o servidor através do comando *ntpdate*:

```
# ntpdate 192.168.100.100
```

3. Caso não exista um servidor NTP na rede, podemos simplesmente reiniciar o daemon do *ntpdate*.

```
# /etc/init.d/ntpdate restart
```

Contabilização de processos

Tem por objetivo apoiar decisões relacionadas a “plano de capacidade” ou a auditoria de um processo que porventura esteja consumindo muito recurso da máquina.

- ▣ Contabilização dos processos pode também possibilitar controle sobre quais processos e quais respectivos usuários.
 - ▣ Para isso, basta usar o comando *lastcomm*, que não vem instalado por padrão na maioria das distribuições Linux. É necessário instalar o pacote *acct*.

Instalação do pacote *acct*, que traz o comando *lastcomm*.

```
# apt-get install acct
```

Verifique se o arquivo `/var/account/pacct`, que é onde fica a informação de contabilização, foi criado.

```
# ls /var/account/pacct
```

Teste o `lastcomm`.

```
# lastcomm
```

Dessa maneira, é mostrada uma visão geral de todos os últimos comandos digitados pelos usuários. Para consultar informações de somente um usuário, execute o `lastcomm` da seguinte forma.

```
# lastcomm root
```

```
# lastcomm toor
```

Podemos também fazer a consulta pelo comando no lugar do usuário.

```
# lastcomm ls
```

Conclusão

Registro de eventos (logs) e auditoria de comandos são atividades demandadas em qualquer ambiente de TI. A importância de se ter um ambiente com a capacidade de ser auditada, com o registro da trilha de todos os comandos, é enorme.

Pensando em uma situação de alta criticidade em que seja necessário responder a um incidente de segurança ou mesmo ter informações para uma perícia forense computacional, ter uma estrutura de logs organizada com réplicas em outro servidor é vital.



Saiba mais

Todas as informações do comando `lastcomm` estão dentro do arquivo `/var/account/pacct` e só podem ser visualizadas através de comandos como o `lastcomm`.

Auditoria com HIDS

Uso de HIDS para identificação de modificações em um servidor:

- Pode ser um mecanismo funcional não somente para identificar mudanças não programadas, mas também mudanças que podem ter ocorrido em um incidente de segurança.

AIDE:

- É um HIDS prático, de implementação simples e com certeza soma em um processo de Hardening, criando a possibilidade de realização de auditoria para verificar mudanças no sistema.

Para instalar o AIDE:

```
apt-get install aide
```

O diretório padrão onde fica o arquivo de configuração é o `/etc/aide`, e o arquivo é o `aide.conf`.

Outro diretório importante é onde ficam as principais regras de onde e como deverão ser identificadas as informações sobre logs:

```
# cd aide.conf.d/
```

Os arquivos da base são criados em `/var/lib/aide` com as informações para auditoria:

```
/var/lib/aide/aide.db -
```

```
/var/lib/aide/aide.db.new
```

É provável que as configurações padrão que indicam onde o AIDE deve verificar não atendam a todos os casos, e consequentemente ajustes deverão ser feitos. Uma interessante modificação é não auditar diretórios que possuem frequentes modificações, como é o caso de arquivos de logs. O AIDE não é a melhor ferramenta nesse caso: o interesse é ter foco em arquivos e diretórios de configuração importantes. Assim sendo, usar os recursos de “exceção” é uma boa solução:

Edite o arquivo `aide.conf` ignorando toda a pasta `/var/log`, `!/var/spool/` e `/var/cache/squid`.

```
#vim aide.conf

All=R+a+sha1+rmd160

/etc p+i+u+g # check somente permissões, inode, usuários
e grupos do etc

/bin All # aplica as regras definidas na variável All para
todos os arqs do /bin

/sbin All # idem /sbin

/var All # idem /var

/usr/bin All # idem /usr/bin

/usr/sbin All # idem /usr/sbin

/etc ConfFiles

#
!/var/log/* # ignora o diretório /var/log
!/var/spool/* # ignora o diretório de spool
!/var/cache/squid.

*Após fazer qualquer modificação, é necessário executar o comando:
# update-aide.conf
```

A ferramenta Ambiente Avançado de Detecção de Intruso (AIDE) cria uma base de dados para comparações posteriores a partir de alguns algoritmos de hash criptográficos e informações de metadados dos arquivos, como grupo, dono, tamanho e hora de modificação.

Uso do AIDE

Criar uma cópia do arquivo de configuração no diretório `/var/lib/aide`, chamado `aide.conf.autogenerated`, que vai ser usado pelo comando gerador de bases do AIDE.

```
# cd /var/lib/aide
# cp aide.conf autogenerated aide.conf
# aide --init
```

Foi gerado o arquivo `/var/lib/aide/db.aide.new`. Para ser feita a auditoria, é necessário renomear o arquivo para `/var/lib/aide/db.aide` ou usar o comando:

```
# aideinit
```



Quando executado, o comando *aideinit* criará o arquivo *var/lib/aide/db.aide.new* e também o */var/lib/aide/db.aide*.

Realização de uma auditoria:

```
# nice -n -19 aide -C --config=/var/lib/aide/aide.conf | tee /etc/  
relatorio.txt
```

Os Host Intrusion Detection System (HIDS) são um mecanismo importante na identificação de malwares e de modificações às configurações do sistema. Dessa forma, mesmo que a norma seja enfática na questão de vírus (que, no contexto de sistemas Like Unix, são diferentes de sistemas Microsoft devido à forma como suas propostas de DAC foram desenhadas), devemos levar em consideração suas recomendações, trazendo-as para o contexto do Sistema Operacional Linux.





Roteiro de Atividades 3

Atividade 3.1 – Hardening Linux – Registro de eventos (logs)

1. Configure a variável HISTTIMEFORMAT no .bashrc dos usuários sysop e syadmin.

2. Configure a variável HISTTIMEFORMAT no .bashrc do /etc/skel.

3. Configure a variável HISTFILE no .bashrc do usuário root para que seja possível ter uma trilha diferente para cada usuário que fizer su para root.

4. Configure o Syslog-ng para estruturar os logs gravando-os e organizando-os em arquivos, com a FACILITY e a PRIORITY. Apresente suas observações.

5. Crie uma regra no Syslog-ng para concentrar em um único arquivo todos os logs.

6. Crie um servidor de Log remoto e realize testes.

7. Crie um canal seguro entre o cliente e o servidor de logs utilizando o STUNNEL.

Atividade 3.2 – Auditoria HIDS

1. Após a configuração, deve-se inicializar o AIDE. Para melhor contextualização, será necessário criar uma simulação de modificação de arquivos estratégicos, modificando metadados e conteúdo. Para isso, execute os seguintes comandos:

```
# echo # >> /etc/hosts.allow  
# echo # >> /etc/hosts.deny  
# echo # >> /etc/services  
# chmod 700 /etc/passwd  
# chmod 700 /etc/shadow  
# chmod 700 /etc/services
```

2. Ativar a contabilização de processos e usar o comando *lastcomm* para listar informações apuradas de contabilização dos processos.



3. Realize uma auditoria com o AIDE.



4

Serviços de Redes – Parte 1

objetivos

Identificar serviços de rede ativos; Reforçar a segurança do serviço SSH; Implementar mecanismo contra ataque de Força Bruta.

conceitos

Menor Privilégio e Menor Recurso para serviços de rede; Serviços TCP e UDP; Hardening de SSH.

Checklist nos serviços do sistema

Instalação e configuração de serviços básicos no Linux:

- Não é complicado. Depois da instalação, já estarão funcionando.
 - No entanto, é preciso avaliar, testar e revisar suas configurações.



Instalar e configurar serviços básicos em um sistema Linux não é tão complicado. Muitos serviços, depois de instalados, já estão funcionando, mas com configuração padrão. Alguns já vêm instalados e habilitados por padrão já na instalação do sistema Linux. Entretanto, ao pensar em segurança, sabemos que não se deve colocar um serviço em produção com configurações padrão sem que estas sejam avaliadas, testadas e revisadas.

Muitos administradores, talvez por serem iniciantes, não sabem tirar proveito de ferramentas clássicas para verificar e avaliar se o serviço está funcionando corretamente, além de verificar em qual porta o serviço está trabalhando e se está recebendo requisições corretamente. Em alguns casos, também é necessário ver se nenhum incidente de segurança motivou alguma mudança no funcionamento do serviço ou mesmo se foi um vetor de ameaça para um acesso indevido que possibilitou um invasor fazer alterações no sistema que poderiam prejudicá-lo.

Diante desse cenário, começaremos este capítulo mostrando alguns comandos que podem formar uma lista de ações e recursos que poderão ser chamados de “checklist”, o que permitirá uma avaliação, embora pontual, mais interessante dos serviços de redes ativos.

Primeiramente, é recomendável identificar a porta definida pelo serviço, mas isso geralmente é uma tarefa simples, pois os serviços possuem portas pré-definidas. Podemos ver algumas portas padrão do sistema no arquivo `/etc/services`. Exemplos de algumas portas:

```
# cat /etc/services
# cat /etc/services | grep -i ssh
```

```
ssh      22/tcp    # SSH Remote Login Protocol  
ssh      22/udp
```

Após ter identificado as informações das portas em que os serviços são ativados, devemos usar algum comando que permita consultar informações de porta, como o *netstat*, *lsof* e *fuser*. Esses comandos vão fornecer algumas informações sobre os serviços de rede do ativo no sistema. É válido lembrar que são encontradas implementações do *netstat* em outros Sistemas Operacionais Unix e também Windows, mas que não necessariamente possuem as mesmas opções ou os recursos funcionarão da mesma forma.

```
# netstat -nltp
```

Onde:

- **-n**: é a opção para fazer o *netstat* não resolver os IPs para nomes;
- **-l**: lista os sockets que estão ouvindo (listen), ou seja, que estão prontos para receber uma conexão;
- **-t**: lista somente os sockets no protocolo TCP. Podemos também utilizar o **-u** para protocolos UDP;
- **-p**: lista informações sobre o processo do serviço vinculado à porta consultada.

Com isso, podemos ver várias informações sobre os sockets que estão aguardando conexões.

Exemplo:

```
Proto Recv-Q Send-Q Endereço Local Endereço Remoto Estado  
tcp      0      0      0.0.0.0:22          0.0.0.0:*      OUÇA
```

- **Proto**: é o protocolo em que esse socket está trabalhando;
- **Endereço Local**: é indicado que a porta 22 pode responder a qualquer interface de rede no sistema (0.0.0.0:22);
- **Endereço Remoto**: mostra que não há nenhuma conexão relacionada àquela porta, cujo Estado é OUÇA, ou seja, aguarda conexões.

Podemos usar o *netstat* com o **-a** no lugar do **-t** para consultar todos os “sockets”, tanto os que estão ouvindo como os que não estão ouvindo e os com “conexões estabelecidas”, o que é muito interessante para o sysadmin. Quando se solicita informações de “conexões estabelecidas”, podemos ver a qual interface e porta (cliente e do serviço) está conectada, e a origem da conexão.

```
# netstat -nat  
  
Proto Recv-Q Send-Q Endereço Local     Endereço Remoto       Estado  
tcp      0      0      10.0.0.1:32778    200.123.123.123:143  ESTABELECIDA  
  
tcp6     0      0      ::ffff:10.0.0.1:22  ::ffff:192.168.0.8:32796  ESTABELECIDA
```

Com a saída do comando *netstat*, anteriormente ilustrada, sabemos quais sockets estão disponíveis e podemos ver qual processo (serviço) está rodando nesse socket com o comando *fuser*.

```
# fuser -v 22/tcp
```

	USER	PID	ACCESS	COMMAND
22/tcp	root	3943	f....	sshd

Onde `-v` é o modo verbose.

Podemos ver qual usuário está rodando o processo, o PID (número do processo) e o programa que está rodando, que nesse caso é o `sshd`. Para cada processo que está rodando, o sistema cria um diretório com seu PID no `/proc`, onde é possível obter algumas de suas informações. Uma delas é o arquivo `cmdline`, que mostra o caminho completo do programa e nos permite saber que aquele programa é real, e não um programa forjado.

```
# cat /proc/3943/cmdline
```

```
/usr/sbin/sshd
```

Outra ferramenta que podemos usar é o `lsof`, que nos mostra informações parecidas com o `netstat` e o `fuser`, só que em uma única ferramenta. O comando `lsof` mostrará os sockets que estão ouvindo, que estão estabelecidos e, entre outros, nos mostrará também o usuário, o PID e o programa que está rodando. Ou seja, percebemos que o comando `lsof` é uma junção entre o `netstat` e o `fuser`. Porém, em algumas distribuições Linux ou mesmo dependendo da instalação, o comando `lsof` pode não ser instalado por padrão, sendo necessário instalá-lo via pacotes ou pelo source.

```
# lsof -i
```

```
firefox-b 4652 user 36u IPv4 22723 TCP 10.0.0.1:38138->64.233.161.19:www (ESTABLISHED)
```

```
ssh 8342 root 3u IPv4 100376 TCP 10.0.0.1:54751->192.168.0.8:ssh (ESTABLISHED)
```

Onde `-i` lista todos os processos ou arquivos relacionados a uma interface de rede. Se não for especificada uma interface de rede, como ilustrado, ele mostrará todas as interfaces.

Uma coisa que muitos administradores também se esquecem de olhar são os raw sockets, sockets que não dependem de um protocolo específico e permitem acesso direto a protocolos de baixo nível como ICMP, TCP, UDP e IP.

Podemos ver se existe algum raw socket aberto no nosso sistema utilizando o `netstat` com outra opção.

```
# netstat -nlw
```

Onde `-w` lista os raw sockets.

Provavelmente não haverá nenhum aberto via Raw Socket, então se retorna uma resposta vazia. Mas deve-se também ter muita atenção com a comunicação Raw Sockets, pois é comum backdoors mais sutis que utilizam o raw socket. Um bom exemplo é uma backdoor que utilize o protocolo ICMP para permitir a execução de comandos remotamente.

Saiba mais

Os raw sockets são considerados por muitos como um potencial foco de problemas de segurança devido ao fato de que poucos administradores verificam as atividades de raw sockets de seus sistemas.

Saiba mais

Veja mais detalhes na documentação do hping.

Saída de uma resposta positiva, utilizando o *netstat* para listar os raw sockets:

Proto	Recv-Q	Send-Q	Endereço Local	Endereço Remoto	Estado
raw	0	0	0.0.0.0:255	0.0.0.0:*	7

Um último programa sugerido, que pode ser usado para validação de portas ativas, é o *nmap*, um **portscanner** muito poderoso para varreduras de portas. Nesse caso, temos a possibilidade de fazer uma análise interna ou externa de um determinado servidor ou grupo de servidores para que seja possível inventariar quais portas/serviços estão disponíveis. O *nmap* não vem por padrão nas distribuições Linux, sendo necessário instalá-lo.

Exemplo de uso para portas TCP:

```
# nmap -sT -n -P0 10.0.0.1
```

PORTE	ESTADO	SERVICE
9/tcp	open	discard
13/tcp	open	daytime
21/tcp	open	ftp
22/tcp	open	ssh

Onde:

- **-sT**: faz a varredura completa de portas TCP;
- **-n**: não resolver nomes via DNS;
- **-P0**: não realizar teste para verificar se o servidor alto está ativo (icmp Request e ack na porta 80);
- **10.0.0.1**: representa seu IP.

Exemplificação de uso *nmap* tendo como alvo a interface loopback:

```
# nmap -sT -n -P0 localhost
```

```
# nmap -sT -n -P0 -p 22 localhost
```

PORTE	ESTADO	SERVICE
22/tcp	open	ssh

Exemplo de uso para portas UDP:

```
# nmap -sU -n -P0 10.0.0.1
```

PORTE	ESTADO	SERVICE
9/udp	open filtered	discard

Portscanner

Ferramenta que faz uma varredura nas portas, testando a segurança do firewall e verificando se as portas estão abertas ou fechadas para acesso através da internet.



```
53/udp    open|filtered domain  
111/udp   open|filtered rpcbind
```

Onde:

- **-sT**: faz a varredura completa de portas TCP;
- **10.0.0.1**: representa seu IP.

No contexto local:

```
# nmap -sU -n -PO localhost  
  
# nmap -sU -n -PO -p 53 localhost  
  
PORT      STATE            SERVICE  
53/udp    open|filtered  domain
```

Exemplo de uso para portas TCP e UDP simultaneamente:

```
# nmap -sU -sT -n -PO 10.0.0.1
```

Em alguns casos, o uso do nmap pode ser interessante para fazer o checklist de todas as portas ativas no servidor.

```
# nmap -sU -sT -n -PO -F 10.0.0.1
```

Onde **-F** representa todas as portas de serviços clássicos no (/etc/services do nmap). Ou usar a opção **-p-** porta a porta até de 1 a 65535. Sendo assim, é possível consultar todas as portas que estão disponíveis em nosso sistema.

```
# nmap -sU -sT -n -PO -p- 10.0.0.1
```

Podemos ver também os banners dos serviços que estão rodando. Para um invasor, essa pode ser uma informação preciosa.

```
# nmap -sV -n -PO 10.0.0.1  
  
PORT      STATE SERVICE VERSION  
22/tcp    open  ssh      OpenSSH 3.8.1p1 (protocol 2.0)  
25/tcp    open  smtp     Exim smtpd 4.50  
111/tcp   open  rpcbind 2 (rpc #100000)  
113/tcp   open  ident    OpenBSD identd  
  
MAC Address: 00:50:BF:63:D0:E7 (Mototech)
```

O nmap possui muitas opções. Podemos consultar em outro momento seu manual, para ver todas as opções de uso.

Não devemos deixar o nmap instalado em um servidor sem controles rígidos de permissionamento, pois, por padrão, um usuário comum pode executá-lo para realização de varredura do tipo TCP Connect. Apesar de ser uma ferramenta que fornece informações preciosas, ele pode fornecer essas mesmas informações para um usuário mal-intencionado ou mesmo para alguém que teve algum tipo de acesso arbitrário ao sistema.

Exercício de fixação 1

Checklist de Rede

Cenário de um incidente de segurança – Estudo de caso

Considere o cenário proposto em um incidente de segurança ocorrido na “Fábrica de Brinquedos – Gift & Gift Ltda.”, onde o administrador de um servidor DNS achou estranho o acesso ao disco, uma vez que já eram mais de 23h de uma sexta-feira, e o acesso está muito grande. Naquela hora não havia ninguém mais na empresa, além dele, que estava acompanhando uma migração de um dos servidores de bancos de dados.

Ao consultar a máquina, o administrador percebeu que, além da porta 53 do DNS e 25 do MTA Exim, destinado a mensagens internas do sistema, havia uma outra porta ativa, a porta 666, como indicado na tabela 4.1.

Conexões Internet Ativas (sem os servidores)

```
Proto Recv-Q Send-Q EndLocal EndRemoto Estado PID/Program name
tcp 0 0 0.0.0.53 0.0.0.* OUÇA 2747/portmap
tcp 0 0 127.0.0.1:25 0.0.0.* OUÇA 3106/exim4
tcp 0 0 0.0.0.666 0.0.0.* OUÇA 3628/inetd
```

Figura 4.1
Tabela de serviços TCP em estado ‘Listen’.

Command Pid User FD Type Device Size Node Name

```
bind 2747 daemon 3u IPv4 10629 UDP *:dns
exim4 3106 Debian-exim 3u IPv4 12046 TCP localhost:smtp (LISTEN)
bind 3323 root 6u IPv4 12845 UDP *:dns
inetd 3676 root 4u IPv4 20467 TCP *:sombra (LISTEN)
```

Figura 4.2
Tabela de saída do comando ‘lsof’.

Conexões Internet Ativas (servidores e estabelecidas)

```
Proto Recv-Q Send-Q EndLocal EndRemoto Estado
tcp 0 0 0.0.0.111 0.0.0.* OUÇA
tcp 0 0 127.0.0.1:25 0.0.0.* OUÇA
tcp 0 0 0.0.0.666 0.0.0.* OUÇA
tcp 4234 0 92.168.0.150:666 212.168.0.171:32776 ESTABELECIDA
tcp 0 978 212.168.0.171:32776 92.168.0.150:666 ESTABELECIDA
```

Tabela 4.3
Tabela de saída do comando ‘netstat’.

Intrigado com a porta 666, o administrador resolveu consultar mais detalhes sobre o processo que estava ativo na respectiva porta. Buscou mais informações com o comando *lsof* – a saída do comando *lsof* é mostrada na tabela 4.2.

Notou um nome estranho e muito sugestivo para o respectivo serviço, o que estava notoriamente qualificando como uma possível backdoor. Com essa identificação, a prova de uma violação do sistema está qualificada, caracterizando um incidente de segurança crítico.

Diante dessas informações, o administrador resolveu coletar mais informações das atividades de conexões estabelecidas com o servidor, buscando identificar se a suposta Backdoor estava sendo utilizada. Usou novamente o comando *netstat*, objetivando ter todas as informações sobre as conexões estabelecidas. A saída do comando *netstat* é mostrada na tabela 4.3.

Antes de tomar qualquer ação reativa, o administrador precisava saber que aplicação estava ativa na respectiva porta. Então, habilmente recorre novamente ao shell do sistema, mas dessa vez usou o comando *fuser* para ter mais informações.

Ao usar o comando *fuser*, como é ilustrado na figura 4.1, ficou claro que o processo vinculado ao deamon inetd na realidade é a shell bash, o que deixava notório que aquela porta era uma clássica backdoor implementada no *inetd.conf*, o que também ratificava o fato de o servidor em questão estar comprometido.

```
xninja# fuser -v 666/tcp
USER PID ACCESS COMMAND
666 root 2321 f . . . . inetd
root 2321 f . . . . bash
xninja#
```

Figura 4.1
Saída do comando 'fuser'

Diante do que foi relatado, assinale V (verdadeiro) ou F (falso):

- () Uma Backdoor como a ilustrada pode ser instalada por um invasor, mas também por um insider (empregado mal-intencionado) ou um gray hat (pseudo consultor prestador de serviço).
- () Uma backdoor dessa categoria é tão engenhosa que não é detectável no sistema, mesmo por um administrador experiente que realize o procedimento de checklist de serviços de rede, relacionando os processos e serviços ativos.
- () É muito comum em uma invasão de sistemas o invasor instalar rootkits que, além de implementarem backdoors no sistema, adulteraram binários de comandos clássicos (*netstat*, *fuser*, *ps*, *lsof* e outros) para evitar que o administrador as identifique. Entretanto, a backdoor encontrada foi implementada utilizando recursos do próprio sistema.

Runlevel

Depois de terem sido realizadas todas as checagens e elencados todos os serviços que estão ativos no nosso sistema, é importante avaliar se realmente é necessário que todos os serviços inicialmente identificados como ativos deveriam estar ativos. Caso não, podemos desativar os que não são necessários, melhorando a performance e a segurança do sistema.

No Debian, podemos consultar os diretórios dos níveis de execução (run level), baseados no System V, que é onde ficam todos os programas que serão ativados na inicialização do sistema.

O nível de execução padrão do Debian é o nível 2. Cada distribuição Linux organiza seu sistema de níveis de execução de uma maneira diferente. Para identificar como cada uma delas trabalha, podemos consultar o arquivo */etc/inittab*, que controla o processo init, o primeiro processo a ser iniciado no sistema. Consultando o diretório do nível de execução padrão do Debian para identificar os serviços inicializados por padrão:

```
# ls -l /etc/rc2.d
```

Repare que são todos links simbólicos que apontam para o diretório `/etc/init.d`, diretório onde ficam todos os scripts que controlam os daemons do sistema. Caso não queira que determinado serviço seja ativado na execução, basta apagar o link simbólico dele do diretório do nível de execução que está sendo utilizando, que nesse exemplo é o runlevel 2.

Como exemplo, podemos remover o link simbólico do exim4, um servidor de e-mail padrão do Debian. Todavia, em muitos casos um MDA interno é relevante para os serviços que ficaram ativos, tudo é uma questão de qual a finalidade do servidor. Imagine que a finalidade nesse exemplo é de fato um servidor firewall: não existe necessidade de deixarmos um servidor de e-mail rodando com um MTA, o que deixa sua porta SMTP (25) aberta, possibilitando que um cracker aproveite-se dela, mas em muitos casos com MDA interno pode ser interessante.

```
# rm -f S20exim4
```

ou

```
# mv S20exim4 K20exim4
```

Com isso, quando o sistema iniciar no nível 2 padrão, o exim4 não iniciará mais. É possível fazer isso com todos os serviços que forem classificados como desnecessários e personalizar também os outros níveis de execução.

Não deve-se esquecer também de verificar o diretório `/etc/rcS.d`, pois nele ficam os links simbólicos dos serviços que são iniciados antes dos serviços do nível de execução padrão. Mesmo que esses serviços sejam primordiais para o funcionamento do seu sistema, é uma boa prática verificar se não existe algum serviço que seja considerado desnecessário para essa ocasião.

Standalone versus Super Daemon

O modelo Inetd (Network Super Daemon), onde `/etc/services` mapeia o nome do serviço a número de porta. Exemplo:

```
ulistserv 372/tcp ulistproc  
/etc/inetd.conf : Arquivo principal de configuração  
ftp stream tcp nowait root /usr/sbin/tcpd proftpd
```

O substituto para inetd e do aplicativo Xinetd desenvolvido originalmente para RedHat e portado para outras distribuições. O Xinetd é considerado mais flexível, e funciona com um mecanismo de controle de acesso mais elaborado, o que torna-o mais seguro.

- ▣ `/etc/xinetd.conf`: arquivo de configuração principal do Xinetd;
- ▣ `/etc/Xinetd/`: contém arquivos para serviços gerenciados pelo Xinetd.

Gerenciamento de serviços de redes em Inetd e Xinetd

Para ativar um serviço via Inetd, basta descomentar a correspondente linha de configuração do serviço no arquivo `inetd.conf` e depois reiniciar o servidor inetd.

Reiniciar o daemon Inetd:

```
# pkill -HUP inetd
```

Outra forma era executar o deamon do inetd:

```
# service inetd restart
```

Caso o Super Deamon seja baseado no Xinetd, que é uma evolução do inetd desenvolvida pela Redhat.

O arquivo de configuração principal é o *xinetd.conf*. Fazer mudanças em xinetd.conf é algo muito específico, pois é um arquivo muito simples, mas habilitar ou desabilitar serviços demanda editar as configurações concentradas no diretório */etc/Xinetd*.

Arquivo de configuração *xinetd.conf*, onde a recomendação trivial é ativar o log via Syslog.

```
defaults

{

    # Please note that you need a log_type line to be able to use
    log_on_success

    # and log_on_failure. The default is the following :
    log_type = SYSLOG daemon info

}

includedir /etc/xinetd.d
```

O Xinetd possibilita melhorias de controle de acesso para serviços, o que é um ótimo motivo para usar Xinetd em vez do antigo inetd. Outro ponto positivo é que as configurações de serviços estão separadas por arquivos, o que traz uma modularidade que possibilita melhor administração e organização.

Toda mudança de configuração, tanto no *xinetd.conf* como nos arquivos dentro do diretório */etc/Xinetd*, demanda que o serviço seja reinicializado:

```
# /etc/rc.d/init.d/xinetd restart
```

Durante o processo de Hardening, deve-se identificar os serviços de rede ativos via Super Deamon e avaliar se realmente devem estar ativos, caso não devem ser desativados. Típicos serviços que devem ser bloqueados, como:

- ▣ Finger, echo e ntalk;
- ▣ Telnet (use o ssh ou um servidor telnet com ssl);
- ▣ FTP (se possível, user sftp ou scp);
- ▣ rwho, rsh , rlogin e rexec (recomenda-se ssh, scp e sftp).

Uma recomendação é verificar o permissionamento dos arquivos do diretório */etc/Xinetd*, em "600":

```
# chmod -v 600 /etc/xinetd.d/*
```

E antes de ir para produção, definir os atributos do arquivos para imutável:

```
# chattr +i /etc/xinetd.d/* && lsattr /etc/xinetd.d/*
```

E também é recomendado rever o permissionamento dos scripts dos deamons que ficam concentrados em `/etc/init.d`. Esse scripts devem estar disponíveis para leitura apenas para o root:

```
# chmod -V 700 /etc/rc.d/init.d/*
```

Gerenciando inicialização de serviços

Em distribuições like Debian, como Ubuntu, existe alguma forma de gerenciar os que devem ou não serem ativados no boot do sistema. Entre as formas de gerenciamento, destacam-se:

- update-rc.d;
- service;
- rcconf;
- invoke-rc.d.

Segurança em serviços de redes



Boas práticas de segurança de redes:

- Aplicação de controles e segregação nos serviços de redes ativos.
 - Usar configuração combinada entre dois arquivos:
 - TCPWRAPPERS.

No contexto de serviços de redes, entre as boas práticas de segurança de redes, é importante a aplicação sempre que possível de controles segregação nos serviços de redes ativos, principalmente nos serviços administrativos. Para isso, pode ser interessante usar a configuração combinada entre dois arquivos, para limitar o uso dos serviços de rede, que configuraram o recurso conhecido como TCPWRAPPERS (`/etc/hosts.deny` e `/etc/hosts.allow`), combinada com limitações de conexão configuradas via o recurso PAM (`/etc/pam.d` e `/etc/security`).

Isso tudo, posteriormente aliado a uma política bem definida de Firewall, e com

uma estrutura de registro de eventos (logs), vai com certeza agregar ainda mais segurança ao sistema.

Utilizando o TCPWRAPPERS, podemos restringir o acesso de todos os serviços de rede e liberá-lo somente aos IPs que desejados.

É importante destacar que o serviço de rede em questão tem de ter sido compilado com suporte para o uso do TCPWRAPPERS; dessa forma, para confirmar, é recomendável usar o comando `ldd`. Veja o exemplo:

```
# ldd /usr/sbin/sshd
```

Primeiro bloqueia-se o serviço de ssh para todos no arquivo `/etc/hosts.deny`.

```
# vi /etc/hosts.deny
sshd: ALL
```

Em seguida, liberamos para quem desejamos.

```
# vi /etc/hosts.allow  
sshd: 10.0.0.1 10.0.0.3
```

Nesse exemplo, está sendo definido que somente esses dois IPs podem fazer conexões SSH no servidor; todos os outros serão bloqueados.

Para validar a configuração, podemos tentar conectar via SSH no servidor a partir de um desses IPs que estão liberados e de um que não está liberado.

Ferramentas de varreduras e levantamento de informação

Devemos levar em consideração no checklist de rede a avaliação de que tipo de informação é fornecida pelo serviços de rede, para que seja possível rever as configurações para removê-las. É fato que a obscuridade de informações de serviços de rede é uma boa prática, mas não é uma ação definitiva que resolva um problema de vulnerabilidade, por exemplo. O administrador tem de ter em mente que não fornecer informações sobre seu sistema via serviços de rede é uma boa prática, mas mantê-los atualizados e devidamente configurados é ainda mais importante.

Bons exemplos de ferramentas úteis para essa coleta de informações: footprint, fingerprint e enumeração, comumente utilizadas por invasores.

Fingerprint de serviço

Submundo da internet: divulgação de ferramentas que fazem fingerprint em serviços é comum.

- ▣ Por isso, é importante que um administrador:
 - ▣ Remova informações desnecessárias.
 - ▣ Saiba validar que tipo de informação está disponibilizando.

É muito comum no universo underground (submundo da internet) a divulgação de ferramentas que realizam fingerprint em serviços, pois a identificação da versão de um serviço é uma informação fundamental para um scriptkid aplicar o uso de um determinado exploit, ainda não sendo incomum ferramentas que fazem esse tipo de consulta em grande escala. Dessa forma, é importante que um administrador remova as informações desnecessárias, mas que também saiba validar que tipo de informação está disponibilizando. Ferramentas como httprint, amap e nmap podem ser muito úteis nesses momentos.

Amap

Scanner de fingerprint de serviço desenvolvido pelo THC (www.thc.org).

Utilização:

```
# amap -sT -d -b -o amap_report.txt 12.18.1.14
```

Onde:

- ▣ **-sT**: define o método TCP de varredura;
- ▣ **-d**: define saída em modo hexadecimal;
- ▣ **-b**: define saída em modo ASCII;
- ▣ **-o**: define que o resultado da varredura deverá ser gravado no arquivo informado após o parâmetro.

Httpprint

Ferramenta útil na identificação ou levantamento de informações do servidor HTTP, como versão, tipo e recurso disponível – SSL e suporte a uma linguagem específica, como PHP.

Exemplo de uso:

```
# httpprint -s signatures.txt -o httpprint_report.html -h 12.18.1.14
```

Onde:

- **-s:** informa o arquivo que contém as assinaturas de fingerprint;
- **-o:** define que o resultado da varredura deverá ser gravado no arquivo informado após o parâmetro;
- **-h:** informa o IP do alvo.

O Nikto.pl é um scanner de vulnerabilidades web que pode ser usado na enumeração e identificação de possíveis vulnerabilidades. É recomendável que antes de seu uso façamos a atualização da sua base de conhecimento de vulnerabilidade, baixando as atualizações dos plug-ins de ataque, utilizando a opção “-update”, conforme ilustrado.

Consulta dos plug-ins disponíveis:

```
# nikto.pl -list-plugins
```

Atualização dos plug-ins:

```
# ./nikto.pl -update
```

Para uma análise simples, basta usar a opção “-host”:

```
# ./nikto.pl -host 12.18.1.14
```

Podemos também gerar um relatório HTML com o resultado da análise:

```
# ./nikto.pl -host 12.18.1.14 -Format htm -output nikto_report.html
```

Onde:

- **-list-plugins:** lista os plug-ins ativos;
- **-update:** conecta no site do projeto para baixar novos plug-ins e atualizações;
- **-host:** define o alive;
- **-format:** define o formato de saída;
- **-output:** define o nome do arquivo de saída do relatório.

As ferramentas que se destacam na realização desse tipo de teste são THC Hydra e Medusa. Ambas possuem suporte a vários tipos de ataques de Força Bruta e a diferentes serviços de redes.

Exemplo do uso de Hydra para um ataque de Força Bruta:

```
# hydra -l root -P PASSFILE.txt -o LOG.txt 12.18.1.14 ssh
```

Onde:

- **-l root:** login alvo;
- **-P PASSFILE.txt:** Lista/dicionário de senhas;
- **-o LOG.txt:** Log criado com informações da execução do ataque;



Saiba mais

Um teste interessante é estressar os serviços de rede que possuem autenticação, tanto para avaliar a sua performance como também para ter uma visão geral sobre o comportamento do serviço sobre um ataque de Força Bruta. Isso possibilita ao administrador avaliar melhor o serviço e as informações de registro de eventos (logs) geradas.



- **12.18.1.14:** IP do alvo;
- **ssh:** nome do módulo do respectivo serviço que será atacado.

Mais um exemplo do uso de Força Bruta, agora utilizando o medusa para o ataque :

```
medusa -u root -P PASSFILE.txt -h 12.18.1.14 -M ssh
```

Onde:

- **-u root:** login alvo;
- **-P PASSFILE.TXT:** lista/dicionário de senhas;
- **-o LOG.txt:** log criado com informações da execução do ataque;
- **-h 12.18.1.14:** IP do alvo;
- **-M ssh:** nome do módulo do respectivo serviço que será atacado.

As ferramentas Hydra e Medusa são projetos distintos mas ambas tem a mesma finalidade e muitas funcionalidades em comum.

Tabela comparativa entre Medusa e Hydra:

Área	Feature	Medusa 2.1	Hydra 7.1
*	Licença	GPL-2	GPL-3
Core	Parallel Method	pthread	fork()
Generic Wrapper Module		sim	
AFP		sim	sim
CVS		sim	sim
FTP	FTP	sim	sim
	Explícito FTPS – módulo AUTH TLS Mode conforme definido na RFC 4217)	sim	sim
	I FTPS (FTP sobre SSL (990/tcp))	sim	sim
HTTP	Basic Auth	sim	sim
	NTLM Auth (Windows Integrated)	sim	sim
	Digest Authentication	MD5, MD5-sess	MD5
HTTP Proxy			sim
ICQ			sim



Área	Feature	Medusa 2.1	Hydra 7.1
IMAP	Method LOGIN Support	sim	sim
	Supporte ao Método – AUTH-PLAIN Support	sim	sim
	Supporte ao Método – AUTH-NTLM	sim	sim
	Supporte a SSL	IMAPS, STARTTLS	IMAPS, STARTTLS
LDAP			sim
Microsoft SQL	Port Auto-Detection	sim	
	MS-SQL	sim	sim
MySQL	Pre-4.1 Authentication	sim	sim
	Pre-4.1 Hash Passing	sim	
	4.1+ Authentication	sim	sim
NCP (NetWare)		sim (ncpfs)	sim (ncpfs)
NNTP		Sim (Original AUTHINFO)	sim (Original AUTHINFO)
Oracle	Database	sim (via Wrapper script)	
	Listener		
	SID		sim
PcAnywhere	Suporte a Encryption Level	None	None
	Suporte a Authentication Mode(s)	Native PCA, ADS, NT, Windows	Native PCA
PCNFS			sim
POP3	Suporte ao método AUTH-USER	sim	sim
	Suporte ao método AUTH-LOGIN	sim	sim
	Suporte ao método AUTH-PLAIN	sim	sim
	Suporte ao método AUTH-NTLM	sim	sim
	Suporte ao SSL	POP3S, STARTTLS	POP3S
PostgreSQL		sim	sim
RDP (Terminal Server)		sim (via Wrapper Script)	sim
REXEC		sim	sim



Área	Feature	Medusa 2.1	Hydra 7.1
RLOGIN	Suporte rhost	sim	
	Suporte Password	sim	sim
RSH		sim	sim
SAPR3			sim
SIP			sim
SMB (Microsoft Windows/Samba)	Authentication Modes	clear-text, LMv1, NTLMv1, LMv2, NTLMv2	clear-text, LMv1, NTLMv1, LMv2, NTLMv2
	Hash Passing	sim	sim
SMTP	Suporte ao método AUTH-LGIN	sim	sim
	Suporte ao método AUTH-PLAIN	sim	sim
	Suporte ao método AUTH-NTLM	sim	sim
	Suporte a SSL	STARTTLS	STARTTLS
	Suporte a VRFY	sim	sim
SNMP		sim	sim
SOCKS5			sim
SSHv2		sim (baseado na libssh2)	sim (baseado na libssh)
SVN		sim	sim
TeamSpeak			sim
Telnet	Generic Telnet	sim	sim
	Cisco (AAA/non-AAA)	sim	sim
	Cisco enable password		sim
	AS/400 (TN5250) Support	sim	
VNC	Suporte ao método Password-less/Password-only	sim	sim
	Suporte ao método Anti-Brute Force Slowdown	sim	
	Suporte ao método Username/Password	sim	



Área	Feature	Medusa 2.1	Hydra 7.1
VmWare Authentication Daemon	Suporte ao método Non-SSL Authentication	sim	sim
	SSL Authentication	sim	
Web Form Module		sim	sim

Exercício de fixação 1

Levantamento de informação e Força Bruta

- Realize um ataque de Força Bruta com o HYDRA, tendo como alvo o servidor SSH;
- Faça um ataque de Força Bruta com o MEDUSA, tendo como alvo o servidor SSH.

Até esse ponto os procedimentos tratados tinham como meio parametrizações de recursos do Sistema Operacional. No entanto, um Hardening não deve se limitar a isso. O Hardening deve, sim, iniciar na camada do Sistema Operacional, mas deve se entender aos serviços que serão providos pelo respectivo servidor. A partir de agora e nos próximos capítulos, serão tratados procedimentos de Hardening de serviços comuns em servidores Unix .

Hardening do serviço SSH

Hardening de serviço:

- Deve ter foco na possibilidade de eliminar recursos disponíveis por padrão, que podem ser um vetor para ameaças, normalmente por causa da falta de controles.

Com relação ao serviço SSH, existem várias parametrizações que podem somar na segurança do serviço.

Uma possibilidade de controle é o uso do recurso disponível nas bibliotecas PAM. Tratando o SSH diretamente, podemos fazer alguns ajustes no seu arquivo de configuração para deixá-lo mais restritivo, combinando mais uma vez com controles via PAM.

Podemos usar novamente o PAM para restringir o acesso ao nosso servidor via SSH em determinados horários. Primeiro devemos habilitar o módulo no arquivo de configuração do programa SSH.

```
# vi /etc/pam.d/ssh
account    required      pam_time.so
```

Logo em seguida, editar o arquivo */etc/security/time.conf* para fazer a restrição de tempo:

```
# vi /etc/security/time.conf
ssh;*:!toor;A10730-1900
```

Nesse exemplo, foi permitido que o acesso via SSH no nosso servidor seja feito todos os dias, mas somente das 7h30 às 19h, exceto para o usuário toor, que poderá acessar em qualquer horário.

Outras ações interessantes e necessárias no serviço, para torná-lo mais seguro:

- Alterar a porta padrão 22 para outra porta alta;
- Configurar o serviço para ouvir somente no IP definido;



Boas práticas recomendadas na ABNT NBR ISO/IEC 27002:2005: é válido destacar que ferramentas como NMAP, NIKTO, HTTPRINT, HTTPSQUASH e AMAP são ferramentas poderosas, que podem ser úteis em um processo de auditoria, mas não devem ser instaladas no servidor, mas sim exclusivamente na estação administrativa para garantir sua integridade e mitigar a possibilidade de mal uso, fazendo conformidade com as diretrizes do item 15.3 da norma.



- ▣ Proibir o login como super-usuário;
- ▣ Utilizar apenas a versão 2 do protocolo SSH;
- ▣ Aplicar restrições de login relacionadas ao tempo de inatividade;
- ▣ Liberar acesso apenas para usuários específicos;
- ▣ Não permitir senhas em branco;
- ▣ Desabilitar o uso de senha;
- ▣ Refinando o uso do SFTP;
- ▣ Ter controles contra ataques de Força Bruta.

Modificação da porta padrão

A porta padrão do SSH é alvo constante de ataque de Força Bruta, sendo significativa a estatística de ataques acumulados anualmente e apresentados no relatório do Cert.br. Diante disso, uma ação simples mas eficaz é alterar a porta padrão 22 para outra à escolha do administrador. Para isso, devemos editar o arquivo de configuração do servidor SSH (*/etc/ssh/sshd_config*).

```
# vi /etc/ssh/sshd_config
```

```
Port 63322
```

Configurar o serviço para ouvir somente nos IPs definidos

Saiba mais

É recomendável que a porta de acesso do SSH seja mudada, definindo uma porta acima de 1024, definida pelo sysadmin, o que traria dificuldade inicial para ações de varreduras de portas. Quando é modificada a porta padrão do SSH para qualquer outra e alguém tentar se conectar em nosso servidor via SSH, ele precisará especificar a porta em que o SSH está trabalhando.

Nesse cenário, teoricamente somente os responsáveis pelo servidor saberão qual a porta definida para conexão ao serviço SSH.

Caso o servidor tenha mais de uma interface de rede, pode-se limitar o servidor para que responda somente na interface que se deseja. `ListenAddress 10.0.0.1`

Proibir o login como super-usuário

Essa é uma das opções primordiais que devem ser mudadas. O SSH, por padrão, permite que o root possa fazer conexões SSH de primeira. Caso não seja desativada essa opção definindo o no, de nada adiantariam as configurações que foram feitas no PAM até agora, salvo se a configuração for feita via PAM no arquivo de configuração */etc/pam.d/ssh*. Uma alternativa é não permitir que o root tenha acesso direto através do SSH.

```
PermitRootLogin no
```

Forçar o uso da versão 2 do protocolo

A versão 1 do protocolo SSH possui falhas seriíssimas de segurança, então deve-se configurar o servidor SSH com suporte exclusivo para a versão 2:

```
Protocol 2
```

Por fim, no arquivo de configuração do ssh, devemos habilitar o uso de um banner, mensagem que aparecerá quando alguém tentar conectar em nosso servidor.

Essa configuração pode ser feita ou via a modificação do */etc/issue.net* com a mensagem que deseja ou usando o opção “banner”, do arquivo de configuração do servidor SSH.

