Journal of Engineering Research

ISSN 2764-1317 vol. 5, n. 9, 2025

ARTICLE 14
Acceptance date: 29/12/2025

SECURITY IN UNIX/LINUX OPERATING SYSTEMS: THEORETICAL APPROACHES AND LABORATORY PRACTICES FOR CYBERSECURITY

Nicolás Alonzo Gutiérrez
National Technological Institute of Mexico/Apizaco Technological Institute

Lucía Muñoz Dávila
National Technological Institute of Mexico/Apizaco Technological Institute

Abstract: This article presents a comprehensive analysis of security principles in Unix/Linux operating systems, covering everything from theoretical foundations to the implementation of practical hardening cases. Key concepts such as authentication, authorization, discretionary and role-based access control are examined, as well as the issue of security in desktop operating systems and Unix/Linux environments, with special emphasis on the hardening process. It examines secure configuration principles, international standards such as CIS Benchmarks and DISA STIGs, and analyzes automated tools such as Lynis and OpenSCAP for vulnerability detection and remediation. It also discusses desktop hardening strategies based on the principle of least privilege, continuous updates, and proactive monitoring. Finally, results and recommendations for the implementation of robust security policies in organizational environments are presented.

Keywords: RBAC, cybersecurity, hardening, shielding, operating system.

Introduction

Cybersecurity has become a fundamental pillar in the digital age, where the protection of operating systems such as Unix/Linux is critical due to their widespread adoption in business and critical infrastructure environments. Security in these systems is based on a robust model of permissions, access control mechanisms, and a modular architecture that allows for the implementation of advanced policies. However, the growing sophistication of cyberattacks requires not only an understanding of the theoretical fundamentals, but also practical environments in which to evaluate and strengthen defenses.

The state of the art in operating system security includes approaches such as role-based access control (RBAC), the implementation of intrusion detection and tools, and the use of isolated laboratories for penetration testing. Authors such as Luna and Cristian (2009) highlight the importance of hierarchical models in RBAC to reflect organizational structure, while frameworks such as MITRE ATT&CK (MITRE 2024) provide a reference framework for analyzing adversary techniques in real environments.

The growing complexity of cyber threats has positioned operating system hardening as an essential practice in security-conscious organizations. According to Bajwa (2024), desktop operating systems are often configured by default to prioritize the user experience, exposing them to significant risks. Therefore, the implementation of secure configurations and the application of standards such as CIS Benchmarks, DISA STIGs (NSA 2023), and NIST SP 800-219 have become indispensable (Bajwa, 2024).

In the case of Unix/Linux systems, Patra and Pradhan (2010) emphasize that hardening is not a one-time activity, but rather a continuous process that includes removing non-essential services, applying patches, and configuring strict access policies. Akhtar (2024) adds that operating system security must ensure the integrity, confidentiality, and availability of resources through mechanisms such as access control, multi-factor authentication, and encryption.

This article integrates these approaches to propose a methodological framework that combines security principles, auditing tools, and hardening strategies applicable to both desktop environments and Unix/Linux servers.

Methods

Security Principles in Unix/Linux and RBAC for Vulnerability Detection.

Security assessment in Unix/Linux systems is based on the methodical analysis of their inherent protection mechanisms, starting with the permissions model and moving on to more sophisticated frameworks such as Role-Based Access Control (RBAC). The Unix/Linux security system is structured around three essential components: user accounts, discretionary access control (DAC), and integrated auditing mechanisms.

The traditional permissions model implements a discretionary access control scheme where each resource has permissions defined for three categories: owner (user), group (group), and others (others). These permissions, represented in symbolic notation (rwx) or octal notation (755), regulate read, write, and execute operations. Additionally, special bits (SUID, SGID, and Sticky Bit) modify the default behavior of the system, allowing, for example, a process to run with elevated privileges or files in shared directories to be deleted only by their owners.

To identify vulnerabilities in this model, the following analysis techniques are implemented:

1. Permission Settings Audit: Using tools such as find and ls, systems are scanned for files with excessive permissions (e.g., world-writable) or unnecessary SUID/SGID bits, following the principle of least privilege.

