Hardening de Sistemas Linux

O que é, quais são os padrões e como proteger seus sistemas Linux

Lucas Rayan Guerra

Introdução

Em um cenário de ameaças cibernéticas em constante evolução, a segurança de sistemas operacionais tornou-se um pilar fundamental para a resiliência de qualquer organização. O hardening, ou endurecimento de sistemas, é o processo de configurar um sistema para reduzir sua superfície de ataque, eliminando vulnerabilidades e restringindo privilégios. Para sistemas Linux, que sustentam a vasta maioria da infraestrutura de nuvem e servidores corporativos, o hardening não é uma opção, mas uma necessidade crítica.

Esta cartilha técnica foi elaborada para fornecer um guia detalhado e prático sobre o hardening de sistemas Linux, alinhado com os mais rigorosos padrões da indústria, como os Benchmarks do Center for Internet Security (CIS) e os Security Technical Implementation Guides (STIGs) da DISA. Abordaremos desde a segurança do kernel até a configuração de firewalls de host, auditoria e criptografia, oferecendo um roteiro completo para transformar uma instalação padrão do Linux em uma fortaleza digital.

O Que é Hardening e Por Que é Essencial?

O hardening de sistemas é um conjunto de práticas e tecnologias para proteger um sistema contra ataques, minimizando sua superfície de ataque. Uma superfície de ataque é o conjunto de todos os pontos (vetores de ataque) onde um invasor pode tentar entrar ou extrair dados de um ambiente. Quanto menor a superfície de ataque, mais seguro é o sistema.

O Problema da Configuração Padrão

A maioria das distribuições Linux é otimizada para facilidade de uso e compatibilidade, não para segurança máxima. Uma instalação padrão geralmente inclui:

- **Serviços desnecessários:** Serviços de rede que não são essenciais para a função do servidor, mas que abrem portas e criam vetores de ataque.

- **Permissões permissivas:** Contas de usuário e arquivos com permissões mais amplas do que o necessário.

- **Protocolos obsoletos:** Protocolos de criptografia fracos ou inseguros que são mantidos por razões de compatibilidade.

Os Pilares do Hardening de Linux

Uma estratégia de hardening eficaz se baseia em múltiplos controles de segurança sobrepostos, um conceito conhecido como defesa em profundidade. Se uma camada de defesa falhar, outras estarão em vigor para impedir o avanço do atacante.

Pilares do Hardening de Linux		
Pilar	**Descrição Técnica**	**Impacto na Segurança**
Funcionalidade Mínima	Remover todos os pacotes, serviços e protocolos que não são estritamente necessários para a função do servidor.	Reduz drasticamente a superfície de ataque, eliminando vulnerabilidades latentes e simplificando a gestão de patches.
Princípio do Menor Privilégio	Conceder a cada usuário, processo e aplicação apenas as permissões mínimas necessárias para realizar sua função.	Limita o dano que um atacante pode causar se uma conta ou processo for comprometido.
Defesa em Profundidade	Implementar múltiplos controles de segurança em diferentes camadas do sistema (kernel, rede, aplicação, dados).	Garante que a falha de um único controle não resulte em um comprometimento total do sistema.
Monitoramento Contínuo	Auditar e registrar todas as atividades relevantes do sistema para detectar e responder a atividades suspeitas.	Fornece a visibilidade necessária para identificar ataques em andamento e realizar análises forenses.

Padrões Globais e Frameworks de Conformidade

Para guiar os esforços de hardening, a indústria de segurança cibernética consolidou padrões técnicos validados por especialistas. Os dois mais importantes são os Benchmarks do **Center for Internet Security (CIS)** e os **Security Technical Implementation Guides (STIGs)** da Defense Information Systems Agency (DISA).

CIS Benchmarks

Os CIS Benchmarks são guias de configuração de segurança desenvolvidos através de um processo de consenso global. Eles oferecem recomendações prescritivas para mais de 25 famílias de produtos, incluindo as principais distribuições Linux (Ubuntu, Debian, RHEL).

- **Nível 1 (Level 1):** Configurações de segurança essenciais que fornecem um benefício claro e têm baixo impacto na funcionalidade. Recomendado para todos os sistemas.

- **Nível 2 (Level 2):** Configurações para ambientes de alta segurança, onde a proteção é primordial, mesmo que isso implique em maior sobrecarga de gerenciamento ou restrições funcionais.

DISA STIGs

Os STIGs são guias de configuração mandatórios para sistemas do Departamento de Defesa dos EUA (DoD) e outras agências federais. Eles são mais rigorosos que os CIS Benchmarks e focam em ambientes de missão crítica. As vulnerabilidades são categorizadas por severidade:

- **CAT I (Categoria I):** Risco máximo. Permite acesso imediato ao sistema ou privilégios de superusuário.

- **CAT II (Categoria II):** Risco médio. Potencial para escalonamento de privilégios ou acesso não autorizado.

- **CAT III (Categoria III):** Risco baixo. Exposição de informações ou falhas operacionais.