Coloque sempre mensagens de aviso, com uma mensagem clara e restritiva de que tudo o que for feito será registrado. Mas atenção: caso seja habilitado o banner no SSH, não se esqueça de mudar o *issue.net*, pois normalmente os sistemas Linux trazem a distribuição e a versão nesse arquivo. Não seria interessante um possível atacante conseguir saber qual é o Sistema Operacional e/ou a distribuição que está sendo usada através de uma simples tentativa de conexão SSH.



```
Banner /etc/issue.net
```

Aplicar restrições de login relacionadas ao tempo de inatividade.

Liberar acesso apenas para usuários específicos

Quando o serviço SSH destinando as ações administrativas e/ou o acesso é restrito a usuários administrativos, é uma ação recomendável restringir o acesso exclusivo aos usuários administrativos que realmente devem ter acesso ao servidor. Isso pode ser parametrizado via conta de usuário ou via grupo, de duas formas, ou definindo quais usuários ou grupos terão acesso ou quais usuários e grupos não terão acesso.

- **AllowUsers:** define quais usuários terão acesso;
- **DenyUsers:** define quais usuários não terão acesso;
- **AllowGroups:** define quais grupos terão acesso;
- **DenyGroups:** define quais grupos não terão acesso.

Para realizar a parametrização, basta colocar os nomes de usuários ou grupos separados por espaço. A parametrização de usuário fica assim:

```
AllowUsers joao syadmin rpnadmin sysop
```

Exemplo de parametrização de grupo Allowgroups administrators:

```
Allowgroups administrators
```

É recomendável parametrizar usando o conceito de definir que tem permissão, ou seja, é preciso criar uma whitelist, pois é mais restritivo e fácil de administrar.

Sendo possível, é recomendável usar somente chaves para autenticação.

Não permitir senhas em branco

Impedir que senhas em branco sejam aceitas:

```
PermitEmptyPasswords no
```

Desabilitar o uso de senha

O esquema de autenticação baseada em senha é o padrão de um servidor SSH; no entanto, é interessante parametrizar para que a autenticação fique vincula a conceitos de chaves simétricas (chave pública e privada). Essa parametrização pode ser definida pela opção PasswordAuthentication, conforme exemplificado:

```
PasswordAuthentication no
```

Para criação das chaves usa-se o comando *ssh-keygen*, que as cria por padrão no diretório .ssh, dentro do home do usuário:

```
ssh-keygen -t rsa -b 4096
```

Após a criação, é necessário enviar a chave pública para a usuário na máquina correspondente:

```
ssh-copy-id -i .ssh/id_rsa.pub usuario@10.0.0.1
```

O próximo login via ssh deve ocorrer sem senha.



Não permitir senhas em branco

Impedir que senhas em branco sejam aceitas:

```
PermitEmptyPasswords no
```

Desabilitar o uso de senha

O esquema de autenticação baseada em senha é o padrão de um servidor SSH. Todavia, é interessante parametrizar para que a autenticação fique vincula a conceitos de chaves simétricas (chave pública e privada). Essa parametrização pode ser definida pela opção "PasswordAuthentication", conforme exemplificado:

```
PasswordAuthentication no
```

Para a criação das chaves, usa-se o comando `ssh-keygen`, que as cria por padrão no diretório `.ssh`, dentro do `/home` do usuário utilizado:

```
ssh-keygen -t rsa -b 4096
```

Durante a geração da chaves, é solicitada a criação de uma senha. Caso seja escondida uma senha, ela será solicitada toda vez que a chave for utilizada. Caso queira conexão direta com chaves, mas sem senha, basta, durante a solicitação de criação de senha, pressionar a tecla `enter`. Esse segundo cenário é interessante quando se deseja execução de scripts remotos.

Após a criação, é necessário enviar a chave pública para o usuário na máquina correspondente:

```
ssh-copy-id -i .ssh/id_rsa.pub usuario@10.0.0.1
```

Refinando o uso do SFTP

É recomendável ativar logs de operações do SFTP, caso esse recurso esteja ativo, conforme o exemplo:

```
Subsystem sftp    /usr/libexec/openssh/sftp-server -l INFO -f AUTH
```

Em situações específicas, podemos desejar desabilitar o serviço. Para isso, basta comentar a linha. Outra possibilidade é parametrizar a conta de um específico usuário para que seja definido o uso exclusivo do SFTP. Para isso, devemos definir o binário do serviço do SFTP como a shell do usuário:

```
# echo "usermod -s /usr/libexec/openssh/sftp-server" >> /etc/shells
# usermod -s /usr/libexec/openssh/sftp-server usersftp
# grep ^# sftp-server /etc/passwd
```

Depois de todas as configurações feitas, é necessário reiniciar o ssh para que as modificações entrem em vigor.

```
# /etc/init.d/ssh restart
```



Mitigação de ataque de Força Bruta

Já foi mencionado neste capítulo o grande número de ataques de Força Bruta tendo como alvo a porta 22. Inicialmente, até foi sugerida a troca da porta 22 por outra porta alta à escolha do servidor. Todavia, independente ou não da troca da porta padrão, é recomendável a instalação de algum controle para conter, registrar e mitigar ataques de Força Bruta tendo como alvo o servidor SSH ou mesmo outros serviços.

Instalação do Fail2ban

```
# sudo apt-get install -y fail2ban
```

A configuração fica centralizada nos arquivos:

- */etc/fail2ban/jail.conf*
- */etc/fail2ban/fail2ban.conf*

No arquivo *fail2ban.conf* são parametrizadas informações inerentes aos registros de eventos:

```
# cd /etc/fail2ban/  
# grep -v ^# fail2ban.conf | grep .  
  
[Definition]  
  
loglevel = 3  
  
logtarget = /var/log/fail2ban.log  
  
socket = /var/run/fail2ban/fail2ban.sock
```

A configuração que define as diretrizes de funcionamento ficam concentradas no arquivo *jail.conf*. Como boa prática, é recomendável fazer uma cópia do arquivo original antes de qualquer alteração.

```
# cd /etc/fail2ban/  
# cp jail.conf jai.conf_ORIGINAL -v
```

O arquivo *jail.conf* é definido em sessões, onde a primeira sessão denominada “default” permite a parametrização do funcionamento do fail2ban através de algumas opções:

Ignoreip: onde são definidos os IPs que não vão ser bloqueados pelo programa, ou seja, uma whitelist;

- **bantime:** onde é definido o tempo em segundos em que o IP ficará banido ou bloqueado;
- **maxretry:** define o número máximo em que o IP pode tentar efetivar um processo de login no servidor SSH até ser bloqueado;
- **Logpath:** define o arquivo de log onde serão registradas as tentativas de login que falharam;
- **destemail:** define o e-mail para o qual deverão ser encaminhadas as notificações de bloqueio;
- **banaction:** define qual ação será tomada.



A segunda parte da configuração do jail.conf onde são parametrizadas informações inerentes aos serviços que podem ser protegidos pelo fail2ban, tanto para o propósito de mitigação contra ataques de Força Bruta, como também ataques de Negação de Serviço baseadas em “flood” de conexões, entre os serviços que já vêm com predefinições:

- Servidor web apache;
- Servidor FTP vsftpd;
- Servidor FTP proftpd;
- Servidor FTP pure-ftpd;
- Servidor FTP wuftpd;
- Servidor de SMTP postfix;
- Servidor de correio couriersmtp e courierauth;
- Serviço de autenticação sasl;
- Servidor de POP3/IMAP dovecot;
- Servidor de DNS (ataques named-refused-tcp).

Exemplo de uma configuração para o servidor SSH:

```
[ssh]  
enabled = true  
port = ssh  
filter = sshd  
logpath = /var/log/auth_fail2ban.log  
maxretry = 6
```

Nessa configuração, é informado que o controle está ativo pelo parâmetro “enabled=true”. Dessa forma, a cada seis tentativas, conforme parametrizado por “maxretry=6”, o IP origem da tentativa de conexão será bloqueado.

O ideal é mudar para “enabled=true” e parametrizar todos os controles de serviços que desejamos que sejam protegidos pelo Fail2ban:







Roteiro de Atividades 4

Atividade 4.1 – # service fail2ban restart

Sistemas de detecção de intrusões (IDS) em redes WLAN

Na prática de Hardening de serviço de rede, o foco é eliminar serviços desnecessários que por padrão estejam ativos e reavaliar as configurações do servidor SSH.

1. Revise todos os serviços ativos TCP e de defina quais devem realmente estar ativos;
2. Revise todos os serviços ativos UDP e de defina quais devem realmente estar ativos;
3. Avalie as conexões baseadas em Rawsocket;
4. Realize um Hardening no SSH.
 - ▣ Mude a porta padrão;
 - ▣ Defina que somente o grupo sysadmin pode realizar login;
 - ▣ Desabilita o login direto via usuário root;
 - ▣ Defina o serviço SSH para ouvir somente no IP definido e no localhost;
 - ▣ Ative registro de eventos do SFTP.
5. Crie controles com FAIL2BAN contra ataques de Força Bruta e Negação de Serviço.
6. Avalie os controles do Fail2ban com o HYDRA.
7. Avalie os controles do Fail2ban com o MEDUSA.
8. Habilite autenticação exclusiva via chaves simétricas.



Guarde as informações levantadas, pois no capítulo 5 será proposto Hardening do Apache e, após realizados os procedimentos, deveremos repetir esses exercícios para comparar e avaliar as melhorias obtidas.

9. Realize uma avaliação de um servidor usando as ferramentas NMAP, AMAP, HTTPPRINT, HTTPSQUASH e NIKTO, no servidor Apache.

9.1. Leitura de banner:

```
# echo "GET http 1.0" | nc <IP ALVO> 80
```

9.2. Leitura de banner via nmap:

```
# nmap -sV -n -P0 -p 80,443 <IP ALVO>
```

9.3. Coleta de informações via fingerprint no serviço http:

```
# cd /pentest/enumeration/www/httpprint/Linux/  
# ./httpprint -h <IP ALVO> -s signatures.txt
```



9.4. Coleta de informações de serviço http:

```
# cd /pentest/enumeration/complemento/httsquash  
#./httsquash -r <IP ALVO>
```

9.5. Coleta de informações via amap:

```
# amap -bq <IP ALVO>80  
# amap -bq <IP ALVO>
```

9.6. Coletando informações via “fingerprint de serviço” com http:

```
# nmap --script http-enum <IP ALVO>
```

9.7. Coletando informações com o nmap usando o script http-enum:

```
# nmap --script http-enum -p <IP ALVO>
```

9.8. Coletando informações com o nmap usando o script http-headers:

```
# nmap - -script http-headers <IP ALVO>
```

9.9. Coletando informações com o nmap usando o script http-methods:

```
# nmap --script http-methods <IP ALVO>
```

9.10. Coletando informações com o nmap usando o script http-php-version:

```
# nmap --script http-php-version <IP ALVO>
```

9.11. Coletando informações com o nmap usando todos os scripts:

```
# nmap --script http-enum,http-headers,http-methods,http-php-  
version <IP ALVO>
```



5

Serviços de Redes – Parte 2

objetivos

Aprender hardening em servidores Apache, PHP e servidores BIND; Fazer configuração do PS-WATCHER; Identificar serviços de rede ativos; Reforçar a segurança do serviço SSH; Implementar mecanismo contra ataque de Força Bruta.

conceitos

Serviço Web, baseado no Apache com PHP; Serviço DNS, baseado no BIND.

Hardening

Primeiro passo:

- Verificar se o serviço está desatualizado.
 - Se estiver desatualizado: vai permitir que ataques de Negação de Serviço (DOS) e/ou de execução remota de comandos possam estar presentes vinculados a potenciais vulnerabilidade publicadas.

Diante disso, recomenda-se verificar informações de vulnerabilidade pontuais com o comando *debsecan*:

```
debsecan | grep apache
```

Caso exista, devemos aplicar a atualização.

É recomendável rever as regras de diretório, definida como “ServerRoot”. É recomendável identificar o diretório padrão das configurações do serviço do Apache, definido na diretriz “ServerRoot” dentro do arquivo *apache2.conf*, pois os ajustes de configuração são feitos nesses arquivos.

Uma maneira simples de identificar como estão distribuídos os arquivos de configuração de um determinado pacote: basta consultar informações do pacote. Em distribuições like Debian, para identificar a estrutura de arquivos de configuração e binários, basta usar o comando *dpkg* no pacote instalado da seguinte forma:

```
# dpkg -L apache2-common
```

Estrutura da organização dos arquivos de configuração do pacote do Apache 2 na distribuição Ubuntu:

```
/etc/apache2: Diretório principal de configuração do apache
```



```
/etc/apache2/conf.d: Diretório de arquivos de configurações adicionais  
/etc/apache2/mods-available : configurações de módulos disponíveis  
/etc/apache2/mods-enabled: configurações de módulos ativos  
/etc/apache2/sites-available : configurações de sites disponíveis  
/etc/apache2/sites-enabled: configurações de sites ativos
```

Normalmente, na maioria das distribuições Linux, as configurações pré-definidas na criação dos pacotes atendem bem. No entanto, segue uma exemplificação da configuração ideal, onde o usuário root deve ser o dono dos arquivos e o permissionamento de grupo deve estar vinculado ao grupo root.

```
cd /etc/apache2  
chown root.root /etc/apache2  
chmod -R 750 /etc/apache2
```

Antes de qualquer modificação, em qualquer arquivo de configuração do Apache, é recomendável manter uma cópia do arquivo original, conforme o exemplo:

```
# cp apache2.conf apache2.conf_ORIGINAL  
# grep -v ^# apache2.conf_ORIGINAL | grep . > apache2.conf
```

É recomendável verificar se o usuário de sistema responsável pelo deamon do Apache (`httpd`) tem uma shell inválida. Adiante, é necessário identificar o usuário na distribuição em uso vinculado ao Apache. No Ubuntu, é o usuário `www-data`.

Para certificar o usuário utilizado pelo Apache, devemos identificar as definições para opções `User` e `Group` no arquivo de configuração `apache.conf`.

```
# These need to be set in /etc/apache2/envvars  
User ${APACHE_RUN_USER}  
Group ${APACHE_RUN_GROUP}
```

Os valores das variáveis são estabelecidos no arquivo `/etc/apache2/envvars:export`

```
APACHE_RUN_USER=www-data  
export APACHE_RUN_GROUP=www-data
```

O serviço do Apache deve ser iniciado pelo usuário root, ou seja, o processo principal. Mas com a definição de “User”, ele muda para o usuário definido pela diretiva. Dessa forma, os processos que atendem as conexões estarão vinculados a um usuário não privilegiado.

Um vez identificado o usuário, devemos verificar qual shell está definida:

```
# grep ^www-data /etc/passwd  
www-data:x:33:33:www-data:/var/www:/bin/sh  
# grep www-data /etc/group  
www-data:x:33:
```



Saiba mais

Em algumas distribuições, as configurações são concentradas em poucos arquivos, mas no Ubuntu a configuração é dividida em vários arquivos. Dessa forma, embora todas as orientações deste capítulo sejam aplicáveis ao Apache versão, o administrador deve primeiro entender como estão organizadas as configurações do Apache em sua distribuição.

Caso seja necessário alterar a shell para uma shell não válida:

```
# ls -l /bin/false  
-rwxr-xr-x 1 root root 22896 Apr 1 2012 /bin/false  
  
# dpkg -S /bin/false  
coreutils: /bin/false  
  
chsh -s /bin/false www-data  
  
# grep www-data /etc/passwd  
www-data:x:33:33:www-data:/var/www:/bin/false
```

Verificar se o serviço está sendo inicializado por um usuário não privilegiado, pois como já sugerido no processo de Hardening de outros serviços, é uma boa prática em sistema Like Unix executar um serviço sempre que possível com um usuário de sistema, ou seja, colocar como meta no processo de Hardening «Menor privilégio, Menor recurso» em qualquer decisão.

Nas distribuições Linux, o nome do usuário responsável pelo deamon do Apache pode ser diferente. Isso não é um problema, desde que não seja um usuário não privilegiado, ou seja, o importante é não ser vinculado à conta root.

A estrutura e Logs do Apache é muito bem elabora, haja vista que é base para processamento de avaliações estatísticas de acesso ao sites. Ele é gerida pelo próprio deamon. No entanto, em algum caso, se for necessário, podemos enviar também logs para o deamon do syslog. Para isso, devemos inserir nas configurações do Apache:

```
#ErrorLog file-path|syslog[:facility]  
ErrorLog syslog:user
```

Ter uma visão das conexões vinculadas ao Apache e também os domínios virtuais que estão sendo hospedados pelo servidor, auxiliar na avaliação de como se encontra o funcionamento do servidor, tendo informações sobre o recurso utilizado:

Habilitação da publicação das informações sobre processos

Para o funcionamento desse recurso, é necessário ter o módulo “status” ativo, e a conexão deve ser restrita aos IPs administrativos. Devemos também inserir a respectivas configurações no apache.conf:<Location /server-status>

```
SetHandler server-status  
Order deny,allow  
Deny from all  
Allow from 127.0.0.1
```



```
Allow from 192.168.56.0/24  
</Location>  
Ativação do modulo "status":# a2enmod status
```

Quando mod_status é carregado para o servidor, a sua capacidade está disponível em todos os arquivos de configuração, incluindo por Diretório arquivos (por exemplo, .htaccess). O módulo mod_status, configurado com acesso “não restrito”, pode fornecer a um potencial invasor informações que podem ser usadas para refinar um ataque ou mesmo elaboração de exploit direcionados. Diante disso, a correta configuração da diretiva “Allow from” torna-se imprescindível na configuração desse recurso.

Para habilitação da publicação das informações gerais do funcionamento do Apache e demais diretrizes de configuração, com o objetivo de ajudar a avaliar como o Apache está funcionando, quais módulos estão ativos, pois permitirá ao sysadmin ter informações suficientes para definir o que poderá ser feito Hardening do apache, quais são as diretivas que deverão ser modificadas ou desabilitadas, quais módulos deverão ser habilitados e quais deverão ser desabilitados. Para o funcionamento desse recurso, é necessário ter o módulo «info» ativo e, da mesma forma que no recurso «server-status», a conexão deve ser restrita aos IPs administrativos. Assim, para configuração do recurso, é necessário inserir a respectivas configurações no apache.conf:<Location /server-info>

```
SetHandler server-info  
Order deny,allow  
Deny from all  
Allow from 127.0.0.1  
Allow from 192.168.56.0/24  
</Location>  
Ativação do módulo "info":# a2enmod info
```

Após inserir as configurações, é necessário reiniciar o Apache e fazer um checklist para verificar se o serviço foi ativado:

Reinicialização do Apache:

```
# service apache2 restart
```

Verificação de informações dos processos ativos na porta 80, que pode ser realizada com os comandos: lsof, netstat e fuser. Veja os exemplos:

```
# fuser -v 80/tcp  
USER          PID ACCESS COMMAND  
80/tcp:  
          root      3921 F.... apache2  
          www-data   3924 F.... apache2  
          www-data   3931 F.... apache2  
# lsof -n -i :80  
COMMAND  PID   USER FD   TYPE DEVICE SIZE/OFF NODE NAME
```

```

apache2 3921 root      3u   IPv4  16609      0t0  TCP *:http (LISTEN)
apache2 3924 www-data  3u   IPv4  16609      0t0  TCP *:http (LISTEN)
apache2 3931 www-data  3u   IPv4  16609      0t0  TCP *:http (LISTEN)

# netstat -nltp | grep :80
tcp  0      0 0.0.0.0:80    0.0.0.0:*      LISTEN      3921/apache2

```

Saiba mais

A cada modificação nos arquivos de configuração do apache2, devemos reinicializar o serviço, e é recomendável realizar o checklist com os comandos *lsof*, *netstat* e *fuser*, para validar se está ativo. Em caso também de existirem sites publicados via conexões HTTP3, o checklist deve ser aplicado à porta 443.

Remoção de módulos não utilizados

Considerando o conceito de «Menor recurso», com as informações disponibilizadas pelo «server-info», o administrador deve avaliar quais são os módulos do Apache que devem realmente estar ativos, para a proposta do servidor em questão. Assim, é recomendável remover alguns módulos, caso eles realmente não sejam necessários para aquele contexto:

Módulo de suporte a Webdav

Uma vez que tenhamos a certeza de que não se demanda o uso do recurso WebDAV, recomendamos desativar módulos WebDAV. É uma ação necessária para melhorar a segurança do servidor web, reduzindo a quantidade de caminhos de código potencialmente vulneráveis expostos à rede e diminuindo o potencial para acesso não autorizado a arquivos via controles padrão ou mesmo mal configurados de acesso WebDAV.

Para desabilitar:

```
# a2dismod dav dav_fs dav_lockMódulo autoindex
```

O módulo autoindex gera automaticamente um “index.html”, que lista o conteúdo de pastas no servidor quando não se tem uma página “index” publicada.

Considerando o conceito de “segurança por obscuridade”, onde se pensa em fornecer sempre o mínimo de informação, é recomendado não ser habilitado o módulo autoindex, pois pode revelar informações úteis para um atacante, como convenções de nomenclatura e caminhos de diretórios, ou mesmo arquivos que accidentalmente não deveriam estar publicados em uma respectiva pasta.

```
# a2dismod autoindex
```

Módulos de proxy

Em projetos de segurança para infraestrutura web, o uso de servidores proxy pode funcionar como um controle de segurança interessante. Todavia, quando devidamente configurado e com conceito de perímetros de rede bem definidos. Mas se o servidor web em questão não será um proxy web para outro servidor, é recomendável identificar se existem módulos de proxy desativados.

```
# a2dismod proxy proxy_ajp proxy_balancer proxy_connect proxy_ftp
proxy_http proxy_scgi
```



Módulo Userdir

Em configurações de servidor web onde não haverá necessidade de publicação de home de usuário, é recomendável desabilitar o módulo Userdir.

```
# a2dismod userdir
```

Suporte a SSIs

É um recurso antigo e na maioria das vezes desnecessário. Dessa forma, caso não esteja sendo utilizado por nenhuma aplicação, é recomendável desabilitá-lo. Recursos que não são utilizados oficialmente e estão ativos podem ser uma oportunidade para que ataques os usem de forma maliciosa, consumindo recursos do servidor:

```
vi /etc/apache2/mods-enabled/mime.conf

#AddType text/html .shtml

#AddOutputFilter INCLUDES .shtml
```

Configuração do módulo Expires

Essa configuração trata mais da questão de performance, pois possibilita definir um tempo de vida para objetos estáticos dos sites publicados via Apache. Habilitar esse módulo ajuda a melhorar qualidade de acesso dos clientes.<IfModule mod_expires.c>

```
ExpiresActive On
ExpiresDefault "access plus 1 years"
ExpiresByType image/gif "access plus 1 years"
ExpiresByType image/jpeg "access plus 1 years"
ExpiresByType image/png "access plus 1 years"
ExpiresByType image/ico "access plus 1 years"
ExpiresByType text/css "access plus 1 years"
ExpiresByType text/txt "access plus 1 years"
ExpiresByType text/javascript "access plus 1 years"
ExpiresByType application/x-unknown-content-type "access plus 1 years"
ExpiresByType application/x-javascript "access plus 1 years"
</IfModule>
```

Ocultando a versão do servidor

Por padrão, o servidor retorna resposta HTTP contendo muitas informações, como a versão do Apache, php e, dependendo da configuração, em alguns casos até mesmo a versão do Sistema Operacional. Isso é ruim, pois não desejamos ter um serviço que forneça informações detalhadas para um atacante: quanto menos informações sobre a implementação do serviço fornecido, melhor. Diante disso, outra ação baseada no conceito de «segurança por obscuridade» é configurar o Apache para ocultar informações sobre o sistema. Essas informações surgem quando uma tela de erro ocorre. Para ocultar as informações



do sistema, é necessário editar o arquivo `/etc/apache2/conf.d/security` e modificar a diretriz "Server Tokens" para "Prod":

Os seguintes valores são possíveis ServerTokens:

- **ServerTokens Prod:** informa somente que o servidor é Apache. Exemplo: "Server: Apache";
- **ServerTokens Major:** informa que o servidor é Apache e a versão. Exemplo: "Server: Apache/2";
- **ServerTokens Minor:** informa que o servidor é Apache e a versão completa. Exemplo: "Server: Apache/2.2";
- **ServerTokens Min displays:** informa que o servidor é Apache, além da versão completa e o release. Exemplo: "Server: Apache/2.2.17";
- **ServerTokens OS displays:** informa que o servidor é Apache, a versão completa, release e Sistema Operacional. Exemplo: "Server: Apache/2.2.17 (Ubuntu)";
- **ServerTokens Full displays:** informa que o servidor é Apache, a versão completa e o release, versão do PHP e informações adicionais do Sistema Operacional. Exemplo: "Server: Apache/2.2.17 (Ubuntu PHP/5.3.5" (If you don't specify any ServerTokens value, this is the default).

Module Security (Modsecurity)

O módulo de segurança, o mod_security, é um módulo do Apache que será instalado para bloquear o monitoramento de requisições e respostas HTTP tanto quanto a negação de pacotes suspeitos.

Instalação do Modsecurity:

```
# apt-get install -y modsecurity-crs libapache2-modsecurity  
Definindo as configurações básicas:# cd /etc/modsecurity  
# grep -v ^# modsecurity.conf-recommended |grep . > modsecurity.conf
```

Ativação do Modsecurity:

```
# a2enmod mod-security  
# service restart apache2
```

Acompanha informações do registro de eventos do Modsecurity em:

```
# tail -f /var/log/apache2/modsec_audit.log  
--fla41278-H--  
Apache-Error: [file "core.c"] [line 3558] [level 3] Invalid URI in  
request GET /help../../../../../../../../../../../../etc/shadow  
HTTP/1.1  
Stopwatch: 1362354524153078 676 (- - -)  
Stopwatch2: 1362354524153078 676; combined=39, p1=0, p2=0, p3=7,  
p4=6, p5=20, sr=0, sw=6, l=0, gc=0  
Response-Body-Transformed: Dechunked  
Producer: ModSecurity for Apache/2.6.3 (http://www.modsecurity.org/).
```

Server: ApacheControle sobre arquivos htaccess

Devemos verificar se está ativo o controle, impossibilitando acesso .htaccess:

```
AccessFileName .htaccess

<Files ~ "\.ht">

    Order allow,deny

    Deny from all

    Satisfy all

</Files>
```

Desabilitar página de erro

A disponibilização de informações sobre o servidor em páginas de erro é habilitada por padrão. É recomendável remover informações do servidor de páginas de erro. Para isso, basta alterar o arquivo `/etc/apache2/conf.d/security` modificando o valor da diretriz "Server-Signature" para "Off":

```
vi /etc/apache2/conf.d/security

ServerSignature Off
```

Incluir restrições para acesso a .old .swp e .bak

É possível indicar cópias de scripts feitas por administradores com nomes de cgis + .swp, .bak e .old, tendo acesso ao código dos scripts.

Remover todos os arquivos do tipo .old, .swp e .bak que possam ser lidos por clientes web e também incluir restrições, como:

```
<Files ~ "\.(bak|old|\$)" >

    Order allow,deny

    Deny from all

</Files>
```

```
<Location /server-status>

    SetHandler server-status

    AuthType Basic

    AuthName "Server Status"

    AuthUserFile /etc/apache2/htpasswd

    Require valid-user

    Order deny,allow

    Deny from all
```



```

        Allow from 127.0.0.1

        Allow from 192.168.56.0/24

    </Location>

<Location /server-info>

    SetHandler server-info

    AuthType Basic

    AuthName "Server Status"

    AuthUserFile /etc/apache2/htpasswd

    Require valid-user

    Order deny,allow

    Deny from all

    Allow from 127.0.0.1

    Allow from 192.168.56.0/24

</Location>

```

O Hardening da linguagem PHP apresentado neste capítulo tem como foco um modelo web. Dessa forma consiste, em muitos momentos, em desabilitar recursos poderosos da linguagem PHP que, para um contexto web, não são necessários. Todavia, não é incomum programadores que não levam isso em consideração. Nesse contexto, devemos solicitar que o código seja revisto, pois não se deve aprovar um código que não tenha boas práticas de programação segura ou mesmo use recursos que não são ideais para uma publicação web.

! Em suma, todas as sugestões deste capítulo devem ser aplicadas sempre que possível, mas não devemos deixar de aplicar uma configuração em detrimento da segurança, para que uma determinada aplicação que foi mal elaborada seja colocada em produção.

É preciso que os programadores busquem informações sobre desenvolvimento usando boas práticas de segurança e customizações de segurança do seu ambiente PHP.

Recomendamos que sejam ativadas na sua área de hospedagem diretivas de PHP inerente à segurança, lembrando que esta poderá afetar diretamente o funcionamento da sua aplicação PHP, pois algumas diretrizes tornam restritivas a ação de algumas funções, o que pode ter como consequência a falha em sua aplicação web, tornando-o parcialmente ou mesmo totalmente inoperante, o que motivará revisão e até mesmo modificação no seu código. Diante disso, devemos avaliar cada mudança que for feita e identificar o impacto em nossa aplicação. Lembramos que não damos suporte à aplicação do cliente.

Seguem algumas recomendações para o php-cgi.ini que se encontra na sua área de hospedagem, que só devem se aplicadas pelo programador do site após homologação da aplicação:

```

# ativar o modo de segurança

safe_mode = on

```

A linguagem PHP pode também ser utilizada para aplicações em ambiente shell, embora não seja a melhor linguagem para esse propósito. Algumas funções são muito poderosas, mas para um contexto web não são recomendadas. Diante disso, é recomendável desativá-las usando a opção “`disable_functions`” para este propósito, conforme o exemplo:

```
#desabilitar funcoes

disable_functions = php_uname, getmyuid, getmypid, passthru, leak,
listen, diskfreespace, tmpfile, link, ignore_user_abord, shell_
exec, dl, set_time_limit, exec, system, highlight_file, source,
show_source, fpaththru, virtual, posix_ctermid, posix_getcwd,
posix_getegid, posix_geteuid, posix_getgid, posix_getgrgid,
posix_getgrnam, posix_getgroups, posix_getlogin, posix_getpgid,
posix_getpgrp, posix_getpid, posix,_getppid, posix_getpwnam,
posix_getpwuid, posix_getrlimit, posix_getsid, posix_getuid, posix_
isatty, posix_kill, posix_mkfifo, posix_setegid, posix_seteuid,
posix_setgid, posix_setpgid, posix_setsid, posix_setuid, posix_
times, posix_ttyname, posix_uname, proc_open, proc_close, proc_
get_status, proc_nice, proc_terminate, phpinfo

#desabilitar mensagens gerais debug - opcional

display_errors = Off

register_globals = Off

# desabilitar funcoes de inclusão de arquivos

allow_url_fopen = off

allow_url_include = off

# desabilitar funcoes de upload de arquivos

file_uploads = Off

# ativar modo de segurança sql

sql.safe_mode = Off
```

Outras recomendações importantes, também inerentes à segurança de sua área de hospedagem:

- Revise todo o código de sua aplicação web e faça um novo upload, com o máximo de brevidade, pois uma vez que um invasor teve acesso à sua área de hospedagem, ele pode ter inserido trechos de códigos em seus arquivos com o objetivo de permitir acesso arbitrário, caso o meio pelo qual ele explorou a vulnerabilidade seja corrigido. Esse tipo de artimanha é facilmente implementado usando funções com “`passthru`”;



- ▣ Atualize seu antivírus e demais programas que possam identificar Malware, pois está cada vez mais comum a proliferação de Malwares desenhados para interceptar senhas ou mesmo contaminar os programas comumente usados para a publicação de sites. Uma vez interceptada a senha, o Malware passa automaticamente a realizar o download de páginas, inserir modificações e fazer um novo upload;
- ▣ Caso utilize SSH, recomendamos que passe a realizar as conexões utilizando chaves, pois mesmo que um Malware consiga interceptar sua senha e enviá-la para um invasor, sem a chave ele não conseguirá efetuar login em sua área de hospedagem;
- ▣ Dispomos de módulos para segurança específicos para o Apache (servidor web), tais como suhosin e mod_security e estes podem ser configurados em sua área de hospedagem.

Diante do que foi relatado, é recomendável que todo o código seja revisto com o máximo de urgência, pois não temos como garantir problemas de segurança vinculados à programação da aplicação do cliente. Podemos, sim, ser reativos e ajudar no momento em que o incidente é identificado, como foi o caso aqui. E esse site se encontra com problemas sérios e vulnerável no que tange à sua programação e na maneira em que as variáveis tratam os dados que são passados. E tenha em mente também que muitas invasões podem motivar problemas maiores, pois uma vez tendo acesso à sua área, o invasor pode usá-la para outros propósitos.

É fato que para eliminar todos os problemas, validando-os de forma a ter certeza que cada vulnerabilidade foi corrigida, é recomendável que o programador durante a fase de testes faça todas as simulações. Por esse motivo, buscamos destacar as linhas de logs mais relevantes do incidente. Dessa forma, o programador da aplicação pode analisar como foi explorado e, se quiser, pode criar um ambiente de simulação para tentar reproduzir o incidente como prova de conceito de que os problemas foram identificados e resolvidos.

Destacamos alguns links relevantes do site do projeto Open Web Application Security Project (OWASP), que reúne informações muito importantes para desenvolvimento de aplicações web.

OWASP – Ataques de PHP Injection:

- ▣ http://www.owasp.org/index.php/Path_Traversal
- ▣ http://www.owasp.org/index.php/Code_Injection
- ▣ http://www.owasp.org/index.php/Top_10_2007-Injection_Flaws
- ▣ http://www.owasp.org/index.php/Comment_Injection_Attack
- ▣ [http://www.owasp.org/index.php/Cross-site_Scripting_\(XSS\)](http://www.owasp.org/index.php/Cross-site_Scripting_(XSS))
- ▣ [http://www.owasp.org/index.php/Testing_for_IMAP/SMTP_Injection_\(OWASP-DV-011\)](http://www.owasp.org/index.php/Testing_for_IMAP/SMTP_Injection_(OWASP-DV-011))
- ▣ http://www.owasp.org/index.php/Parameter_Delimiter

OWASP – Ataques de SQL Injection:

- ▣ http://www.owasp.org/index.php/SQL_Injection
- ▣ [http://www.owasp.org/index.php/Testing_for_SQL_Injection_\(OWASP-DV-005\)](http://www.owasp.org/index.php/Testing_for_SQL_Injection_(OWASP-DV-005))

Hardening do serviço DNS/BIND

Um servidor de DNS baseado no BIND normalmente exerce uma das quatro funções em uma rede:

- ▣ **DNS Autoritativo:** sendo o servidor principal de um ou mais domínios;
- ▣ **DNS Secundário:** responsável pela réplica de domínios;



- **DNS Cache:** atuar somente como servidor de consulta;
- **DNS Forwarder:** encaminha consultas para outros servidores.

O ideal para implementação de servidores DNS é ter um servidor dedicado para cada função separadamente de outros serviços; dessa forma, a instalação de um Sistema Operacional em um servidor que vai suportar um serviço DNS deve ser a mais enxuta possível, devidamente direcionada para seu propósito.

Além das boas práticas de implementação, devemos configurar o BIND com ajuste finos que ajudem e reforcem a segurança de um servidor DNS.

Configurações recomendadas

A seguir, serão elencadas algumas configurações recomendadas na configuração do BIND:

- Executar o BIND como usuário;
- Evitar as técnicas de enumeração de verão;
- Limitar consultas recursivas;
- Limitar consultas sempre que possível;
- Limitar consulta ao cache do servidor sempre que possível;
- Uso de listas negras para evitar ataques de origem conhecida;
- Configuração de registros de eventos (logs).

Configurar da diretriz ‘version’

Verifique se os processos do servidor BIND que atenderão as consultas estão sendo executados por um usuário de sistema destinado para o fim, que não tenha home definido e nem shell válida, ou seja, o BIND não deve ser executado pelo usuário com ID 0 (usuário root).

A configuração da diretriz “option” é recomendável, pois por padrão um servidor BIND fornece informação da versão utilizada, assim o administrador pode impossibilitar a enumeração. Para a diretriz “version” ter efeito sobre todo o servidor, ela deve ser inserida no arquivo *named.conf*, dentro da sessão “options”, ou em arquivo que esteja diretamente vinculado via diretriz “include”.

```
options {
    version "None";
};
```

Limita consulta recursivas

Explicitar que o recurso de consultas recursivas estarão limitados a IP pré-definidos.

```
acl "localhost" {
    127.0.0.1;
};

acl "clients_corp"
{      192.168.0.0/24;
```

```

        192.168.1.0/24;
        `92.168.2.0/24;
    };

allow-recursion {
    localhost;
    clients_corp;
};

```

Onde “clients_corp” é uma ACL pré-definida na qual estarão relacionados os IP e Redes que poderão realizar uma pesquisa recursiva. Para a diretriz “allow-recursion” ter efeito sobre todo o servidor, ela deve ser inserida no arquivo *named.conf* dentro da sessão “options” ou em arquivo que esteja diretamente vinculado via diretriz “include”.

Limitação de consultas ao cache do servidor DNS

```

allow-query {
    localhost;
    clients_corp;
};

```

A diretriz “allow-query” limita as consultas ao servidor de Cache. Essa diretriz restringe somente a lista de servidores informada o direito a consultas. Para que a diretriz “allow-query” tenha efeito sobre todo o servidor, ela deve ser inserida no arquivo *named.conf* dentro da sessão “options” ou em arquivo que esteja diretamente vinculado via diretriz “include”.

Tratando-se de um servidor DNS Cache, pode ser interessante ativar a diretriz *allow-query-cache* na sessão “options” da mesma forma. Exemplo:

```

allow-query-cache {
    localhost;
    clients_corp;
};

```

Restrição de transferência de zona

É recomendável para qualquer tipo de servidor DNS a limitação de transferência de zona. Para isso, devemos utilizar a diretriz *allow-transfer*.

```

allow-transfer {
    localhost;
    clients_corp;
};

```

É recomendável centralizar o uso da diretriz “allow-transfer” na sessão *zone*, ou seja, onde são definidas as configurações das zonas primárias.



Lista negra

O conceito de lista negra no mínimo deve ser utilizado para eliminar a possibilidade de perda de tempo com consultas forjadas com IP origem inválido (não roteáveis segundo a RFC 1918). No entanto, também pode ser útil para o sysadmin restringir uma determinada range de IPs a realizar consultas a um servidor DNS específico.

```
acl "privados" {  
    10/8;  
    192.168/16;  
    172.16/12;  
};  
  
blackhole {  
    privados;  
};
```

Configuração de registros de eventos (logs)

O uso da diretriz “logging” é recomendável tanto pelo fato de que o registro de evento é uma necessidade para conformidade com qualquer norma ou boas práticas de segurança, quanto pelo fato de possibilitar ao administrador avaliar de forma palpável o funcionamento do serviço, diagnosticar problemas e também identificar possíveis abusos e ataques maliciosos.

A configuração é dividida em duas partes. A primeira parte, que é definida pela diretriz “channel”, consiste em definir o nível do log, como ele será rotacionado e em que arquivo será armazenado. Importante lembrar que a diretriz channel deve estar inserida na sessão “loggin”.

```
logging {  
  
    channel auditoria_log {  
        file "/var/run/named/auditoria.log" versions 10 size  
        20m;  
        severity debug;  
        print-time yes;  
    };
```

Quando se usa a diretriz “severity”, é possível ter oito níveis de logs:

- ▣ **critical**: logs de nível de criatividade mais alto;
- ▣ **error**: informações de erro;
- ▣ **warning**: notificações de problemas;
- ▣ **notice**: log de informações básicas;
- ▣ **info**: logs de informações gerais do funcionamento do servidor;

Saiba mais

- **debug [level]:** logs de erros muito detalhados para auxiliar a depuração de um problema;
- **none:** nenhum nível de severidade

Uma vez definido o “channel”, cria-se a regra de log, relacionada com a categoria que se deseja. A seguir, alguns exemplos.

Log de informações sobre o processamento dos arquivos de configuração do Bind:

```
category config {  
    audtiria_log;  
};
```

Log de informações de DNSSEC, caso esteja habilitado no servidor:

```
category dnssec {  
    audtiria_log;  
};
```

Informações gerais de comunicação de rede:

```
category network {  
    audtiria_log;  
};
```

Concentra informações gerais de consultas que foram requisitadas ao servidor Bind:

```
category security {  
    audtiria_log;  
};
```

Informações sobre atualizações de zonas:

```
category update {  
    audtiria_log;  
};
```

Parametrização para logs de transferências de zonas recebidas:

```
category xfer-in {  
    audtiria_log;  
};
```

Parametrização de logs de transferência de zona enviadas:

```
category xfer-out {  
    audtiria_log;  
};
```

A seguir, outro exemplo definindo o channel para logs para “debug”, mas habilitando apenas as categorias de logs padrões (default) e logs gerais (general):



```

// Optional debug log file, may be enabled dynamically.

channel debug_log {
    file “/var/run/named/debug.log”;
    severity dynamic;
    print-time yes;
};

category default {
    debug_log;
};

category general {
    debug_log;
};

};


```

Considerando os exemplos apresentados de configuração “logs”, no final, utilizando todos, teríamos:

```

logging {

channel auditria_log {
    file “/var/run/named/auditria.log” versions 10 size
20m;
    severity debug;
    print-time yes;

category config {
    auditria_log;
};

category dnssec {
    auditria_log;
};

category network {
    auditria_log;
};

category security {

```



```
        audtiria_log;  
    };  
  
category update {  
    audtiria_log;  
};  
  
category xfer-in {  
    audtiria_log;  
};  
  
category xfer-out {  
    audtiria_log;  
};  
  
channel debug_log {  
    file "/var/run/named/debug.log";  
    severity dynamic;  
    print-time yes;  
};  
  
category default {  
    debug_log;  
};  
  
category general {  
    debug_log;  
};  
};  
};
```

PS-Watcher: Monitoracão de serviços ativos

Monitora os serviços, com o objetivo de mantê-los ativos.

- Ação muito importante.
 - Deve ser uma ação em um processo de Hardening de serviço de redes, pois a "disponibilidade" também é um pilar de segurança.

Criar controle que possa aumentar a garantia de que os serviços estejam disponíveis é um valor agregador no processo de Hardening.



É possível monitorar computadores e serviços de rede com uma ampla variedade de soluções diferentes. Entre elas existem boas soluções FOSS de monitoramento, como o Nagios, Zabbix ou OpenNMS, mas ainda assim, requer um planejamento e ajustes. Todavia, quando precisa resolver problemas menores ou pontuais, com dados de processo em um sistema, o processo de monitoramento pode ser realizado pela ferramenta ps-watcher.

O ps-watcher é um dos bons exemplos de ferramenta do mundo Unix que faz uma coisa – e faz bem. Ele permite que sejam acessadas todas as informações do processo em um sistema e que seja possível tomar alguma ação com base nessas informações. O ps-watcher fornece uma interface consistente (com algumas ressalvas) para as informações do processo variadas, disponível em diferentes máquinas Unix e Linux, já tendo pacotes prontos para algumas das distribuições Linux, entre elas Ubuntu e Debian.

Depois de ter ps-watcher instalado, o próximo passo é criar um arquivo de configuração que contém as regras que desejamos usar para monitorar ou agir sobre o que o comando ps fornece de informação. Junto com o arquivo de configuração fornecido, podemos usar a linha de comando *opções* para ajustar o comportamento de ps-watcher e ajudar na depuração.

Periodicamente é extraída uma lista de processos obtidos via “ps”, onde cada item da lista contém o nome do processo (apenas o que está listado na “cmd” campo, não o comando completo e argumentos) e seu ID de processo (PID).

Em um arquivo de configuração, é especificada uma lista de expressões regulares para extrair informações dos processos para posterior análise. Para cada conjunto de informações extraídas do processo, é realizada uma avaliação. Cada conjunto de informações avaliada pode se referir a variáveis que são estabelecidas pelo PS e dizem respeito ao processo combinado avaliado, como, por exemplo, a quantidade de memória consumida pelo processo, ou o tempo total em que este está em execução. Algumas outras variáveis podem ser definidas, tal como o número de vezes em que o processo foi executado. Durante a análise, se é identificada uma instabilidade, o processo pode ser morto e reinicializado novamente.