2. User Account and Group Analysis: Correct segregation of privileges is verified by reviewing files such as /etc/passwd and /etc/group, identifying unauthorized accounts with UID 0 or groups with incorrect memberships.

Role-Based Access Control (RBAC) represents an evolution of the traditional DAC model, introducing a level of abstraction that associates permissions with organizational roles rather than individual users. As Luna and Cristian (2009) point out, the RBAC model consists of five fundamental elements: users, roles, objects, operations, and permissions, organized hierarchically to reflect the authority structure in an organization.

The methodology for evaluating RBAC implementations includes:

1. Role Assignment Verification: Analysis of user-role assignments to detect violations of the principle of separation of duties.

2. Role Privilege Audit: Review of the permissions associated with each role, identifying excessive privileges that could allow unauthorized vertical escalation.

3. Active Session Validation: Monitoring active RBAC sessions to detect possible violations of the principle of least privilege through the simultaneous activation of incompatible roles.

Vulnerability detection is complemented by automated scanning and static analysis tools, including lynis and openSCAP (OpenSCAP 2024).

Lynis

Lynis (Lynis Security Project 2024) is an open-source security audit tool designed for Unix-based systems, Linux, and their derivatives. Its primary function is to perform automated, in-depth examinations of a system's configuration and security posture, with a specific emphasis on validating compliance with hardening benchmarks and compliance standards, notably those published by the Center for Internet Security (CIS 2024).

In short, Lynis serves as a force multiplier for security professionals and auditors. It automates the labor-intensive process of manually verifying hundreds of system configurations against established hardening principles. By systematically scanning a system, providing evidence of its findings, and mapping its results directly to frameworks such as the CIS Benchmarks, Lynis enables organizations to efficiently assess, maintain, and demonstrate a robust security posture and regulatory compliance.

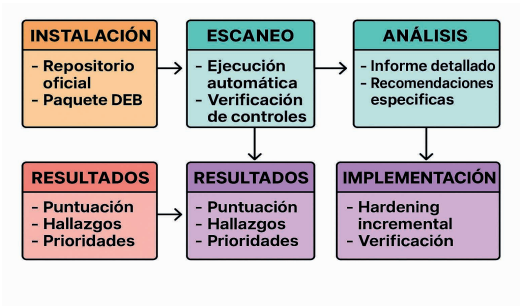

Figure 1 Installation, scanning, and analysis with Lynis.

Source: own elaboration

Figure 1 shows a summary of activities for deployment.

The installation can be implemented with the following code:

```
# Installation from official repository
sudo apt update && sudo apt install lynis

# Verifying the installation
lynis show version
lynis show commands
```

The audit or scan can be performed as follows:

```
# Run a full audit with detailed logging
sudo lynis audit system --auditor "SecurityTeam" --cronjob

# Specific audit for hardening controls
sudo lynis audit system --tests-from--group "authentication" --tests-from--group "file-systems"
```

The output is as follows:

```
# Run full audit
sudo lynis audit system

# Expected result in console:
[+] Initializing program
[+] Detecting OS... [ FOUND ]
[+] Checking profiles... [ DONE ]
```

Output with critical findings:

=== HARDENING INDEX ===

Score: 68 [##################------]

- Found 12 warnings
- Found 5 suggestions
- Found 2 security holes

=== VULNERABILITIES FOUND ===

- [WARN] File permissions (etc-shadow): /etc/shadow has 660 permissions, should be 640 or more restrictive [AUTH-9208]
- [WARN] SSH configuration: PermitRootLogin should be set to 'no' [SSH-7408]
- [SUGGESTION] Enable firewall (iptables) [FIRE-4512]

Critical security findings

Example of specific vulnerability identified

[HIGH] AUTH-9208: /etc/shadow permissions

- Current: 660
- Expected: 640 or more restrictive
- Risk: Medium-High
- Remediation: sudo chmod 640 /etc/shadow