Automação da Conformidade com OpenSCAP

Verificar manualmente a conformidade com centenas de controles é impraticável. O OpenSCAP é uma suíte de ferramentas de código aberto que automatiza a verificação de conformidade. Ele utiliza o protocolo SCAP (Security Content Automation Protocol) para escanear um sistema e compará-lo com as políticas definidas nos benchmarks CIS ou STIGs, gerando relatórios detalhados de conformidade e scripts de remediação.

Segurança do Kernel e Parametrização via sysctl

O kernel é o coração do sistema operacional. Seu endurecimento é crucial para mitigar ataques de exploração de memória e fortalecer a pilha de rede. O utilitário ***sysctl*** permite modificar parâmetros do kernel em tempo de execução.

Parâmetro *sysctl*	Valor Recomendado	Função de Segurança
kernel.randomize_va_space	2	Ativa o ASLR (Address Space Layout Randomization) completo para dificultar ataques de exploração de memória como buffer overflow.
kernel.kptr_restrict	2	Oculta endereços de ponteiros do kernel de usuários não privilegiados, dificultando a criação de exploits.
net.ipv4.conf.all.rp_filter	1	Ativa a validação de endereço de origem (anti-spoofing), mitigando ataques que falsificam o IP de origem.

Parâmetro *sysctl*	Valor Recomendado	Função de Segurança
net.ipv4.conf.all.accept_redirects	0	Recusa mensagens de redirecionamento ICMP, que podem ser usadas em ataques de Man-in-the-Middle.
fs.protected_fifos	2	Impede a criação de FIFOs (pipes nomeados) em diretórios com permissões de escrita para todos, um vetor de ataque comum.

Para aplicar essas configurações de forma persistente, crie um arquivo em ***/etc/sysctl.d/99-hardening.conf*** com os parâmetros desejados e execute ***sysctl -p***.

Autenticação Robusta com PAM

O **Pluggable Authentication Modules (PAM)** é um framework modular que gerencia a autenticação de usuários e políticas de sessão. O endurecimento do PAM é essencial para impor políticas de senhas fortes e mecanismos de bloqueio de contas.

Políticas de Senha Fortes

Utilize o módulo ***pam_pwquality.so*** para impor requisitos de complexidade de senha. Edite o arquivo ***/etc/security/pwquality.conf***:

```
minlen = 14
dcredit = -1
ucredit = -1
lcredit = -1
ocredit = -1
```

Isso exige senhas de no mínimo 14 caracteres, com pelo menos um dígito, uma letra maiúscula, uma minúscula e um caractere especial.

Bloqueio de Contas

Use o módulo ***pam_faillock.so*** para bloquear contas após tentativas de login falhas. Adicione as seguintes linhas ao início dos arquivos em ***/etc/pam.d/*** (como ***system-auth*** e ***password-auth***):

```
auth required pam_faillock.so preauth silent audit deny=3 unlock_time=900
auth [success=1 default=bad] pam_unix.so
auth [default=die] pam_faillock.so authfail audit deny=3 unlock_time=900
```

Isso bloqueará a conta por 15 minutos (900 segundos) após 3 tentativas falhas.

Autenticação Multifator (MFA)

A autenticação por senha sozinha não é mais suficiente. Implemente MFA:

- **TOTP (Time-based One-Time Password):** Use o módulo ***libpam-google-authenticator*** para exigir um código de um aplicativo como o Google Authenticator.
- **FIDO2/U2F:** Use o módulo ***pam_u2f.so*** para exigir uma chave de segurança de hardware (como uma YubiKey), oferecendo a proteção mais forte contra phishing.

Configuração Segura do Protocolo SSH

O SSH é a principal interface de gerenciamento e um dos serviços mais atacados. Seu endurecimento é crítico. Edite o arquivo ***/etc/ssh/sshd_config***:

Diretiva *sshd_config*	Valor Recomendado	Objetivo Técnico
PasswordAuthentication	no	Desativa a autenticação por senha, forçando o uso de chaves criptográficas, que são muito mais seguras.

Diretiva *sshd_config*	Valor Recomendado	Objetivo Técnico
PermitRootLogin	no	Impede o login direto do usuário root, forçando o uso de ***sudo*** e criando uma trilha de auditoria clara.
Protocol	2	Força o uso da versão 2 do protocolo SSH, que é muito mais segura que a versão 1.
Ciphers, **KEXs**, **MACs**	(listas restritas)	Restringe os algoritmos criptográficos a opções modernas e seguras, como aes256-gcm@openssh.com e Ed25519.
AllowUsers	user1 user2	Restringe o acesso SSH a uma lista explícita de usuários autorizados.

Use Chaves Ed25519: Ao gerar chaves SSH, prefira o algoritmo Ed25519, que é mais seguro e eficiente que o RSA tradicional:

```
ssh-keygen -t ed25519 -a 100
```

Controle de Acesso Obrigatório (MAC)

O Controle de Acesso Discricionário (DAC) padrão do Linux permite que o dono de um arquivo defina suas permissões. Se um processo for comprometido, ele herda as permissões do usuário. O **Controle de Acesso Obrigatório (MAC)** resolve isso, impondo políticas em nível de kernel que restringem o que cada processo pode fazer, independentemente do usuário.