Roteiro de Atividades 5

Atividade 5.1 – Sistemas de detecção de intrusões (IDS) em redes WLAN

Na prática, serão testadas as configurações refinadas para melhorar a segurança de serviços clássicos, como HTTP e o suporte a PHP e BIND.

1. Aplique todas as configurações recomendadas neste capítulo no servidor WEB.
2. Realize todos os testes propostos no capítulo 4 e avalie o que as configurações de Hardening proporcionaram de melhorias.
3. Realize o Hardening no BIND, aplicando todas as configurações propostas.
4. Configure o PS-WATCHER para controlar SSH, HTTP, BIND e FAIL2BAN, e realize testes para avaliar o seu funcionamento.





6

Hardening em sistema Linux – Proxy Web

objetivos

Aprender conceitos de servidor Proxy Web; Instalar o SQUID; Entender os conceitos de ACL; Conhecer ACL de Autenticação; Gerar relatório.

conceitos

Servidor de Proxy Web; Instalação do SQUID Proxy Web; Configuração de um servidor de Logs; Contabilização de Processos; Detecção de alteração no sistema – HIDS.

Introdução

Neste capítulo, serão apresentados conceitos de Proxy Web e também como executar a implementação de um servidor com a finalidade de Proxy Web baseado na ferramenta FOSS Squid. Vários aspectos da configuração, controle de conteúdo e autenticação também serão mostrados neste capítulo.

Exercício de nívelamento 1

Proxy web

Como você avalia o uso indiscriminado da web em uma corporação?

Existe controle de acesso ao conteúdo em sua empresa?

Avalie a possibilidade de geração de relatórios informando o conteúdo dos usuários de uma empresa como ferramenta de apoio à gestão do uso da web em ambiente corporativo.





Proxy:

- Serve como ponto intermediário entre um cliente e um servidor.

Podemos criar proxy para vários tipos de protocolo:

- smtp, pop3, http etc.

Servidor proxy Squid:

- Tem função de proxy web.

- Mantém um cache de conteúdo acessado de todos os clientes web de uma rede.

A função básica de um proxy na rede é servir como um ponto intermediário, um procurador entre a conexão de um cliente e um servidor. Normalmente, entre uma rede local e um serviço de rede na internet, existe a possibilidade de criação de proxy para vários tipos de protocolo, como smtp, pop3, http, entre outros. O servidor proxy Squid tem a função de proxy web, ou seja, mantém um cache de conteúdo web acessado de todos os clientes web de uma determinada rede.

O Squid é uma solução de software livre para proxy cache para a Web com suporte a protocolos HTTP, HTTPS e FTP, entre os principais. A implementação desse servidor em uma rede possibilita reduzir a largura de banda e melhora os tempos de resposta fazendo cache e reutilização de páginas da web frequentemente acessadas. Permite também a definição de controles de acesso amplos e pode ser um acelerador web. Existem versões para diversos Sistemas Operacionais, incluindo Windows, e está licenciado sob a GNU GPL.

Embora sua principal característica seja o proxy-caching sob os protocolos HTTP e FTP, o Squid suporta também:

- Proxying de conexões SSL;
- Hierarquia de cache;
- Aceleração HTTP;
- Caching de pesquisas DNS;
- SNMP.

O squid, como servidor proxy cache, funciona de forma objetiva, onde quando uma solicitação de uma estação na rede solicitar um mesmo conteúdo web estático, ele será fornecido via estrutura de cache. Para que isso funcione adequadamente, é recomendável definir regras de Firewall no perímetro da rede que redirecione conexões web para a porta do servidor proxy squid utilizada, que normalmente é a porta padrão 3128. O uso de servidor proxy web soma valor à segurança da rede, além de possibilitar o melhor uso do link sso, pois a única máquina que fará conexão web diretamente para a internet é o servidor proxy, e os clientes da rede se comunicam diretamente com ele.

Porém, se o conteúdo não tiver sido acessado antes e por consequência não estiver presente no cache, então o servidor vai baixar esse conteúdo para o seu cache. Dessa forma, para que lá exista uma cópia para uma consulta futura proveniente dos clientes, reduzindo assim o tempo de acesso e o consumo de banda para uso de internet.



Controle de conteúdo acessado na web

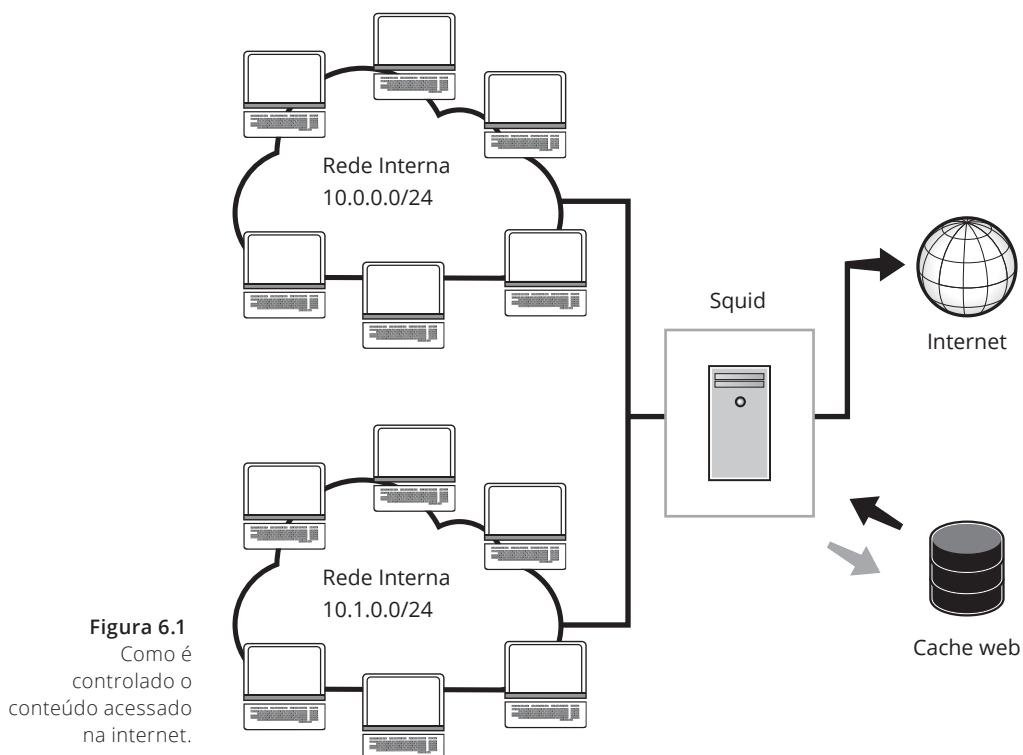


Figura 6.1
Como é controlado o conteúdo acessado na internet.

Implementação de um Proxy Web com Squid – Instalação e Configuração do Proxy Cache

Squid:

- Ferramenta open source para configuração de um proxy cache.
 - Suporta dados de protocolos como FTP, Gopher e HTTP.
 - Tem suporte a SSL, métodos de autenticação e políticas de controle de conexão baseadas em ACL (Access Control List – Lista de Controle de Acesso).

Para realizar a configuração de um proxy cache, será apresentada uma ferramenta open source de grande utilização para esse tipo de procedimento, que é o Squid, um dos melhores programas para a realização de proxy cache para clientes web. Suporta dados de protocolos como FTP, Gopher e HTTP. Também traz suporte a SSL, métodos de autenticação e políticas de controle de conexão baseadas em ACL (Access Control List – Lista de Controle de Acesso).

A próxima etapa a ser realizada é a instalação dos pacotes usados para a configuração do Squid, que será baseada em Debian GNU/Linux (embora o Squid possa ser encontrado em formato para compilação ou em outras distribuições que fazem uso de kernel Linux). Portanto, aconselhamos que o conhecimento dos métodos de gerenciamento de pacotes seja um pré-requisito para a configuração na distribuição que mais agrada ao usuário.

Na instalação dos pacotes, o gerenciamento será feito via APT. Poderá ser usado qualquer mirror FTP ou HTTP de árvore stable do Debian GNU/Linux que contenha o pacote do servidor Squid abordado nessa configuração. Após a realização de todo esse procedimento realizado, será instalado o pacote do Squid:

```
# apt-get install -y squid
```

Um procedimento normal de verificação em um sistema Ubuntu Linux, para saber se o pacote foi instalado ou não, pode ser realizado através do comando `dpkg`:

```
# dpkg -l | grep squid
```

Configurações básicas como Proxy Cache

Antes de serem iniciadas as configurações, um procedimento importante a ser executado é o backup dos arquivos de configurações originais, para manter uma cópia de segurança, caso ocorra algum imprevisto durante o procedimento de configuração do servidor. Por isso, devemos manter uma cópia do arquivo original do servidor Squid, que é o `squid.conf`.

Geralmente, em um servidor com Debian GNU/Linux, o arquivo de configuração do servidor squid, o `squid.conf`, está localizado em “/etc/squid”. Geralmente, essa é a localização padrão, mas também podemos encontrar no diretório “/etc”. Tudo vai depender de como o pacote da distribuição utilizada foi criado:

Cópia de segurança do arquivo de configuração original e criação de um arquivo sem os comentários.

```
# cd /etc/squid  
# cp squid.conf squid.conf.original  
# cat squid.conf.original | grep -v ^# | grep . > squid.conf
```

Dentro do arquivo de configuração do squid, em `/etc/squid/squid.conf`, o primeiro parâmetro a ser gerenciado ou criado será aquele que indica o Squid para trabalhar como servidor de cache web apenas para conexões a partir da interface voltada para a saída da rede local. Isso porque, uma vez definida a política de acesso excepcionalmente para a rede local, é recomendável que o Squid deve ser habilitado exclusivamente para rede local, ou seja, o socket de conexão (porta) não será publicado via interface externa (internet).

A finalidade é ter uma proteção maior contra a possibilidade de o servidor ser atribuído com a função de proxy anônimo e fazer parte de listas de proxy anônimos publicadas na internet. Outro motivador é que no cenário de proxy para rede LAN não é necessária a publicação para interface externa, colocando mais uma vez em prática o conceito de “menor privilégio e menor recurso”.

Essa característica é utilizada muitas vezes por invasores para mascarar os IPs de origem da invasão ou por usuários mal-intencionados que desejam burlar políticas de controle de acesso pré-estabelecidas.

Para isso, basta habilitar a porta para acesso ao proxy, que por padrão é listada na porta 3128 (supondo que a interface do proxy voltada para a rede esteja com o IP 10.0.0.1). Além disso, é recomendável também ativar o squid na interface loopback:

```
http_port 127.0.0.1:3128  
http_port 10.0.0.1:3128
```

Também é interessante a configuração do nome de exibição do servidor, pois quando o Squid retornar algum erro durante o acesso, o nome do servidor estará presente no rodapé dessas páginas de erro:

```
visible_hostname servidor.empresacom.br
```

Caso o nome completo do servidor seja identificado na configuração, esse nome deverá estar presente no arquivo de resolução local do servidor, o `/etc/hosts` relacionado com o IP do servidor ao qual está associado esse nome.

Somente essas alterações são necessárias para a simples ativação do Squid como um proxy cache básico. Agora o serviço do Squid pode ser iniciado através do script de inicialização dentro do diretório `/etc/init.d`:

```
# /etc/init.d/squid stop  
# /etc/init.d/squid start
```

Outra forma de iniciar o squid é via o comando `service`:

```
# service squid stop  
# service squid start
```

Após a inicialização do serviço, devemos verificar as portas ativas na pilha de rede do servidor.:

```
# netstat -nltp | more  
# netstat -nlup | more
```

Também é importante a verificação dos processos ativos nas respectivas portas alocadas pelo Squid, usando o fuser:

```
# fuser -v 3128/tcp  
# fuser -v 3130/udp  
# fuser -v 32768/udp
```

Uma terceira forma de consultar processos vinculados à porta é com o comando `lsof`:

```
# lsof -n -i :3128  
# lsof -n -i :3130  
# lsof -n -i :32768
```

Demandase também utilizar o comando `Nmap` para consultar se as portas estão ativas para conexões externas e também ativas na interface Loopback:

```
# nmap -sT -sU -n -P0 -p 3128,3130,33768 10.0.0.1  
# nmap -sT -sU -n -P0 -p 3128,3130,33768 127.0.0.1
```

Após isso, os clientes de acesso web, como os browsers das máquinas da rede, já podem ser configurados para acessarem a internet através do proxy. Os acessos dos clientes podem ser monitorados através dos logs relacionados ao Squid, que se localizam em `/var/log/squid`, utilizando por meio dos terminais inativos `tty10`, `tty11` e `tty12` para visualização da saída do log.:

```
# tail -f /var/log/squid/cache.log >> /dev/tty10 &  
# tail -f /var/log/squid/store.log >> /dev/tty11 &  
# tail -f /var/log/squid/access.log >> /dev/tty12 &
```

Uso de ACLs no Squid

Um dos principais conceitos presentes na configuração do Squid é o uso de Listas de Controle de Acesso (ACLs), para determinar diferentes regras de acesso através de vários métodos. Elas podem estabelecer níveis de controle de acesso aos determinados conteúdos requisitados pelos clientes.

As ACLs sempre possuirão um formato específico dentro do arquivo de configuração do proxy.:

```
acl nome tipo [“arquivo” | string]
```

Onde:

- **acl**: declara que será uma lista de controle de acesso;
- **nome**: é a identificação da ACL, o nome relativo que referencia essa regra;
- **Tipo**: é dado o formato de ACL a ser analisado, onde entre esses tipos os quais podemos ter, por exemplo, controle por IP de origem, por destino, por data ou hora, por palavra, por protocolo etc.;
- **String ou “arquivo”**: definimos qual a palavra-chave, o domínio, o IP ou a outra forma de controle (através do tipo de ACL) ou define-se um conjunto deles dentro de um arquivo no sistema.

Agora, dentro do arquivo de configuração (*/etc/squid/squid.conf*), poderia ser alocada uma linha com uma nova ACL, que, por exemplo, controla tudo o que se origina da rede local:

```
acl minharede src 192.168.0.0/255.255.255.0
```

Nessa linha, foi criada a ACL chamada minharede, que controla todas as requisições que possuem um determinado IP de origem (src); no caso, todo o range da rede 192.168.0.0/255.255.255.0.

Com essa ACL, define-se esse tipo de controle para a rede. Mas como o Squid vai trabalhar esse controle em específico? Ele sabe que deve analisar tudo que se origina na rede 192.168.0.0, mas o que fazer com essas requisições?

Para isso é que existem os controles dentro do arquivo do Squid chamados `http_access`, que determinam como essas requisições serão tratadas (permitidas ou não) em relação ao acesso ao documento requisitado.

O formato do parâmetro `http_access` é:

```
http_access [deny|allow] acl
```

Dentro do arquivo de configuração do Squid, será incluída uma nova linha que identifica o controle presente na ACL chamada minharede, para que seja permitido o seu acesso.:

```
http_access allow minharede
```

Estrutura de CACHE do Squid

No Squid, podemos definir diversos parâmetros para o armazenamento de cache de páginas, onde poderão ser definidos os vários níveis de diretórios ou partições diferentes existentes para controlar o acesso às informações em cache.

O diretório de cache seria o “pai” dos níveis de diretórios de swap do proxy. Pode ser usada também uma partição, mas o Squid não criará esse diretório e nem montará a partição: ambos devem estar disponíveis previamente.



- **# Mbytes:** espaço em disco (MB) para esse diretório (padrão é 100MB);
- **# Nível-1:** o número do primeiro nível de diretórios criado após o diretório “pai”;
- **# Nível-2:** o número do segundo nível de diretórios criado após o nível-1.

```
# cache_dir Diretório Mbytes Nivel-1 Nivel-2
```

Onde:

- **Mbytes:** é o espaço em disco (MB) para esse diretório (o padrão é 100 MB);
- **Nível-1:** é o número do primeiro nível de diretórios criado após o diretório “pai”;
- **Nível-2:** é o número do segundo nível de diretórios criado após o nível-1.

Pode-se identificar uma estrutura de cache específica para o proxy, na linha seguinte, por esse exemplo, com 512 MB, 128 diretórios e 256 subdiretórios.:

```
cache_dir /var/spool/squid 512 128 256
```

Depois de editados os parâmetros correspondentes, o arquivo de configuração final deverá ter um aspecto similar ao da página seguinte:

```
http_port 10.0.0.1:3128
icp_port 0
hierarchy_stoplist cgi-bin ?
acl QUERY urlpath_regex cgi-bin \?
no_cache deny QUERY
cache_dir ufs /var/spool/squid 500 32 256
refresh_pattern ^ftp: 1440 20% 10080
refresh_pattern ^gopher:1440 0% 1440
refresh_pattern . 0 20% 4320

acl all src 0.0.0.0/0.0.0.0
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl to_localhost dst 127.0.0.0/8
acl SSL_ports port 443 563 # https, snews
acl SSL_ports port 873 # rsync
acl Safe_ports port 80 # http
acl Safe_ports port 21 # ftp
acl Safe_ports port 443 563 # https, snews
acl Safe_ports port 70 # gopher
acl Safe_ports port 210 # wais
```



```

acl Safe_ports port 1025-65535 # unregistered ports
acl Safe_ports port 280          # http-mgmt
acl Safe_ports port 488          # gss-http
acl Safe_ports port 591          # filemaker
acl Safe_ports port 777          # multiling http
acl Safe_ports port 631          # cups
acl Safe_ports port 873          # rsync
acl Safe_ports port 901          # SWAT
acl purge method PURGE
acl CONNECT method CONNECT
acl minharede src 10.0.0.0/255.255.255.0
acl minhamaquina src 10.10.0.3
http_access allow manager localhost
http_access deny manager
http_access allow purge localhost
http_access deny purge
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports
http_access allow localhost
http_access allow minharede
http_access allow minhamaquina
http_access deny all
http_reply_access allow all
coredump_dir /var/spool/squid
visible_hostname proxy.morpheio.xxx.br

```

Registro de atividades

No que diz respeito à segurança de rede, a utilização de um proxy cache em uma permite também que sejam gerados registros de atividades dos usuários e que os mesmos registros sejam mantidos por tempo definido para corporação, possibilitando investigações futuras e também a monitoração do controle de acesso. Esses registros mantém a seguinte estrutura de dados:

- Identificação dos usuários:** no Squid, por padrão tem-se o registro de IP de acesso pelo cliente; porém, com o proxy utilizando autenticação, são vinculadas informações com o respectivo login do usuário;

- b. **Data e horários de entrada:** no Squid, esses dados são registrados por padrão;
- c. **Identidade do terminal ou, quando possível, sua localização:** no Squid, essa diretriz de recomendação não é aplicável;
- d. **Registro das tentativas de acesso aos aceitos e rejeitados:** no Squid, esses dados são registrados por padrão;
- e. **Registro das tentativas de acesso a outros recursos e dados aceitos e rejeitados:** no Squid, esses dados são registrados por padrão.

Estrutura de registros de eventos do Squid

Embora seja um padrão de configuração nativa a estrutura de arquivos de logs sugerida a seguir, ela será definida no arquivo `/etc/squid/squid.conf` para estabelecer a estrutura de logs.

Por padrão, determina-se a diretiva que identifica o arquivo de registros de conexões HTTP de clientes ativos no proxy, gravados em `/var/log/squid/access.log`:

```
cache_access_log /var/log/squid/access.log
```

A seguir se estabelece o parâmetro contendo o arquivo de registro de informações como hora e data em que o cache foi inicializado e o que foi armazenado, informações presentes em `/var/log/squid/cache.log`:

```
cache_log /var/log/squid/cache.log
```

Outra estrutura muito importante é o arquivo com o registro dos objetos que foram armazenados em consultas estabelecidas pelos clientes (páginas, figuras etc.), que deverá ser armazenado em `/var/log/squid/store.log`:

```
cache_store_log /var/log/squid/store.log
```

Criação de ACLs

Possibilita filtrar o conteúdo web dos usuários.

Existem três tipos de ACLs:

- ▣ `urlpath_regex`.
- ▣ `url_regex`.
- ▣ `dstdomain`.

Outro recurso motivador para o uso de um proxy-web é o conceito de filtro de conteúdo, que possibilita filtrar o conteúdo web dos usuários. Para realizar essa configuração, serão definidos três tipos de ACLs, combinando seus diversos recursos. Os tipos utilizados serão:

- ▣ **`urlpath_regex`:** filtro de uma string na URL;
- ▣ **`url_regex`:** filtro de complemento de uma URL;
- ▣ **`dstdomain`:** filtro de uma URL.

Para uma efetiva organização, define-se a política de conteúdo controlado em diversos arquivos localizados dentro de um diretório chamado `/etc/squid//listanegra`, que será criado:



```
# mkdir /etc/squid/listanegra
```

E a estrutura de arquivos dentro desse diretório seguirá a seguinte lógica:

- **palavraquente**: terá palavras-chave que serão filtradas;
- **excecoes**: sites que serão exceções à regra;
- **download**: extensões que serão filtradas;
- **sitequente**: sites de conteúdo pornográfico (exemplo: www.uol.com.br/playboy);
- **urlquente**: URL de sites pornográficos;
- **audiovideo**: sites com conteúdo de downloads e vídeos;
- **dominios_agressivos**: sites com conteúdo que tenha como tema agressões;
- **drogas**: sites sobre drogas;
- **jogospassatemplos**: sites de jogos como cassinos on-line;
- **violencia**: sites com fotos de violência ou acidentes;
- **warez**: sites de pirataria;
- **hacking**: sites de conteúdo hacking, com vírus, trojans e crackings.

As ACLs serão declaradas na sequência dentro do arquivo de configuração do Squid, nesta sequência, desta forma:

Relação de sites que serão exceções às políticas definidas:

```
acl excecoes dstdomain "/etc/squid/listanegra/site_excecoes"
```

Tratamento especial através de string dentro de uma URL:

```
acl palavraquente url_regex -i "/etc/squid/listanegra/palavraquente"  
acl download url_regex -i "/etc/squid/listanegra/download"  
acl sitequente url_regex -i "/etc/squid/listanegra/sitequente"
```

Tratamento a partir de uma URL ou endereçamento IP:

```
acl urlquente dstdomain "/etc/squid/listanegra/urlquente"  
acl audiovideo dstdomain "/etc/squid/listanegra/ audiovideo"  
acl aggressivos dstdomain "/etc/squid/listanegra/dominios_agressivos"  
acl drogas dstdomain "/etc/squid/listanegra/ drogas"  
acl jogospassatemplos dstdomain "/etc/squid/listanegra/  
jogospassatemplos"  
acl violencia dstdomain "/etc/squid/listanegra/violencia"  
acl warez dstdomain "/etc/squid/listanegra/warez"  
acl hacking dstdomain "/etc/squid/listanegra/hacking "
```

E agora serão feitas as chamadas das políticas de acesso para as ACLs criadas, através da diretiva `http_access`:

Relação de sites que serão exceções às políticas definidas:

```
http_access allow excecoes !palavraquente !download !sitequente  
!urlquente !audiovideo !aggressivos !drogas !jogospassatemplos
```



```
!violencia !warez !hacking
```

Outra forma muito útil de uso das ACLs de conteúdo é gerar controles de bloqueio de download baseado em extensões. Veja o exemplo:

```
acl download_extensoes url_regex -i \.exe$ \.com$ \.bat$ \.pif$  
    \.scr$ \.mp3$ \.vqf$ \.tar.gz$ \.gz$ \.rpm$ \.zip$ \.rar$ \.avi$ \.mpeg$  
    \.mpe$ \.mpg$ \.qt$ \.ram$ \.rm$ \.iso$ \.raw$ \.wav$ \.mov$ \.7z$
```

O uso de recursos de expressão regular (regex) é para informar que há um terminal com a extensão. Por isso o uso do “\$” e a contra barra, “\”, antes do ponto. O ponto deve ser interpretado como ponto, pois em regex é um caractere curinga.

Exemplo de uso:

```
http_access deny download_extensoes
```

Exercício de Fixação

Proxy web

Avalie o valor agregado que o controle de conteúdo web pode trazer em uma corporação.

Calcule o custo direto de um grupo de funcionários, considerando que eles perdem 2h diretas com o uso inadequado da web, ou seja, acessando conteúdo fora do foco. Assuma o valor homem hora médio de R\$ 3 para um cenário de 500 funcionários. Qual o custo fixo para uma corporação?

Considere o valor calculado na questão anterior, multiplique pelas 44 semanas úteis de um ano, para ter o valor dentro de um ano:

ACL Time – Regras temporizadas

Controle de acesso baseado em tempo é um dos recursos mais interessantes do Squid. Ao usar esse tipo de ACL, podemos especificar um período de tempo, na forma de dia(s) ou intervalo de tempo. Em seguida, os pedidos durante esse período de tempo serão correspondidos ou identificados por essa ACL.

O formato do tipo tempo ACL é:



```
acl ACL_NAME time [dia -abreviação] [hora_inicial-hora_final]
```

Dias	Abreviação	Tradução
Sunday	S	Domingo
Monday	M	Segunda-feira
Tuesday	T	Terça-feira
Wednesday	W	Quarta-feira
Thursday	H	Quinta-feira
Friday	F	Sexta-feira
Saturday	A	Sábado
All Weekdays	D	Dias úteis da semana

Tabela 6.1
Criação de ACLs.

Exemplo de uso:

Criação de três ACLs, uma para o horário da manhã, outra para o horário do almoço e outra para o horário da tarde.

```
acl manha time MTWHF 09:00-12:59  
acl almoco time D 13:00-13:59  
acl tarde time MTWHF 14:00-18:00
```

Para exemplificar o uso, serão criadas ACLs de conteúdo:

```
acl tranqueira dstdomain youtube.com facebook.com orkut.com bbb.  
globo.com
```

Os domínios da ACL “tranqueira” serão liberados exclusivamente no horário do almoço.

Primeiro exemplo:

```
http_access allow youtube !manha !tarde
```

Outra forma:

```
http_access allow youtube almoco
```

Outras diretivas importantes para funcionamento do proxy

Define-se que a limpeza automática de cache seja semanal. Caso isso não seja definido, o padrão é uma vez por mês. Isso será definido através da diretiva reference_age:

```
reference_age 1 week
```

Outro fator de segurança importante é a impossibilidade de se realizar cache de páginas seguras, para evitar que informações inválidas sejam consultadas posteriormente. Define-se esse padrão através do parâmetro no_cache:

```
no_cache deny SSL_ports
```

Quando o servidor de proxy web é configurado, podemos usar a autenticação, pois é um recurso muito interessante para controle de acesso.

O recurso de autenticação torna o gerenciamento dos “logs de acesso” muito mais prático e

interessante, pois possibilita a visão clara do uso do recurso pelo usuário. O Squid trabalha com o conceito de autenticadores externos para realizar tal função, ou seja, programas à parte para realizar os padrões de autenticação suportados pelo próprio proxy.

Para se especificar um autenticador externo para acesso ao cache do Squid – muito útil para manter o controle de usuários em relação ao nome/senha para acesso ao proxy –, devemos informar ao Squid quem é o programa externo responsável pela tarefa. Dos tipos de autenticadores externos, podem ser usados vários tipos, como autenticadores de padrão NCSA (como em servidores web), em uma base LDAP, em uma base Samba ou um domínio Microsoft.

Neste exemplo, é utilizado o autenticador padrão de servidor web, o ncsa_auth, por fazer parte da base do próprio Squid:

```
auth_param basic program /usr/lib/squid/ncsa_auth /etc/squid/passwd
```

O arquivo *squid.conf* vem comentado, informando outros métodos de autenticação, e a forma como o Squid foi concebido torna fácil para o administrador utilizar um outro método para autenticar usuários. Para exemplificar o que é dito, seguem instruções sobre como proceder para autenticar usuários Squid na base de usuários LDAP, que em servidores Linux é comumente implementado utilizando o OpenLDAP.

```
# vi /etc/squid/squid.conf  
  
auth_param basic program /usr/lib/squid/ldap_auth -b  
dc=teste,dc=xxx,dc=br -f uid=%s 192.168.0.124
```

Devemos definir o número de processos-filho para a realização do procedimento de autenticação, pois valores mais baixos servem para dificultar o uso de programas de “bruteforce” que tentam, através de ataques de dicionários, “adivinar” a senha do usuário:

```
auth_param basic children 5
```

Outra importante definição é o tempo de expiração da senha para a autenticação do usuário através de uma conexão cliente (esse tempo é dado em segundos):

```
authenticate_ttl 300
```

Agora será determinada uma ACL chamada password, do tipo proxy_auth, para ativar o recurso de autenticação.:

```
acl password proxy_auth REQUERID
```

Depois disso, o controle de acesso via diretiva http_access deve ser especificado, para fazer a chamada da ACL. No caso do Squid, a ordem das ACLs presentes é importante durante a análise do arquivo pelo servidor; por consequência, essa ACL deverá ser invocada antes de qualquer outra, se não a autenticação não será requerida corretamente durante o acesso pelos clientes:

```
http_access allow password
```

E, por fim, faz-se necessária a presença de uma ACL estratégica, no final do arquivo de configuração, negando qualquer possibilidade que não se enquadrou nas ACLs anteriores:

```
http_access deny all
```

Após todos os procedimentos, o arquivo de configuração do Squid pode ser salvo e o serviço ser recarregado, como feito anteriormente, para especificar todas as novas mudanças. O teste a ser realizado deverá ser o mesmo efetuado na configuração anterior, através da configu-



ração dos browsers clientes apontando para o servidor proxy.

Na prática existem duas formas de fazer o Squid se autenticar em um servidor Windows com Active Directory. Em uma delas, a feita por LDAP, o usuário terá de digitar a senha do domínio e, quando quiser navegar na internet, ele deverá se autenticar no proxy (Squid). Esse tipo de autenticação é feita por ldap. Outra forma é por NTLM, mas neste capítulo será exemplificada somente a autenticação por LDAP, pois também é a sugerida para autenticação no OpenLDAP.

O modo via AD é uma autenticação interessante. Basta colocar as seguintes linhas no *squid.conf*:

```
# vi /etc/squid/squid.conf
```

Linha para autenticação de usuários:

```
auth_param basic program /usr/lib/squid/ldap_auth -R -b
dc=dominio,dc=com,dc=br -D cn=squid,dc=dominio,dc=com,dc=br -w
"winserver" -f sAMAccountName=%s -h 192.168.0.1

# linha para autenticação de grupos
external_acl_type ldap_group %LOGIN /usr/lib/squid/squid_ldap_
group -R -b dc=dominio,dc=com,dc=br -h 192.168.0.1 -D cn=squid,d
c=dominio,dc=com,dc=br -w winserver -f "(&(cn=%a)(member=%v))" -F
"(sAMAccountName=%s)"
```

Caso seja um servidor OpenLDAP, podemos autenticar:

```
auth_param basic program /usr/lib/squid/ldap_auth -b
dc=teste,dc=xxx,dc=br -f uid=%s 192.168.0.124
```

Após incluir essas linhas, basta reiniciar o serviço do Squid que a autenticação já estará funcionando.

```
# /etc/init.d/squid stop
# /etc/init.d/squid start
```

A adoção de um Proxy Cache em uma rede é um procedimento extremamente viável dentro de uma topologia de rede, pois através dele podem-se estabelecer diversos controles sobre o acesso dos usuários e clientes da rede aos conteúdos externos, o que facilita o processo de auditoria e gerenciamento das possíveis consultas realizadas. Além disso, essa configuração visa a um melhor aproveitamento de recursos importantes da rede, como o uso de link externo, modelando o acesso desses mesmos clientes a fim de se ter ganho maior em performance, disponibilidade e simplicidade, tanto no acesso pelos usuários quanto pelos controles específicos dos administradores.

Outro fator importante que justifica o uso de um proxy web é ter controle sobre o conteúdo de acesso na corporação, pois um proxy web possibilita também gerar registros para auditoria de atividades dos usuários que podem ser mantidos por tempo definido dentro da corporação, possibilitando investigações futuras e também a monitoração do controle de acesso.

Esses registros devem manter a seguinte estrutura de dados:

- **Identificação dos usuários:** no Squid, por padrão, temos o registro de IP de acesso pelo cliente; porém, com o proxy utilizando autenticação, são vinculadas informações com o respectivo login do usuário;



Saiba mais

Não é foco desse treinamento a utilização de diretórios baseados no protocolo LDAP, mas é interessante lembrar que o Squid é uma solução madura e tem suporte a um grande número de métodos de autenticação, entre eles os diretórios baseado em LDAP, como OpenLDAP e Microsoft Active Directory.



A ABNT NBR ISO/IEC 27002:2008 recomenda a segregação de rede. Dessa forma, a utilização de um proxy web é um dispositivo de rede que, além de ajudar a estabelecer a segurança de perímetro, também coloca a rede em conformidade com a 27002.

- ▣ **Data e horários de entrada:** no Squid, esses dados são registrados por padrão;
- ▣ **Registro das tentativas de acesso aos aceitos e rejeitados:** no Squid, esses dados são registrados por padrão;
- ▣ **Registro das tentativas de acesso a outros recursos e dados aceitos e rejeitados:** no Squid, esses dados são registrados por padrão.

Diante disso, o uso de proxy web baseado no Squid possibilita também conformidade com o item 10.10.1 da normal.

Monitoramento dos logs do Squid com SARG

Gerenciar as atividades do usuários diretamente nos logs gerados pelo Squid é uma tarefa possível, mas é interessante tratar os logs de acesso via relatórios html gerados pela ferramenta Sarg. O Squid Analysis Report Generator (Sarg) foi desenvolvido pelo brasileiro Pedro Orso.

Para instalar o Sarg, devemos usar o comando *apt-get*:

```
# apt-get install -y sarg
```

A configuração do Sarg é centralizada no arquivo *sarg.conf*. Devemos alterar algumas definições dentro do arquivo *sarg.conf*, que é encontrado normalmente dentro do diretório */etc/squid*:

language Portuguese:

```
access_log /var/log/squid/access.log
title "Relatório SARG"
temporary_dir /tmp
output_dir /var/www/[seu-nome]/squid-reports
resolve_ip no
user_ip no
```

Depois de configurar o *sarg.conf*, basta gerar os relatórios com o comando:

```
# sarg
```

Após a execução, é necessário verificar a página criada em */var/www/[seu-nome]/squid-reports* e usá-la em seu navegador.

O pacote sarg traz consigo um script denominado “*sarg-reports*”. Após configurar o sarg, é possível gerar relatório de forma simples.

Relatório diário:

```
# sarg-reports daily"
```

Relatório semanal:

```
# sarg-reports weekly"
```

Além disso, faz parte do pacote a geração do arquivo */etc/crontab.d/sarg*, que faz o agendamento da geração dos relatórios via crontab.

Instalação do filtro de conteúdo Dansguardian

O Dansguardian é um filtro de conteúdo que pode ser integrado ao Squid para a filtragem de



“conteúdo”, muito mais arrojado que as regras baseadas em ACL disponíveis no Squid.

Para o processo de instalação e configuração do Dansguardian, é recomendável que sejam feitas cópias de segurança do arquivo *squid.conf*. Para isso, basta acessar o diretório:

```
# cd /etc/squid/
```

Fazer a cópia do *squid.conf* original:

```
# cp squid.conf squid.conf.ORIGINAL
```

A instalação do Dansguardian pode ser feita pelo repositório, através da ferramenta apt:

```
# apt-get install dansguardian
```

Os arquivos de configurações do Dansguardian estarão em */etc/dansguardian*:

```
# cd /etc/dansguardian  
# ls -l
```

É recomendável fazer uma cópia de segurança do arquivo de configuração *dansguardian.conf* antes de qualquer alteração:

```
# cp /etc/dansguardian/dansguardian.conf /etc/dansguardian/  
dansguardian.conf.ORIGINAL
```

Dentro do arquivo *dansguardian.conf* necessário: procurar a tag “UNCONFIGURED” e alterar para:

```
# UNCONFIGURED
```

Alterar a diretriz filterip para:

```
filterip=
```

Alterar a diretriz filterport para:

```
filterport=8080
```

Alterar a diretriz proxyip para:

```
proxyip=127.0.0.1
```

Alterar a diretriz proxyport para:

```
proxyport=3128
```

Após as alterações, devemos reiniciar o serviço do dansguardian:

```
service dansguardian stop  
service dansguardian start
```

Uma vez com o Dansguardian ativado, devemos realizar os testes via browser e avaliar os resultados fazendo simulações de acesso a site indevidos.

```
sudo /etc/init.d/dansguardian restart
```

Agora é só configurar os browsers para usarem o proxy.





Roteiro de Atividades 6

Prática de instalação, configuração dos controles de conteúdo, autenticação e geração de relatórios. Será necessário que o aluno utilize a máquina virtual preparada como Virtuabox. Dessa forma, será possível aplicar todos os conhecimentos (técnicas e ferramentas).

Atividade 6.1 – Hardening Linux – Proxy web – Squid

1. Configure um Squid para funcionar como um proxy web exclusivo de cache. Reinicie o Squid, teste-o e valide seu funcionamento e os logs gerados.
2. Salve as configurações do exercício 1 e atribua controle de conteúdo, reinicie o Squid, teste-o e valide seu funcionamento e os logs gerados.
3. Salve as configurações do exercício 2 e atribua controle utilizando a ACL TIME, reinicie o squid, teste-o e valide seu funcionamento e os logs gerados.
4. Salve as configurações do exercício 3 e atribua controle utilizando Autentica NCSA_AUTH, reinicie o Squid, teste-o e valide seu funcionamento e os logs gerados.
5. Configure o Sarg para gerar relatórios HTML.
6. Configure a automação de relatórios via crontab para todo dia, 3h.





7

Firewall – Parte 1

objetivos

Classificar um firewall quanto à sua atuação; Desenhar topologias que contemplem o uso de Firewall; Executar comandos de administração do IPtables; Configurar regras de filtro de pacotes do IPtables.

conceitos

Tipos de firewall e topologia de redes com firewall; Segurança de Perímetro e uso do recurso do Kernel do Linux denominado Netfilter, para implementação de firewall com auxílio da ferramenta Iptables; Parametrização e criação de chains, administração das tabelas; Exemplificações de como tratar certos serviços e protocolos (TCP, UDP e ICMP).

Introdução

Firewalls:

- Sozinhos, não têm como garantir a segurança de uma rede de computadores.
- Citados na norma internacional de Segurança da ISO/IEC 27002:2008.

Sistemas de firewalls são importantes em um projeto de segurança; todavia, sozinhos não têm como garantir a segurança de uma rede de computadores. Demandam-se outros mecanismos como também uma administração proativa. Mas a sua importância é notória, o que já é tratado na norma internacional de Segurança da ISO/IEC 27002:2008, ou seja, entre os controles da rede, há a recomendação de segregação da rede, assim como a implementação de proteção dos serviços disponibilizados contra acessos não autorizados – um trabalho para um sistema de filtros (firewall), para criar perímetros de redes devidamente protegidos e gerenciados.

Neste capítulo, será utilizado o Iptables, ferramenta que surgiu no Kernel 2.4, evoluiu, foi mantida no Kernel 2.6 e posteriormente também permaneceu na série 3.0. O IPtables possibilita ao administrador manipular as “capacidades” do Netfilter, o firewall de tecnologia State Full nativa no Linux.

Nosso objetivo será compreender como as comunicações de rede acontecem no contexto do protocolo, ou seja, trabalhar a matéria-prima, que é o TCP/IP, através de um método didático amigavelmente “Lego”, devido ao fato de que no processo inicial de aprendizado tratamos cada ação importante separadamente, como peças de um quebra-cabeça, para no final juntá-las com coerência e construir um modelo de políticas de firewall realmente funcional.

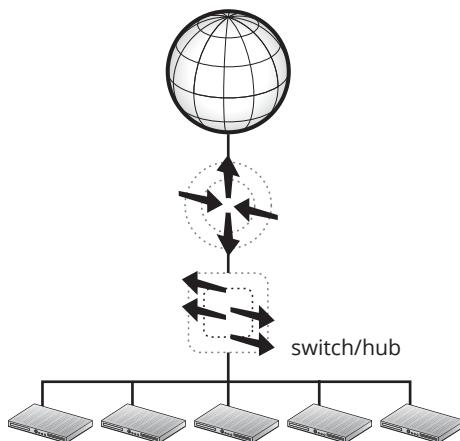


Arquitetura de firewall

Normalmente, as empresas preferem implementar um firewall baseado apenas em uma máquina, seja um host PC ou um roteador. Entretanto, os firewalls mais robustos são compostos por várias partes. Veja algumas arquiteturas a seguir.

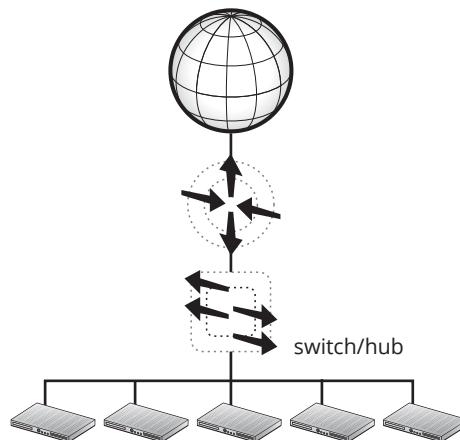
Roteador com Triagem

(Screening Router)



Roteador com Triagem

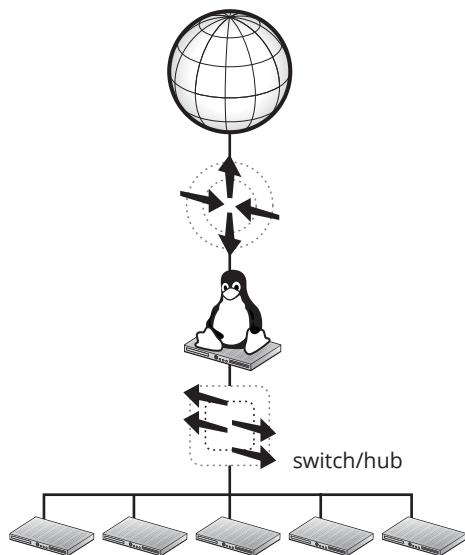
(Screening Router)



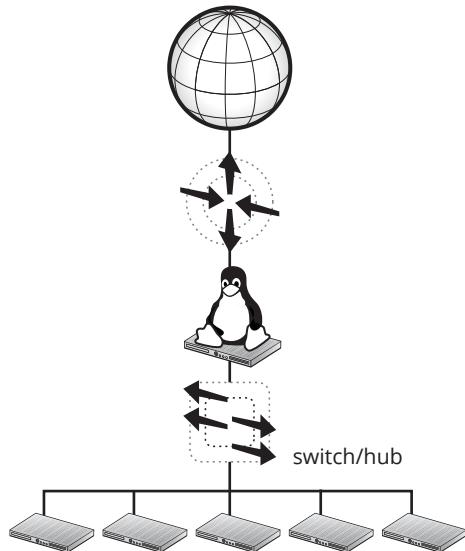
Boas Práticas recomendadas na ABNT NBR ISO/IEC 27002:2008: sistemas de firewall são importantes em um projeto de segurança; no entanto, sozinhos não têm como garantir a segurança de uma rede de computadores. Demandam-se outros mecanismos, além de administração proativa. Entretanto, sua importância é notória, a ponto de a norma internacional de Segurança da Informação BS7799, vinculada no Brasil pela ABNT como NBR ISO/IEC 17799:2001, no item 8.5.1, Controles da rede, recomendar a segregação da rede, assim como a implementação de proteção dos serviços disponibilizados contra acessos não autorizados. Isso significa um trabalho para um sistema de filtros (firewall) que, mais enfaticamente, é citado na NBR ISO/IEC 17799:2001 nas alíneas e e f do item 9.4.2, Rota de rede obrigatória.