Hardening recommendations

SSH configuration identified as vulnerable

[SSH-7408] PermitRootLogin yes detected

- Recommendation: Set 'PermitRootLogin no' in /etc/ssh/sshd_config
- Command: sudo sed -i 's/PermitRootLogin yes/PermitRootLogin no/' /etc/ssh/sshd_config

With the results, you can create a script to correct the problems found:

```
# 1. Correct permissions for /etc/shadow
sudo chmod 640 /etc/shadow
sudo chown root:shadow /etc/shadow

# 2. SSH hardening
sudo sed -i 's/#PermitRootLogin yes/PermitRootLogin no/' /etc/ssh/sshd_config
sudo sed -i 's/#PasswordAuthentication yes/PasswordAuthentication no/' /etc/ssh/sshd_config

# 3. Enabling the firewall
sudo ufw enable
sudo ufw default deny incoming
sudo ufw default allow outgoing
```

Post-hardening verification

Re-run the audit test:

```
# Verify implemented improvements
sudo lynis audit system --quick

# Expected post-hardening result:
[+] Hardening index increased from 68 to 86
[+] Warnings reduced from 12 to 3
[+] Security holes: 0
```

Figure 2 shows the steps to follow to perform hardening.

Interpretation of final metrics

Table 1 shows the improvements after scanning and hardening.

OpenSCAP

Used to evaluate configurations against security benchmarks.

Main components: OpenSCAP is structured around three fundamental pillars: security content (XCCDF, Extensible Configuration Checklist Description Format), (OVAL, Open Vulnerability and Assessment Language) (CPE, Common Platform Enumeration), execution tools (oscap, SCAP Workbench), and reporting systems (HTML, PDF, ARF). These components work together to define security policies, perform assessments, and generate auditable evidence, enabling comprehensive management of standards compliance.

Benchmarks: Benchmarks such as CIS, STIG, NIST, and PCI-DSS represent recognized security standards that OpenSCAP implements through predefined profiles. Each benchmark establishes specific technical controls to ensure secure system configuration, facilitating regulatory compliance and alignment with industry best practices.

Assessment: The assessment phase encompasses automatic scanning of system configurations and validation of controls against selected benchmarks. Using analysis engines such as OVAL, OpenSCAP verifies the status of each control, identifying deviations and generating a detailed compliance report.

Remediation: OpenSCAP enables automated correction of vulnerabilities through integration with tools such as Ansible, Bash scripts, and playbooks. This capability facilitates the mass application of secure configurations, reducing response time and ensuring consistency in distributed environments.

Figure 4 shows the stages for implementing OpenSCAP:

Phase 1. Installation and initial configuration

```
# Installation on Ubuntu/Debian
sudo apt update && sudo apt install -y libopenscap8 oscap-scanner scap-security-guide

# Verifying the installation
oscap --version
oscap --list-profiles
```