SELinux (Security-Enhanced Linux)

- **Como funciona:** Baseado em rótulos. Cada arquivo, processo e porta de rede tem um contexto de segurança. O kernel bloqueia qualquer ação não permitida pela política carregada.

- **Distribuições:** Padrão em RHEL, CentOS, Fedora.

- **Modos:**
  - **Enforcing:** Bloqueia violações de política.
  - **Permissive:** Registra violações, mas não bloqueia (útil para depuração).
  - **Disabled:** Desativado.

AppArmor (Application Armor)

- **Como funciona:** Baseado em caminhos de arquivo. Define perfis que especificam quais arquivos um programa pode ler, escrever e executar.

- **Distribuições:** Padrão em Ubuntu, Debian, SUSE.

- **Vantagem:** Considerado mais fácil de gerenciar que o SELinux.

Comparação entre o SELinux e o AppArmor		
Característica	**SELinux**	**AppArmor**
Modelo	Baseado em Rótulos (Inodes)	Baseado em Caminhos (Paths)
Granularidade	Extremamente granular	Mais simples, focado em aplicações
Complexidade	Alta	Média
Distribuições	RHEL, Fedora, CentOS	Ubuntu, Debian, SUSE

Firewall de Host com nftables

Um firewall de host é a última linha de defesa da rede. O ***nftables*** é o framework moderno que substitui o antigo ***iptables***.

Exemplo de Ruleset Básico para um Servidor Web (***/etc/nftables.conf***):

```
flush ruleset

table inet filter {
        chain input {
                type filter hook input priority 0; policy drop;

                # Permitir tráfego estabelecido e relacionado
                ct state established,related accept

                # Permitir tráfego de loopback
                iifname "lo" accept

                # Permitir ICMP (ping)
                ip protocol icmp accept

                # Permitir SSH (porta 22)
                tcp dport 22 accept

                # Permitir HTTP e HTTPS
                tcp dport { 80, 443 } accept
        }

        chain forward {
                type filter hook forward priority 0; policy drop;
        }

        chain output {
                type filter hook output priority 0; policy accept;
        }
}
```

Este ruleset implementa uma política de "negar por padrão" (***policy drop***) e permite explicitamente apenas o tráfego necessário.

Auditoria com auditd

O ***auditd*** é o subsistema de auditoria do Linux. Ele registra eventos de segurança com base em regras predefinidas, criando uma trilha de auditoria detalhada.

Exemplos de Regras de Auditoria (***/etc/audit/rules.d/audit.rules***):

```
# Monitorar alterações em arquivos de configuração críticos
-w /etc/passwd -p wa -k identity
-w /etc/sudoers -p wa -k privileges

# Monitorar chamadas de sistema relacionadas à criação de processos
-a always,exit -F arch=b64 -S execve -k exec

# Monitorar tentativas de login falhas
-w /var/log/faillog -p wa -k logins
```

Use ***ausearch*** para pesquisar logs e ***aureport*** para gerar relatórios sumários.

Criptografia de Disco com LUKS e TPM

A criptografia de dados em repouso protege contra o acesso físico não autorizado ao disco.

- **LUKS (Linux Unified Key Setup):** É o padrão para criptografia de disco no Linux. Ele cria um container criptografado que pode ser desbloqueado com uma senha.

- **TPM (Trusted Platform Module):** É um chip de segurança na placa-mãe que pode armazenar chaves criptográficas de forma segura. O LUKS pode ser integrado ao TPM para desbloquear o disco automaticamente na inicialização, mas apenas se o processo de boot não tiver sido adulterado (verificado pelo Secure Boot).

Essa combinação (LUKS + TPM + Secure Boot) garante que os dados permaneçam criptografados e inacessíveis se o disco for roubado ou se um atacante tentar inicializar um sistema operacional não autorizado.

Conclusão

O hardening de sistemas Linux é uma disciplina essencial que transforma um sistema operacional de propósito geral em uma plataforma segura e resiliente. Não é um evento único, mas um processo contínuo de avaliação, configuração, monitoramento e automação. Ao adotar uma abordagem de defesa em profundidade e seguir os padrões da indústria como CIS e STIG, as organizações podem reduzir drasticamente sua superfície de ataque e proteger seus ativos mais críticos contra as ameaças cibernéticas de hoje e de amanhã.

Referências

[1] Center for Internet Security (CIS). (2024). CIS Benchmarks.

[2] Defense Information Systems Agency (DISA). (2024). Security Technical Implementation Guides (STIGs).

[3] OpenSCAP. (2024). OpenSCAP Security Compliance.

[4] Red Hat. (2023). Security Hardening Guide for RHEL.

[5] ArchWiki. (2024). Security.

[6] Netwrix. (2024). Linux Server Hardening and Security Best Practices.

[7] SUSE. (2024). Security and Hardening Guide.

[8] Ubuntu. (2024). Security for Ubuntu.