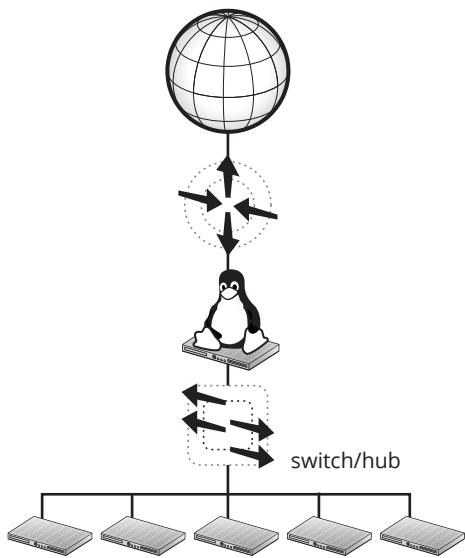
Gateway de Base Dupla
(Dual Homed Gateway)



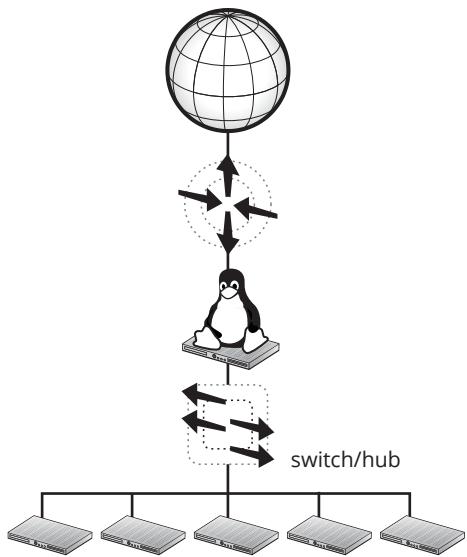
Gateway Host com Triagem
(Screened Host Gateway)



Gateway de Base Dupla
(Dual Homed Gateway)



Gateway Host com Triagem
(Screened Host Gateway)



**Gateway Host com Triagem
(Screened Subnet)**

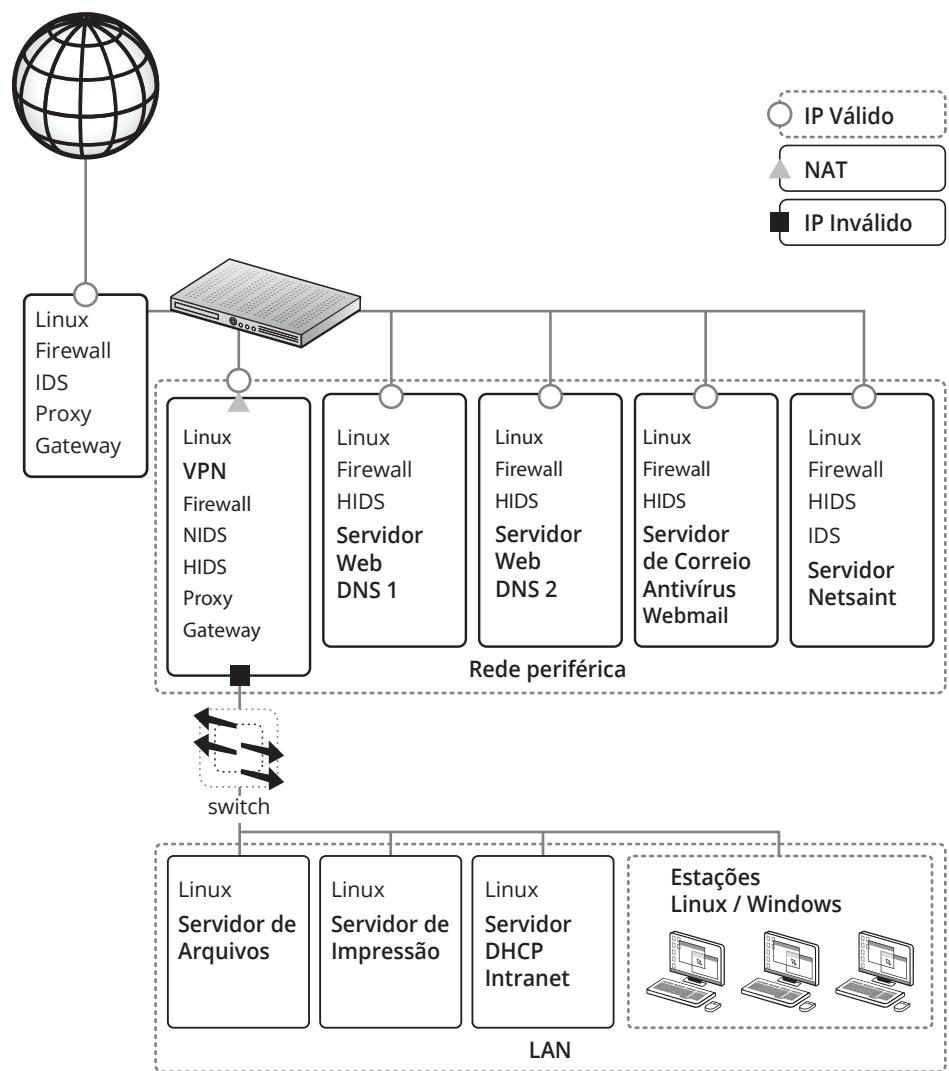
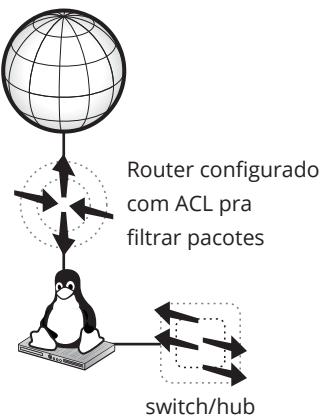


Figura 7.1
Composição dos firewalls mais robustos.

Tipos de firewall e como eles trabalham na estrutura de camada OSI.

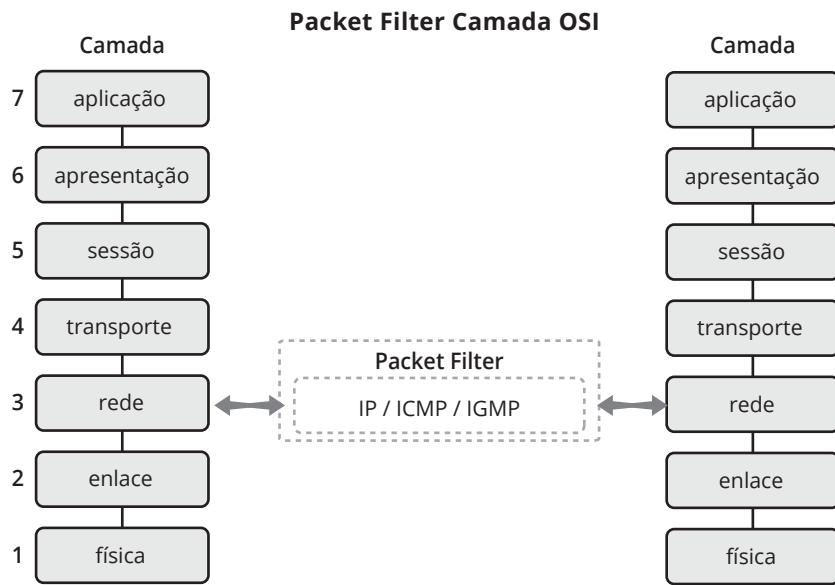


Figura 7.2
Firewalls e a estrutura de camada OSI.

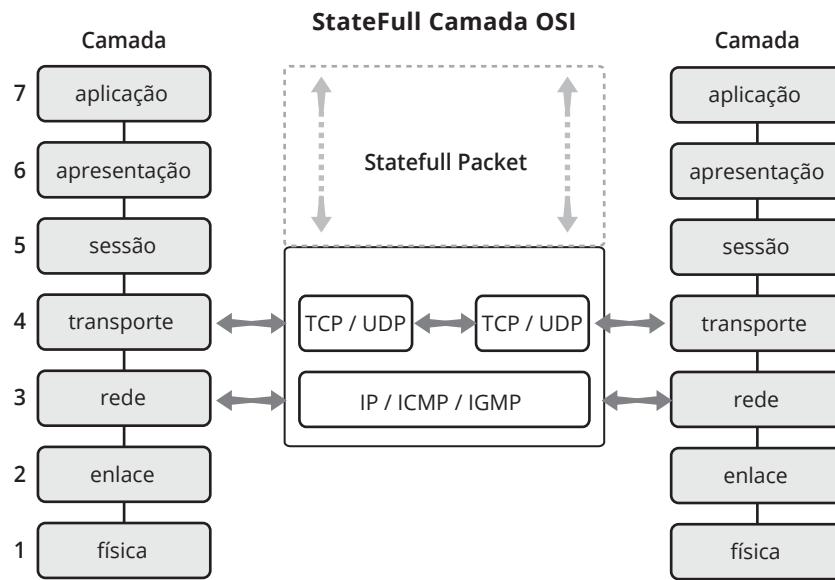


Figura 7.3
StateFull Camada OSI.

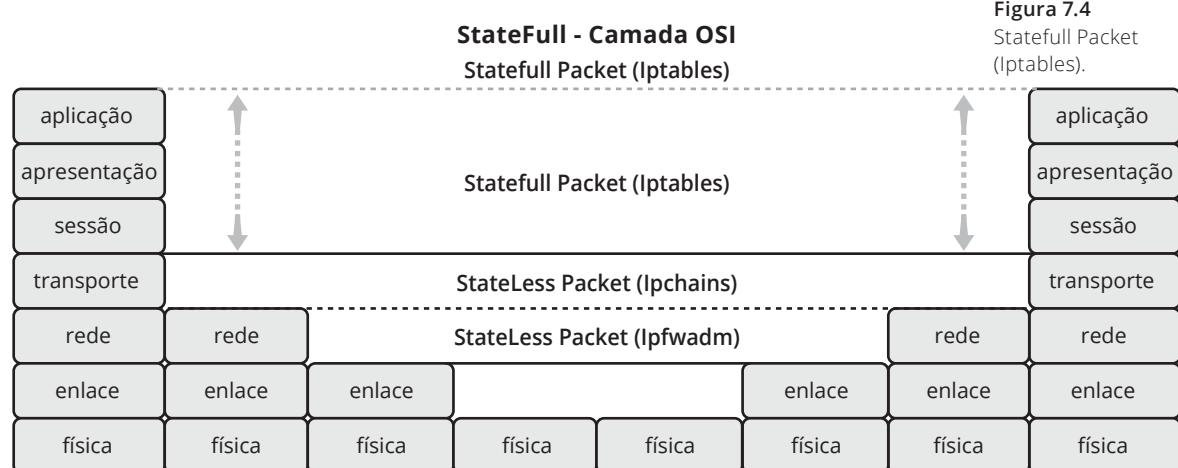


Figura 7.4
Statefull Packet (Iptables).

Uma visão a partir do datagrama

De uma forma direta e didática, pode-se classificar um firewall de acordo com seu nível de atuação em um datagrama, ou seja, o quanto em média do datagrama é efetivamente processado, ou quais os campos do respectivo cabeçalho podem ser relevantes para ação do firewall.

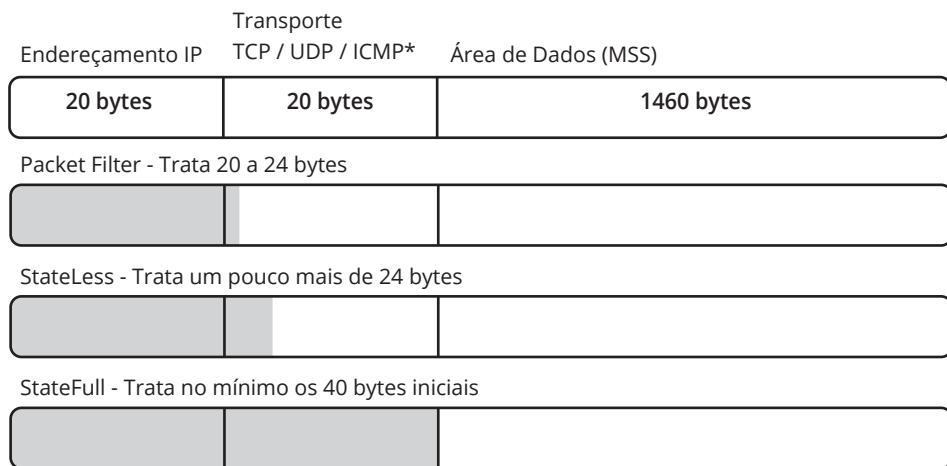


Figura 7.5
O firewall e o datagrama.

* O protocolo de transporte pode ter o cabeçalho de até 20 bytes

* Valor máximo do Datagrama (MTU) 1500 bytes

- ▣ **Packet Filter:** trata em média de 20 a 24 bytes iniciais de um pacote, ou seja, esse tipo de firewall trata e tem recursos para tratar todo o cabeçalho IP (os primeiros 20 bytes do pacote) e a parte do cabeçalho de transporte no que diz respeito à porta origem e porta destino. Lembrando que o campo “porta” tem 2 bytes (16 bits que correspondem a $2^{16}=65536$ portas);
- ▣ **StateLess:** trata em média um pouco mais que 24 bytes iniciais de um pacote, ou seja, esse tipo de firewall seria um Packet Filter melhorado. Sendo relevante lembrar que esse tipo de firewall pode até ter um tratamento para algum campo específico do cabeçalho de transporte, mas não é capaz de tirar proveito do conceito de Estado de Conexão;
- ▣ **StateFull:** essa categoria de firewall trata no mínimo os 40 bytes iniciais de um datagrama, ou seja, todos o cabeçalho IP e, seja qual for o protocolo de transporte (UDP ou TCP), são capazes de tratar o cabeçalho de Transporte. Sendo capaz de aproveitar as informações de estado de conexão do cabeçalho TCP.

Fundamentos para o IPtables

Antes mesmo de serem abordados conceitos mais específicos do IPtables, é interessante que estejamos familiarizados com alguns fundamentos básicos relacionados ao Netfilter e o kernel do Linux.

O Netfilter é a porção do Kernel do Linux que vai compor as capacidades de firewall na Kernel space e que será configurado e gerenciado pela ferramenta Iptables na userspace. Todavia, quando se menciona o firewall do Linux, é comum simplesmente denominá-lo Iptables.

Projeto em evolução, a cada nova versão o Netfilter torna-se cada vez melhor e mais competitivo, traz novos recursos para tratamento de pacotes e seguramente, pela sua qualidade, torna-se em muitos momentos ótima opção em relação aos firewalls comerciais. No entanto, quando uma empresa de grande porte faz a opção por uma solução de firewall baseada em software livre, não o faz pelo custo. Faz independentemente do preço. Colocamos em

primeiro lugar a funcionalidade e maturidade da tecnologia. Dessa forma, soluções de firewall baseadas em Linux ou mesmo baseadas em Unix BSD são usadas por serem funcionais e confiáveis.

Algumas das principais características do firewall IPTables

- ▣ Possibilidade de delimitar portas/endereço de origem/destino;
- ▣ Suporte a protocolos TCP/UDP/ICMP e IGMP, entre outros;
- ▣ Possibilita tratamento de pacotes por interface origem/destino;
- ▣ Redirecionamento;
- ▣ O conceito de tratamento de tráfego dividindo tabelas o torna extremamente versátil, possibilitando ao administrador definir políticas claras para cada situação, seja entrada, saída ou repasse de pacotes com tradução ou não de endereçamento;
- ▣ É estável e rápido;
- ▣ Possibilita o tratamento do estado da conexão, sendo muito fácil bloquear ou rejeitar pacotes manipulados por ataques (mal formados);
- ▣ O fato de sua estrutura se basear em módulos o torna um projeto facilmente escalável;
- ▣ Suporte a roteamento de pacotes, tratados em uma área diferente de tráfego padrão;
- ▣ Suporte à manipulação de campos de um datagrama;
- ▣ Capacidade de trabalhar com fragmentação;
- ▣ Possibilidade de personalizar o registro de eventos;
- ▣ Suporte a NAT;
- ▣ Contabilização de pacotes;
- ▣ Controle sobre os limites de pacote.

Trabalhando com IPTables

Projeto Netfilter:

- ▣ Possui a interface Iptables.
 - ▣ Extremamente arrojado.



Conforme já mencionado, o projeto Netfilter traz consigo a interface Iptables, que possibilita a manipulação de recursos do Kernel do Linux. Atualmente, essa é a opção de firewall dos administradores Linux. A proposta deste capítulo é abordar as capacidades do IPTables de forma didática, por meio de situações específicas, tentando sempre exemplificar e não apenas definir parâmetros. Todavia, o IPTables é extremamente arrojado, e está sendo melhorado gradativamente desde a sua primeira versão; por isso, não se tem a pretensão de abordar todos os seus recursos neste capítulo ou mesmo neste livro.

O Iptables trabalha com o conceito de tabelas, no qual cada uma delas trata o pacote de acordo com a situação vinculada, ou seja, com a direção e origem. Tabelas que filtram pacotes:

- ▣ **INPUT:** pacotes roteados para o firewall; o destino é o próprio firewall;
- ▣ **OUTPUT:** pacotes gerados localmente pelo firewall; tudo que a máquina firewall envia;
- ▣ **FORWARD:** tratam os pacotes que vão atravessar o firewall; de uma rede para outra, onde não existe a necessidade de tradução de endereçamento (regras de NAT).



Tabelas que tratam de NAT:

- **PREROUTING**: pacote entrando pelo firewall para uma rede interna, antes da decisão de roteamento. Em suma, a NAT de 1:1.
- **POSTROUTING**: saída dos pacotes da rede através do firewall para a internet.

Regras

De forma objetiva, pode-se definir um conjunto de regras pelo comando *IPtables* para realizar algum tipo de ação em relação de uma comunicação de rede (grupo de pacotes). Essas regras são mantidas em chains e permanecem até uma nova definição do administrador ou reinicialização do sistema. Um bom exemplo de regra seria bloquear requisições oriundas do IP 10.0.0.1 para a porta de destino 80/TCP através da interface eth1 com destino à própria máquina Firewall.

A sintaxe dessa regra para o *Iptables* fica assim:

```
# iptables -A INPUT -j DROP -i eth1 -s 10.0.0.1 -p tcp --dport 80
```

Dante disso, também pode-se definir como um conjunto de regras que especificam determinadas ações conforme características de uma comunicação de rede e sua direção (origem/destino).

Conceitos de chains

No *Iptables*, as chains são locais onde são armazenadas as regras do firewall, ou seja, é a forma como o *Iptables* organiza os critérios de decisões pré-definidas para o tráfego de pacotes de uma comunicação de redes. Essas regras são interpretadas de forma top-down (de cima para baixo), dando o devido tratamento ao pacote.

Existem dois tipos de chain: as nativas, ou seja, built-in, e as embutidas – as parametrizadas e definidas pelo administrador do sistema. As chains built-in são pré-estabelecidas no próprio *IPtables*, enquanto que as embutidas podem ser criadas/modificadas/removidas pelo administrador.

A tabela Filter possui três chains built-in, que são INPUT, OUTPUT e FORWARD. Veja o exemplo:

```
# iptables -n -L -t filter

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
```

Pelo fato de ser a tabela padrão, não é necessário passá-la como parâmetro, conforme explicado:

```
# iptables -n -L
```



Chain	Descrição
INPUT	Destinada a pacotes cujo destino a máquina <i>firewall</i> .
FORWARD	Destinada a pacotes que atravessam a máquina <i>firewall</i> .
OUTPUT	Destinada a pacotes cujo origem na <i>firewall</i> .

Tabela 7.1
Chais da tabela filter.

Tabelas

Tabelas são onde ficam armazenadas uma ou mais chains que têm características similares. Existem três tipos de tabelas no Iptables: a tabela FILTER, já definida, a tabela NAT (tradução de endereçamento) e a tabela Mangle (manipulação de campos do datagrama). Podemos referenciá-las através do parâmetro -t tabela.

```
# iptables -n -L -t nat

Chain PREROUTING (policy ACCEPT)
target     prot opt source          destination

Chain INPUT (policy ACCEPT)
target     prot opt source          destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source          destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source          destination
```

Chain	Descrição
PREROUTING	Destinada a pacotes que precisam de tradução antes do roteamento, ou seja, NAT Reverso.
POSTROUTING	Destinada a pacotes que precisam de tradução depois do roteamento.
OUTPUT	Destinado a pacotes gerados no próprio <i>firewall</i> , mas que precisam de tradução antes do roteamento, ou seja, um tipo Prerouting para pacotes com origem no próprio Firewall.

Tabela 7.2
Chais da tabela NAT.

A tabela Mangle é utilizada em situações onde houver a necessidade de alterações especiais em pacotes. A tabela Mangle originalmente tem cinco chains: PREROUTING, POSTROUTING, INPUT, OUTPUT e FORWARD. Na prática, cada uma dessas chains é processada antes da chain correspondente na tabela filter e nat, para definir opções especiais, como, por exemplo, opções como o Tipo de Serviço (TOS).



Caminho de um pacote nas tabelas e chains

A figura a seguir ilustra o fluxo e a sequência de processamento que um pacote segue quando entra no firewall IPtables. Onde as chains são analisadas sequencialmente de cima para baixo conforme é ilustrado na figura. Quando um pacote confere com uma regra, a ação especificada na regra é executada no pacote.

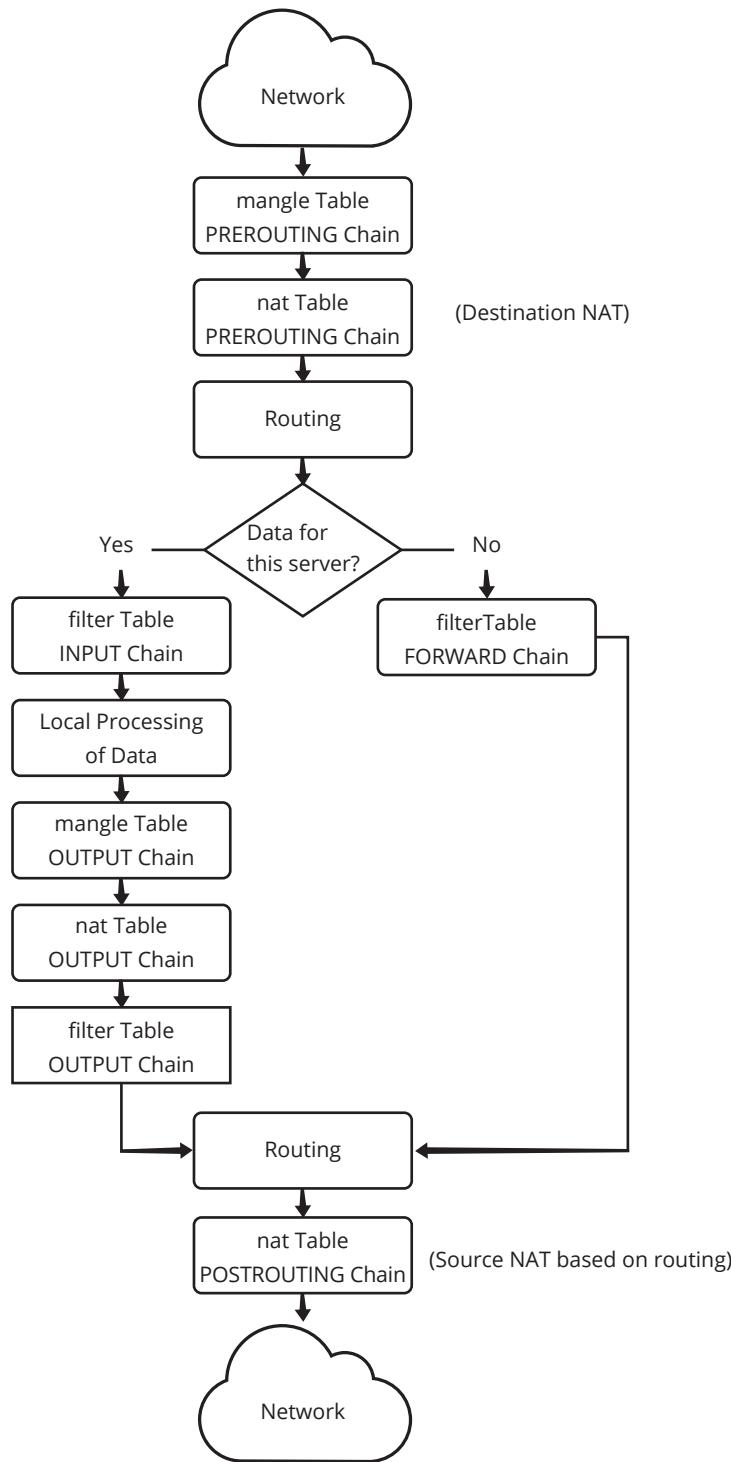


Figura 7.6
Fluxo e sequência de processamento que um pacote segue quando entra no firewall IPtables.



Como adicionar regras em uma chain

O parâmetro **-A (append)** é o responsável por adicionar regras às chains, onde a última regra adicionada via esse comando será a última a ser processada, pois cada nova regra adicionada vai para o fim da fila.

```
Sintaxe: IPtables -t tabela -A chain [especificação da regra] [opções]
```

Exemplificação de uma regra negando qualquer pacote ICMP com destino a interface lo (loopback).

```
# iptables -A INPUT -j DROP -i lo -p icmp
```

Ou:

```
# iptables -t filter -A INPUT -j DROP -i lo -p icmp
```

Listando (-L) as regras em tabelas e chains

Conforme já exemplificado, pode-se verificar as regras ativas em memória de todas as chains de uma tabela ou em uma chain específica através do parâmetro **-L**.

```
Sintaxe: IPtables -t tabela -L [chain] [opções]
```

As opções são parâmetros que são passados para a elaboração de um regra mais refinada e detalhada. A tabela mostra as principais opções:

Opção	Descrição
-v	Modo verbose, mostra informações mais detalhadas sobre a regra.
-n	Apresenta informações em formato numérico, como por exemplo: IP e Portas.
-x	Exibe informações absolutas.
--line-numbers	Exibe o número de cada regra.

Tabela 7.3
Opções das chains.

Exemplificação de uma listagem completa de informações ativas da tabela Mangle:

```
# iptables -t mangle -n -L -v --line-numbers

Chain PREROUTING (policy ACCEPT 1545 packets, 635K bytes)
num  pkts bytes target     prot opt in      out      source
destination

Chain INPUT (policy ACCEPT 1545 packets, 635K bytes)
num  pkts bytes target     prot opt in      out      source
destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
```



```
num pkts bytes target      prot opt in     out      source  
destination
```

```
Chain OUTPUT (policy ACCEPT 808 packets, 62046 bytes)
```

```
num pkts bytes target      prot opt in     out      source  
destination
```

```
Chain POSTROUTING (policy ACCEPT 826 packets, 63337 bytes)
```

```
num pkts bytes target      prot opt in     out      source  
destination
```

Os campos exibidos na listagem têm os seguintes significados:

Campo	Descrição
Num	Número da regra na chain.
Pkts	Número de pacotes que bateram com a regra.
Bytes	Número de bytes relacionados com a respectiva regra.
Target	O alvo a ser realizado caso um pacote confira com a regra.
Prot	Protocolo tratado pela regra criada.
Opt	Opções extras passadas na regra i criada.
In	Interface de entrada tratada.
Out	Interface de saída tratada.
Source	Endereço origem.
Destination	Endereço destino.
Outras opções	Demais campos complementares, como por exemplo: porta de origem/destino, match que está sendo utilizada.

Tabela 7.4
Campos da listagem de informações ativas da tabela Mangle.

Apagando (-D) uma regra

Com o “-D” é possível apagar uma chain de duas formas: através do índice ou número da regra (representado por num), ou reescrevendo a regra novamente mas trocando o parâmetro “-A” ou “-I” pelo parâmetro “-D”. Veja os exemplos:

```
# iptables -D INPUT 8  
# iptables -t nat -D POSTROUTING 5  
# iptables -D FORWARD -j ACCEPT -s 10.0.0.0/8 -d 172.16.0.0/16
```

Inserindo (-I) uma regra

Possibilita inserir um regra em uma posição definida em uma chain.



Sintaxe: IPtables -t tabela -I chain [índice] [especificação da regra] [opções]

```
# iptables -I FORWARD 3 -m state --state NEW -d 10.0.0.0/0 -j ACCEPT
```

Substituindo (-R) uma regra

Funciona de forma similar à inserção de regra, mas com o propósito de substituir uma determinada regra. Vide exemplos:

```
# iptables -R OUTPUT 15 -j ACCEPT -o eth0 -p icmp --icmp-type echo-request  
# iptables -R INPUT    16 -j ACCEPT -i eth0 -p icmp --icmp-type echo-reply
```

Criando uma nova (-N) chain

A possibilidade de criar chain proporciona a oportunidade de o administrador criar firewalls mais arrojados e eficientes, pois pode criar chains para tratar situações específicas. O parâmetro utilizado para criar uma nova chain é o -N, conforme ilustrado:

Sintaxe: IPtables -t tabela -N [nome]

Após a criação, a chain pode ser manipulada como qualquer outra chain, como inserir, remover, substituir ou adicionar:

```
# iptables -N DROP_NET  
# iptables -A DROP_NET -j DROP -i eth0 -s 172.16.0.0/16  
# iptables -A DROP_NET -j DROP -i eth0 -s 172.17.0.0/16
```

Uma vez definido qual será o tratamento ao pacote na respectiva chain, deve-se criar uma regra na tabelas principais para que o fluxo de tratamento seja desviado para a chain criada.

```
# iptables -A INPUT -j DROP_NET -s 172.16.0.0/16  
# iptables -A INPUT -j DROP_NET -s 172.16.0.0/16
```

Renomeando (-E) uma chain

Pode-se renomear o nome de uma chain através do parâmetro “-E”:

Sintaxe: IPtables -t tabela -E [nome-existente] [novo-nome]



Isso só é possível para chains criadas, e não se aplica às chains padrão do IPtables.

Apagar (-F) as regras de uma chain

Existem duas maneiras de apagar as regras existentes em uma chain: ou apagamos as regras de cada chain individualmente ou apagam-se todas as regras das chains existentes em uma tabela.

```
# iptables -t filter -F DROP_NET  
# iptables -t filter -F
```

Zerando (-Z) contadores de chains

O recurso de contabilidade de pacotes processados pelas chains pode ser muito interessante para administradores que desejam ter uma visão estatística a partir de informações do firewall. Esse recurso é ativado por padrão e pode ser consultado com o uso do -v ou -x na listagem de regras de uma chain. Para reiniciar os contadores, pode-se usar o parâmetro -Z, conforme ilustrado:

```
# iptables -Z INPUT  
# iptables -t mangle -Z OUTPUT
```

Excluindo (-X) chains criadas por usuários

Quando desejamos excluir chains criadas pelo administrador individualmente ou todas as existentes em uma tabela:

```
# iptables -t tabela -X  
# iptables -t tabela -X DROP_NET
```

Política básica

A política básica, também chamada de política padrão, tem como objetivo definir qual o tratamento do pacote caso ele não confira com qualquer regra em uma chain. Classicamente existem dois modos, o modo permissivo e o modo restritivo.

- **Modo permissivo (ACCEPT):** o firewall é definido para aceitar qualquer pacote caso o pacote não tenha sido tratado por qualquer outra regra;
- **Modo restritivo (DROP):** firewall é definido para negar qualquer pacote caso o pacote não tenha sido tratado por qualquer outra regra.

No Iptables as regras padrão são definidas pelo comando -P:

```
# iptables -P INPUT DROP  
# iptables -P OUTPUT DROP  
# iptables -P FORWARD DRO
```

Especificando interfaces de entrada e saída

Os parâmetros -i e -o definem interfaces de entrada (origem) e saída (destino), respectivamente, para os pacotes de rede. Sua sintaxe longa para esses parâmetros são --in-interface e --out-interface. Na criação de regras mais restritivas a um determinado fluxo de pacotes, a parametrização da interface é um recurso muito útil.

Entretanto, nem toda chain aceita os dois parâmetros simultaneamente, ou seja, tem de se levar em consideração o fluxo de pacotes, pois não tem sentido por exemplo especificar uma interface de saída na chain INPUT da tabela filter que vai tratar o fluxo de entrada de pacotes. A tabela a seguir faz uma relação da possibilidade de utilização dos parâmetros -i e -o em todas as chains.



Tabela	Chain	Interface Entrada (-i)	Interface Saída (-o)
Filter	INPUT	SIM	NÃO
	FORWARD	SIM	SIM
	OUTPUT	NÃO	SIM
Nat	PREROUTING	SIM	NÃO
	POSTROUTING	NÃO	SIM
	OUTPUT	NÃO	SIM
RAW	PREROUTING	SIM	NÃO
	OUTPUT	NÃO	SIM
Mangle	INPUT	SIM	NÃO
	FORWARD	SIM	SIM
	OUTPUT	NÃO	SIM
	PREROUTING	SIM	NÃO
	POSTROUTING	NÃO	SIM

Tabela 7.5
Opções de chains para cada tabela do Netfilter.

A tabela RAM só pode ser usada nas chains PREROUTING e OUTPUT. Não há suporte para outras chains porque essa forma de tratamento só é necessária uma vez que esses são os únicos lugares em que se pode lidar com os pacotes antes que eles realmente fechem uma trilha de conexão.

Proposta de regras

- Uso do Iptables para configurar uma máquina com os serviços de firewall e Proxy Web.
- Pode fazer o papel de firewall de fronteira de uma pequena ou grande rede.



Este capítulo terá como objetivo também primeiro conceituar e exemplificar o uso do Iptables para configurar uma máquina com os serviços de firewall e Proxy Web. Normalmente, um servidor configurado para essa finalidade pode fazer o papel de firewall de fronteira de uma pequena ou mesmo grande rede; dessa forma, poderá filtrar todo o tráfego da internet para a rede interna, e da rede interna para a internet. Já um proxy pode ser utilizado como filtro de conteúdo web.

A metodologia utilizada para a implementação do firewall será a seguinte: iremos “negar” todo o tráfego para as CHAINS de INPUT, OUTPUT e FORWARD da tabela filter e, posteriormente, o gerente de TI nos passará a relação dos serviços que devem ser liberados no Firewall, o que chamaremos de exceções. Todo o tráfego de pacotes que as minhas exceções não cobrirem estarão bloqueados por padrão. Em suma, o que não for oficialmente permitido já está expressamente negado. Para finalizar, é interessante que façamos uma auditoria em cima de tentativas de intrusão e varreduras qualificadas como regras de controle.



Organização das políticas

Figura 7.7
Depois das políticas definidas em outras chains, as políticas básicas.



As políticas básicas para o firewall são tratadas por último depois que o pacote passar todas as políticas definidas nas outras chains; todavia, elas são definidas primeiro para depois serem definidas as chains de políticas de exceções e os controles.

Parte-se do princípio de que seu sistema Linux está devidamente configurado, ou seja, um hardening restritivo foi executado no servidor e o suporte a IPtables está implementado.

Para tornar o estudo mais didático, inicialmente serão conceituadas as políticas dividindo-as em:

- **Básicas:** inerentes a tratamentos da base do firewall, ou partiremos do princípio de negar tudo o que não é permitido, já que está expressamente negado;
- **Exceções:** serão todas as políticas destinadas a permitir algum tipo de acesso;
- **Controles:** são políticas que vão possibilitar a geração de registros de eventos, ou seja, todas a políticas diretamente vinculadas a logs.

Inicialmente, partir para três chains básicas, negando qualquer acesso para INPUT, OUTPUT e FORWARD. Para isso, usaremos o comando -P.

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

Existe a possibilidade de, em vez de negar um pacote (DROP), rejeitá-lo (REJECT), o que não aplica as regras básicas onde somente pode-se ou aceitar ou negar; entretanto, é válido lembrar que o ato de rejeitar pacotes tem o custo de que para cada pacote rejeitado é enviada uma resposta padrão ICMP do tipo ICMP3/3.

Seguindo a mesma lógica, se for desejado ter um servidor totalmente permissível de forma suicida na internet, bastaria delimitar as políticas básicas aceitando qualquer conexão, o que com certeza em momento algum é recomendável; todavia, tal cenário poderia ser definido assim:

```
iptables -P INPUT ACCEPT  
iptables -P OUTPUT ACCEPT  
iptables -P FORWARD ACCEPT
```

Regras adicionais a regras básicas

Com a política básica para três chains fixadas em "drop", ou seja, descartando silenciosamente pacotes, teríamos um servidor seguro. Este não estaria recebendo nenhum pacote, mas por outro lado também não estaria enviando nada. Embora seguro, não tem sentido e não é o que se deseja, mas essa é a política básica recomendada para construção de firewall;



assim, a partir desse momento, o administrador pode adicionar políticas de exceções de forma incremental, ou seja: na ordem em que forem adicionadas, elas serão tratadas pelo kernel do Linux e posteriormente políticas de controles para motivar registro de eventos (logs) indesejados, pois mesmo já negados, é recomendável ter o controle sobre as ocorrências para fins de auditoria e avaliação da segurança.

É importante lembrar que de uma forma geral a construção de regras de firewall são fundamentadas nas “políticas básicas”, sendo as regras de exceção e controles (logs) que tratarão boa parte do tráfego de redes. Ocorrerá também a necessidade de tratar ainda assim as exceções. Exemplo: o administrador liberar comunicação via protocolo ICMP, mas mesmo assim será necessário dizer qual o tipo de ICMP e talvez a quantidade.

Esse último exemplo ratifica a necessidade de uma análise detalhada do que tem de se fazer e também o melhor uso da ferramenta. Ter um firewall com um grande número de regras não define se é bom ou ruim. Quase sempre se o número de regras é muito grande é bem provável que muita coisa pode ser otimizada.

Além do que já foi mencionado, existe a necessidade de tratar melhor um determinado datagrama ou mesmo reescrever um determinado campo de um protocolo.

Firewalls também são responsáveis pela relação de IPs válidos na internet versus os não roteáveis quando a comunicação é baseada em IPv4, ou seja, recursos com PAT e NAT que serão explicados no capítulo 8.

E, por fim, o controle de Banda passa ser uma função que em muitos momentos pode e deve ser combinada com as capacidades do Firewall utilizado. A pirâmide a seguir ilustrar esses conceitos e a inter-relação entre eles.



Figura 7.8
Controle de Banda.

Essa ação pode ser automatizada, e é o ideal. Dessa forma, segue uma exemplificação de um script denominado drop.sh para definir a política padrão:



```
# vi drop.sh
# ./drop.sh
```

Figura 7.9
Definindo a política básica.

Para limpar as regras em todas as chains e voltar com a política padrão ACCEPT, eis um exemplo de script que customiza a tarefa. Ele foi denominado com o nome “limpa.sh”.

```
# vi limpa.sh
```

```
#!/bin/bash

PF=$(which IPtables)

if [ -z $PF ] ; then
    echo "Comando IPtables nao encontrado"
    exit
fi

nega ()
{
    $PF -P INPUT DROP
    $PF -P OUTPUT DROP
    $PF -P FORWARD DROP
}

nega
#!/bin/bash

PF=$(which IPtables)

if [ -z $PF ] ; then
    echo "Comando IPtables nao encontrado"
    exit
fi

limpa ()
{
    $PF -F
    $PF -P INPUT ACCEPT
    $PF -P OUTPUT ACCEPT
    $PF -P FORWARD ACCEPT
```



```
}
```

```
limpa
```

Normalmente, inicia-se um script definindo a shell que interpretará o seu conteúdo; todavia, a linha `#!/bin/bash` não é inserida, as instruções do script são interpretadas pela shell corrente ou pode ser definida a shell durante a execução, precedendo o nome do script pela shell desejada conforme exemplo ilustrativo: `# bash script.sh`

Exemplo de um script para visualizar as políticas na tabela FILTER, denominado como `status.sh`:

```
# vi status.sh

PF=$(which IPtables)

if [ -z $PF ] ; then
    echo "Comando IPtables nao encontrado"
    exit
fi
ver ()
{
$PF -nL -t filter
}
ver

# ./status.sh
```

Como foi definido o modo restritivo nas políticas básicas, ou seja, foi fechado (DROP) para todas as chains que definem as políticas básicas do firewall, será necessário criar políticas de exceções, ou seja, regras de liberação (ACCEPT), para a comunicações pré-definidas pelo administrador.



Figura 7.10
Regras de liberação
(ACCEPT).

Para melhor contextualizar, será proposto um cenário onde desejamos a execução do comando `ping`. Dessa forma, é interessante sempre liberarmos a saída e a entrada de dados para a própria máquina; para tanto, temos a interface loopback "lo".

```
iptables -A INPUT -i lo -d 127.0.0.1 -j ACCEPT
iptables -A OUTPUT -o lo -s 127.0.0.1 -j ACCEPT
```



Com essa política, a máquina consegue realizar “ping” a si própria, ou seja, origem e destino na própria máquina.

Exemplo de script de customização para definir uma política para a interface de loopback:

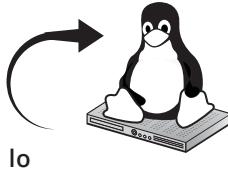


Figura 7.11
Política para a interface de loopback.

```
# vi loop.sh

PF=$(which IPtables)

if [ -z $PF ] ; then
    echo "Comando IPtables não encontrado"
    exit
fi

L0="127.0.0.1"

loop ()
{
    $PF -A INPUT -i lo -d $L0 -j ACCEPT
    $PF -A OUTPUT -o lo -d $L0 -j ACCEPT
}
loop

# ./loop.sh
```

Mas essa política anterior é egoísta, pois só possibilita “pingar” o próprio host, mas nenhum host ou mesmo outras interfaces do próprio host. Considere que está definido o IP 10.0.0.1 na interface eth0 e deseja-se que o servidor seja capaz de encaminhar a consulta de ping para qualquer outros host. Dessa forma, é necessário adicionar políticas para echo-request e echo-reply

```
# iptables -A INPUT -j ACCEPT -p icmp --icmp-type 0 -d 10.0.0.1 -s
0/0 -i eth0

# iptables -A OUTPUT -j ACCEPT -p icmp --icmp-type 8 -s 10.0.0.1 -d
0/0 -o eth0
```

Segue uma customização em scripts shell para ativar a política de ping pela interface eth0.

```
# vi ping2.sh

PF=$(which IPtables)
```



```

if [ -z $PF ] ; then
    echo "Comando IPtables nao encontrado"
    exit
fi
ETH0="10.0.0.1"
ping2()
{
    IPtables -A INPUT -j ACCEPT -p icmp --icmp-type 0 -d $ETH0 -s 0/0 -i
    eth0
    IPtables -A OUTPUT -j ACCEPT -p icmp --icmp-type 8 -s $ETH0 -d 0/0
    -o eth0

    $PF -A INPUT -i lo -d $LO -j ACCEPT
    $PF -A OUTPUT -o lo -d $LO -j ACCEPT
}
ping

# ./ping2.sh

```

Seguimos essa lógica para criar uma política de exceção para que seja possível traçar rotas usando o comando *mtr* do Linux, que é uma ferramenta de traceroute que utiliza icmp (encaminha pacotes echo-request). Porém, sabemos que ele utiliza uma técnica simplória, ainda que mais funcional, para enviar pacotes icmp echo-resquest com valores de ttl, os quais causarão a morte súbita do pacote cada vez que passarem por um roteador, lembrando que o atestado de óbito do pacote é o icmp de tempo excedido (time-exceeded) tipo 11.