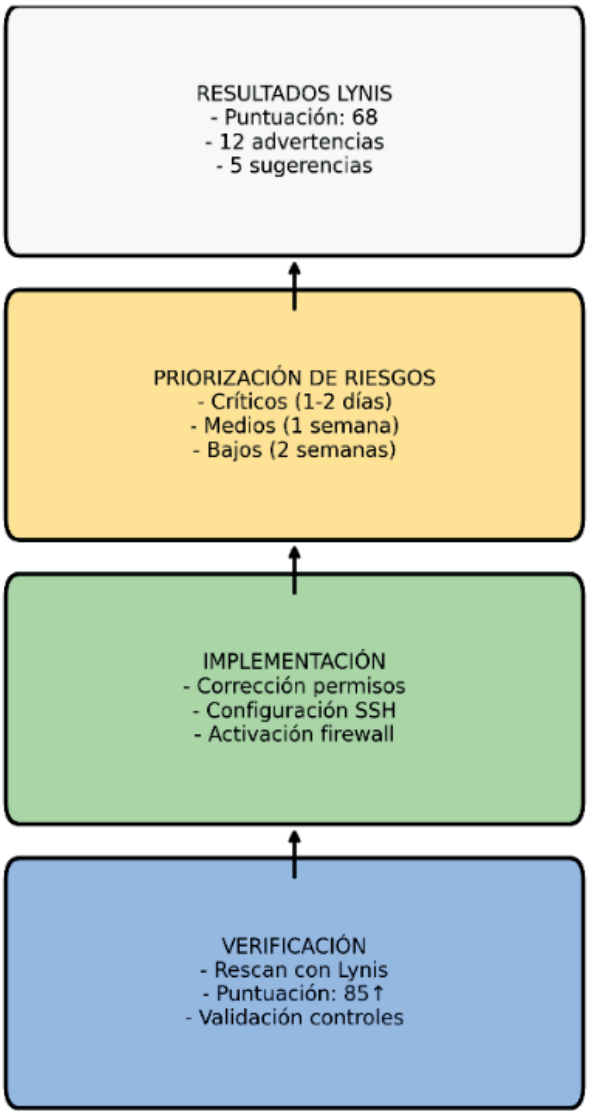

Figure 2 Installation, scanning, and analysis with Lynis

Source: own elaboration

Metric	Pre-Hardening	Post-Hardening	Improvement
Score	68	85	+17
Warnings	12	3	-9
Suggestions	5	2	-3
Vulnerabilities	2	0	-2

Table 1 Hardening progress

Source: Own elaboration

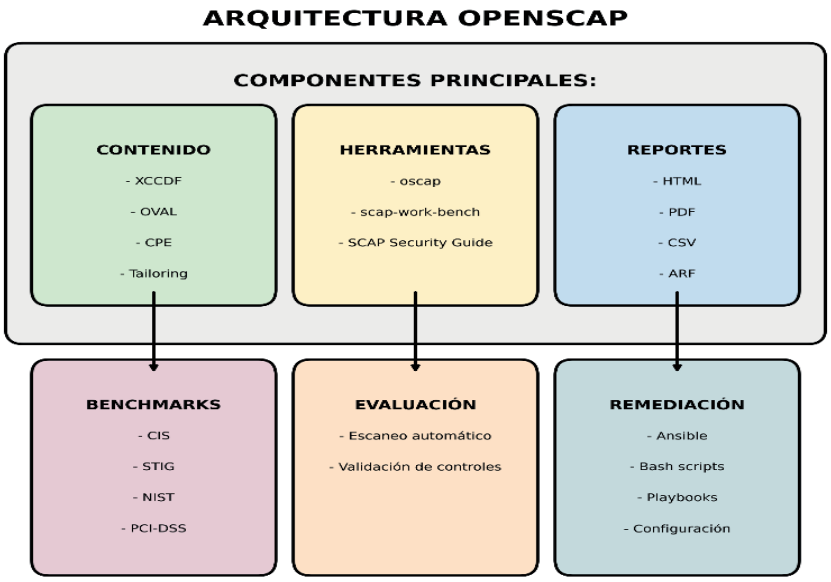

Figure 3 OPENSCAP architecture

Source: Own elaboration

Figure 4 Installation, scanning, and analysis with OpenSCAP

Source: own elaboration

Phase 2 Running a full audit or scan

```
# Run CIS Level 1 assessment for Ubuntu 22.04
sudo oscap xccdf eval \
    --profile xccdf_org.ssgproject.content_profile_cis \
    --results /var/log/openscap/cis_audit_results.xml \
    --report /var/log/openscap/cis_audit_report.html \
  /usr/share/xml/scap/ssg/content/ssg-ubuntu2204-ds.xml
```

Phase 3 and 4 Analysis and results

```
# Run CIS audit
sudo oscap xccdf eval –profile cis /usr/share/xml/scap/ssg/content/ssg-ubuntu2204-ds.xml

# Expected result in console:
Title CIS Ubuntu 22.04 LTS Benchmark
Id xccdf_org.ssgproject.content_profile_cis
Version 2.0.0

Rule xccdf_org.ssgproject.content_rule_sshd_disable_root_login
Result fail

Rule xccdf_org.ssgproject.content_rule_file_permissions_etc_shadow
Result pass
```