Nosso objetivo seria liberar a execução do comando *mtr ip.ip.ip.ip* a partir da máquina 10.0.0.1. Como a saída de icmp echo-resquest já foi liberada anteriormente, simplesmente bastaria aceitar ICMP tipo 11

Exemplo:

```
iptables -A input -j ACCEPT -p icmp --icmp-type 11 -d 0/0 -i eth0
```

Exemplo de script para definir políticas para poder utilização do programa *mtr*:

```

PF=$(which IPtables)

if [ -z $PF ] ; then
    echo "Comando IPtables nao encontrado"
    exit

```

```

fi

NET="0/0"

ETH0="192.168.200.8"

mtr ()

{

$PF -A INPUT -p icmp --icmp-type 11 -s $NET -d $ETH0 -j ACCEPT

}

mtr

# ./mtr.sh

```

Com a regra de ICMP 11 ativa é possível traçar rotas com mtr tendo como alvo um número IP; todavia, se desejar fazer o mesmo com o traceroute do Linux, sabendo que ele utiliza pacotes UDP tendo como destino portas altas, demanda-se criar uma política de exceção. Nesse caso, por se tratar de um protocolo de transporte, temos de trabalhar com as portas origem (--sport) e com as portas destino (--dport). Diante desse cenário, fica claro que é necessário abrir uma exceção para a comunicação de pacotes UDP. Deve-se também levar em consideração que o traceroute do Linux trabalha de forma incremental para a porta destino, ao iniciar com o valor de porta alta a partir 33435, incrementando em razão direta aos números de hops (saltos) e usa a mesma porta de origem, a qual, quando é definida, acaba sendo uma porta livre acima de 1024. Diante disso, a regra deverá atender a todos esses detalhes.

Exemplo

```
# iptables -A INPUT -p 17 -s 10.0.0.1 --sport 1024:65535 -d 0/0
--dport 33435:33465 -j ACCEPT
```

 O valor 17 é referente ao protocolo UDP, vide /etc/protocols).

Em suma, sendo liberado o range de 33435 a 33465, o que dá um total de 30 saltos, o suficiente para dar várias voltas na internet.

Imagine que é desejado responder a solicitações do comando *ping* outros hosts. Pode-se criar uma política para o ping da seguinte forma: cria-se uma política de exceção para a saída de pacotes icmp echo-reply (tipo 0) e para de entrada de icmp echo-request (tipo 8). Considerando a máquina em questão com o IP 10.0.0.1.

Exemplos:

```
# iptables -A INPUT -j ACCEPT -p icmp --icmp-type 8 -d 10.0.0.1 -s 0/0
# iptables -A OUTPUT -j ACCEPT -p icmp --icmp-type 0 -s 10.0.0.1 -d 0/0
```

Assim, foram criadas duas exceções, uma para a saída de pacotes de ICMP, que é identificado como protocolo tipo 1 (vide a tabela completa /etc/protocols), cujo destino é a minha máquina, indicada por -s que lemos com source, ou seja, origem e destino são indicados com -d, e estão sendo aceitas.





Exemplo de script de automação denominado ping_response.sh para liberar a entrada de pacotes ICMP echo request e a saída de pacotes ICMP echo reply:

```
# vi ping_response.sh

PF=$(which IPtables)

if [ -z $PF ] ; then
    echo "Comando IPtables nao encontrado"
    exit
fi

NET="0/0"
ETH0="192.168.200.8"

ping_response ()
{
$PF -A INPUT -p icmp --icmp-type 0 -s $NET -d $ETH0 -j ACCEPT
$PF -A OUTPUT -p icmp --icmp-type 8 -s $ETH0 -d $NET -j ACCEPT
}

ping

# ./ping_response.sh
```

Mas nas políticas de exceção já exemplificadas para traçar as rotas existe uma limitação. Devido à política básica de DROP, não é possível fazer nenhuma resolução de nome, ou seja, até o momento o traceroute está liberado somente para IP, não sendo possível até então fazer um mtr ou traceroute para um www.dominio.com.br. Para que isso seja possível, deve-se liberar consultas DNS via UDP na porta de servidor, que por padrão é 53, para a máquina em questão, a fim de que esta possa fazer a resolução de nomes. Exemplos:

```
Iptables -A OUTPUT -p 17 -s 10.0.0.1 - -sport 1024:65535 -d 0/0 -
-dport 53 -j ACCEPT

Iptables -A INPUT -p 17 -d 10.0.0.1 - -dport 1024:65535 -s 0/0 -
-sport 53 -j ACCEP
```

Exemplo de script denominado dns.sh para que seja se comunicar com servidores DNS:

```
# vi dns.sh
```

```

PF=$(which IPtables)

if [ -z $PF ] ; then
    echo "Comando IPtables nao encontrado"
    exit
fi

NET="0/0"

ETH0="10.0.0.1"

PA="1024:65535"

dns ()
{
$PF -A INPUT -p 17 -s $NET --sport 53 -d $ETH0 --dport $PA -j ACCEPT
$PF -A INPUT -p 1 --icmp-type 3 -s $NET -d $ETH0 -j ACCEPT
$PF -A OUTPUT -p 17 -s $ETH0 --sport $PA -d $NET --dport 53 -j ACCEPT
}

dns

# ./dns.sh

```

Liberação, entrada e saída de serviços TCP e UDP, sem o controle de estado de conexão ou controle de pacotes relacionados respectivamente, é uma forma simplista de fazer tratamento do fluxo de pacotes de comunicação de redes, ou seja, mesmo sendo um o Iptables, um o firewall do tipo Statefull, são regras do tipo Packet Filter.

Segue um exemplo de regra de cliente assumindo o 10.0.0.1 para o serviço portmap cuja porta padrão é 111. Segue a mesma lógica de serviços TCP: o cliente se conecta em uma porta de serviço tendo a resposta em uma porta alta livre:

```

iptables -A INPUT -j ACCEPT -d 10.0.0.1 -p udp --sport 111 --dport 1024:65535

iptables -A OUTPUT -j ACCEPT -s 10.0.0.1 -p udp --sport 1024:65535 --dport 111

```

Considerando uma regra, sendo a máquina com a regra de firewall e um servidor:

```

iptables -A INPUT -j ACCEPT -d 10.0.0.1 -p udp --sport 111 --dport 1024:65535

iptables -A OUTPUT -j ACCEPT -s 10.0.0.1 -p udp --sport 1024:65535 --dport 111

```

Considerando a regra que possibilita comunicação com o serviço DNS já adicionada, pode-se executar a resolução de nomes. Continuando a lógica de firewall, já que é possível traçar rotas e resolver nomes, basta agora liberar a navegação web realizada através do protocolo http, o qual é transportado via TCP e, por padrão, utiliza a porta de servidor 80.

```
Iptables -A OUTPUT -j ACCEPT -p 6 -s 10.0.0.1 --sport 1024:65535  
-d 0/0 - -dport 80  
  
Iptables -A INPUT -j ACCEPT -p 6 -d 10.0.0.1 --dport 1024:65535  
-s 0/0 - -sport
```

1024:65535



Exemplo de script denominado http.sh para automatizar regras para acessar servidores web:

```
# vi http.sh  
  
PF=$(which iptables)  
  
if [ -z $PF ] ; then  
    echo "Comando iptables nao encontrado"  
    exit  
fi  
  
NET="0/0"  
  
ETH0="10.0.0.1"  
  
PA="1024:65535"  
  
http ()  
{  
    $PF -A OUTPUT -p 6 -s $ETH0 --sport $PA -d $NET --dport 80 -j ACCEPT  
    $PF -A INPUT -p 6 -s $NET --sport 80 -d $ETH0 --dport $PA -j ACCEPT  
}  
  
http  
  
# ./http.sh
```

Toda lógica de políticas que aplicamos aqui diz respeito a uma única máquina, um conceito de firewall home; mas as políticas podem ser traduzidas para uma situação de rede, o que faremos a seguir.



Com conceito de políticas de exceção, pode-se criar exceções para os protocolos mais utilizados. As exceções a seguir partirão do princípio de que a máquina é um cliente. Vejamos exemplos de políticas de exceção para os protocolos SMTP, POP3, IMAP, TELNET, SSH, FTP, HTTPS, HTTP, considerando o host 10.0.0.1 como host cliente.

```
Serviço - SMTP (porta 25) e porta 587  
iptables -A INPUT -j ACCEPT -d 10.0.0.1 -p tcp --sport 25 --dport 1024:65535  
iptables -A OUTPUT -j ACCEPT -s 10.0.0.1 -p tcp --sport 1024:65535 --dport 25  
iptables -A INPUT -j ACCEPT -d 10.0.0.1 -p tcp --sport 587 --dport 1024:65535  
iptables -A OUTPUT -j ACCEPT -s 10.0.0.1 -p tcp --sport 1024:65535 --dport 587
```

Mudando de ponto de vista, imagine-se desenvolvendo um firewall para uma máquina que vai disponibilizar vários serviços – assuma-se para fins didáticos, pois uma máquina “multiuso” não é recomendável; quando projetam-se servidores, é necessário ter em mente o conceito de Bastion Host, pois consiste em problemas de segurança em uma máquina e reúne vários serviços juntos, pelo de fato de um serviço comprometido poder significar todos os demais também comprometidos. Todavia, para ilustrar a liberação de serviços via firewall para clientes, será assumido que o host em questão deseja liberar para qualquer cliente os respectivos serviços:

```
Servidor web (porta 80)  
iptables -A INPUT -j ACCEPT -d 10.0.0.1 -p tcp --sport 1024:65535 --dport 80  
iptables -A OUTPUT -j ACCEPT -s 10.0.0.1 -p tcp --sport 80 --dport 1024:65535
```

As políticas anteriores devem ser utilizadas como base para um firewall home para cada servidor, negando todos os acessos possíveis, exceto o do serviço que o servidor oferece.

Nesse exemplo, não colocamos portas essenciais como 23 (telnet) e 22 (ssh), por serem serviços de login remoto, e propomos em nossas políticas de firewall sermos mais restritivos, liberando acesso a serviços desse nível somente a IPs conhecidos.

Uma política interessante é liberar serviços específicos somente para IPs específicos. O que queremos é liberar serviços com SSH, WEBMIN, NTP, SNMP, NESSUS somente para as máquinas “amigas”, impossibilitando ao máximo a exploração desse tipo de serviço.

Para compreender melhor, considere a máquina 192.168.10.1 como a máquina do administrador, e 10.0.0.1 como uma máquina remota confiável. Veja os exemplos.

Liberando restritivamente o SSH:

```
iptable -A INPUT -s 10.0.0.1 --sport 1024:65535 -d 192.168.10.1  
--dport 22 -j ACCEPT  
iptable -A OUTPUT -d 10.0.0.1 --dport 1024:65535 -s 192.168.10.1  
--sport 22 -j ACCEPT
```



Com as políticas citadas, já é possível pensar em um firewall de fronteira entre duas ou mais redes. Para melhor contextualizar a situação, assuma um cenário onde é necessário liberar o repasse de pacote entre duas redes locais 192.168.10.0/24 e 10.0.0.0/8, supondo que esse firewall seja o gateway que atua na fronteira das redes supracitadas. Contudo, para facilitar a parametrização da regras, será usado o recurso de “multiport”, o qual nos possibilita declarar até 15 portas numa única regra; dessa vez, não será necessário escrever uma regra para cada protocolo em questão.

```
iptables -A FORWARD -p 6 -s 10.0.0.0/8 -m multiport --sport
20,21,22,23,25,80,110,143,443 -d 192.168.0.0/24 -m multiport --dport
1024:65535 -j ACCEPT
```

É possível também, em vez de declarar literalmente o protocolo, usar o número correspondente, como, por exemplo: 1 para ICMP, 6 para TCP ou 17 para UDP. Caso deseja-se saber os números dos demais protocolos, basta consultar o arquivo */etc/protocols*.

Ou poderíamos fazer da forma clássica:

```
iptables -A FORWARD -d 192.168.10.0/24 -p tcp --sport 80 --dport
1024:65535 -j ACCEPT

iptables -A FORWARD -d 192.168.10.0/24 -p tcp --sport 110 --dport
1024:65535 -j ACCEPT

iptables -A FORWARD -d 192.168.10.0/24 -p tcp --sport 143 --dport
1024:65535 -j ACCEPT
```

Trabalhando com ICMP (RFC 792)

Customização de políticas de exceção para tratamento do protocolo ICMP, que é um protocolo muito útil na gerência de rede, mas devido ao grande número de tipo de respostas, motiva administradores que têm poucos conhecimentos de TCP/IP a criarem políticas não adequadas. Outro fato é que esse protocolo é muito utilizado por ataques para a elaboração de Ataques de Deny of Service (DOS) e técnicas de Fingerprint. A tabela a seguir servirá de referência. Mais detalhes, vide RFC792:

TYPE	CODE	Description	Query	Error
0	0	Echo Reply	x	
3	0	Network Unreachable		x
3	1	Host Unreachable		x
3	2	Protocol Unreachable		x
3	3	Port Unreachable		x
3	4	Fragmentation needed but no frag. bit set		x
3	5	Source routing failed		x
3	6	Destination network unknown		x
3	7	Destination host unknown		x
3	8	Source host isolated (obsolete)		x

Tabela 7.5
Tipos e códigos de retorno inclusos no cabeçalho do protocolo ICMP.



TYPE	CODE	Description	Query	Error
3	9	Destination network administratively prohibited		x
3	10	Destination host administratively prohibited		x
3	11	Network unreachable for TOS		x
3	12	Host unreachable for TOS		x
3	13	Communication administratively prohibited by filtering		x
3	14	Host precedence violation		x
3	15	Precedence cutoff in effect		x
4	0	Source quench		
5	0	Redirect for network		
5	1	Redirect for host		
5	2	Redirect for TOS and network		
5	3	Redirect for TOS and host		
8	0	Echo request	x	
9	0	Router advertisement		
10	0	Route solicitation		
11	0	TTL equals 0 during transit		x
11	1	TTL equals 0 during reassembly		x
12	0	IP header bad (catchall error)		x
12	1	Required options missing		x
13	0	Timestamp request (obsolete)	x	
14	0	Timestamp reply (obsolete)	x	
15	0	Information request (obsolete)	x	
16	0	Information reply (obsolete)	x	
17	0	Address mask request	x	
18	0	Address mask reply	x	

O protocolo ICMP, por ser um protocolo de mensagem, como o próprio nome já diz, não é utilizado normalmente para a troca de dados, e sim para teste e notificações, sendo a maioria entre roteadores. Não pode-se negar que por ser aparentemente inocente, ele pode ser utilizado em ataques engenhosos. Um bom exemplo de um ataque clássico via a famosa ferramenta Tríbo Flood Network (TFN) é uma ferramenta DDoS que utiliza pacotes icmp echo-reply para enviar a ordem de ataque da máquina master para as máquinas zumbis, com informações como o tipo de ataque e o endereço da vítima para as máquinas-zumbis, mostrando que até mesmo um datagrama supostamente inocente pode trazer perigo para a rede.



O fato interessante é que muitos ataques DOS utilizam datagramas ICMP, como o Ping da Morte, Jolt e Smurf.

Entretanto, simplesmente negar tráfego de datagramas ICMP de forma arbitrária não é inteligente; mas de forma restritiva, especificando exatamente os tipos de mensagens ICMP que você julgar conveniente, é algo que se pode fazer.

No início do capítulo, foram apresentados exemplos interessantes do uso de mensagens ICMP. Um dos exemplos é o uso do comando *ping*, que usa duas mensagens: requisição ICMP ECHO-REQUEST (tipo 8) e resposta ICMP ECHO-REPLY (tipo 0). Outro exemplo é o atestado de óbito de pacotes quando executamos qualquer programa que seja capaz de trancar a rota, pois quando o valor de TTL zera, o pacote é destruído e a notificação da morte vem através de um ICMP TIME-EXCEEDED (tipo 11), que pode ser dois tipos de código de mensagem (TTL-ZERO-DURING-TRANSIT e TTL-ZERO-DURING-REASSEMBLY).

Embora com avanço das tecnologias de redes de computadores no que tange a infraestrutura de internet, é cada vez mais incomum a necessidade de fragmentação de datagramas motivados por valores de MTU inferiores a 1500 bytes. Mas mesmo assim, quando existe uma fragmentação de IP o protocolo ICMP volta à cena, pois quando a fragmentação é demandada, o roteador utiliza um datagrama ICMP DESTINATION-UNREACHABLE (tipo 3), mais especificamente de código FRAGMENTATION (tipo 3, código 4).

O código utilizado também em ICMP do tipo é o código de PORT-UNREACHABLE (tipo 3, código 3), usado para notificar que uma porta não possui serviço UDP, uma vez que a esta recebe um datagrama UDP.

Para finalizar, não é recomendável bloquear mensagens do tipo HOST-UNREACHABLE (tipo 3, código 1) e SOURCE QUENCH (tipo 4, código 0); caso aconteça, poderemos ter problemas no controle das conexões. Respectivamente, a primeira mensagem notifica quando o destino é inalcançável e a segunda, que o host está sobrecarregado.

Entretanto, para a concepção de políticas de exceção para datagramas ICMP, o IPtables é capaz de tratar cada tipo de mensagem e seus respectivos códigos; para isso, utilize o comando a seguir para ver a lista de mensagens e códigos, e consulte as RFC do ICMP para mais detalhes.

```
# iptables -p icmp -h
```

Alguns exemplos genéricos:

```
# iptables -A INPUT -j ACCEPT -s 0/0 -d 10.0.0.1 -p icmp --icmp-type time-exceeded  
# iptables -A INPUT -j ACCEPT -s 0/0 -d 10.0.0.1 -p icmp --icmp-type echo-replay  
# iptables -A INPUT -j ACCEPT -s 0/0 -d 10.0.0.1 -p icmp --icmp-type destination-unreachable  
# iptables -A OUTPUT -j ACCEPT -s 0/0 -d 10.0.0.1 -p icmp --icmp-type echo-request
```

Para exemplificar de forma bem específica, será ilustrado o tratamento das mensagens ICMP e o código de mensagem inerente para políticas de repasse (FORWARD).

Tratando ICMP 3 (Unreachable), liberando o repasse, mas com limitação de um datagrama por segundo:

Liberando código 1 (HOST UNREACHABLE)

```
# iptables -A FORWARD -j ACCEPT -p 1 --icmp-type 3/1 -m limit --limit 1/s
```

Liberando o código 3 (PORT UNREACHABLE), resposta de porta UDP fechada:

```
iptables -A FORWARD -j ACCEPT -p 1 --icmp-type 3/3 -m limit --limit 1/s
```

Liberando o código 4 (FRAGMENTATION NEEDED), datagrama utilizado pelo roteador para notificar que haverá fragmentação a partir dele, ou seja, que o valor de MTU no próximo enlace é menor:

```
# iptables -A FORWARD -j ACCEPT -p 1 --icmp-type 3/4 -m limit --limit 1/s
```

Tratando ICMP 11 (TIME EXCEEDED), que é responsável pela notificação da morte de um pacote. No caso de programas que traçam rotas, o código padrão de resposta é 0. Vide o tratamento a seguir:

Liberando o código 0 (TTL EQUALS 0 DURING TRANSIT), notificação da morte do pacote:

```
# iptables -A FORWARD -j ACCEPT -p 1 --icmp-type 11/0 -m limit --limit 1/s
```

Liberando o código 1 (TTL EQUALS 0 DURING REASSEMBLY), notifica a morte do pacote durante a sua reconstrução em um roteador. Não muito comum, mas conveniente liberar.

```
# iptables -A FORWARD -j ACCEPT -p 1 --icmp-type 11/1 -m limit --limit 1/s
```

Tratando ICMP 4 (SOURCE QUENCH):

```
# iptables -A FORWARD -j ACCEPT -p 1 --icmp-type 4/0 -m limit --limit 1/s
```

Tratando o comando *PING*, liberando respectivamente o ECHO-REPLY tipo 0 código 0, e o ECHO REQUEST tipo 8 código 0:

```
# iptables -A FORWARD -j ACCEPT -p 1 --icmp-type 0/0 -m limit --limit 1/s
```

```
# iptables -A FORWARD -j ACCEPT -p 1 --icmp-type 8/0 -m limit --limit 1/s -j ACCEPT
```

Exemplificação de um script para automatizar e gerar as regras para tratamento aos principais tipos de ICMP:

```
# vi /root/rascunho/exec.icmp
```

```
PF=$(which iptables)
```

```
NET=0/0
```

```
ETH0=192.168.200.8
```



```
if [ -z $PF ]; then
    echo "Comando iptables nao encontrado"
    exit
fi
for tipo in 0 3/0 3/1 3/2 3/3 3/4 4 5 11 12
do
$PF -A INPUT -p icmp -s $NET -d $ETH0 --icmp-type $tipo -j ACCEPT -m
limit --limit 1/s
done
```





Roteiro de Atividades 7

Prática de parametrização de políticas para tráfego de rede utilizando o recurso Netfilter do Kernel do Linux através da ferramenta IPtables. Será necessário que o aluno utilize a máquina virtual preparada como Virtuabox. Dessa forma, será possível aplicar todos os conhecimentos (técnicas e ferramentas) de Hardening propostos até o momento.

Atividade 7.1 – Hardening Linux – Firewall Linux

1. Construir um script de firewall para tratar:

Políticas de acesso aos protocolos SSH, HTTP, SNMP, considerado a máquina firewall com servidor (alvo).

Políticas de acesso aos protocolos SSH, DNS, HTTP, NTP considerando a máquina firewall origem/cliente.

Teste as regras e valide-as.

2. Desenvolva um shell script para tratar os principais tipos de ICMP que um servidor deve receber. Considere que seja um gateway de fronteira.

Técnicas de Traceroute são utilizadas para traçar a rota da origem até o destino, ou seja, verificar quantos roteadores há entre a origem e o destino. É sabido que o mtr envia pacotes icmp echo request e espera como resposta o icmp 11 (tempo excedido). Dessa forma, teremos a exceção para ICMP (para entrada/ input)

TTL equals 0 during transit- TTL equals 0 during reassembly

Trataremos como exceção para entrada e repasse também ICMP do TIP em nosso Firewall, entre eles:

- **Port Unreachable:** resposta padrão de porta UDP fechada;
- **Host Unreachable:** resposta padrão para um host fora de alcance;
- **Network Unreachable:** que é a resposta padrão para rede fora de alcance;
- **Port Unreachable:** que é a resposta padrão de porta UDP fechada;
- **Fragmentation needed:** também conhecido como ICMP PATH Discovery.

Para possibilitar o uso do “ping” de forma egoísta, ou seja, somente o servidor firewall pode realizar consulta de Ping, mas não responder.

- Tratar a saída de Echo-request.
- Tratar a entrada de Echo-reply.
- Tratar também a entrada de mensagens do tipo ICMP Source-quench.

Embora seja uma situação rara, elabore uma política de exceção de entrada e repasse para mensagens IP header bad.

Para evitar possibilidades de ataques de Flood a partir dos tipos de respostas para as quais foram geradas políticas de exceção, utilize o conceito de limitar datagramas.

Elabore um terceiro script reunindo as funcionalidade dos dois anteriores, copie o script para o diretório `/etc/init.d` e faça-o funcionar automaticamente no runlevel 2.





8

Segurança de Perímetro – Parte 2

objetivos

Aprender conceitos de políticas de controles (registros de eventos/log); Conhecer regras baseadas em pacotes de conexões relacionadas e regras baseadas em Estado de Conexão (Statefull); Entender tradução de Endereçamento IP (NAT); Aprender manipulação de campos de Datagrama; Implementar Port Knocking.

conceitos

Regras de controle; Regras Statefull, NAT e Port Knocking para implementação de Firewall.

Exercício de nivelamento 1

Firewalls

Existem vantagens em registrar eventos arbitrários via Firewall?

Uma rede baseada em um gateway que realizar NAT para todas as estações de trabalho consegue prover segurança suficiente para impossibilitar ataques com origem na internet? Por quê?

Na sua opinião, a configuração de um Port Knocking é uma alternativa para acesso a recursos administrativos?

Neste capítulo, serão apresentados conceitos e regras de controle para registro de possíveis eventos arbitrários, políticas com o conceito de State Full, implementação de Port Knocking, tradução de endereçamento IP (NAT) e manipulação de datagrama via tabela Mangle.



Regras de controle



Firewall: faz o registro das atividades que não são permitidas.

- Assim, o administrador, por meio dos logs, pode analisar comportamentos hostis.

Uma tarefa importante do firewall é justamente fazer o registro das atividades que não são permitidas, para que através dos logs o administrador possa analisar comportamentos hostis e tentar antecipar-se quanto às atividades de invasores como Script Kiddies, insider, Black Hats e GrayHat. Os exercícios a seguir têm por objetivo exemplificar a construção de políticas de controle.

Uma estratégia que pode ser adotada é criar regras para registrar eventos que possam caracterizar as atividades que estejam relacionadas com ações escusas motivadas pelos agentes como invasores e/ou malware.

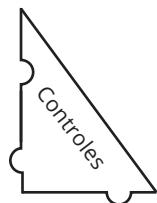


Figura 8.1
Criação de regras para registro de eventos: foco no controle.

A criação de uma regra de controle que avalie pacotes “perdidos”, ou seja, pacotes que não pertencem às conexões estabelecidas ou mesmo ataques de pacotes mal formados é recomendável, pois a incidência desse tipo de pacote pode caracterizar um ataque de negação de serviços (DOS) em execução:

```
# vi /root/rascunho/malformados

PF=$(which iptables)
malformados()
{
    for CHAINS in INPUT FORWARD
    do
        $PF -A $CHAINS -m state --state INVALID -j LOG --log-prefix
        "pkt_bad"
        $PF -A $CHAINS -m state --state INVALID -j DROP
    done
}
malformados
```

Comandos para teste do script o script e listagem das regras ativas:

```
# bash malformados.fw
# iptables -L -n
```

Outro comportamento que pode ser monitorado é o uso de combinações de flags TCP não convencionais, que podem ser utilizados em ataques de varreduras e negação de serviço.

Script para automatizar definição de políticas de controles de ataque de pacotes com flags furadas:

```
# vi /root/rascunho/flags.tcp
PF=$(which iptables)

flags_furado()
{
for FLAGS in $(cat /etc/fw/flags.tcp | grep -v ^#)
do
    for CHAINS in INPUT FORWARD
    do
        $PF -A $CHAINS -p 6 --tcp-flags $FLAGS $FLAGS -j LOG
--log-prefix "flag_$FLAGS"
        $PF -A $CHAINS -p 6 --tcp-flags $FLAGS $FLAGS -j DROP
    done
done
}

flags_furado
# vim /etc/fw/flags.tcp
NONE
ALL
FIN,RST
SYN,RST
FIN,URG,PSH
```

Comandos para teste do script o script e listagem das regras ativas:

```
# bash flags.tcp
# iptables -L -n
```

Vigiar portas clássicas utilizadas por malware ou mesmo ferramentas indesejadas. Pois mesmo que o firewall tenha o conceito restritivo em sua política básica, ainda sim negando o acesso às portas que não foram expressamente liberadas, caso ocorram ações relacionadas a essas portas, elas devem ser registradas para auditoria.



Exemplificação de um script para definir políticas de controles para acesso a portas indevidas:

```
# vi /root/rascunho/indevidas_tcp_udp  
# vim /etc/fw/indevidas_tcp_udp  
# netbus  
12345  
12346  
# IRC  
6660:6669  
# BO  
31337  
31336  
666
```

Comandos para teste do script o script e listagem das regras ativas:

```
# cd /root/rascunho/  
# bash indevidas_tcp_udp  
# iptables -L -n
```

Seguindo a mesma lógica de portas indevidas, vale um tratamento especial para porta de ferramentas P2P, onde ao invés de simplesmente bloquear, pode ser interessante liberar, mas limitar o tráfego de tal forma que seja ruim o uso desse tipo de ferramenta, pois bloquear diretamente pode motivar usuários mais arrojados a tentar meios de contornar o firewall. Liberar com limitação pode desmotivá-los a usar a ferramenta, pois o download será muito lento.

Exemplo de script para definir políticas de controles para aplicações P2P:

```
# vi /root/rascunho/p2p  
PF=$(which iptables)  
  
indevidas_tcp_udp()  
{  
    for PORTAS in $(cat /etc/fw/indevidas_tcp_udp | grep -v ^#)  
    do  
        for CHAINS in INPUT FORWARD  
        do  
            for PROTO in tcp udp  
            do  
                $PF -A $CHAINS -p $PROTO --dport $PORTAS -j LOG --log-prefix  
                "indevidas_$PORTAS"
```



```

done

done

done

}

indevidas_tcp_udp

# vim /etc/fw/p2p

# GNUTella

6346

6246

#Kazaa

1214

5000

8888

6257

Comandos para teste do script o script e listagem das regras
ativas:

# cd /root/rascunho/
# bash p2p
# iptables -L -n

```

Deve ser de conhecimento de todos os administradores de rede os conceitos de classe de rede. Dessa forma, bloquear as ranges de IP não roteáveis na internet, que são definidas pelo IANA, pode ser uma ação interessante para mitigar ataques baseado em falsificação de IP (Ip Spoofing).

Segue um exemplo de script para definir políticas de controles para pacotes com endereçoamento reservado do IANA, que possivelmente são pacotes forjados:

```

PF=$(which iptables)

iana()
{
for IP in $(cat /etc/fw/iana | grep -v ^#)
do
for CHAINS in INPUT
do

```

```

$PF -A $CHAINS -i eth0 -p 0 -s $IP -j LOG --log-prefix
“IANA_${IP}”

$PF -A $CHAINS -i eth0 -p 0 -s $IP -j DROP

done

}

iana

Arquivo /etc/fw/iana.

```

0.0.0.0/8	90.0.0.0/8	218.0.0.0/8
1.0.0.0/8	91.0.0.0/8	219.0.0.0/8
2.0.0.0/8	92.0.0.0/8	220.0.0.0/8
5.0.0.0/8	93.0.0.0/8	221.0.0.0/8
7.0.0.0/8	94.0.0.0/8	222.0.0.0/8
23.0.0.0/8	95.0.0.0/8	223.0.0.0/8
27.0.0.0/8	96.0.0.0/8	224.0.0.0/4
31.0.0.0/8	97.0.0.0/8	240.0.0.0/8
36.0.0.0/8	98.0.0.0/8	240.0.0.0/5
37.0.0.0/8	99.0.0.0/8	241.0.0.0/8
39.0.0.0/8	100.0.0.0/8	242.0.0.0/8
41.0.0.0/8	101.0.0.0/8	243.0.0.0/8
42.0.0.0/8	102.0.0.0/8	244.0.0.0/8
58.0.0.0/8	103.0.0.0/8	245.0.0.0/8
59.0.0.0/8	104.0.0.0/8	246.0.0.0/8
60.0.0.0/8	105.0.0.0/8	247.0.0.0/8
67.0.0.0/8	106.0.0.0/8	248.0.0.0/8
68.0.0.0/8	107.0.0.0/8	249.0.0.0/8
69.0.0.0/8	108.0.0.0/8	250.0.0.0/8
70.0.0.0/8	109.0.0.0/8	251.0.0.0/8
71.0.0.0/8	110.0.0.0/8	252.0.0.0/8
72.0.0.0/8	111.0.0.0/8	253.0.0.0/8
73.0.0.0/8	112.0.0.0/8	254.0.0.0/8
74.0.0.0/8	113.0.0.0/8	255.0.0.0/8
75.0.0.0/8	114.0.0.0/8	
76.0.0.0/8	115.0.0.0/8	
77.0.0.0/8	116.0.0.0/8	
78.0.0.0/8	117.0.0.0/8	
79.0.0.0/8	118.0.0.0/8	
80.0.0.0/8	119.0.0.0/8	
81.0.0.0/8	120.0.0.0/8	
82.0.0.0/8	121.0.0.0/8	
83.0.0.0/8	122.0.0.0/8	
84.0.0.0/8	123.0.0.0/8	
85.0.0.0/8	124.0.0.0/8	
86.0.0.0/8	125.0.0.0/8	
87.0.0.0/8	126.0.0.0/8	
88.0.0.0/8	197.0.0.0/8	
89.0.0.0/8	201.0.0.0/8	

Tabela 8.1
O arquivo
'/etc/fw/iana'.



Comandos para teste do script o script e listagem das regras ativas:

```
# cd /root/rascunho/  
# bash iana.ip  
# iptables -L -n
```

É natural que o seu firewall de fronteira seja vítima de varreduras. Uma bastante comum é a varredura furtiva, baseadas nas flags urg, psh e rst do segmento TCP, ou em pacotes TCP que não têm nenhuma flag habilitada. Isso posto, implemente uma regra de controle para “registrar” esse tipo de pacote e posteriormente “DROP” o pacote. Faça um shell script com função para essa regra.

```
PF=$(which iptables)  
  
liberatudo()  
{  
  
    for CHAINS in INPUT OUTPUT FORWARD  
    do  
        $PF -P $CHAINS ACCEPT  
  
    done  
  
    $PF -F  
}  
  
liberatudo
```

Comandos para teste do script o script e listagem das regras ativas:

```
# cd /root/rascunho  
# bash liberatudo  
# iptables -nL
```

Criando Chains

1 - Criar um script com CHAIN para aceitar somente a entrada de pacotes ICMP tipo 0, 11, 3 e 4:

a) Gerando uma função usando recurso de chains para tratar conexões ICMP:

```
# vi icmp_type.sh  
  
NET="0/0"  
  
ET0="192.168.0.24"  
  
PF=$(which iptables)  
  
  
icmptype( )  
{  
  
    $PF -N ALLOW_ICMP
```

```

T="0 3 11 4"

for TYPE in $T
do
    $PF -A ALLOW_ICMP -p 1 --icmp-type $TYPE -s $NET -d $ETO -j
ACCEPT
done

$PF -A INPUT -p 1 -j ALLOW_ICMP
}

```

Comandos para teste do script o script e listagem das regras ativas:

```

# ./ icmp_type.sh
# iptables -n -L

```

Gerando uma função usando recurso de chains para tratar conexões ICMP sendo explícito quanto aos tipos que devemos tratar, que foram estudados anteriormente.

```

$PF -N ALLOW_ICMP
NET="0/0"
ETO="192.168.200.9"
PA="1024:65535"
PF=$(which iptables)
$PF -A ALLOW_ICMP -j ACCEPT -m limit --limit 2/s
for TIPO in 0 3/0 3/1 3/2 3/3 3/4 4 5 11 12
do
    $PF -A INPUT -j ALLOW_ICMP -p 1 -s $NET -d $ETO --icmp-type $TIPO
done
$PF -A OUTPUT -j ALLOW_ICMP -p 1 -d $NET -s $ETO --icmp-type 8

```

O recurso para a criação de uma chains específica que possibilite tornar mais arrojado o tratamento dos pacotes. Uma boa situação para ser parametrizada em uma chain particular são as conexões TCP, onde pode-se combinar o recurso de tratamento statefull com uma chain específica. Assim, todas as conexões TCP expressamente permitidas podem ter o seu fluxo de pacote transferido para essa chains, otimizando o firewall.

Exemplo de um script customizado para esse fim, considerando o contexto onde as regras atuarão nas mesmas máquinas e proverão o serviço tendo como cliente exclusivamente a rede 192.168.100.0/24. Aproveitando ainda o conceito de “estado de conexão”, dessa forma as regras serão restritivas para as portas-alvo, no que tange o conceito de handshake tcp, mas

a resposta (OUTPUT) será tratada por uma regra simples que considerará somente pacotes relacionados e de uma conexão já estabelecida. Isso economiza regras e processamento:

```
# vi state_tcp1.sh

$PF=$(which iptables)

statetcp1()
{
    $PF -N ALLOW_TCP

    $PF -A ALLOW_TCP -p 6 --syn -j ACCEPT

    $PF -A ALLOW_TCP -p 6 -m state --state ESTABLISHED,RELATED -j ACCEPT

    $PF -A ALLOW_TCP -p 6 ! --syn -m state --state NEW -j LOG \
        --log-prefix "New not syn:"

    $PF -A ALLOW_TCP -p tcp ! --syn -m state --state NEW -j DROP
}

for PORTA in $(cat /etc/fw/portasservico.tcp | grep -v ^#)
do

    $PF -A INPUT -j ALLOW_TCP -s 192.168.100.0/24 --dport $PORTA

    Done

    $PF -A OUTPUT -j ACCEPT TCP -p 6 -m state --state
ESTABLISHED,RELATED

    $PF -A INPUT -j ACCEPT TCP -p 6 -m state --state
ESTABLISHED,RELATED

# ./state_tcp1.sh
```

Segundo exemplo, similar ao anterior, mas onde a customização tem o ponto de vista de cliente de uma lista definida de serviços:

```
vi state_tcp2.sh

$PF=$(which iptables)

statetcp2()
{
    $PF -N ALLOW_TCP
```



```

$PF -A ALLOW_TCP -p 6 --syn -j ACCEPT
$PF -A ALLOW_TCP -p 6 -m state --state ESTABLISHED,RELATED -j ACCEPT
$PF -A ALLOW_TCP -p 6 ! --syn -m state --state NEW -j LOG \
--log-prefix "New not syn:"
$PF -A ALLOW_TCP -p tcp ! --syn -m state --state NEW -j DROP

for PORTA in $(cat /etc/fw/portascliente.tcp | grep -v ^#)
do

$PF -A OUTPUT -j ALLOW_TCP -d 192.168.100.0/24 --dport $PORTA

Done

$PF -A INPUT -j ACCEPT -p 6 -m state --state
ESTABLISHED,RELATED

}

#/state_tcp2.sh

```

Construindo um script de firewall

Configuração de script para oficializar regras do firewall:

- ▣ Depois de testado, deve ser inserido no /etc/init.d
- ▣ E definido para ser inicializado com um script de runlevel.



A configuração de um script para oficializar a regras do firewall é a melhor solução. Esse script, uma vez testado, deve ser inserido no /etc/init.d e definido para ser inicializado com um script de runlevel.

Colocar o script para ser carregado durante o boot no nível padrão e inicialização do seu Linux, lembrando que as distribuições like Debian e RedHat seguem o modelo System V, e Like Slackware segue o modelo BSD. No caso de distribuições Like Debian, como Ubuntu, o recomendado é o runlevel 2 e, para distribuições Like Redhat, como Fedora e Suse, o recomendável é o runlevel 3.

Segue um exemplo de um firewall Home, destinado à defesa exclusiva da própria máquina nesse exemplo, considerando que é um servidor Web e FTP:

```

#!/bin/bash
FW=/sbin/iptables

```

```

##-----
clear
echo INICIANDO FIREWALL
echo Fecha tudo
$FW -F

for TABELA in INPUT OUTPUT FORWARD
do
$FW -P $TABELA DROP
done

##-----
echo Política LOOPBACK
$FW -A INPUT -j ACCEPT -i lo
$FW -A OUTPUT -j ACCEPT -o lo

##-----
echo Política PING EGOISTA
$FW -A INPUT -j ACCEPT -d 10.197.25.228 -p icmp -m icmp --icmp-type 0
$FW -A OUTPUT -j ACCEPT -s 10.197.25.228 -p icmp -m icmp --icmp-type 8

##-----
#ATESTADO DE MORTE DO PACOTE
echo Política Traceroute
$FW -A INPUT -j ACCEPT -d 10.197.25.228 -p icmp -m icmp --icmp-type 11

##-----
echo Política DHCP
$FW -A OUTPUT -j ACCEPT -p udp -m udp --dport 68
$FW -A INPUT -j ACCEPT -p udp -m udp --sport 68

##-----

```



```

echo Política CLIENTE UDP

for PORTA in 53 514
do

$FW -A OUTPUT -j ACCEPT -s 10.197.25.228 -p udp -m udp --sport
1024:65535 --dport $PORTA

$FW -A INPUT -j ACCEPT -d 10.197.25.228 -p udp -m udp --sport
$PORTA --dport 1024:65535

done

##-----


echo Política de SERVIDOR TCP

for PORTA in 21 80 443
do

$FW -I INPUT -j ACCEPT -p tcp -d 10.197.25.228 -m state --state
NEW,ESTABLISHED --sport $PORTA --dport 1024:65535

done

$FW -I OUTPUT -j ACCEPT -p tcp -s 10.197.25.228 -m state --state
ESTABLISHED --sport 1024:65535

```

Outro exemplo do esqueleto de um script de Runlevel notoriamente baseado em System V com o cabeçalho padrão reconhecido por ferramentas como ntsysv disponível em distribuição Like Redhat:

```

#!/bin/sh

# chkconfig: 345 10 10
# description: Controle de Filtros Pacotes e Estado de Conexao
SF=/sbin/iptables
MEUIP="192.168.100.1"
CLIENTE="1024:65535"
#=====
#AQUI ENTRAM AS FUNCOES
#=====

case "$1" in
    start|-s)
        #Aqui sao ativadas as funcoes

```

```

;;
stop|-p)
# funcao que limpa as regras do firewall da memoria
;;

restart|-r)
$0 stop
$0 start
;;

status|-c)
$PF -L -n
;;
*)
echo "Use: $0 { |start|stop|restart|status| }"
exit 1
;;
esac
exit

```

Segue um terceiro exemplo, agora um script completo, para uma possível solução de Script de Firewall de Fronteira:

```

#!/bin/sh
#
# Packetfilter Controle de regras básica de Filtro (police)
# chkconfig: 345 10 10
# description: Controle de regras básicas de Filtros (police) -
# Designer by 4NIX - www.4NIX.com.br
#####
##### Coloque aqui as variáveis

PF=/sbin/iptables
NET="0/0"
ET0="192.168.200.9"
PA="1024:65535"

```



```

#####
##### Coloque aqui as funções

liberadns()
{
$PF -A INPUT -p 17 -s $NET --sport 53 -d $ETO --dport $PA -j ACCEPT
$PF -A INPUT -p 1 --icmp-type 3/3 -s $NET -d $ETO -j ACCEPT
$PF -A OUTPUT -p 17 -s $ETO --sport $PA -d $NET --dport 53 -j ACCEPT
}

trojan()
{
for PORT in $(cat /root/firewall/trojanporta.txt)
do
    for TABELA in INPUT FORWARD
    do
        for PROTO in udp tcp
        do
            $PF -A $TABELA -j LOG --log-prefix "TROJAN $PORT" -p
$PROTO --dport $PORT
            $PF -A $TABELA -j DROP -p $PROTO --dport $PORT
        done
    done
done
}

negatudo()
{
$PF -P INPUT DROP
$PF -P OUTPUT DROP
$PF -P FORWARD DROP
}

```

```

liberatudo()
{
$PF -P INPUT ACCEPT
$PF -P OUTPUT ACCEPT
$PF -P FORWARD ACCEPT
}

liberamtr()
{
$PF -A INPUT -p icmp --icmp-type 11 -s $NET -d $ETO -j ACCEPT
}

liberaping()
{
$PF -A INPUT -p icmp --icmp-type 0 -s $NET -d $ETO -j ACCEPT
$PF -A OUTPUT -p icmp --icmp-type 8 -s $ETO -d $NET -j ACCEPT
}

reject_tcp()
{
$PF -A INPUT -j REJECT --reject-with tcp-reset -p 6 -s $NET --dport
22
$PF -A OUTPUT -j ACCEPT -p 6 --sport 22 -d $NET
}

reject_udp()
{
$PF -A INPUT -j REJECT -p 17 -s $NET --dport 514
$PF -A OUTPUT -j ACCEPT -p 1 --icmp-type 3/3 -m limit --limit 1/s -d
$NET
}

```



```

state_icmp()
{
$PF -N ALLOW_ICMP
$PF -A ALLOW_ICMP -j ACCEPT -m limit --limit 2/s
for TIPO in 0 3/0 3/1 3/2 3/3 3/4 4 5 11 12
do
$PF -A INPUT -j ALLOW_ICMP -p 1 -s $NET -d $ETO --icmp-type $TIPO
done
$PF -A OUTPUT -j ALLOW_ICMP -p 1 -d $NET -s $ETO --icmp-type 8
}

state_tcp()
{
$PF -N ALLOW_TCP
$PF -A ALLOW_TCP -p 6 --syn -j ACCEPT
$PF -A ALLOW_TCP -p 6 -m state --state NEW,ESTABLISHED,RELATED -j
ACCEPT
$PF -A ALLOW_TCP -p 6 ! --syn -m state --state NEW -j LOG --log-
prefix "New not syn:"
$PF -A ALLOW_TCP -p 6 ! --syn -m state --state NEW -j DROP
$PF -A INPUT -j ALLOW_TCP
$PF -A OUTPUT -j ALLOW_TCP
}

limpatudo()
{
$PF -F
$PF -Z
$PF -X
}

status()

```

```

{

$PF -nL -t filter

}

#### Inicio do Case

case "$1" in

start)
    setup_kernel
    limpatudo
    negatudo
    liberadns
    liberamtr
    liberaping
    state_tcp
    state_icmp
    reject_tcp
    reject_udp

;;
stop)
    limpatudo
    liberatudo
;;
fechado)
    limpatudo
    negatudo
;;
liberado)
    limpatudo
    liberatudo
;;
restart)

```



```

$0 stop
$0 start
;;
status)
status
;;
*)
echo "Use: $0 {start|stop|restart|status|liberado|fechado}"
exit 1
;;
esac

exit 0

```

Trabalhando com tradução de endereçamento IP

Um recurso muito útil para redes baseadas no protocolo IPv4, que também está disponível no Iptables, é o recurso de NAT.

O recurso de NAT é amplamente utilizado e talvez tenha sido um dos recursos desenvolvidos para o protocolo IPv4 que por consequência aumentou a vida do IPv4 e adiou a implementação do IPv6.

Outro ponto do NAT que é relevante é o fato de que muitos administradores assume que o NAT sozinho é um fator de segurança inabalável. Na prática, ter uma rede atrás do NAT pode ajudar o conceito de segurança de Perímetro em alguns momentos, mas uma rede atrás de um NAT não está inalcançável: existem várias ameaças baseadas em técnicas de ataques a clientes que provam isso, como por exemplo vulnerabilidades em navegadores. Outro bom exemplo são os malwares que utilizam a técnica de “connect back” para possibilitar acesso remoto.

O IPtables tem a tabela NAT dedicada a controlam o fluxo de pacotes que demanda a tradução dos endereços que atravessam o código de roteamento feito pelo kernel do Linux. Esse recurso é possível de ser implementado de três formas, as quais podemos chamar de “chains”:

- ▣ **PREROUTING:** manipulação dos datagramas que chegam da internet para um servidor em um perímetro interno, normalmente utilizada para Nat reverso;
- ▣ **OUTPUT:** é a manipulação dos datagramas que têm origem na própria máquina firewall e até podem atender à regras de repasse que necessitem de tradução IP;
- ▣ **POSTROUTING:** manipulação dos datagramas de um perímetro interno com destino à internet, necessitando de tradução IP.



Exemplo de parametrização de um Mascaramento (IP masquerading), da rede interna 10.0.0.0/16 para a internet:

```
# iptables -A FORWARD -s 10.0.0.0/16 -j ACCEPT  
# iptables -t nat -A POSTROUTING -o eth1 -s 10.0.0.0/16 -j MASQUERADE  
# echo "1" >/proc/sys/net/ipv4/ip_forward
```

A regra de mascaramento anterior tem como objetivo permitir que toda a rede 10.0.0.0/16 possa sair para a internet com o fluxo de pacotes com saída definida pela interface eth1, que é a interface com IP de internet nesse exemplo.

A realização de tradução de IP inválido (não roteável na internet) para válido (roteável na internet) na saída e o inverso no retorno de cada datagrama pode ser realizada com o módulo MASQUERADE, mas é válido lembrar que o MASQUERADE é uma forma especial “Source Nat/SNAT”. O MASQUERADE deve ser usado quando o IP da interface que está conectada à internet é configurado dinamicamente. Um exemplo clássico são as conexões ADSL. Quando o IP for fixo, como é comum em um link dedicado, o recomendável é usar o SNAT. Segue um exemplo feito com o SNAT equivalente à solução com MASQUERADE exemplificada:

```
# iptables -A FORWARD -s 10.0.0.0/16 -j ACCEPT  
# iptables -t nat -A POSTROUTING -s 10.0.0.1 -o eth0 -j SNAT --to 200.1.2.3  
# echo "1" >/proc/sys/net/ipv4/ip_forward
```

 Observe que no caso do SNAT o IP já é pré-definido e, no caso do MASQUERADE, haverá uma consulta para identificar o IP da interface.

Para listar as políticas de NAT ativas, deve-se informar a tabela, conforme exemplificado:

```
# iptables -t nat -n -L
```

Outra possibilidade de tradução de endereço de datagramas através do uso do recurso SNAT é especificar faixas de endereços e portas que serão substituídas na saída do datagrama usado pela interface definida:

```
# iptables -t nat -A POSTROUTING -s 10.0.0.0/16 -o eth0 -j SNAT --to 200.1.2.1-200.1.2.3.4
```

Pode-se ainda predefinir uma faixa de portas:

```
# iptables -t nat -A POSTROUTING -p 6 -s 10.0.0.0/16 -o eth0 -j SNAT --to 200.1.2.1-200.1.2.3.4:35000-35500
```

Quando é definido uma range de portas e IPs, internamente usamos um algoritmo baseado em circular (round robin) para tradução do endereçamento (NAT) de cada conexão.

Outra forma útil de NAT é o Reverso, onde se deseja publicar uma porta de um servidor interno para a internet. Veja os exemplos:

Regra para entrada do fluxo de pacotes:

```
# iptables -t nat -A PREROUTING -i eth1 -p 6 -d 200.1.2.1 --dport 80 -j DNAT --to-destination 10.0.0.2:80
```



Regra para saída do fluxo de pacotes:

```
# iptables -t nat -A POSTROUTING -o eth0 -p 6 -s 10.0.0.1 --sport 80  
-j SNAT --to-source 200.170.203.68:80
```

Caso o administrador queira fazer o tratamento do NAT de uma rede de forma mais restritiva, basta apenas definir as portas e os protocolos os quais deseja-se permitir que suas respectivas comunicações sejam tratadas pela regra de NAT para ter acesso à internet. Veja o exemplo das portas 80, 110 e 1423:

```
iptables -t nat -A POSTROUTING -s 10.0.0.0/16 -p 6 --sport 80 --dport  
1024:65535 -j SNAT --to 200.1.2.3
```

```
iptables -t nat -A POSTROUTING -s 10.0.0.0/16 -p 6 --sport 110  
--dport 1024:65535 -j SNAT --to 200.1.2.3
```

```
iptables -t nat -A POSTROUTING -s 10.0.0.0/16 -p 6 --sport 143  
--dport 1024:65535 -j SNAT --to 200.1.2.3
```

Podemos usar o recurso de “multiport” combinando também com “POSTROUTING”.

Exemplo:

```
iptables -t nat -A POSTROUTING -p 6 -s 10.0.0.0/16 -m  
multiport --sport 1024:65535 -d 0/0 -m multiport --dport  
20,21,22,23,25,80,110,143,443 -j SNAT --to 200.1.2.
```

Trabalhando com redirecionamento de portas

O redirecionamento de portas possibilita modificar conexões com destino a uma porta definida para outra porta na mesma máquina. O recurso de REDIRECT é usado para fazer o redirecionamento parametrizando a porta com opção “--to-port”.

Um exemplo bem prático de utilização de redirecionamento seria para estabelecer um proxy transparente, redirecionando qualquer conexão destinada a 80 para a porta do Proxy. Veja o exemplo a seguir:

```
# iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j  
REDIRECT --to-port 3128
```

Cenário NAT 1: uma máquina tem duas interfaces de rede eth0 e eth1. Esse Host está se comunicando com a rede interna (LAN) de uma empresa e com a internet ao mesmo tempo, sendo que o endereço IP da interface eth0 é 192.168.200.35, essa interface se comunica com a rede interna, o endereço IP da interface eth1 é 200.204.0.164 (IP válido) e essa interface está direto na internet. O objetivo é compartilhar a internet para todas as máquinas da rede 192.168.200.0/24. Elabore um script que execute essa tarefa:

```
# vi nat.sh  
  
PF=$(which iptables)  
if [ -z $PF ] ; then
```

```

        echo "Comando iptables nao encontrado"
        exit
    fi
    func_nat ()
    {
        $PF -t nat -A POSTROUTING -s 192.168.200.0/24 -o eth1 -j SNAT --to
        200.204.0.164
    }
    func_nat

# ./nat.sh

```

Cenário NAT 2: exemplificando o uso de criação de chain e o uso de regras com estado de conexão para criar uma função que vai incrementar as portas a partir de um arquivo externo, considerando os pacotes oriundos na rede interna com destino à internet:

```

$PF=$(which iptables)

$PF -N ALLOW_TCP

NET="0/0"

MINHANET="192.168.200.0/24"

CLI="1024:65535"

$PF -A ALLOW_TCP -p 6 --syn -j ACCEPT

$PF -A ALLOW_TCP -p 6 -m state --state NEW,ESTABLISHED,RELATED -j
ACCEPT

$PF -A ALLOW_TCP -p 6 ! --syn -m state --state NEW -j LOG --log-
prefix "New not syn:"

$PF -A ALLOW_TCP -p 6 ! --syn -m state --state NEW -j DROP

for PORTA in $(cat /etc/fw/portascli.tcp | grep -v ^#)
do
    $PF -A FORWARD -J ALLOW_TCP -s $MINHANET -d 0/0 - -sport $CLI - -
dport $PORT
done

$PF -t nat -A POSTROUTING -s $MINHANET -o eth1 -j SNAT --to 200.204.0.164

```



Monitoramento das conexões feitas na tabela NAT

Toda a atividade de tradução de IPs é registrada no arquivo `/proc/net/ip_conntrack`, sendo possível consultar esse registro. Para visualizar o uso do comando:

```
# cat /proc/net/ip_conntrack
```

Serão listadas todas as conexões que estão sendo tratadas pelo módulo NAT. Outra forma interessante é deixar fixada a saída em um terminal que não está em uso:

```
# tail -f /proc/net/ip_conntrack >> /dev/tty12 &.
```

Usando a Tabela Mangle

Um recurso poderoso do iptables é a capacidade de reescrever valores de campos essenciais de um datagrama. É evidente que deixamos de simplesmente fazer um roteamento inteligente a partir de políticas predefinidas. Nesse nível de tratamento, o datagrama será “reconstruído/reescrito”, ou seja, reescrito, o que, inicialmente, observando de uma forma mais direta, é um ótimo recurso, mas se você é uma daquelas pessoas que gosta de resuscitar máquinas com Linux, entenda que a manipulação de datagramas demanda mais um pouco de processamento. Dependendo do nível de tráfego, talvez não seja interessante utilizar os recursos da tabela Mangle; ou, ainda, se a máquina em questão for um firewall de fronteira; por outro lado, ser for um hardware com boa capacidade de processamento, usufrua desse recurso. Pode-se utilizar os recursos da tabela mangle com PREROUTING, FORWARD e OUTPUT.

Um bom exemplo de utilização da tabela mangle é a possibilidade de manipulação do campo *TOS* (*Type of Service*) utilizando a opção `--set-tos TOS`, que define a nova prioridade dos pacotes tendo como base o TOS. Os valores aceitos são os seguintes para TOS:

- ▣ **Espera Mínima:** é Minimize-Delay, 16 ou 0x10;
- ▣ **Máximo Processamento:** é Maximize-Throughput, 8 ou 0x08;
- ▣ **Máxima Confiança:** é Maximize-Reliability, 4 ou 0x04;
- ▣ **Custo Mínimo:** é Minimize-Cost, 2 ou 0x02;
- ▣ **Prioridade Normal:** é Normal-Service, 0 ou 0x00.

Um datagrama é definido por padrão com o valor TOS ajustado como prioridade normal (bits ajustados para 0x00). Com o recurso, podemos manipular esse campo aumentando a prioridade a ser dada ao datagrama; por outro lado, essa manipulação não faz grande diferença na internet, pois não é política dos roteadores de fronteira e muito menos de sistemas autônomos tratarem desse campo. Pelo menos no seu gateway essa política fará diferença.

Exemplificando o uso do mangle ao definir prioridade mínima para os respectivos protocolos:

Protocolo FTP:

Datagramas montados na própria máquina Firewall:

```
# iptables -t mangle -A OUTPUT -o eth0 -p 6 --dport 21 -j TOS --set-tos 0x10
```

Datagramas que atravessam a máquina para outra rede:

```
# iptables -t mangle -A FORWARD -o eth0 -p 6 --dport 21 -j TOS --set-tos 0x10
```



Datagramas que atravessam a máquina saindo para a internet:

```
# iptables -t mangle -A POSTROUTING -o eth0 -p 6 --dport 21 -j TOS  
--set-tos 0x10
```

Protocolo Telnet

Datagramas montados na própria máquina firewall:

```
# iptables -t mangle -A OUTPUT -o eth0 -p 6 --dport 23 -j TOS --set-  
tos 0x10
```

Datagramas que atravessam a máquina para outra rede:

```
# iptables -t mangle -A FORWARD -o eth0 -p 6 --dport 23 -j TOS  
--set-tos 0x10
```

Datagramas que atravessam a máquina saindo para a internet:

```
# iptables -t mangle -A POSTROUTING -o eth0 -p 6 --dport 23 -j TOS  
--set-tos 0x10
```

Protocolo DNS

Datagramas montados na própria máquina firewall:

```
# iptables -t mangle -A OUTPUT -o eth0 -p 17 --dport 53 -j TOS  
--set-tos 0x10
```

Datagramas que atravessam a máquina para outra rede:

```
# iptables -t mangle -A FORWARD -o eth0 -p 17 --dport 53 -j TOS  
--set-tos 0x10
```

Datagramas que atravessam a máquina saindo para a internet:

```
# iptables -t mangle -A POSTROUTING -o eth0 -p 17 --dport 53 -j TOS  
--set-tos 0x10
```

Protocolo IRC

Datagramas montados na própria máquina firewall:

```
# iptables -t mangle -A OUTPUT -o eth0 -p 6 --dport 6666-6668 -j TOS  
--set-tos 16
```

Datagramas que atravessam a máquina para outra rede:

```
# iptables -t mangle -A FORWARD -o eth0 -p 6 --dport 6666-6668 -j  
TOS --set-tos 16
```

Datagramas que atravessam a máquina saindo para a internet:

```
# iptables -t mangle -A POSTROUTING -o eth0 -p 6 --dport 6666-6668  
-j TOS --set-tos 16
```



Protocolo FTP-Data

Datagramas montados na própria máquina Firewall:

```
# iptables -t mangle -A OUTPUT -o eth0 -p 6 --dport 20 -j TOS --set-tos 8
```

Datagramas que atravessam a máquina para outra rede:

```
# iptables -t mangle -A FORWARD -o eth0 -p 6 --dport 20 -j TOS --set-tos 8
```

Datagramas que atravessam a máquina saindo para a internet:

```
# iptables -t mangle -A POSTROUTING -o eth0 -p 6 --dport 20 -j TOS --set-tos 8
```

Protocolo ICQ

Datagramas montados na própria máquina firewall:

```
# iptables -t mangle -A OUTPUT -o eth0 -p 17 --dport 4000 -j TOS --set-tos 0
```

Datagramas que atravessam a máquina para outra rede:

```
# iptables -t mangle -A FORWARD -o eth0 -p 17 --dport 4000 -j TOS --set-tos 0x10
```

Datagramas que atravessam a máquina saindo para a internet:

```
# iptables -t mangle -A POSTROUTING -o eth0 -p 17 --dport 40001 -j TOS --set-tos 0x10
```

Protocolo SSH

Datagramas que atravessam a máquina para outra rede:

```
# iptables -t mangle -A FORWARD -o eth0 -p 6 --dport 22 -j TOS --set-tos 16
```

Datagramas que atravessam a máquina, saindo para a internet:

```
# iptables -t mangle -A POSTROUTING -o eth0 -p 6 --dport 22 -j TOS --set-tos 16
```

Para listar as políticas definidas na tabela mangle:

```
# iptables -nL -t mangle
```

Definindo políticas a partir do Mac Address, exemplo onde confere com a máquina o endereço ethernet igual a 00:0C:6E:17:1A:E6.

```
# iptables -t filter -A INPUT -m mac --mac-source 00:80:AD:B2:60:0B -j DROP
```



Analisando o conteúdo dos datagramas

O IPTables possibilita a verificação do conteúdo de um datagrama através do módulo String e a definição do que será feito com esse pacote, ou seja, nesse contexto, o Iptables torna-se um firewall que atua na Camada 7 (OSI/ISO).

Embora a proposta seja de uma análise rápida, o tratamento demanda uma análise do conteúdo de cada pacote que atenda à política estabelecida. Veja o exemplo a seguir:

Kazaa

```
# iptables -A FORWARD -m string --algo bm --string "X-Kaza " -j DROP  
  
# iptables -A FORWARD -m string --algo bm --string "X-Kazaa" -j LOG -log-prefix "KAZAA"
```

Arquivos .exe

```
# iptables -A INPUT -m string --algo bm --string ! ".exe" -j DROP  
  
# iptables -A INPUT -m string --algo bm --string ! ".exe" -j LOG -log-prefix "ARQ .EXE"
```

Arquivos .exe

```
# iptables -A INPUT -m string --algo bm --string ! ".exe" -j DROP  
  
# iptables -A INPUT -m string --algo bm --string ! ".exe" -j LOG -log-prefix "ARQ .EXE"
```

Arquivos .com

```
# iptables -A INPUT -m string --algo bm --string ! ".com" -j DROP  
  
# iptables -A INPUT -m string --algo bm --string ! ".com" -j LOG -log-prefix "ARQ .COM"
```

Arquivos .pif

```
# iptables -A INPUT -m string --algo bm --string ! ".pif" -j DROP  
  
# iptables -A INPUT -m string --algo bm --string ! ".pif" -j LOG -log-prefix "ARQ .PIF"
```

Exemplo de uso do módulo “string” para bloquear o Facebook:

```
iptables -I FORWARD -m string --algo bm --string "facebook.com" -j DROP  
iptables -I OUTPUT -m string --algo bm --string "facebook.com" -j DROP
```

Embora seja possível bloquear esses tipos de arquivos via IPTables, na maioria dos casos é recomendável realizar esse tipo de tratamento com um servidor Proxy web como o squid.



Todavia, também é recomendado implementar regras rígidas nas estações de trabalho para primeiro impossibilitar instalações de aplicativos pelo usuários e também limitar a execução de binários. Em suma, não se deve deixar esses critérios de segurança exclusivamente sobre a responsabilidade dos dispositivos que cuidam da segurança no perímetro da rede, mesmo porque hoje, com os dispositivos móveis, o perímetro de segurança também se estende às pessoas.

Port Knocking – Conceito e prática

É um método utilizado para, a partir de uma sequência de pacotes específicos, executar uma regra pré-definida no firewall para possibilitar a realização de uma conexão que até então estaria fechada. Normalmente, um administrador usa esse recurso para portas de serviços de login remoto como SSH.

Um ferramenta interessante para a realização dessa configuração é o KNOCKD, seu arquivo de configuração `/etc/knockd.conf`:

```
[options]
logfile = /var/log/knockd.log

[openSSH]
sequence    = 7000,8000,9000
seq_timeout = 5
command     = /sbin/iptables -A INPUT -s %IP% -p tcp --dport 22
-j ACCEPT
tcpflags    = syn

[closeSSH]
sequence    = 9000,8000,7000
seq_timeout = 5
command     = /sbin/iptables -D INPUT -s %IP% -p tcp --dport 22
-j ACCEPT
tcpflags    = fin

Importante lembrar que é necessário alterar o valor da variável booleana START_KNOCKD no arquivo de configuração /etc/default/knockd.
#####
## knockd's default file, for generic sys config
#####

# control if we start knockd at init or not
```

```
# 1 = start  
# anything else = don't start  
START_KNOCKD=1  
  
# command line optionsUtilizando a porta:  
hping -p 7000 -c 1 --syn 192.168.1.111  
hping -p 8000 -c 1 --syn 192.168.1.111  
hping -p 9000 -c 1 --syn 192.168.1.111  
  
Desativando a porta:  
hping -p 9000 -c 1 --fin 192.168.1.111  
hping -p 8000 -c 1 --fin 192.168.1.111  
hping -p 7000 -c 1 --fin 192.168.1.111
```







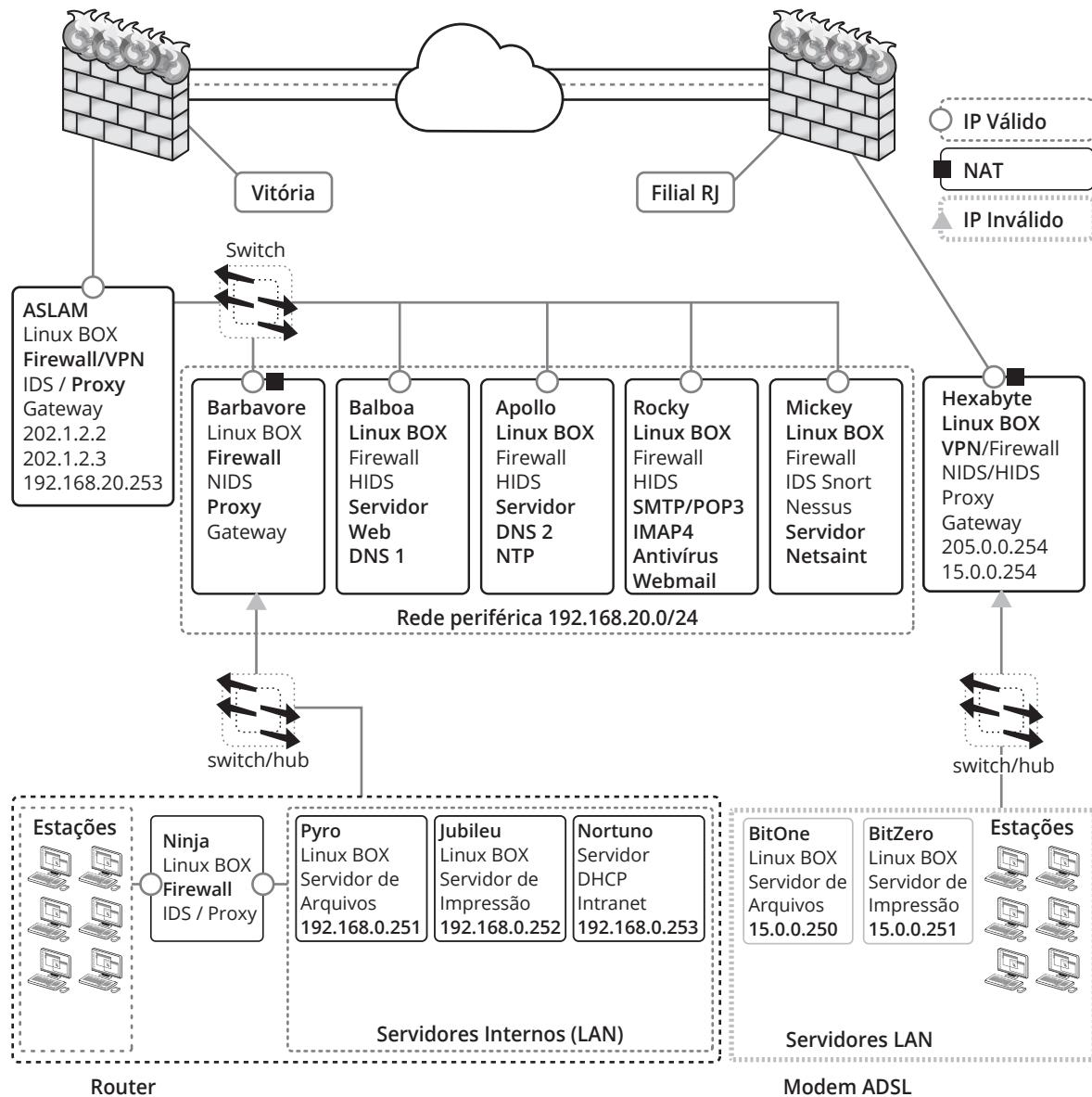
Roteiro de Atividades 8

Prática de customização dos critérios de autenticação e uso de alguns recursos através dos módulos de Kernel e a ferramenta IPTables. Será necessário que o aluno utilize a máquina virtual preparada como Virtuabox. Dessa forma, será possível aplicar todos os conhecimentos (técnicas e ferramentas) de Segurança de Perímetro propostos até o momento.

Figura 8.2
Criação de script de firewall para o Servidor Aslam.

Atividade 8.1 – Hardening Linux – Firewall Linux

1. Considerando o layout da rede da figura a seguir, crie um script de firewall para o Servidor Aslam, que deverá ser instalado no /etc/init.d e funcionar no boot da máquina automaticamente no Runlevel definido, atento às respectivas políticas:



- A Política base será: "Tudo o que não for declarado como permitido, já está expressamente negado" (DROP);
- Todas as conexões como origem ou destino na Loopback são permitidas;
- Só permitirá tráfego para a internet oriundo da rede 192.168.0.0/24, que será roteada pelo Servidor Barbavore com destino à internet, tratando o estado de conexão em tabela à parte (usando o recurso de "chains"), com exceção de conexões destinadas aos serviços HTTP, HTTPS e FTP;
- Aceita conexão VPN IPSEC entre os servidores Aslam (IP 202.1.2.3) e Hexabyte (205.0.0.254), tratando AH e ESP;
- O servidor Aslam (IP 202.1.2.3) também será proxy transparente para conexões com destino a serviços HTTP, HTTPS e FTP, aceitando conexões somente oriundas da rede 192.168.0.0/24, que serão roteadas pelo servidor Barbavore;
- Qualquer atividade de tentativa de conexão cujo destino seja o servidor Aslam deverá ser registrada, seja com origem na internet ou na rede interna, com exceção do servidor Mickey (192.168.20.4);
- Somente Servidor Apollo realizará sincronização com servidores NTP externos. Os demais servidores da Rede Periférica sincronizarão seus relógios a partir do servidor Apollo;
- O Servidor Mickey poderá estabelecer conexões com servidor do projeto Nessus e do Nikto para atualizações dos plugins;
- Em quais atividades as portas administrativas (vide tabela 1) deverão ser registradas (logs) e rejeitadas com controle de limite, com exceção a conexões iniciadas da partir do servidor Mickey (192.168.20.4);

Porta	Protocolo	Rejeitar com
514 - Syslog	UDP	ICMP 3/3
22 - SSH	TCP	TCP/RST
23 - Telnet	TCP	TCP/RST
8000 - Ntop	TCP	TCP/RST
10000 - Webmin	TCP	TCP/RST

- Somente aceitará conexões de clientes oriundas da internet para os da Rede Periférica destinada às suas respectivas portas de serviço declaradas (exclusivamente) na tabela a seguir, tratando essas conexões na tabela de Nat.

Servidor	IP Interno	IP Externo	Protocolo	Porta de Serviço
Balboa	192.168.20.1	202.1.2.2	TCP	80 - HTTP
			UDP	443 - HTTPS
Apollo	192.168.20.2	202.1.2.3	UDP	53 - DNS
Rocky	192.168.20.3	202.1.2.3	TCP	53 - DNS
			TCP	25 - SMTP
			TCP	110 - POP3
			TCP	143 - IMAP
			TCP	443 - HTTPS

2. Elabore uma configuração para o servidor Barbarvore, assumindo:

Que somente ouvirá na Interface eth0, terá o IP: 192.168.0.254, na outra interface eth1 com o IP: 192.168.20.254 e seu gateway default será o servidor ASLAM na interface eth0, com o IP 192.168.20.253;

3. Considerando o layout da rede da figura 8.2, crie um script de firewall para o servidor Barbarvore que deverá ser instalado no /etc/init.d e funcionar no boot da máquina automaticamente no Runlevel definido, atendendo às respectivas políticas;
4. Configure um Port Knocking para porta 22 no servidor Barbarvore.





9

Tuning de Kernel

objetivos

Usar a ferramenta sysctl para Tuning de Kernel; Conhecer Tuning na Pilha TCP/IP com foco no protocolo TCP, ICMP e IP; Entender o Modelo MAC de segurança; Conhecer as ferramentas de MAC para Linux; Usar os módulos Yama e Tomoyo de Segurança; Aprender a parametrizar o Kernel do Linux com a ferramenta sysctl, o Kernel com foco na Pilha TCP/IP e o arquivo /etc/sysctl.conf; Avaliar procedimentos baseados no modelo DAC e MAC de segurança; Ativar alguns controles de MAC baseados no Kernel do Linux.

conceitos

Tuning de Kernel; Tuning da pilha TCP/IP com foco no protocolo TCP; Tuning da pilha TCP/IP com foco no protocolo ICMP.

Tuning de Kernel

Firewall:

- Importante em um projeto de rede.
 - Mas deve estar sempre combinado com outros para melhorar a segurança.

Tuning de sistema Gnu/Linux para um servidor firewall.

O administrador de Sistema Gnu/Linux, ao pensar em implementar um firewall com Linux, tem de ter em mente que um firewall é fundamental em um projeto de rede, mas é um elemento que apenas combinado com outros vai melhorar a segurança proposta. Diante disso, é necessário saber que para melhor funcionamento do firewall, é necessário combinar recursos importantes do Kernel com o que foi definido através do comando *Iptables*. O objetivo deste capítulo é apresentar uma proposta de Tuning de um sistema Gnu/Linux para um servidor firewall.

Este capítulo é diretamente vinculado ao capítulo de firewall; entretanto, pode ser lido separadamente, pois sendo o foco ajuste finos através ferramenta sysctl, ele torna-se um caminho de aprendizado para a utilização desse importante recurso do Sistema Operacional Gnu/Linux.

Contextualizando: quando fala-se em realizar um Tuning, ou seja, ajustes finos, temos que entender que o assunto é bem vasto, entretanto, a proposta desse capítulo é ser um complemento ao capítulo que tratou da possibilidade de implementação de firewall com Linux.

Serão abordados procedimentos importantes para ajustes finos do Sistema Operacional Linux tendo em mente um firewall como objetivo. Entretanto, o processo capacita o administrador a manipular todas as capacidades disponíveis do Kernel que possibilitam modificação em tempo de execução.



Para termos a funcionalidade para modificação de parâmetros e variáveis de Kernel em tempo de execução, precisamos de CONFIG_SYSCTL definido no Kernel. Destacamos que esse recurso é padrão em um Kernel 2.4 ou superior.

Sendo o objetivo o Tuning da Stack TCP/IP (Pilha TCP/IP), vamos manter o foco em três pontos:

- **Tuning TCP**: diretrizes importantes para o tratamento de pacote TCP;
- **Tuning ICMP**: diretrizes importantes para o tratamento de pacote ICMP;
- **Tuning IP**: diretrizes importantes para o tratamento características de troca de pacotes relevantes.

A importância do Tuning é muito grande, pois um firewall implementado com Linux, além das políticas objetivas que são definidas com o Iptables, demanda também definir algumas opções de tratamento dos datagramas na Pilha TCP/IP de seu Sistema Operacional para melhorar o desempenho do firewall e a segurança proposta. Se um administrador não realiza um Tuning para a implementação de um firewall, não está usando o melhor do sistema.

Tuning TCP

Protocolo de TCP:

- Tem características peculiares que devem ser lembradas durante o processo de Tuning.
- É composto por 20 bytes.
 - Possuem informações fundamentais para dificultar ou evitar ataques de Negação de Serviço e técnicas de IP Spoofing.

Devido ao estado de conexão, o Protocolo de TCP tem características peculiares que devem ser lembradas durante o processo de Tuning, pois os 20 bytes que compõe o cabeçalho têm informações interessantes para um firewall de estado de conexão (statefull) que são fundamentais para dificultar ou mesmo em muitos casos evitar ataques como DOS (da sigla em inglês: Denial Of Service – Negação de Serviço) e técnicas de IP Spoofing (falsificação de cabeçalho IP).

Quando estamos configurando políticas com Iptables usando o módulo “state”, estamos utilizando as capacidades de tratamento de datagramas vinculados às propriedades de estado de conexão que foram implementadas no Netfilter.

O Kernel do Linux possibilita-nos manipular algumas opções do Kernel vinculados ao estado da conexão.

Manipulando o valor de ‘Timeout de TCP/FIN’

Verificando o valor para a diretriz “tcp_fin_timeout”. Lembrando que pode variar de acordo com a distribuição Linux e também em relação aos outros Sistemas Operacionais. Verificando valor atual:

```
# sysctl -a | grep net.ipv4.tcp_fin_timeout
```

É recomendável manter um valor próximo ao do valor padrão, que é 60.

```
# sysctl -w net.ipv4.tcp_fin_timeout=50
```



Saiba mais

Para a manipulação consciente dessas capacidades, é preciso conhecer a estrutura do sistema de arquivos proc. Os parâmetros de Kernel podem ser alterados através do sistema de arquivos /proc ou utilizando a ferramenta sysctl.

Para que essa diretriz seja definitiva, é necessário inseri-la no arquivo `/etc/sysctl.conf`. Verifique se a definição já existe.

```
# cat /etc/sysctl.conf | grep -v ^# | grep .
```

Caso não, insira:

```
# echo "net.ipv4.tcp_fin_timeout =50" >> /etc/sysctl.conf
```

Ativando a novas opções com opção “-p” do sysctl.

```
# sysctl -p
```

Manipulando o valor do ‘Keep Alive TCP’

A opção `net.ipv4.tcp_keepalive_time` possibilita manipular o valor, que por padrão no Kernel 2.6 é de 7.200 segundos, ou seja, 2h. Mais um valor importante para regras elaboradas usando o recurso “StateFull” do Iptables.

Verifique o valor para a diretriz “`ip_default_ttl`”, lembrando que pode variar de acordo com a distribuição Linux e também em relação aos outros Sistemas Operacionais. Verificando valor atual:

```
# sysctl -a | grep net.ipv4.tcp_keepalive
```

O recomendável é manter o valor genérico de sistemas, que é de 7.200 segundos. Dessa forma, vamos atender melhor o tratamento de conexões estabelecidas. Em momentos críticos onde o firewall esteja trabalhando com uma quantidade muito grande de conexões e o preço para manter as informações do estado de conexão esteja motivando queda na performance, o administrador pode diminuir esse tempo de forma a manter por um tempo mínimo as informações de estado de conexão. Diante desse cenário, é sugerido como valor mínimo aceito 1.800 segundos. Todavia, essa manipulação deve ser feita com cautela.

```
# sysctl -w net.ipv4.tcp_keepalive=7200
```

Para que essa diretriz seja definitiva, é necessário inseri-la no arquivo `/etc/sysctl.conf`.

Verifique se a definição já existe:

```
# cat /etc/sysctl.conf | grep -v ^# | grep .
```

Caso não, insira:

```
# echo "net.ipv4.tcp_keepalive=7200=0" >> /etc/sysctl.conf
```

Ativando a novas opções com opção “-p” do sysctl.

```
# sysctl -p
```

Desligando o ‘tcp_window_scaling’

Verifique o valor para a diretriz “`tcp_window_scaling`”, que é booleano. Recomenda-se desabilitar para dificultar técnicas de fingerprint.

Valor padrão: `net.ipv4.tcp_window_scaling=0`. Verificando valor atual:

```
# sysctl -a | grep tcp_window_scaling
```



O recomendável é manter o valor de 0 (zero).

```
# sysctl -w net.ipv4.tcp_window_scaling=0
```

Para que essa diretriz seja definitiva, é necessário inseri-la no arquivo `/etc/sysctl.conf`.

Verifique se a definição já existe:

```
# cat /etc/sysctl.conf | grep -v ^# | grep .
```

Caso não, insira:

```
# echo "net.ipv4.tcp_window_scaling=0" >> /etc/sysctl.conf
```

Ativando a novas opções com opção “-p” do sysctl.

```
# sysctl -p
```

Desligando ‘tcp_sack’

Verifique o valor para a diretriz “tcp_sack”, que também possui valor booleano, e para um Firewall – recomenda-se o valor 0. É interessante para reforçar a questão de tratamento de handshakes.

Verificando valor atual:

```
# sysctl -a | grep tcp_sack
```

O recomendável é manter o valor 0 (zero):

```
# sysctl -w net.ipv4.tcp_sack=0
```

Para que essa diretriz seja definitiva, é necessário inseri-la no arquivo `/etc/sysctl.conf`.

Verifique se a definição já existe:

```
# cat /etc/sysctl.conf | grep -v ^# | grep .
```

Caso não, insira:

```
# echo "net.ipv4.tcp_sack=0" >> /etc/sysctl.conf
```

Ativando a novas opções com opção “-p” do sysctl.

```
# sysctl -p
```

Desligando a opção “tcp_timestamps”:

Verifique o valor para a diretriz “tcp_timestamps”, que tem valor booleano e deve estar desabilitada em um Firewall Interessante para evitar fingerprint passivo.

```
# sysctl -a | grep tcp_timestamps
```

O recomendável é manter o valor genérico de sistemas Unix like atuante como firewall, que deve ser 0 (zero).

```
# sysctl -w net.ipv4.tcp_timestamps=0
```

Para que essa diretriz seja definitiva, é necessário inseri-la no arquivo `/etc/sysctl.conf`.

Verifique se a definição já existe:

```
# cat /etc/sysctl.conf | grep -v ^# | grep .
```

Caso não, insira:

```
# echo "net.ipv4.tcp_timestamps=0" >> /etc/sysctl.conf
```

Ativando a novas opções com opção “-p” do sysctl.

```
# sysctl -p
```

Desabilitando o valor de ECN

O ECN diz respeito a Notificação Explícita do Congestion (ECN), é descrito na RFC 3168 e é um padrão proposto da internet. Todavia, temos observado problemas estranhos sendo relatados em listas de discussão sobre o uso dessa opção para acesso a alguns websites com roteadores que não fornecem esse tipo de informação.

Verifique o valor para a diretriz “tcp_ecn”, lembrando que são valores que podem ser passados via sysctl:

- ▣ **0:** desativa suporte ao ECN;
- ▣ **1:** desativa suporte ao ECN;
- ▣ **2:** só anunciar apoio ECN quando solicitado.

Verificando o valor atual:

```
# sysctl -a | grep tcp_ecn
```

Modificando o valor:

```
# sysctl -w net.ipv4.tcp_ecn =0
```

Para que essa diretriz seja definitiva, é necessário inseri-la no arquivo */etc/sysctl.conf*.

Verifique se a definição já existe:

```
# cat /etc/sysctl.conf | grep -v ^# | grep .
```

Caso não, insira:

```
# echo "net.ipv4.tcp_ecn=0" >> /etc/sysctl.conf
```

Ativando a novas opções com opção “-p” do sysctl:

```
# sysctl -p
```

Melhorando o tratamento para início de conexão

Muitos administradores são vítimas de ataques de inundação de pacotes de início de conexão, tecnicamente conhecidos com Synflood, uma categoria de ataque de Negação de Serviço que sutilmente se aproveita da característica de como os Sistemas Operacionais realizam o HandShake TCP/IP.

Para combater esse tipo de ataque em sistema Linux em muitos cenários, é preciso usar o recurso denominado Syncookies, que pode ser ativado no Kernel do Linux através da diretriz “syncookies” e deve ser combinado com as diretrizes “tcp_max_syn_backlog”, “tcp_synack_retries” e “tcp_abort_on_overflow”.

A opção “tcp_synccookies” é de valor booleano e quando ativada possibilita ativar o recurso syncookies para TCP, uma vez que em distribuições Linux é comum que o Kernel venha compilado com CONFIG_SYN_COOKIES ativo, o que permite que o recurso syncookies seja usado pelo Sistema Operacional toda vez que a fila de syn exceda o valor máximo.

Verifique o valor para a diretriz “tcp_syncookies” verificando o valor atual:

```
# sysctl -a | grep tcp_syncookies
```

O recomendável é manter o valor 1; dessa forma, é ativado mais um mecanismo que ajuda o Sistema Operacional, podendo ser útil em muitas situações de ataques de Inundação de pacotes de Início de Conexão (Synflood).

```
# sysctl -w net.ipv4.tcp_syncookies =1
```

Para que essa diretriz seja definitiva, é necessário inseri-la no arquivo */etc/sysctl.conf*.

Verifique se a definição já existe:

```
# cat /etc/sysctl.conf | grep -v ^# | grep .
```

Caso não, insira:

```
# echo "net.ipv4.tcp_syncookies =1" >> /etc/sysctl.conf
```

Ativando a novas opções com opção “-p” do sysctl.

```
# sysctl -p
```

Definindo “tcp_abort_on_overflow”: possibilita o cancelamento de conexões se os serviços ativos forem demasiadamente lentos e incapazes de prosseguir e aceitar a conexão. Lembrando que esse comportamento não é permitido por padrão. Significa que se o excesso ocorrer devido a um estouro (overflow), a conexão poderá ser retomada.

Verifique o valor para a diretriz “tcp_abort_on_overflow” verificando o valor atual:

```
# sysctl -a | grep tcp_abort_on_overflow
```

O recomendável é manter o valor 1; dessa forma, é ativado mais um mecanismo que ajuda o Sistema Operacional, podendo ser útil em muitas situações de ataques de Inundação de pacotes que buscam Negação de Serviço (DOS).

```
# sysctl -w net.ipv4. tcp_abort_on_overflow =1
```

Para que essa diretriz seja definitiva, é necessário inseri-la no arquivo */etc/sysctl.conf*.

Verifique se a definição já existe:

```
# cat /etc/sysctl.conf | grep -v ^# | grep .
```

Caso não, insira:

```
# echo "net.ipv4. tcp_abort_on_overflow =1" >> /etc/sysctl.conf
```

Ativando a novas opções com opção “-p” do sysctl.

```
# sysctl -p
```

Manipulando a opção ‘tcp_synack_retries’

Número máximo de tempo para que um segmento de SYN/ACK para uma conexão passiva do TCP seja retransmitido. Esse número não deve ser mais elevado que 255. O valor de padrão normalmente habilitado no Kernel do Linux é 5, que corresponde a 180 segundos.

Verifique o valor para diretriz “tcp_synack_retries”, checando o valor atual:

```
# sysctl -a | grep tcp_synack_retries
```

O recomendável é manter o valor padrão de 5.

```
# sysctl -w net.ipv4.tcp_synack_retries=5
```

Para que essa diretriz seja definitiva, é necessário inseri-la no arquivo `/etc/sysctl.conf`.

Verifique se a definição já existe:

```
# cat /etc/sysctl.conf | grep -v ^# | grep .
```

Caso não, insira:

```
# echo "net.ipv4.tcp_synack_retries =5" >> /etc/sysctl.conf
```

Ativando a novas opções com opção “-p” do sysctl.

```
# sysctl -p
```

Manipulando ‘tcp_max_syn_backlog’

O número máximo dos pedidos de conexão enfileirados que não têm recebido ainda um reconhecimento do cliente conectando. Se esse número for excedido, o Kernel começará a deixar de atender pedidos de conexão.

O valor padrão é de 256 e é aumentado para 1024 quando a memória atual no sistema é adequada, ou seja, maior ou igual a 128Mb, e reduzido para 128 quando o sistema tem memória muito baixa, ou seja, menor ou igual a 32Mb.

Esse tipo de informação remete o administrador a uma decisão de melhor dimensionamento da memória do servidor que atuará como firewall.

Verifique o valor para a diretriz “tcp_max_syn_backlog”, verificando valor atual:

```
# sysctl -a | grep tcp_max_syn_backlog
```

O recomendável é manter o valor padrão de 5.

```
# sysctl -w net.ipv4.tcp_max_syn_backlog= 1024
```

Para que essa diretriz seja definitiva, é necessário inseri-la no arquivo `/etc/sysctl.conf`.

Verifique se a definição já existe:

```
# cat /etc/sysctl.conf | grep -v ^# | grep .
```

Caso não, insira:

```
# echo "net.ipv4.tcp_max_syn_backlog=1024" >> /etc/sysctl.conf
```

Ativando a novas opções com opção “-p” do sysctl.

```
# sysctl -p
```

Manipulando a opção ‘tcp_max_tw_buckets’

O número máximo de “sockets” no estado de TIME_WAIT permitido no sistema. Esse limite existe para buscar impedir simples de ataques de Negação de Serviços (DOS).

Verifique o valor para a diretriz “tcp_max_tw_buckets” padrão do sistema, que geralmente é 180000. Todavia, diminuir esse valor pode implicar em problema de performance em servidores de grande quantidade de acesso. Verificando o valor atual:

```
# sysctl -a | grep tcp_max_tw_buckets
```

O recomendável é manter o valor padrão.

```
# sysctl -w net.ipv4.tcp_max_tw_buckets =180000
```

Para que essa diretriz seja definitiva, é necessário inseri-la no arquivo `/etc/sysctl.conf`.

Verifique se a definição já existe:

```
# cat /etc/sysctl.conf | grep -v ^# | grep .
```

Caso não, insira:

```
# echo "net.ipv4.ip_default_ttl=255=0" >> /etc/sysctl.conf
```

Ativando a novas opções com opção “-p” do sysctl.

```
# sysctl -p
```

Tuning ICMP

Definido resposta a Ping

É possível, habilitando a opção “`icmp_echo_ignore_all`”, que trabalhava de forma “booleana”, ou seja, colocando 1 (um) ativa a opção, 0 (zero) desativa. Com o valor 1 (um), definir com que o Kernel GNU/Linux ignore todas as mensagens Internet Control Message Protocol (ICMP) do tipo 8 (Echo Request), não retornando a mensagem ICMP tipo 0 (Echo Reply) para o solicitante. Em suma, com o bit ativo 1 (um) nessa opção, faremos com que nenhuma interface de rede (ETH*), e também a própria interface Loopback, responda ao comando `ping`.

Uma boa motivação para isso é que um pacote de IP carregando uma mensagem ICMP pode conter um payload com informações diferentes do padrão ou mesmo pode ser utilizado com o propósito de Fingerprint remoto via TTL.

Por padrão, o valor dessa opção é 0 (zero), ou seja, é desativado quando o Kernel é iniciado. Muitos administradores não gostam de utilizar essa opção com bit 1, pois uma vez ativa, não é possível definir exceções. Simplesmente a host não responde mais a ping.

Na maioria das situações de uma LAN, é recomendável que seja deixado o valor padrão que até o momento vem definido como 0 (zero) e via Política de Firewall seja restritivo, definido para quais hosts a respectiva máquina responderá a solicitações de ICMP 8. Pois para os administradores que usam o ping como uma ferramenta de diagnóstico, o recurso continuará disponível. Entretanto, não há motivo para que uma solicitação de fora de um host qualquer na internet, por exemplo, necessite de resposta ao comando `ping`.

Devemos verificar o valor para a diretriz “`icmp_echo_ignore_all`” com o comando `sysctl`:

```
# sysctl -a | grep ignore_all
```

O valor ideal para um firewall é 0 (desabilitado), pois uma vez habilitado, todas as mensagens ICMP Echo Request serão ignoradas. Caso seja necessário modificar o valor é possível em tempo execução.

```
# sysctl -w net.ipv4.icmp_echo_ignore_all=0
```

Para que essa diretriz seja definitiva, é necessário inseri-la no arquivo `/etc/sysctl.conf`.

Verifique se a definição já existe.

```
# cat /etc/sysctl.conf | grep -v ^# | grep .
```

Caso não, insira:

```
# echo“net.ipv4. net.ipv4.icmp_echo_ignore_all =0” >> /etc/sysctl.conf
```

Após inserir uma nova diretriz no arquivo */etc/sysctl.conf*, use a opção “-p” para ativá-la.

```
# sysctl -p
```

Desligando resposta de broadcast

Por ser um protocolo de teste ICMP, naturalmente suporta tráfego broadcast. Isso significa que é possível ser encaminhada uma solicitação de ICMP tipo 8 (Echo Request) tanto para endereço broadcast como para multicast de uma rede. No caso da solicitação encaminhada para endereço de broadcast é sabido que todos os hosts que estiverem ativos responderão.

Podemos utilizar a opção “icmp_echo_ignore_broadcasts”, com o comando *sysctl*, o que possibilita desabilitar resposta broadcasts de ICMP, o que para um firewall, roteador, gateway e mesmo hosts que se encontram em uma DMZ é extremamente recomendável.

Essa recomendação vem dos primórdios da internet, onde um ataque sutil que se aproveitava dessa característica de resposta de broadcast tornou-me muito comum: o ataque ficou conhecido como Smurf Attack.

Verifique o valor para a diretriz “icmp_echo_ignore_broadcasts”, lembrando que o valor para um firewall deve ser 1.

```
# sysctl -a | grep ignore_broadcasts
```

O valor ideal para um firewall é 0 (desabilitado), pois uma vez habilitado, todas as mensagens ICMP Echo Request serão ignoradas. Caso seja necessário modificar o valor é possível em tempo execução.

```
# sysctl -w net.ipv4.icmp_echo_ignore_broadcasts=1
```

Para que essa diretriz seja definitiva, é necessário inseri-la no arquivo */etc/sysctl.conf*.

Verifique se a definição já existe.

```
# cat /etc/sysctl.conf | grep -v ^# | grep .
```

Caso não, insira:

```
# echo “net.ipv4.icmp_echo_ignore_all=1” >> /etc/sysctl.conf
```

Após inserir uma nova diretriz no arquivo */etc/sysctl.conf* utilize a opção “-p” para ativá-la.

```
# sysctl -p
```

Definido limites no tratamento de resposta a datagramas ICMP

Partindo do princípio de que um servidor responderá a ping (ICMP), é recomendável definir quantos pacotes no máximo serão tratados pelo Kernel através da opção “icmp_ratelimit”. Com essa opção, pode-se definir a quantidade de pacotes que serão tratados na respectiva janela de tempo. Essa opção trabalha em conjunto com a opção “icmp_ratemask”, que possibilita parametrizar quais os tipos de ICMP que deverão ser tratados por esse critério.

Verifique o valor para a diretriz “icmp_ratelimit”:

```
# sysctl -a | grep icmp_ratelimit
```

O valor ideal para um firewall é 100 jiffies (1 jiffy=1/100 sc), e uma vez habilitado, todas as mensagens ICMP serão limitadas. Caso seja necessário modificar o valor, é possível em tempo execução.

```
# sysctl -w net.ipv4.icmp_ratelimit=200
```

Para que essa diretriz seja definitiva, é necessário inserir essa definição no arquivo */etc/sysctl.conf*. Verifique se a definição já existe.

```
# cat /etc/sysctl.conf | grep -v ^# | grep .
```

Caso não, insira:

```
# echo "net.ipv4.icmp_ratelimit=100" >> /etc/sysctl.conf
```

Após inserir uma nova diretriz no arquivo */etc/sysctl.conf*, utilize a opção “-p” para ativá-la.

```
# sysctl -p
```

Verifique o valor para diretriz “icmp_ratemask”:

```
# sysctl -a | grep icmp_ratemask
```

O valor padrão “icmp_ratemask” é 6168, resultante do valor 2 elevado a “n” para cada tipo ICMP, ou seja:

```
icmp_ratemask=(2^3 + 2^4 + 2^11 + 2^12)=6168
```

Assim, somente os datagramas ICMP do tipo 3, 4, 11, 12 serão limitados pelo valor da diretriz “icmp_ratelimit”.

Dessa forma, se quisermos limitar ICMP Echo-reply, a conta deve ser:

```
icmp_ratemask=(2^0 + 2^3 + 2^4 + 2^11 + 2^12)=6169
```

```
# sysctl -w net.ipv4.icmp_ratemask=6169
```

Para que essa diretriz seja definitiva, é necessário inseri-la no arquivo */etc/sysctl.conf*.

Verifique se a definição já existe.

```
# cat /etc/sysctl.conf | grep -v ^# | grep .
```

Caso não, insira:

```
# echo "net.ipv4.icmp_ratemask=6169" >> /etc/sysctl.conf
```

Após inserir uma nova diretriz no arquivo */etc/sysctl.conf*, utilize a opção “-p” para ativá-la.

```
# sysctl -p
```

Tratando mensagem inválida ICMP de roteadores

A opção “icmp_ignore_bogus_error_responses”, que por padrão é desabilitada, deve ser ativada em um servidor que atuará como firewall, Gateway ou um Roteador. Sendo necessária, pois algumas vezes um roteador envia em broadcast frames contendo respostas inválidas e, como resultado, todos esses eventos são registrados (log) pelo Kernel. Para evitar o armazenamento desnecessário de logs, é aconselhado habilitar esse parâmetro para que o Kernel ignore essas mensagens inválidas.



Verifique o valor para a diretriz “`icmp_ignore_bogus_error_responses`”, lembrando que o valor para um firewall deve ser 1.

```
# sysctl -a | grep icmp_ignore_bogus_error_responses
```

O valor ideal para um firewall é 0 (desabilitado), pois uma vez habilitado, todas as mensagens ICMP Echo Request serão ignoradas. Caso seja necessário modificar o valor, é possível em tempo execução.

```
# sysctl -w net.ipv4.icmp_ignore_bogus_error_responses=1
```

Para que essa diretriz seja definitiva, é necessário inseri-la no arquivo `/etc/sysctl.conf`. Verifique se a definição já existe.

```
# cat /etc/sysctl.conf | grep -v ^# | grep .
```

Caso não, insira:

```
# echo "net.ipv4.icmp_ignore_bogus_error_responses=1" >>/etc/sysctl.conf
```

Após inserir uma nova diretriz no arquivo `/etc/sysctl.conf`, utilize a opção “-p” para ativá-la.

```
# sysctl -p
```

Tuning IP

Habilitando repasse de Pacotes (IP_FORWARD)

Certifique-se de que o encaminhamento de IP esteja desligado. Nós só precisamos disso para um host hospedado em vários lugares. É aconselhável ligar ou desligar essa opção antes de outras opções, já que ela liga ou desliga outras opções também.

Verifique o valor para a diretriz “`ip_forward`”, lembrando que o valor para um firewall deve ser 1.

```
# sysctl -a | grep ip_forward
```

O valor ideal para um firewall é 0 (desabilitado), pois uma vez habilitadas, todas as mensagens ICMP Echo Request serão ignoradas. Caso seja necessário modificar o valor, é possível em tempo execução.

```
# sysctl -w net.ipv4.ip_forward =1
```

Para que essa diretriz seja definitiva, é necessário inseri-la no arquivo `/etc/sysctl.conf`. Verifique se a definição já existe.

```
# cat /etc/sysctl.conf | grep -v ^# | grep .
```

Caso não, insira:

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

Após inserir uma nova diretriz no arquivo `/etc/sysctl.conf`, utilize a opção “-p” para ativá-la.

```
# sysctl -p
```



Manipulação da opção de ‘source_route’

Essas opções vinculadas às interfaces de rede possibilitam aceite de pacotes roteados pela origem. Em um firewall, recomenda-se desabilitar essa possibilidade.

Consultando o valor ativo:

```
# cat /etc/sysctl.conf | grep source_route
```

Definindo valor recomendável para um firewall para 0 (desabilitado). Caso seja necessário, modificar o valor é possível em tempo execução:

```
# sysctl -w net.ipv4.conf.all.accept_source_route=0  
# sysctl -w net.ipv4.conf.lo.accept_source_route=0  
# sysctl -w net.ipv4.conf.eth0.accept_source_route=0  
# sysctl -w net.ipv4.conf.default.accept_source_route=0
```

Para que essa diretriz seja definitiva, é necessário inseri-la no arquivo */etc/sysctl.conf*.

Verifique se a definição já existe.

```
# cat /etc/sysctl.conf | grep -v ^# | grep .
```

Caso não, insira:

```
# echo "net.ipv4.conf.all.accept_source_route=0" >> /etc/sysctl.conf  
# echo "net.ipv4.conf.lo.accept_source_route=0" >> /etc/sysctl.conf  
# echo "net.ipv4.conf.eth0.accept_source_route=0" >> /etc/sysctl.conf  
# echo "net.ipv4.conf.default.accept_source_route=0" >> /etc/sysctl.conf
```

Ativando a novas opções com opção “-p” do sysctl.

```
# sysctl -p
```

Habilitando Rp-filter

Possibilita o filtro de caminho reverso. Isso ajuda a ter a certeza de que os pacotes usam endereços de fonte legítimos ao rejeitar pacotes de entrada automaticamente, se a entrada da tabela de roteamento para o endereço fonte não bater com a interface de rede em que chegam.

Esse procedimento pode evitar ataques de IP Spoofing. No entanto, o administrador deve ficar atento com problemas que possam ser causados em caso de roteamento assimétrico ou mesmo balanceamento, devido ao algoritmo de injeção dos pacotes nas interfaces físicas provocar uma situação em que o rp_filter impedirá o tráfego. Nesse contexto, demanda-se que o suporte a rp_filter seja desabilitado nas interfaces de rede envolvidas; do contrário, utilize-o.

Verifique o valor para a diretriz “ip_default_ttl”,

```
# sysctl -a | grep rp_filter
```

É recomendável manter o valor genérico de sistemas Unix like, que é 255. Dessa forma, minimizamos a possibilidade técnica de fingerprint de Sistemas Operacionais baseadas em TTL.

```
# sysctl -w net.ipv4.conf.all.rp_filter=1  
# sysctl -w net.ipv4.conf.lo.rp_filter=1  
# sysctl -w net.ipv4.conf.eth0.rp_filter=1  
# sysctl -w net.ipv4.conf.default.rp_filter=1
```

Para que essa diretriz seja definitiva, é necessário inseri-la no arquivo */etc/sysctl.conf*. Verifique se a definição já existe:

```
# cat /etc/sysctl.conf | grep -v ^# | grep .
```

Caso não, insira:

```
# echo "net.ipv4.conf.all.rp_filter=1" >> /etc/sysctl.conf  
# echo "net.ipv4.conf.lo.rp_filter=1" >> /etc/sysctl.conf  
# echo "net.ipv4.conf.eth0.rp_filter=1" >> /etc/sysctl.conf  
# echo "net.ipv4.conf.default.rp_filter=1" >> /etc/sysctl.conf
```

Ativando a novas opções com opção “-p” do sysctl.

```
# sysctl -p
```

Habilitando o ‘Log_martian’

Essa opção não é interessante para alguns administradores, devido à grande quantidade de registro de eventos (logs) que pode gerar, pois possibilita que qualquer pacote de origem suspeita ou desconhecida (como pacotes forjados) sejam registrados (log) pelo próprio Kernel.

Verifique o valor para a diretriz “log_martians” em seu sistema:

```
# sysctl -a | grep log_martians
```

O recomendável é manter o valor 1.

```
# sysctl -w net.ipv4.conf.log_martians =1  
# sysctl -w net.ipv4.conf.lo. log_martians =1  
# sysctl -w net.ipv4.conf.eth0. log_martians =1  
# sysctl -w net.ipv4.conf.default. log_martians =1
```

Para que essa diretriz seja definitiva, é necessário inseri-la no arquivo */etc/sysctl.conf*.

Verifique se a definição já existe:

```
# cat /etc/sysctl.conf | grep -v ^# | grep .
```

Caso não, insira:

```
# echo "net.ipv4.conf.all.log_martians =1" >> /etc/sysctl.conf  
# echo "net.ipv4.conf.lo.log_martians =1" >> /etc/sysctl.conf  
# echo "net.ipv4.conf.eth0.log_martians =1" >> /etc/sysctl.conf  
# echo "net.ipv4.conf.default.log_martians =1" >> /etc/sysctl.conf
```



Ativando a novas opções com opção “-p” do sysctl.

```
# sysctl -p
```

Manipulando Valor de TTL

Verifique o valor para a diretriz “ip_default_ttl”, lembrando que ele pode variar de acordo com a distribuição Linux e também em relação aos outros Sistemas Operacionais. Verificando valor atual:

```
# sysctl -a | grep ip_default_ttl
```

O recomendável é manter o valor genérico de sistemas Unix like, que é 255. Dessa forma, minimizamos dificuldades técnicas de fingerprint baseadas em TTL.

```
# sysctl -w net.ipv4.ip_default_ttl=255
```

! Distribuições Linux em geral mantêm o valor 64 para TTL, igual a sistema Unix BSD, Mac OS, entre outros.

Para que essa diretriz seja definitiva, é necessário inseri-la no arquivo */etc/sysctl.conf*.

Verifique se a definição já existe:

```
# cat /etc/sysctl.conf | grep -v ^# | grep .
```

Caso não, insira:

```
# echo "net.ipv4.ip_default_ttl=255=0" >> /etc/sysctl.conf
```

Ativando a novas opções com opção “-p” do sysctl.

```
# sysctl -p
```

Manipulando a opção ‘shared_media’

O valor default é 1 dessa variável, que é booleana. O principal objetivo dessa variável é informar ao Kernel se deve realmente enviar mensagens ICMP redirects para uma rede específica ou não.

Essa variável informa ao Kernel se a interface física está compartilhada ou não, ou seja, se diferentes redes IP com diferentes máscaras operam sobre uma mesma mídia ou não.

Caso não se utilize IP virtual, o valor deve ser 0 (zero).

Verifique o valor para diretriz “shared_media”:

```
# sysctl -a | grep shared_media
```

O recomendável é manter 1:

```
# sysctl -w net.ipv4.conf.shared_media=1  
# sysctl -w net.ipv4.conf.lo.shared_media =1  
# sysctl -w net.ipv4.conf.eth0.shared_media =1  
# sysctl -w net.ipv4.conf.default.shared_media =1
```

Para que essa diretriz seja definitiva, é necessário inseri-la no arquivo `/etc/sysctl.conf`. Verifique se a definição já existe:

```
# cat /etc/sysctl.conf | grep -v ^# | grep .
```

Caso não, insira:

```
# echo "net.ipv4.conf.all.shared_media =1" >> /etc/sysctl.conf  
# echo " net.ipv4.conf.lo.shared_media =1" >> /etc/sysctl.conf  
# echo " net.ipv4.conf.eth0.shared_media =1" >> /etc/sysctl.conf  
# echo "net.ipv4.conf.default.shared_media =1" >> /etc/sysctl.conf
```

Ativando a novas opções com a opção “-p” do sysctl.

```
# sysctl -p
```

Manipulando a opção ‘ip_local_port_range’

Para essa opção, devem ser passados dois valores inteiros que definem a range de portas de cliente. Sendo definição de portas de clientes, deve-se parametrizar valores entre 1024 e 65535.

Devendo ficar claro que essas portas não vão conflitar com as portas usadas para NAT (traduções de endereçamento). Sabendo que o Sistema Operacional tem controle sobre isso e a sobreposição é praticamente nula.

Verifique o valor para diretriz “ip_local_port_range”:

```
# sysctl -a | grep ip_local_port_range
```

O recomendável é manter:

```
# sysctl -w ip_local_port_range=1024 65000
```

Para que essa diretriz seja definitiva, é necessário inseri-la no arquivo `/etc/sysctl.conf`.

Verifique se a definição já existe:

```
# cat /etc/sysctl.conf | grep -v ^# | grep .
```

Caso não, insira:

```
# echo "net.ipv4.ip_local_port_range=1024 65000" >> /etc/sysctl.conf
```

Ativando a novas opções com opção “-p” do sysctl.

```
# sysctl -p
```

A seguir, um exemplo de uma função Shell Script (para sh ou bash) que trabalha com o sysctl. Essa função é para definir algumas capacidades do Kernel, para que o sistema seja adequado à função de firewall.

```
# vi tuning_kernel.sh
```

Conteúdo:

```
tuning_kernel()  
{
```



```

#-----
#-----

echo “ - Desabilitando ataques de IP Spoofing.”
sysctl -w net.ipv4.conf.all.rp_filter=2

# Habilitando FORWARDing
sysctl -w net.ipv4.ip_forward=1

echo “Kill Timestamps”
sysctl -w net.ipv4.tcp_timestamps=0

echo “ - Habilitando protecao a Cookie TCP SYN.”
sysctl -w net.ipv4.tcp_syncookies=1

echo “ - Habilitando configuracoes de ICMP.”
echo “Desabilitando protecao a echo de broadcast ICMP.”
sysctl -w net.ipv4.icmp_echo_ignore_broadcasts=1

echo “Habilitando protecao a mensagem de bad error”
sysctl -w net.ipv4.icmp_ignore_bogus_error_responses=1

echo “Desabilitando redirecionamentos de ICMP”
sysctl -w net.ipv4.conf.all.accept_redirects=0

echo “ - Certifique que pacotes roteados na origem foram descartados “
sysctl -w net.ipv4.conf.all.accept_source_route=0

echo “ - Logar pacotes spoofed, roteados na origem ou redirecionados “
sysctl -w net.ipv4.conf.all.log_martians=1

echo “Valores recomendados para tratamento de Datagrama TCP
pensando em DOS e DRDOS”
sysctl -w net.ipv4.tcp_fin_timeout=30

```

```

sysctl -w net.ipv4.tcp_keepalive_time="7200"
sysctl -w net.ipv4.tcp_window_scaling=0
sysctl -w net.ipv4.tcp_sack=0

echo "Alterando o valor do TTL para 255 para ficar padrao do Unix"
sysctl -w net.ipv4.default_ttl=255

echo "Alterando o valor do ratemask para englobar o tratamento para
os"
echo "icmps 0 3 4 5 8 11 12"
sysctl -w net.ipv4.icmp_ratemask=6457
#-----
#-----
}

```

Executando a função:

```
# sh tuning_kernel.sh
```

Segurança de Kernel – Segurança na última fronteira

Tratativa de segurança na última fronteira (camada de Kernel no Linux):

- Estará vinculada aos Módulos de Segurança do Linux (LSM).



Qualquer tratativa de segurança na última fronteira, ou seja, na camada de Kernel no Sistema Operacional Linux, estará vinculada aos Módulos de Segurança do Linux (LSM), que são melhorias desenvolvidas nos últimos anos com foco na segurança do sistema. Essas melhorias são recursos também para conformidade com as recomendações de segurança dos documentos do Posix 1e e também para possibilitar ao sistema Linux ter a capacidade de implementação de controles de segurança baseados no meio Mandatory Access Control (MAC).

Classicamente, em um ambiente MAC, todos os pedidos para o acesso aos recursos são automaticamente sujeitos aos controles de acesso. Em tais ambientes, todos os usuários e recursos são classificados e recebem uma ou mais etiquetas de segurança, tais como: "unclassified", "secret" e "topsecret".

Quando os usuários pedem um recurso, as etiquetas de segurança associadas são examinadas e o acesso está permitido somente se a etiqueta do usuário é igual ou maior do que aquele do recurso. Por exemplo, a um usuário com acesso "Secret" é permitido acessar um documento "Unclassified", mas um usuário com uma etiqueta de "Unclassified" não obtém acesso para uma informação "Secret".

Em ambiente MAC, apenas os indivíduos com privilégios administrativos podem gerenciar os controles de acesso.

Saiba mais



O ambiente MAC é o regime mais restrito de controle de acesso, sendo mais indicado em ambientes onde são demandados controles mais elevados de segurança. Entretanto, os administradores mais experientes têm a oportunidade de colocar a segurança dos servidores que proveem serviços na internet em um outro nível.



Definição de área do usuário

Área do usuário, também chamada de “user mode”, “userland” ou “userspace”, é a porção do sistema de memória onde são executadas as aplicações de usuário, mas que depende diretamente da área destinada às instruções de Kernel. Esse contraste com a camada de Kernel que é essa porção de memória na qual o Kernel executa e supre esses serviços.

O conteúdo da memória RAM, também denominada Memória Principal, pode ser acessado (exemplo: leitura e escrita) em uma velocidade extremamente alta, mas retém temporariamente a informação (quando em uso ou quando há restos de suprimentos de força).

Esse contraste com armazenagem na Memória Secundária (disco rígido) que tem uma velocidade de acesso bem menor, porém a informação é mantida após seu desligamento.

Um processo em execução é um exemplo de um programa. Os processos dos usuários são exemplos para todos os programas, como o Kernel (utilização e aplicação dos programas). Quando um programa vai ser rodado, ele é copiado da armazenagem para o espaço do usuário, então ele pode ser acessado em grande velocidade para a CPU.

O Kernel é um programa que constitui o núcleo central do Sistema Operacional do computador. Ele não é um processo, mas preferencialmente um controlador de processos. Tem o controle completo de tudo o que ocorre no sistema. Isso inclui a administração individual dos processos dos usuários dentro de um espaço do usuário e a prevenção da interferência de outro.

A divisão do sistema de memória de um Sistema Operacional como UNIX para espaço do usuário tem papel importante na manutenção do sistema e estabilidade de segurança.

Definição de camada de Kernel

O sistema de memória em Linux pode ser dividido em duas regiões distintas: área de Kernel e a área de usuário.

A área do Kernel é onde se localiza o núcleo do Sistema Operacional e onde são executados e providos os serviços para as aplicações que estão na área do usuário – a área de Kernel também pode ser chamada de “Kernel mode”, “Kerenlspace” ou “kernelland”.

A área do usuário é a maior porção de memória onde ficam residentes as instruções das aplicações dos usuários e em alguns casos até mesmo instruções complementares do kernel, quando este é baseado em uma arquitetura de microkernel ou híbrida. Essa área também pode ser chamada de “user mode”, “userspace” ou “userland”.

A memória consiste nas células RAM, cujo conteúdo pode ser acessado em uma velocidade extremamente alta, porém só retém temporariamente (por exemplo, quando em uso é necessário trocar a bateria). O propósito é controlar programas e dados que são correntemente usados e, desse modo, servir com velocidade intermediária a CPU, e mais devagar o armazenamento, que consiste em um ou mais discos rígidos.

A camada do usuário é o local onde estão localizados as memórias com os processos dos usuários (todos os outros como o Kernel roda), um processo em execução no instante do programa. Um dos papéis do Kernel é administrar os processos individuais dos usuários com esse espaço e prevenir as interferências de outros.

Vários projetos de Módulo de Segurança Linux (LSM) surgiram e alguns deles inclusive são oficialmente parte do projeto do Kernel Linux.



Saiba mais

A kerneland pode ser acessada por usuários apenas com uso do sistema de chamada. O sistema de chamada requer um Unix operante por um processo ativo para um serviço realizado pelo Kernel., como imput, output ou processo de criação. É um processo ativo que é correntemente progressivo pela CPU, em contraste com um processo que aguarda um próximo passo na CPU. Input/Output é outro programa operação ou serviço que transfere dados para ou da CPU para ou de dispositivos periféricos (disc drive, teclado, mouse e impressora).



Módulos de Segurança do Linux (LSM), a API para os quadros de controle de acesso:

- **AppArmor**: módulo de segurança que proporciona ao sistema um controle de acesso baseado em MAC;
- **Security Enhanced Linux (SELinux)**: uma proposta de segurança flexível e refinada que possibilita a configuração de controles MAC, desenvolvida pela NSA. É um recurso completo para implementação de controle MAC para serviços e dispositivos do sistema;
- **Smack**: é um módulo que proporciona a criação de alguns controles de acesso simplificados de MAC (Simplified MAC Kernel – Simplified Mandatory Access Control Kernel) na camada kernel, embora denominado um módulo simplificado de MAC. O Smack fornece um mecanismo de criação de controles para proteger os processos e dados (arquivos), controles esses que definem as regras de acessos do que deve ser permitido ou negado;
- **Tomoyo**: outro caminho baseado em sistema de controle de acesso (LiveCD disponível);
- **Grsecurity**: é um patch de segurança que não faz parte do código oficial do kernel do Linux, é um projeto externo, mas que traz aprimoramento à segurança para o Kernel Linux, tanto considerando o modelo de segurança MAC como também o modelo RBAC. Entre as capacidades que o Grsecurity possui destaca-se: endurecimento chroot, recursos para auditoria e randomização de proteção de pilha de execução;
- Conjunto de regras de controle de acesso baseado em (RSBAC), Linux Kernel patch implementação de uma estrutura de segurança;
- **FBAC-LSM**: visa proporcionar facilidade de configuração (funcionalidade-based) de controles na camada de Kernel restrições de aplicação;
- **Yama**: um novo módulo de Kernel que foi inserido na 3.4.x, acrescenta restrições para ptrace, proporcionando uma forma programática para declarar relações entre processos. É autoria de um desenvolvedor da empresa responsável pela distribuição Ubuntu (Canonical), sendo mais uma alternativa para um processo de hardening do sistema na camada de Kernel. Melhorando a segurança, proporcionando um nível maior que o tradicional modelo DAC oferta.

Utilização do módulo Yama

Pequeno número de proteções.

- Similar ao que é provido no sistema Solaris.

Para identificar se o suporte está no Kernel, basta executar o comando *grep* no arquivo */boot/config-<kernel version>*.

```
#grep -i yama /boot/config-3.2.0-29-generic
CONFIG_SECURITY_YAMA=y
# CONFIG_DEFAULT_SECURITY_YAMA is not set
```

Os recursos de segurança providos pelo módulo Yama são:

- YAMA_HARDLINKS
- YAMA_PTRACE
- YAMA_SYMLINKS

YAMA_HARDLINKS

Boas práticas recomendadas na ABNT NBR ISO/IEC 27002:2008: com implementação de segurança na camada de Kernel, elevando os controles dos nativos do Sistema Operacional que são baseados no modelo DAC para o modelo MAC, é atendida a conformidade mais uma vez do que foi solicitado no item 10.10.3, onde é enfatizado que os recursos e informações de registros (logs) sejam protegidos contra falsificação e acesso não autorizado. Também possibilita conformidade com o que é citado no item 10.9.3, que diz que a integridade das informações disponíveis em sistemas publicamente acessíveis seja protegida para prevenir alterações não autorizadas.

Esse recurso protege o sistema contra a criação de hardlinks para arquivos que o usuário não tem acesso. Por alguma estranha razão, o padrão POSIX ainda exige esse comportamento. Esse controle é o similar mais próximo Solaris à remoção do privilégio “file_link_any”.

YAMA_PTRACE

Esse parâmetro de proteção do módulo YAMA é projetado para impedir a possibilidade de execução de processo com o mesmo uid de um outro processo que está em execução memória, mas possibilitando também rastreá-los usando ptrace. Sem esse tipo de controle, cria-se um cenário onde é possível gerar uma ameaça aproveitando-se de uma fraqueza particularmente preocupante das interfaces do processo de Linux, onde um usuário poderia ser capaz de examinar a memória e o estado de execução de qualquer de seus processos. Por exemplo, durante a execução de um aplicativo como o Firefox, que foi comprometido, seria possível a um atacante anexar a outros processos em execução para extrair as credenciais adicionais e continuar a expandir o alcance de seu ataque na memória.

YAMA_SYMLINKS

Devido a um histórico longo de problemas de segurança vinculado aos links simbólicos, mais comumente visto em diretórios globalmente graváveis, como “/tmp”.

Os recursos do módulo Yama podem ser ativados ou desativados por meio da ferramenta sysctl.

```
# sysctl -a | grep -i yama
kernel.yama.protected_sticky_symlinks=1
kernel.yama.protected_nonaccess_hardlinks=1
kernel.yama.ptrace_scope=1
```

Tomoyo

O Tomoyo é uma solução para MAC que também faz parte oficialmente do Kernel do Linux, incluído a partir da versão 2.6.30. Vale lembrar que foi somente a partir da versão 2.x do Tomoyo, quando foi reestruturado seguindo as diretrizes do padrão LSM, que passou a ser oficial – até então era um patch externo.

O modelo MAC consiste em criar no Sistema Operacional uma camada de controle onde é possível inserir políticas para permitir ou negar acesso de determinados processos a arquivos e/ou dispositivos, sendo importante lembrar que para um sistema Like Unix um dispositivo é “tratado como um arquivo”. Existem diferentes formas no modelo MAC de tratar o acesso aos arquivos: via informação do “caminho do arquivo” e “labels – rótulos atribuídos” baseados nos atributos estendidos (xattrs). O Tomoyo utiliza o conceito de identificação baseada no “caminho do arquivo”, chamada tecnicamente de “Pathname-based Security”.

O módulo MAC de segurança Tomoyo foi idealizado com objetivo de possibilitar definir regras de acesso considerando o comportamento de um processo de um sistema. Um processo é criado para realizar alguma tarefa, e durante sua execução, além do espaço em memória e do tempo de CPU, um processo lê, abre e fecha arquivos. Através do Tomoyo, cria-se uma camada de controle onde é necessário declarar o comportamento do processo, ou seja, é necessário informar que arquivos e dispositivos o processo pode acessar e qual o nível de acesso será permitido. Em suma, de uma certa forma chamadas para aberturas, leituras, acesso a diretório e fechamento de arquivos serão filtradas, levando em consideração as políticas previamente definidas.

O controle MAC como Tomoyo, de uma forma analógica, cria uma fronteira controlada entre as atividades dos processos na Userland (área de memória de aplicativos de usuário) e a Kerneland (área de memória protegida de instruções do Kernel), ou seja, estabelece segurança na última fronteira para as aplicações. Dessa forma, um processo só poderá estabelecer um comportamento de execução e acesso já previamente definidos.

Configuração e administração do Tomoyo

Para identificar se o suporte está ativo no Kernel, basta executar o comando `grep` no arquivo `/boot/config-<kernel version>`, pois embora o Tomoyo seja habilitado por padrão em várias distribuições Linux, caso o Kernel em uso não tenha o suporte habilitado, será demandado a compilação do Kernel.

```
# grep -i tomoyo /boot/config-3.2.0-4-486
CONFIG_SECURITY_TOMOYO=y
CONFIG_SECURITY_TOMOYO_MAX_ACCEPT_ENTRY=2048
CONFIG_SECURITY_TOMOYO_MAX_AUDIT_LOG=1024
# CONFIG_SECURITY_TOMOYO OMIT_USERSPACE_LOADER is not set
CONFIG_SECURITY_TOMOYO_POLICY_LOADER="/sbin/tomoyo-init"
CONFIG_SECURITY_TOMOYO_ACTIVATION_TRIGGER="/sbin/init"
# CONFIG_DEFAULT_SECURITY_TOMOYO is not set
```

Embora seja identificada a linha “`CONFIG_SECURITY_TOMOYO=y`”, é necessário habilitar o módulo Tomoyo como módulo de segurança padrão. Para isso, é necessário editar o arquivo `/boot/grub/grub.cfg`, inserindo o parâmetro “`security=tomoyo`” na linha do boot do Kernel correspondente. Veja o exemplo:

```
linux    /vmlinuz-3.2.0-4-486 root=/dev/mapper/LVM_RNP-BARRA ro
security=tomoyo quiet
```

Ferramentas da userland

Tenha a certeza de que o suporte ao Tomoyo está ativado.

Com o suporte ativado no Kernel, deve-se validar se as ferramentas já instaladas de gerenciamento do Tomoyo estão instaladas:

```
# dpkg -l | grep tomoyo
```

Caso não, consulte os pacotes disponíveis no repositório remoto via apt:

```
# apt-cache search tomoyo | grep tomoyo
libtomoyotools3 - Lightweight and easy-use Mandatory Access Control
for Linux (shared libraries)

tomoyo-tools - Lightweight and easy-use Mandatory Access Control
for Linux
```

E instale os pacotes necessários:

```
# apt-get install -y libtomoyotools3 tomoyo-tools
```

Para ativar o Tomoyo, use o script de configuração automático que já inicia o Tomoyo com regras básicas que estão concentradas em `/etc/tomoyo`, que antes da execução do script é um diretório vazio. Após a execução, além de geração das regras básicas, são criados também todos os arquivos de configuração.

```
# /usr/lib/tomoyo/init_policy

Creating policy directory... OK

Creating configuration directory... OK

Creating exception policy... OK.

Creating domain policy... OK.

Creating manager policy... OK.

Creating default profile... OK.

Creating stat policy... OK.

Creating configuration file for tomoyo-editpolicy ... OK.

Creating configuration file for tomoyo-auditd ... OK.

Creating configuration file for tomoyo-patternize ... OK.

Creating configuration file for tomoyo-notifyd ... OK.

A lista de arquivos e diretórios criados:

# ls -l

rwxrwxrwx 1 root root 33 Ago 18 00:32 domain_policy.conf ->
policy/current/domain_policy.conf

1rwxrwxrwx 1 root root 36 Ago 18 00:32 exception_policy.conf ->
policy/current/exception_policy.conf

1rwxrwxrwx 1 root root 27 Ago 18 00:32 manager.conf -> policy/
current/manager.conf

drwx----- 3 root root 4096 Ago 18 00:32 policy

1rwxrwxrwx 1 root root 27 Ago 18 00:32 profile.conf -> policy/
current/profile.conf

-rw-r--r-- 1 root root 158 Ago 18 00:32 stat.conf

drwx----- 2 root root 4096 Ago 18 00:32 tools
```

A estrutura de arquivos e diretórios criada durante a execução do script `init_policy` será usada para concentrar informações das políticas e demais detalhes, e não deverão ser editadas diretamente, e sim via ferramentas de gerenciamento do Tomoyo:

Modo de aprendizado

O primeiro passo na utilização do Tomoyo é ativar o modo de aprendizado, inicializando o Tomoyo:

```
# tomoyo-editpolicy /etc/tomoyo/
```



Surgirá a tela de edição de políticas, mas será somente com informações básicas da primeira execução, que são denominadas “domínios”, ou seja, os processos gerenciados pelo Tomoyo, onde o nome de um domínio corresponde a linha completa de execução.

O nome do domínio é separado por espaços. O segundo zero na barra verde do utilitário tomoyo-editpolicy indica que o perfil usado é o 0 (proteção do Tomoyo desativada).

```
<<< Domain Transition Editor >>>      1 domain      '?' for help  
<kernel>  
0: 0      <kernel>
```

Na primeira execução, não há nenhum perfil de política ativo, assim sendo, neste momento devemos iniciar o Tomoyo no modo de aprendizado, para que sejam avaliados todos os processos ativos. Para isso, devemos reiniciar o sistema com o Kernel que tem o suporte ao Tomoyo, pois os sistemas já estão prontos para execução do Tomoyo, uma vez que todos os arquivos de /etc/tomoyo já estão ativos. Dessa forma, para sair do editor, digite “q” e depois “reboot” para inicializar o sistema.

Para visualizar as teclas que possibilitam acesso as outras telas, basta digitar “w” e serão mostrados as opções para alternar em cada tela:

```
e      <<< Exception Policy Editor >>> - informações que ficam no  
arquivo exception_policy.conf  
  
d      <<< Domain Transition Editor >>>  
  
a      <<< Domain Policy Editor >>> - informações que ficam no  
arquivo domain_policy.conf  
  
p      <<< Profile Editor >>> - informações que ficam no arquivo  
profile.conf  
  
m      <<< Manager Policy Editor >>> informações que ficam no  
arquivo manager.conf  
  
n      <<< Namespace Selector >>>  
  
s      <<< Statistics >>>  
  
q      Quit this editor.
```

ou seja:

- ▣ **digite e:** tela de edição de Políticas de Exceção;
- ▣ **digite d:** tela de edição de Domínios de Transição (deletar um domínio);
- ▣ **digite a:** tela de edição de Políticas de Domínios;
- ▣ **digite p:** tela de edição de Perfis;
- ▣ **digite m:** tela de edição do Gerenciador de Políticas;
- ▣ **digite n:** tela de edição de seleção de Namespace;
- ▣ **digite s:** tela de Estatísticas;
- ▣ **digite q:** para sair do editor.

Após o reboot, basta abrir o editor de políticas do Tomoyo: tomoyo-editpolicy, sem parâmetros, e serão mostradas todas a políticas (policies) ativas considerando os processos em execução.



Suponha que se deseja definir políticas para o funcionamento do serviço SSH. Então, é demandado que o utilize durante o procedimento de “aprendizado”, realizando todas as ações comuns, como desativar e ativar o serviço, e realizar uma conexão, pois todas ações estarão sendo interceptadas pelo Tomoyo e as políticas correspondentes geradas.

Devido à característica de configuração do Tomoyo, é recomendável que todo o procedimento seja realizado no processo de teste e homologação das aplicações, para que, ao colocar o servidor em produção, as políticas necessárias já estejam válidas e, se possível, é importante ter um ambiente similar, que pode ser virtual, para que se possa validar novas mudanças para caso seja necessário. Assim, com segurança, o administrador pode realizar todos testes necessários em um ambiente similar, para ter segurança de elaborar um plano de mudança para o servidor em produção.

Nessa primeira execução, ainda não existirá nenhum perfil — até porque ainda nem estamos usando um Kernel com Tomoyo ativado. Nesse momento, deve-se iniciar o Tomoyo para, no modo de aprendizado, entendermos como o sistema funciona por padrão. Para isso, reinicie o sistema com o comando *reboot*, ou por meio da interface gráfica, ou com o comando *shutdown -r now*.

Ao iniciar o primeiro aplicativo que se deseja proteger, por exemplo, o Apache, inicie-o com o comando padrão da sua distribuição.

```
# service apache2 restart
```

Em seguida, abra o editor de políticas do Tomoyo: *tomoyo>EditPolicy*.

No editor, use as setas para ir até o perfil <kernel> /usr/sbin/httpd (o nome pode ser diferente, dependendo da forma como o Apache foi chamado). Se preferir, procure na lista usando a tecla “F”.

```
21: 1 *      /etc/init.d/apache2
22: 0          /bin/echo
23: 0          /bin/run-parts
24: 0          /usr/bin/env
25: 0          /usr/sbin/apache2ctl
26: 0          /bin/chmod
27: 0          /bin/chown
28: 0          /bin/mkdir
29: 0          /bin/mktemp
30: 0          /bin/mv
31: 0          /bin/rm
32: 0          /usr/bin/stat
33: 0          /usr/sbin/apache2
34: 0          /usr/bin/tput
```

Uma vez no modo de aprendizado, o Tomoyo vai iniciar a criação da Polices no arquivo *domain_policy.conf*. A partir desse arquivo, são habilitadas as políticas; todavia, para efetivar o uso da regra, é necessário via *tomoyo-editpolicy* identificar a linha do domínio do Apache2. Pressione “S”, depois “2”, e por último “Enter”, ou seja , alterar o perfil de 1 (learning mode, ou modo de aprendizado) para 2 (permissive mode, ou modo permissivo). Depois, basta sair do editor de políticas do Tomoyo pressionando “Q”.

Após esse procedimento, o Apache será observado “policiado” pelo Tomoyo, e vai enviar mensagens de alerta caso qualquer uma de suas políticas seja violada. As mensagens serão exibidas no console do sistema (*/dev/console*).







Roteiro de Atividades 9

Prática de customização dos critérios de autenticação e uso de alguns recursos através da ferramenta sysctl para Tuning de Kernel. Será necessário que o aluno utilize a máquina virtual preparada como Virtuabox. Dessa forma, será possível aplicar todos os conhecimentos (técnicas e ferramentas) de Hardening propostos até o momento.

Atividade 9.1 – Hardening Linux – Tuning de Kernel

1. Configure o /etc/sysctl.conf com as parametrizações recomendadas neste capítulo.
2. Ative os recursos de segurança do módulo YAMA.
3. Ative o Tomoyo para aprender o funcionamento do Apache.
4. Revise as políticas geradas no Tomoyo.
5. Ative as políticas geradas para o servidor SSH.
6. Ative o Tomoyo para aprender o funcionamento do SSH.
7. Revise as políticas geradas no Tomoyo para o servidor SSH.
8. Utilize o serviço ssh com um usuário definido e avalie as informações de registros de eventos (logs), geradas pelo Tomoyo.





10

Auditoria na camada de Kernel

objetivos

Registrar ações realizadas no sistema a partir das syscall executadas; Criar regras específicas para diretórios-chave; Auditar comandos específicos; Customizar regras para monitorar o sistema a partir de syscall; Customizar regras para monitorar um determinado diretório.

conceitos

Trilha de comandos a partir de registro de syscalls pelo Audit; Definição de regras de auditoria; Geração de relatórios; Contabilização de processo; Auditoria de modificações de um sistema Linux.

Neste capítulo, mostraremos como usar e parametrizar regras para a geração de registros de eventos através do Audit. O Audit é um recurso do Kernel do Linux. Dessa forma, podemos assumir que ao usá-lo estamos estabelecendo segurança na última fronteira.

Exercício de nivelamento 1 Auditoria na camada de Kernel

Avalie o quanto agrega um mecanismo de segurança que possibilita a realização de auditoria na camada de Kernel através de controles sobre as chamadas de sistema realizadas:

Que Sistemas Operacionais possuem auditoria baseada nas chamadas de sistemas?

Auditoria na camada de Kernel

Trilha de comandos: forma de controle importante.



- Mas o que o "history" oferece ainda é pouco.

Audit:

- Framework de auditoria para Linux disponível na série 2.6.x do Kernel.
- Oferece conjunto de recursos que possibilita plena conformidade com o Controlled Access Protection Profiles (CAPP).



Trilha de comandos é uma forma de controle importante, entretanto, o que o "history" oferece ainda é muito pouco para uma gestão de atividades e até mesmo apoio a Gerências de Mudanças, além do fato de que o histórico de comando do "history" poder ser facilmente comprometido. Sendo a proposta deste livro um modelo de Hardening, a auditoria de trilha de comandos merecer uma abordagem mais ampla.

O **Audit** oferece um conjunto de recursos que possibilita plena conformidade com o Controlled Access Protection Profiles (CAPP), pois possibilita criar controles que coletam informações sobre qualquer evento que um administrador julgar relevante. Dessa forma, os registros de auditoria gerados podem ser examinados para determinar se qualquer violação das políticas de segurança ocorreram e por quem foram cometidas.

Devido aos seus recursos, uma vez que é parte do kernel, além de proporcionar uma auditoria na última fronteira, o Audit Framework torna-se para sistema Linux um importante requisito para a certificação Common Criteria-Controlled Access Protection Profiles/Evaluation Assurance Level (CC-CAPP/EAL).

O Common Criteria tem dois conjuntos de requisitos de avaliação, requisitos funcionais e segurança. Os requisitos funcionais descrevem os atributos de segurança do produto em fase de avaliação e são resumidas sob os perfis de acesso controlado de proteção denominado Controlled Access Protection Profiles (CAPP). Requisitos de garantia são resumidos sob o Evaluation Assurance Level (EAL). O EAL descreve todas as atividades que devem ocorrer para os avaliadores estarem confiantes de que os atributos de segurança estão presentes, eficazes e implementados. Exemplos de atividades desse tipo de pesquisa incluem a documentação dos desenvolvedores para vulnerabilidades de segurança, o processo de correção e de testes.

O Common Criteria é um padrão reconhecido internacionalmente e utilizado pelo governo americano e de outras organizações para avaliar a segurança e garantia de produtos de tecnologia. No esquema de Critérios Comuns, EAL representa a profundidade e o rigor da avaliação, dando aos usuários de um respectivo Sistema Operacional a confiança nos produtos especificados em um nível específico de exigências de garantia de segurança associados ao respectivo nível atribuído. A conformidade com esses padrões é muito importante para o mundo corporativo. O Redhat Linux recebeu o Common Criteria Certification com o nível EAL4+. Outros exemplos de sistemas Like Unix que também têm este nível de segurança: VMware ESXi, FreeBSD, Oracle Linux, AIX, HP-UX e Solaris.

O Framework Audit ajuda a tornar o sistema Linux mais seguro, fornecendo os meios para analisar o que está acontecendo no sistema de forma detalhada. Mas sendo uma ferramenta de auditoria, isso não significa, no entanto, fornecer segurança adicional, mas sim fornecer um meio ainda que reativo para rastrear problemas e ajudar ao administrador a realizar uma Resposta a Incidente, possibilitando-o ter conhecimento suficiente para tomar medidas de segurança realmente efetivas.

O Framework Audit é composto por vários componentes, cada um contribuindo com uma funcionalidade importante. Cada módulo tem sua função definida e possibilita interceptação das chamadas de sistema para o Kernel e, a partir de políticas definidas, registra os eventos relevantes. O daemon audit escreve os relatórios é a ferramenta na userland que possibilita a administração do recurso e a captura dos dados relevantes dos eventos e os encaminha para um arquivo de log.

Audit

Framework de auditoria para Linux disponível na série 2.6.x do Kernel, desenvolvido pelo programador Steve Grubb, do time de segurança da RedHat.



Saiba mais

O Common Criteria é um relevante padrão internacional para avaliações de segurança independentes que ajuda os especialistas de segurança a avaliar o nível de segurança de qualquer produto de TI que pretendam implantar em qualquer projeto, inclusive de missão crítica. Diante de tudo isso, é um recurso recomendado em vários documentos de segurança, inclusive os da RedHat e Suse Linux.



Componentes de auditoria

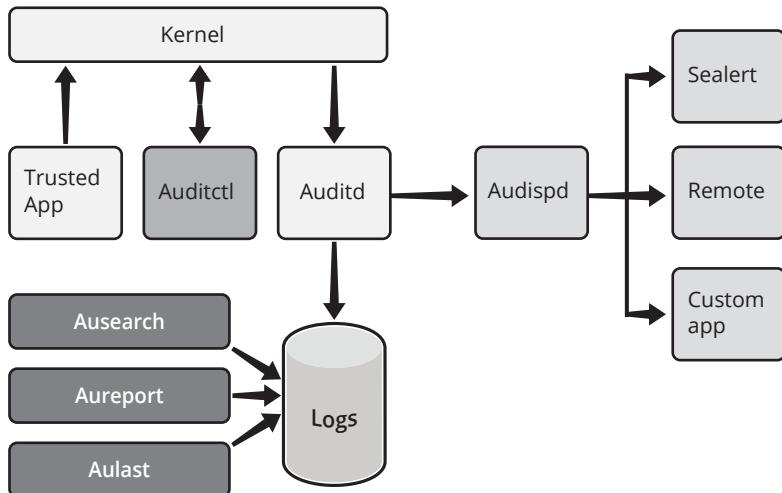


Figura 10.1
Componentes do Framework Audit.

O uso do Framework Audit possibilitará o levantamento a partir de registros (logs) de todas as atividades realizadas em uma determinada janela de tempo, inerente a ações pré-definidas. Outro ponto importante é o fato de que a auditoria é feita a partir do controle da syscalls (chamada de sistema), o que pode ampliar de forma significativa o poder de auditar e o que possibilita facilmente definir regras que coloquem o sistema em conformidade com qualquer regulamentação ou norma (exemplo: PCI, Sarbanes-Oxley, ISO 27002 e HIPAA).

O Framework Audit é um recurso do Kernel que na camada de userland é ativado pelo daemon auditd. Dessa forma, atuando na userland, o sysadmin pode parametrizar todas as regras que serão ativadas na inicialização ou pelo próprio sysadmin. Essas regras são configuradas no "/etc/audit/audit.rules".

Estado da arte

O Framework Audit possibilita consultar em sua base de registros filtrando atividades por:

- ▣ Usuário;
- ▣ Grupo;
- ▣ ID do audit;
- ▣ Hostnames;
- ▣ Endereçamento IP;
- ▣ Nome de arquivo;
- ▣ Operação em arquivos;
- ▣ Falhas;
- ▣ Chamadas de sistema;
- ▣ Argumentos de chamadas de sistema.

O Framework Audit é parte do kernel e tem o conjunto de ferramentas nas userland que possibilita a parametrização e consultas à base de logs em caso de necessidade de auditoria, o que possibilita estabelecer controles de segurança na última fronteira, ou seja, na camada de kernel.



Instalação do Framework Audit:

Através da base apt oficial do repositório do Ubuntu ou Debian, utilize o comando *apt-get* para instalação do Framework Audit:

```
# apt-get install -y auditd
```

Inicializando o Framework Audit através do Daemon auditd:

```
# service auditd restart
```

Para garantir que o deamon do Framework Audit será carregado durante a inicialização do Sistema Operacional:

```
# apt-get install -y rcconf  
# rcconf
```

Arquivos importantes:

- ▣ **/etc/audit/auditd.conf**: arquivo de principal de configuração do audit;
- ▣ **/etc/audit/auditd.rules**: arquivo de principal de configuração de definições para auditctl;
- ▣ **/etc/audisp/audisp.conf**: arquivo de configuração do audisp;
- ▣ **/etc/sysconfig/auditd**: arquivo de parametrizações extras do audit.

Ferramentas de Userland para parametrizar e utilizar as capacidades do Audit:

- ▣ **auditctl**: utilitário de controle do sistema de auditoria do kernel. Você pode consultar o status e deletar ou adicionar regras para o sistema de auditoria do kernel. Configurar o sistema para monitorar um arquivo é feito usando o comando:
 - ▣ **ausearch**: comando para consultar o daemon do Framework Audit baseado em eventos com vários critérios de pesquisa;
 - ▣ **aureport**: ferramenta que produz relatórios resumidos dos logs do sistema Framework Audit (audit.log).

Exemplos de utilização da ferramenta auditctl:

Criando um regra de auditoria para o /etc/passwd com a palavra “usuários” como chave de pesquisa:

```
# auditctl -w /etc/passwd -p war -k usuários
```

Onde:

- ▣ **-w /etc/passwd**: definir uma regra de auditoria para o arquivo */etc/passwd*. Nesse caso, monitorar o arquivo chamado */etc/passwd*;
- ▣ **-p war**: configurar os filtros de permissão para o monitoramento de um sistema de arquivos.

Os valores comumente utilizados são:

- ▣ **r**: para leitura;
- ▣ **w**: para escrita;
- ▣ **x**: para execução;
- ▣ **a**: para anexar (append);
- ▣ **-k <palavra-chave>**: possibilita definir uma palavra-chave para filtragem que pode ser utilizada em consultas futuras com o comando *ausearch*. Essa palavra-chave não deve exceder o valor máximo de 31 bytes.



Em resumo, quando se deseja monitorar um determinado arquivo ou mesmo diretório, é possível criar uma regra para registrar qualquer ação como alteração, escrita ou execução. Tudo é feito registrando as syscalls vinculadas ao respectivo arquivo ou diretório:

Exemplo de criação de uma política para auditar atividades inerentes ao arquivo */etc/shadow*:

```
# auditctl -w /etc/shadow -k senhas -p rwx -k seclinux
```

Exemplo de criação de uma política para auditar todas chamadas de sistema (syscall) – nesse caso, do tipo mount:

```
# auditctl -a exit,never -S mount -k seclinux
```

Exemplo de criação de uma política para auditar toda atividade de processos executados vinculados ao diretório */var/tmp*:

```
# auditctl -w /tmp -p wax -k deamon-tmp
```

Exemplo de criação de uma política para auditar todas as system call vinculadas a um processo cujo PID é 1945:

```
# auditctl -a entry,always -S all -F pid=1945
```

Exemplo de criação de uma política para auditar todas as chamadas de sistema (syscall). Nesse caso, do tipo unlink e unlinkat vinculadas ao diretório */var/tmp*:

```
# auditctl -a exit,always -F dir=/var/tmp -F arch=i386 -S unlink -S unlinkat -k seclinux
```

```
# auditctl -a exit,always -F dir=/var/tmp -F arch=x86_64 -S unlink -S unlinkat -k seclinux
```

Um outra forma é inserir essas regras diretamente no arquivo *audit.rules*, onde deve-se iniciar apagando todas a políticas em memória com a opção “-d”, para na sequência inserir as novas políticas.

```
-D  
-b 1024  
  
-w /etc/shadow -k senhas -p rwx  
-a exit,never -S mount  
  
-w /tmp -p e -k seclinux  
  
-a entry,always -S all -F pid=1945  
  
-a exit,always -F dir=/var/tmp -F arch=i386 -S unlink -S unlinkat -k seclinux  
  
-a exit,always -F dir=/var/tmp -F arch=x86_64 -S unlink -S unlinkat -k seclinux
```

A seguir, exemplos de configurações de regras para serem utilizadas no *audit.rules* para informações sensíveis:

Configuração de Regras do Framework Audit – Conformidade com o modelo de Segurança Controlled Access Protection Profile (CAPP).

Parametrizando a deleção das regras ativas em memória:

```
-D
```

Este parâmetro define Buffer de mensagens:

```
## Feel free to increase this if the machine panic's  
-b 8192
```

Definindo regras do modo panic:

```
-f 2
```

Definição de políticas para log do próprio Framework Audit:

```
-w /var/log/audit/ -k loca_audit  
-w /var/log/audit/audit.log -k  
-w /etc/libaudit.conf -k loca-changemanagement -p wa -k loca_audit  
-w /etc/audisp/ -p wa -k loca-changemanagement -k loca_audit  
-w /etc/audit/ -p wa -k loca_audit  
-w /etc/audit/auditd.conf -p wa -k loca-changemanagement -k loca_audit  
-w /etc/audit/audit.rules -p wa -k loca-changemanagement -k loca_audit
```

Exemplos de regras para auditoria a system call específicas:

```
-a entry,always -S creat -S open -S truncate -S ftruncate  
-a entry,always -S truncate64 -k loca_syscall  
-a entry,always -S ftruncate64 -k loca_syscall
```

Exemplo de conjunto de regras para Auditoria de System Call vinculadas a CHMOD e CHOWN:

```
-a entry,always -F arch=b32 -S chmod -S fchmod -S fchmodat -k loca_syscall  
-a entry,always -F arch=b64 -S chmod -S fchmod -S fchmodat -k loca_syscall  
-a entry,always -F arch=b32 -S chown -S fchown -S fchownat -S lchown -k loca_syscall  
-a entry,always -F arch=b64 -S chown -S fchown -S fchownat -S lchown -k loca_syscall  
-a entry,always -F arch=b32 -S fchown32 -S chown32 -S lchown32 -k loca_syscall  
-a entry,always -S unlink -S rename -S link -S symlink -k loca_syscall  
-a entry,always -S mknod -k loca_syscall  
-a entry,always -S mount -k loca_syscall  
-a entry,always -S clone -k loca_syscall
```

```
-a entry,always -S fork -k loca_syscall  
-a entry,always -S vfork -k loca_syscall  
-a entry,always -S clone2 -k loca_syscall  
-a entry,always -S umask -k loca_syscall  
-a entry,always -S adjtimex -S settimeofday -k loca_syscall
```

Exemplo de regras para auditoria de System Calls vinculadas a operações de “move, remover e link” de arquivo:

```
-a entry,always -F arch=b32 -S unlink -S unlinkat -S rename -S  
renameat -k loca_syscall  
-a entry,always -F arch=b64 -S unlink -S unlinkat -S rename -S  
renameat -k loca_syscall  
-a entry,always -F arch=b32 -S link -S linkat -S symlink -S symlinkat  
-k loca_syscall  
-a entry,always -F arch=b64 -S link -S linkat -S symlink -S symlinkat  
-k loca_syscall
```

Exemplo de regras para auditoria de System Calls vinculadas a operações de Manipulação de atributos de arquivos:

```
-a entry,always -F arch=b32 -S setxattr -S lsetxattr -S fsetxattr -S  
removexattr -S lremovexattr -S fremovexattr -k loca_syscall  
-a entry,always -F arch=b64 -S setxattr -S lsetxattr -S fsetxattr -S  
removexattr -S lremovexattr -S fremovexattr -k loca_syscall
```

Exemplo de regras para auditoria de System Calls vinculadas a operações com arquivos de dispositivos:

```
-a entry,always -F arch=b32 -S mknod -S mknodat -k loca_syscall  
-a entry,always -F arch=b64 -S mknod -S mknodat -k loca_syscall
```

Exemplo de regras para auditoria de System Calls vinculadas a operações com diretórios:

```
-a entry,always -F arch=b32 -S mkdir -S mkdirat -S rmdir -k loca_  
syscall  
-a entry,always -F arch=b64 -S mkdir -S mkdirat -S rmdir -k loca_  
syscall
```

Exemplo de regras para auditoria de System Calls vinculadas a operações de montagem:

```
-a entry,always -S umount -k loca_syscall -k loca_syscall  
-a entry,always -S umount2 -k loca_syscall -k loca_syscall  
-a entry,always -F arch=b32 -S mount -S umount -S umount2 -k loca_  
syscall  
-a entry,always -F arch=b64 -S mount -S umount -S umount2 -k loca_  
syscall
```

Exemplo de regras para auditoria de System Calls vinculadas a IPC SYSV (filas de mensagens entre componentes do kernel):

```
-a entry,always -S ipc -F a0=14 -k loca_syscall  
-a entry,always -S ipc -F a0=13 -k loca_syscall
```

Exemplo de regras para auditoria de System Calls vinculadas a IPC SYSV semaphores:

```
-a entry,always -S ipc -F a0=3 -k loca_syscall  
-a entry,always -S ipc -F a0=2 -k loca_syscall  
-a entry,always -S ipc -F a0=1 -k loca_syscall  
-a entry,always -S ipc -F a0=4 -k loca_syscall
```

Exemplo de regras para auditoria de System Calls vinculadas à IPC SYSV shared memory:

```
-a entry,always -S ipc -F a0=24 -k loca_syscall  
-a entry,always -S ipc -F a0=23 -k loca_syscall
```

Exemplo de regras para auditoria de System Calls vinculadas à criação de processos (fork) ou de thread (clone), para uso muito específico:

```
-a entry,always -F arch=b32 -S clone -k loca_syscall  
-a entry,always -F arch=b64 -S clone -k loca_syscall  
-a entry,always -F arch=b32 -S fork -S vfork -k loca_syscall  
-a entry,always -F arch=b64 -S fork -S vfork -k loca_syscall
```

Exemplo de regras para auditoria de System Calls vinculadas a UMASK:

```
-a entry,always -F arch=b32 -S umask -k loca_syscall  
-a entry,always -F arch=b64 -S umask -k loca_syscall
```

Exemplo de regras para auditoria de System Calls vinculadas à modificação de horário no sistema:

```
-a entry,always -F arch=b32 -S adjtimex -S settimofday -S clock_  
settime -k loca_syscall  
-a entry,always -F arch=b64 -S adjtimex -S settimofday -S clock_  
settime -k loca_syscall
```

Exemplo de regras para auditoria de System Calls vinculadas a PTRACE:

```
-a entry,always -F arch=b32 -S ptrace -k loca_syscall
```

Exemplo de regras para auditoria de System Calls:

```
-a exit,always -F arch=b32 -S personality -k loca_syscall
```

Exemplo de regras para auditoria de gerência de atividades de CRON / AT / ANACRON"

```
-w /var/spool/at  
-w /etc/at.allow -p wa -k loca-changemanagement -k loca_cron  
-w /etc/at.deny -p wa -k loca-changemanagement -k loca_cron
```

```
-w /etc/cron.allow -p wa -k loca-changemanagement -k loca_cron  
-w /etc/cron.deny -p wa -k loca-changemanagement -k loca_cron  
-w /etc/cron.d/ -p wa -k loca-changemanagement -k loca_cron  
-w /etc/cron.daily/ -p wa -k loca-changemanagement -k loca_cron  
-w /etc/cron.hourly/ -p wa -k loca-changemanagement -k loca_cron  
-w /etc/cron.monthly/ -p wa -k loca-changemanagement -k loca_cron  
-w /etc/cron.weekly/ -p wa -k loca-changemanagement -k loca_cron  
-w /etc/crontab -p wa -k loca-changemanagement -k loca_cron  
-w /var/spool/cron/root -p wa -k loca-changemanagement -k loca_cron  
-w /etc/anacrontab -p wa -k loca-changemanagement -k loca_cron
```

Exemplo de regras para auditoria de configurações gerais inerentes à autenticação de usuários:

```
-w /etc/group -k loca_autenticacao -k loca-changemanagement  
-w /etc/passwd -k loca_autenticacao -k loca-changemanagement  
-w /etc/passwd- -k loca_autenticacao -k loca-changemanagement  
-w /etc/gshadow -k loca_autenticacao -k loca-changemanagement  
-w /etc/shadow -k loca_autenticacao -k loca-changemanagement  
-w /etc/shadow- -k loca_autenticacao -k loca-changemanagement  
-w /etc/login.defs -p wa -k loca_autenticacao -k loca-changemanagement
```

Exemplo de regras para auditoria de configurações com recursos de autenticação:

```
-w /etc/sudoers -p wa -k loca_autenticacao  
-w /etc/securetty -p wa -k loca-changemanagement  
-w /var/log/faillog -p wa -k loca-changemanagement  
-w /var/log/lastlog -p wa -k loca-changemanagement  
-w /etc/shells -p wa -k loca-changemanagement  
-w /etc/profile -k -p wax loca-changemanagement  
-w /etc/bashrc -p wa -k loca-changemanagement  
-w /etc/csh.cshrc -p wa -k loca-changemanagement  
-w /etc/csh.login -p wa -k loca-changemanagement  
-w /etc/sysconfig/ -p wa -k loca-changemanagement
```

Exemplo de regras para auditoria de configuração do /etc:

```
-w /etc/inittab -p wa -k loca-changemanagement  
-w /etc/rc.local -p wa -k loca-changemanagement
```

```
-w /etc/rc.sysinit -p wa -k loca-changemanagement  
-w /etc/ld.so.conf -p wa -k loca-changemanagement  
-w /etc/ld.so.conf.d/ -p wa -k loca-changemanagement  
-w /etc/localtime -p wa -k loca-changemanagement  
-w /etc/sysctl.conf -p wa -k loca-changemanagement  
-w /etc/modprobe.conf -p wa -k loca-changemanagement  
-w /etc/issue -p wa -k loca-changemanagement  
-w /etc/issue.net -p wa -k loca-changemanagement  
-w /etc/motd -p wa -k loca-changemanagement  
-w /etc/pam_smb.conf -p wa -k loca-changemanagement
```

Exemplo de regras para auditoria de configurações PAM:

```
-w /etc/pam.d/ -p wa -k loca-changemanagement -p pam  
-w /etc/security/limits.conf -p wa -k loca-changemanagement -p pam  
-w /etc/security/pam_env.conf -p wa -k loca-changemanagement -p pam  
-w /etc/security/namespace.conf -p wa -k loca-changemanagement -p pam  
-w /etc/security/namespace.init -p wa -k loca-changemanagement -p pam
```

Exemplo de regras para auditoria de configuração para auditoria de serviços de rede:

```
-w /etc/hosts -p wa -k loca-changemanagement -k loca-network  
-w /etc/sysconfig/network-scripts/ -p wa -k loca-changemanagement  
-k loca-network  
-w /etc/dhcpd.conf -p wa -k loca-changemanagement -k loca-network  
-w /etc/dhclient-eth0.conf -p wa -k loca-changemanagement -k loca-  
network  
-w /etc/xinetd.conf -p wa -k loca-changemanagement -k loca-network  
-w /etc/xinetd.d/ -p wa -k loca-changemanagement -k loca-network  
-w /etc/hosts.allow -p wa -k loca-changemanagement -k loca-network  
-w /etc/hosts.deny -p wa -k loca-changemanagement -k loca-network  
-w /etc/hosts -p wa -k loca-changemanagement -k loca-network t  
-w /etc/resolv.conf -p wa -k loca-changemanagement -k loca-network  
-w /etc/nsswitch.conf -p wa -k loca-changemanagement -k loca-network
```



Exemplo de regras para auditoria de configurações de rede do serviço NFS / AUTOMOUNT:

```
-w /etc/auto.master -p wa -k loca-changemanagement -k nfs  
-w /etc/auto.misc -p wa -k loca-changemanagement -k nfs  
-w /etc/exports -p wa -k loca-changemanagement -k nfs
```

Exemplo de regras para auditoria de configurações de rede do serviço Postfix:

```
-w /etc/postfix/ -p wa -k loca-changemanagement -k postfix
```

Exemplo de regras para auditoria de configurações de rede do serviço NTP:

```
-w /etc/ntp.conf -k change_management -k ntp
```

Regras para Auditoria em configurações de rede do serviço SSH:

```
-w /etc/ssh/sshd_config -p wa -k loca-changemanagement -k ssh
```

Exemplo de regras para auditoria do serviço STUNNEL:

```
-w /etc/stunnel/stunnel.conf -p wa -k loca-changemanagement -k stunnel  
-w /etc/stunnel/stunnel.pem -p wa -k loca-changemanagement -k stunnel
```

Exemplo de regras para auditoria de configurações de rede do serviço FTP / VSFTP:

```
-w /etc/vsftpd.ftpusers -p wa -k loca-changemanagement -k ftpusers  
-w /etc/vsftpd/vsftpd.conf -p wa -k loca-changemanagement -k vsftpd
```

Exemplo de regras para auditoria de configurações de rede do serviço HTTP / APACHE:

```
-w /etc/httpd/conf/ -p wa -k loca-changemanagement -k apache  
-w /etc/httpd/conf.d/ -p wa -k loca-changemanagement -k apache
```

Exemplo de regras para auditoria de configurações de rede do serviço SAMBA:

```
-w /etc/samba/smb.conf -p wa -k loca-changemanagement -k samba  
-w /etc/samba/smbpasswd -p wa -k loca-changemanagement -k samba  
-w /etc/samba/secrets.tdb -p wa -k loca-changemanagement -k samba  
-w /etc/my.conf -p wa -k loca-changemanagement -k samba
```

Exemplo de regras para auditoria de configurações gerais de e-mail para o sendmail:

```
-w /etc/aliases -p wa -k loca-changemanagement -k sendmail -k postfix  
-w /etc/mail/access -p wa -k loca-changemanagement -k sendmail  
-w /etc/mail/access.db -p wa -k loca-changemanagement -k sendmail  
-w /etc/mail/aliases -p wa -k loca-changemanagement -k sendmail  
-w /etc/mail/domaintable -p wa -k loca-changemanagement -k sendmail  
-w /etc/mail/domaintable.db -p wa -k loca-changemanagement -k sendmail  
-w /etc/mail/helpfile -p wa -k loca-changemanagement -k sendmail  
-w /etc/mail/local-host-names -p wa -k loca-changemanagement -k sendmail
```



```
-w /etc/mail/mailertable -p wa -k loca-changemanagement -k sendmail  
-w /etc/mail/mailertable.db -p wa -k loca-changemanagement -k sendmail  
-w /etc/mail/Makefile -p wa -k loca-changemanagement -k sendmail  
-w /etc/mail/sendmail.cf -p wa -k loca-changemanagement -k sendmail  
-w /etc/mail/sendmail.mc -p wa -k loca-changemanagement -k sendmail  
-w /etc/mail/submit.cf -p wa -k loca-changemanagement -k sendmail  
-w /etc/mail/submit.mc -p wa -k loca-changemanagement -k sendmail  
-w /etc/mail/trusted-users -p wa -k loca-changemanagement -k sendmail  
-w /etc/mail/virtusertable -p wa -k loca-changemanagement -k sendmail  
-w /etc/mail/virtusertable.db -p wa -k loca-changemanagement -k sendmail
```

Exemplo de regras para auditoria de configurações de rede do serviço LOG / SNMP:

```
-w /etc/syslog.conf -p wa -k loca-changemanagement -k syslog  
-w /etc/snmp/snmpd.conf -p wa -k loca-changemanagement -k snmp
```

Exemplo de regras para auditoria de configurações de rede do serviço NIS:

```
-w /etc/yp.conf -k loca-changemanagement -k nis  
-w /var/yp/bindingv -k loca-changemanagement -k nis
```

Exemplo de regras para auditoria de configurações de rede do serviço ldap:

```
-w /etc/ldap.conf -k loca-changemanagement -k ldap
```

Exemplo de regras para auditoria de configurações de rede do serviço kerberos:

```
-w /etc/krb5.conf -p wa -k loca-changemanagement -k kerberos  
-w /etc/krb.conf -p wa -k loca-changemanagement -k kerberos  
-w /etc/krb.realms -p wa -k loca-changemanagement -k kerberos  
-w /etc/initlog.conf -p wa -k loca-changemanagement  
-w /etc/default/ -p wa -k loca-changemanagement  
-w /etc/firmware/microcode.dat -p wa -k loca-changemanagement  
-w /etc/fstab -p wa -k loca-changemanagement
```

Tornando imutável as regras do Framework Audit:

```
-e 2
```

Realização de auditoria para identificação de alterações acesso ao arquivo */etc/passwd* com o comando *ausearch*.

Três exemplificações de como usar o comando *ausearch*:

```
# ausearch -f /etc/passwd
```

```
# ausearch -f /etc/passwd | less  
# ausearch -f /etc/passwd -i | less
```

Onde:

- ▣ * **-f /etc/passwd**: apenas procura por esse arquivo;
- ▣ * **-i**: interpreta entidades numéricas no texto. Por exemplo, UIDs são convertidos para nomes de usuário.

Saída:

```
type=PATH msg=audit(03/16/2007 14:52:59.985:55) : name=/etc/passwd  
flags=follow,open  
  
inode=23087346 dev=08:02 mode=file,644 uid=root ogid=root rdev=00:00  
  
type=CWD msg=audit(03/16/2007 14:52:59.985:55) : cwd=/webroot/home/  
lighttpd  
  
type=FS_INODE msg=audit(03/16/2007 14:52:59.985:55) : inode=23087346  
inode_uid=root  
  
inode_gid=root inode_dev=08:02 inode_rdev=00:00  
  
type=FS_WATCH msg=audit(03/16/2007 14:52:59.985:55) : watch_  
inode=23087346 watch=passwd  
  
filterkey=password-file perm=read,write,append perm_mask=read  
  
type=SYSCALL msg=audit(03/16/2007 14:52:59.985:55) : arch=x86_64  
syscall=open success=yes  
  
exit=3 a0=7fbffffcb4 a1=0 a2=2 a3=6171d0 items=1 pid=12551  
auid=unknown(4294967295)  
  
uid=lighttpd gid=lighttpd euid=lighttpd suid=lighttpd fsuid=lighttpd  
egid=lighttpd  
  
sgid=lighttpd fsgid=lighttpd comm=grep exe=/bin/grep
```

Como deve-se analisar a saída:

- ▣ * **audit(03/16/2007 14:52:59.985:55)**: data do registro da auditoria;
- ▣ * **uid=lighttpd gid=lighttpd**: UIDs e GIDs convertidos para formato de texto. Passando a opção **-i**, quase todos os dados numéricos são convertidos para dados humanamente legíveis. No exemplo, o usuário lighttpd usou o comando *grep* para abrir o arquivo;
- ▣ * **exe="/bin/grep"**: comando utilizado para acessar o arquivo */etc/passwd*;
- ▣ * **perm_mask=read**: o arquivo foi aberto para leitura;

Verifica-se que a partir dos arquivos de log do Framework Audit podemos rastrear quem consultou um arquivo com grep ou gravou alterações utilizando o editor vi/vim. Os logs proveem grande quantidade de informações. Precisamos ler as páginas de manual (man pages) e demais documentações para entender o formato de log “cru”.



Outros exemplos úteis

Pesquisar eventos com data e hora específica. Se a data for omitida, a data de hoje é assumida. Se a hora for omitida, “agora” é assumido por padrão. Use a notação de 24 horas em vez da notação AM/PM para especificar a hora. Exemplo: data 24/20/05, hora 18:00:00.

```
# ausearch -ts today -k password-file  
# ausearch -ts 3/12/07 -k password-file
```

Pesquisar por um evento casando com o utilitário especificado utilizando a opção -x. Por exemplo, verificar quem acessou o arquivo */etc/passwd* usando rm.

```
# ausearch -ts today -k password-file -x rm  
# ausearch -ts 3/12/07 -k password-file -x rm
```

Pesquisar por um evento de determinado usuário pelo UID. Verificar se o usuário neo (UID 876) tentou acessar o arquivo */etc/passwd*:

```
# ausearch -ts today -k password-file -x rm -ui 876  
# ausearch -k password-file -ui 506
```

Geração de relatórios com a ferramenta AUREPORT.

Exemplos de consultas à base de registros do Framework Audit com aureport, de extração de informações sobre a contabilização:

```
# aureport -if /var/log/audit/audit.log  
  
Summary Report  
=====  
Range of time in logs: 27-09-2010 11:24:46.596 - 28-09-2010  
13:16:04.690  
Selected time for report: 27-09-2010 11:24:46 - 28-09-2010  
13:16:04.690  
Number of changes in configuration: 0  
Number of changes to accounts, groups, or roles: 5254  
Number of logins: 0  
Number of failed logins: 0  
Number of authentications: 12  
Number of failed authentications: 729  
Number of users: 12  
Number of terminals: 5  
Number of host names: 21  
Number of executables: 7
```

```
Number of files: 0
Number of AVC's: 0
Number of MAC events: 0
Number of failed syscalls: 0
Number of anomaly events: 0
Number of responses to anomaly events: 0
Number of crypto events: 0
Number of keys: 0
Number of process IDs: 7617
Number of events: 17661
```

```
# aureport --failed

Failed Summary Report
=====
Range of time in logs: 23-09-2009 05:41:55.438 - 28-09-2009 13:20:56.660
Selected time for report: 23-09-2009 05:41:55 - 28-09-2009 13:20:56.660
Number of changes in configuration: 0
Number of changes to accounts, groups, or roles: 25895
Number of logins: 0
Number of failed logins: 0
Number of authentications: 0
Number of failed authentications: 6288
Number of users: 0
Number of terminals: 5
Number of host names: 45
Number of executables: 3
Number of files: 0
Number of AVC's: 0
Number of MAC events: 0
Number of failed syscalls: 0
Number of anomaly events: 0
Number of responses to anomaly events: 0
```



```
Number of crypto events: 0  
Number of keys: 0  
Number of process IDs: 20681  
Number of events: 32212
```

```
# aureport --success  
  
Success Summary Report  
=====  
Range of time in logs: 23-09-2009 05:41:55.438 - 28-09-2009  
13:29:34.423  
Selected time for report: 23-09-2009 05:41:55 - 28-09-2009  
13:29:34.423  
Number of changes in configuration: 0  
Number of changes to accounts, groups, or roles: 633  
Number of logins: 0  
Number of failed logins: 0  
Number of authentications: 128  
Number of failed authentications: 0  
Number of users: 13  
Number of terminals: 10  
Number of host names: 48  
Number of executables: 8  
Number of files: 0  
Number of AVC's: 0  
Number of MAC events: 0  
Number of failed syscalls: 0  
Number of anomaly events: 0  
Number of responses to anomaly events: 0  
Number of crypto events: 0  
Number of keys: 0  
Number of process IDs: 11190  
Number of events: 58292
```



```
aureport -u --summary
```

User Summary Report

```
=====
total   auid
=====
11560  0
600   3000
80    3209
20    50537
12    3589
10    50549
10    50507
10    50540
10    3572
10    3246
10    3097
10    3270
2     3154
```

Saída parcial:

```
aureport -e
```

Event Report

```
=====
# date time event type auid success
=====
1. 23-09-2009 05:41:55 4143669 CRED_ACQ -1 yes
2. 23-09-2009 05:41:55 4143670 USER_START -1 yes
3. 23-09-2009 05:41:55 4143671 USER_END -1 yes
4. 23-09-2009 05:41:55 4143672 USER_CMD -1 yes
5. 23-09-2009 05:42:56 4143673 CRED_ACQ -1 yes
6. 23-09-2009 05:42:56 4143674 USER_START -1 yes
7. 23-09-2009 05:42:56 4143675 USER_END -1 yes
```



```
8. 23-09-2009 05:42:56 4143676 USER_CMD -1 yes
9. 23-09-2009 05:43:34 4143677 CRED_ACQ -1 yes
10. 23-09-2009 05:43:34 4143678 USER_START -1 yes
11. 23-09-2009 05:43:34 4143679 USER_END -1 yes
12. 23-09-2009 05:43:34 4143680 USER_CMD -1 yes
13. 23-09-2009 05:43:34 4143681 CRED_ACQ -1 yes
14. 23-09-2009 05:43:34 4143682 USER_START -1 yes
15. 23-09-2009 05:43:34 4143683 USER_END -1 yes
```

Parcial:

```
# aureport -p

Process ID Report
=====
# date time pid exe syscall auid event
=====
1. 23-09-2009 05:41:55 14397 /usr/bin/sudo 0 -1 4143669
2. 23-09-2009 05:41:55 14397 /usr/bin/sudo 0 -1 4143670
3. 23-09-2009 05:41:55 14397 /usr/bin/sudo 0 -1 4143671
4. 23-09-2009 05:41:55 14397 ? 0 -1 4143672
5. 23-09-2009 05:42:56 14503 /usr/bin/sudo 0 -1 4143673
6. 23-09-2009 05:42:56 14503 /usr/bin/sudo 0 -1 4143674
7. 23-09-2009 05:42:56 14503 /usr/bin/sudo 0 -1 4143675
8. 23-09-2009 05:42:56 14503 ? 0 -1 4143676
9. 23-09-2009 05:43:34 14572 /usr/bin/sudo 0 -1 4143677
10. 23-09-2009 05:43:34 14572 /usr/bin/sudo 0 -1 4143678
11. 23-09-2009 05:43:34 14572 /usr/bin/sudo 0 -1 4143679
12. 23-09-2009 05:43:34 14572 ? 0 -1 4143680
13. 23-09-2009 05:43:34 14574 /usr/bin/sudo 0 -1 4143681
14. 23-09-2009 05:43:34 14574 /usr/bin/sudo 0 -1 4143682
15. 23-09-2009 05:43:34 14574 /usr/bin/sudo 0 -1 4143683
```

```
#aureport -x | head -n 25
```

Executable Report

```
=====
# date time exe term host auid event
=====
1. 23-09-2009 05:41:55 /usr/bin/sudo ? hm1208 -1 4143669
2. 23-09-2009 05:41:55 /usr/bin/sudo ? hm1208 -1 4143670
3. 23-09-2009 05:41:55 /usr/bin/sudo ? hm1208 -1 4143671
4. 23-09-2009 05:42:56 /usr/bin/sudo ? hm1208 -1 4143673
5. 23-09-2009 05:42:56 /usr/bin/sudo ? hm1208 -1 4143674
6. 23-09-2009 05:42:56 /usr/bin/sudo ? hm1208 -1 4143675
7. 23-09-2009 05:43:34 /usr/bin/sudo ? hm1208 -1 4143677
8. 23-09-2009 05:43:34 /usr/bin/sudo ? hm1208 -1 4143678
9. 23-09-2009 05:43:34 /usr/bin/sudo ? hm1208 -1 4143679
10. 23-09-2009 05:43:34 /usr/bin/sudo ? hm1208 -1 4143681
11. 23-09-2009 05:43:34 /usr/bin/sudo ? hm1208 -1 4143682
12. 23-09-2009 05:43:34 /usr/bin/sudo ? hm1208 -1 4143683
13. 23-09-2009 05:43:58 /usr/bin/sudo ? hm1208 -1 4143685
14. 23-09-2009 05:43:58 /usr/bin/sudo ? hm1208 -1 4143686
15. 23-09-2009 05:43:58 /usr/bin/sudo ? hm1208 -1 4143687
16. 23-09-2009 05:44:58 /usr/bin/sudo ? hm1208 -1 4143689
17. 23-09-2009 05:44:58 /usr/bin/sudo ? hm1208 -1 4143690
18. 23-09-2009 05:44:58 /usr/bin/sudo ? hm1208 -1 4143691
19. 23-09-2009 05:45:01 /usr/sbin/crond cron ? -1 4143693
20. 23-09-2009 05:45:01 /usr/sbin/crond cron ? -1 4143694
```

```
# aureport -u | tail
```

```
/sbin/audispd permissions should be 0750
84. 09/23/2009 14:48:31 -1 ? ? ? 3
85. 09/23/2009 14:53:00 -1 (none) ? /usr/lib/virtualbox/VirtualBox 4
86. 09/23/2009 14:56:22 -1 ? ? ? 5
87. 09/23/2009 14:56:51 -1 (none) ? /usr/lib/virtualbox/VirtualBox 6
88. 09/23/2009 15:16:23 -1 (none) ? /usr/lib/virtualbox/VirtualBox 7
```



```
89. 09/23/2009 15:43:54 -1 ? ? ? 8
90. 09/24/2009 15:17:47 -1 pts8 ? /usr/sbin/tcpdump 9
91. 09/24/2009 15:18:03 -1 pts8 ? /usr/sbin/tcpdump 10
92. 09/28/2009 10:46:01 -1 pts6 ? /usr/sbin/tcpdump 11
93. 09/28/2009 10:51:51 -1 pts6 ? /usr/sbin/tcpdump 12
```

Log Time Range Report

```
=====
/var/log/audit/audit.log.3: 23-09-2009 05:41:55.438 - 24-09-2009
17:41:06.885
/var/log/audit/audit.log.2: 24-09-2009 17:41:06.936 - 26-09-2009
03:10:37.737
/var/log/audit/audit.log.1: 26-09-2009 03:10:37.771 - 27-09-2009
11:24:46.596
/var/log/audit/audit.log: 27-09-2009 11:24:46.596 - 28-09-2009
14:15:25.638
```





Roteiro de Atividades 10

Atividade 10.1 – Hardening em Sistema Linux

Na prática de customização, regras de registro de eventos a partir da syscalls com o Framework Audit. Será necessário que o aluno utilize a máquina virtual preparada como Virtuabox. Dessa forma, será possível aplicar todos os conhecimentos (técnicas e ferramentas) de Hardening propostos até o momento.

Atividade 10.2 – Hardening Linux – Registro de Eventos (logs) e Auditoria

1. Instale e configure o deamon do Audit.
2. Crie regra no `/etc/audit/audit.conf` para auditoria do `/tmp`, registrando qualquer ação do tipo: "escrita, execução e alteração".
3. Crie regra no `/etc/audit/audit.conf` para auditoria do `/root`, registrando qualquer ação do tipo: "escrita, execução e alteração".
4. Crie regra no `/etc/audit/audit.conf` para auditoria para registrar ação do tipo: "escrita, execução e alteração" no respectivos diretórios: `/etc/logrotate/`, `/etc/hosts.allow` `/etc/hosts.deny`, `/etc/hosts`, `/etc/xinetd.d/`, `/etc/passwd` e `/etc/shadow`.
5. Crie uma regra no `/etc/audit/audit.conf` para monitorar execução dos comandos: `shutdown`, `reboot`, `halt`, `shred`, `nc`, `tcpdump` e `nmap`.
6. Configure o Framework Audit para enviar logs para o Syslog.







Doutorando no TIDD-PUC/SP, Mestre em Engenharia da Computação no IPT/USP, Mestre em Engenharia de Redes pelo IPT/USP; Pós-graduado em Administração de Redes Linux pela UFLA-MG; Pós-graduado em Análise de Sistemas e Graduação em Processamento de Dados pela Universidade Mackenzie, atua na área de TI desde de 1997, realizando neste período vários projetos de implantação de serviços de rede e segurança. Atualmente atua como Coordenador do Curso de Redes de Computadores na BANDTEC responsável pela cadeira de Sistemas Operacionais, Computação Distribuída e Centralizada e Segurança e Auditoria. É um evangelista do Software Livre, sendo embaixador Fedora, Proctor BSDA e LPI, também atua como professor convidado responsável por cátedras inerentes a Segurança e Computação Forense em sistema Linux, já tendo a oportunidade de atuar na UFLA/ARL (MG), IBTA (Campinas), UNISALES (ES), Faculdade Pitágoras (Guarapari-ES, Teixeira de Freitas-BA), Universidade Potiguar (RN), ITA (SP), IEASAM (PA), Uniron (RO), FAAR(RO), FACID (PI) entre outras. Sendo idealizador e coordenador dos cursos de Pós Graduação Segurança e Computação Forense nas entidades: Universidade Maurício de Nassau (AL) , Faculdade FACID (PI), Faculdade CET(PI), Faculdade Atual (RR), Iquali/EEMBA (BA); Instrutor autorizado FreeBSDBrasil e convidado da Academia Clavis..

LIVRO DE APOIO AO CURSO

O livro apresenta os conceitos teóricos e práticos sobre a análise de ameaças, minimização de riscos e execução de atividades corretivas identificadas nos servidores de rede e seus serviços básicos, como também, formas de reforço e garantia da segurança para tais equipamentos e serviços. Ao final do curso, o aluno será de capaz de aplicar diversas técnicas, procedimentos e utilizar ferramentas específicas para que os servidores possuam a segurança de seus serviços elevada atendendo os requisitos de segurança necessários. Este livro inclui os roteiros de atividades práticas e o conteúdo dos slides apresentados em sala de aula, apoiando profissionais na disseminação deste conhecimento em suas organizações ou localidades de origem.