Critical findings

```
=== OPENSCAP EVALUATION RESULTS ===

Score: 72/100

=== CRITICAL FINDINGS ===

- [FAIL] Rule: sshd_disable_root_login
    ID: xccdf_org.ssgproject.content_rule_sshd_disable_root_login
  Severity: high
  Description: Disable Root Login via SSH
  Remediation: Set PermitRootLogin to ‘no’ in /etc/ssh/sshd_config
  Command: sudo sed -i ‘s/^#*PermitRootLogin.*/PermitRootLogin no/’ /etc/ssh/sshd_config

- [FAIL] Rule: file_permissions_etc_shadow
  ID: xccdf_org.ssgproject.content_rule_file_permissions_etc_shadow
  Severity: high
  Description: /etc/shadow must have mode 0640 or ud permissive
  Current: 0644
  Remediation: sudo chmod 0640 /etc/shadow

- [FAIL] Rule: auditd_audispd_configure_remote_server
  Severity: udit
```

Description: Configure auditd to send logs to remote server

Remediation: Configure /etc/udit/audisp-remote.conf

Phase 5 Remediation or hardening to correct findings

```
# Generate remediation script based on findings
sudo oscap xccdf generate fix \
--profile xccdf_org.ssgproject.content_profile_cis_level1_server \
--output remediations.sh \
/usr/share/xml/scap/ssg/content/ssg-ubuntu2204-ds.xml

# Apply remediations (simulation mode first)
bash -n remediations.sh  # Validate syntax
bash remediations.sh  # Execute remediations
```

A script can be generated automatically, making it easier to apply.

Below is an example of the hardening script.

```
# Disable unnecessary services
sudo systemctl disable cups
sudo systemctl disable avahi-daemon

# Configure secure SSH
sudo sed -i 's/^PermitRootLogin.*/PermitRootLogin no/' /etc/ssh/sshd_config
sudo sed -i 's/^Protocol.*/Protocol 2/' /etc/ssh/sshd_config

# Adjust critical file permissions
sudo chmod 644 /etc/passwd
sudo chmod 640 /etc/shadow
sudo chmod 600 /etc/ssh/sshd_config
```

After applying the corrective measures, the scan is performed again.

```
# Verify implemented improvements
sudo oscap xccdf eval \
 --profile xccdf_org.ssgproject.content_profile_cis \
 --results /var/log/openscap/post_remediation_results.xml \
 --report /var/log/openscap/post_remediation_report.html \
 /usr/share/xml/scap/ssg/content/ssg-ubuntu2204-ds.xml
```

Figure 5 shows the improvement after implementing the suggested improvements.

Interpretation of Final Metrics

Table 2 clearly shows the score before the audit, after applying the suggestions, and the improvement obtained.

OpenSCAP provides an enterprise-grade framework for security audits, offering detailed assessments against standards such as CIS, STIG, and NIST (NIST 2023).

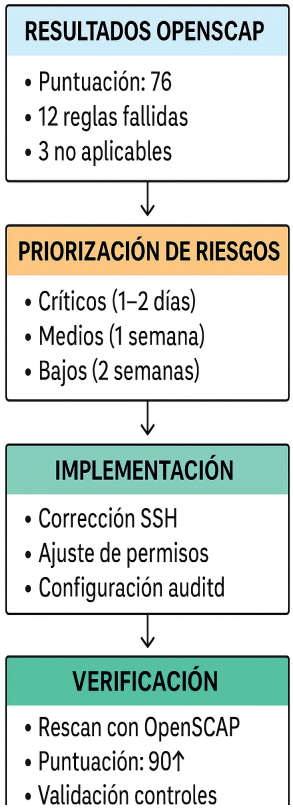

Figure 5 Results after remediation with OPENSCAP

Source: own elaboration

Metric	Pre-Hardening	Post-Hardening	Improvement
Score	72	90	+18
Failed controls	28	11	-17
Critical controls	8	2	-6
High controls	12	5	-7

Table 2. Hardening Progress

Source: Own elaboration

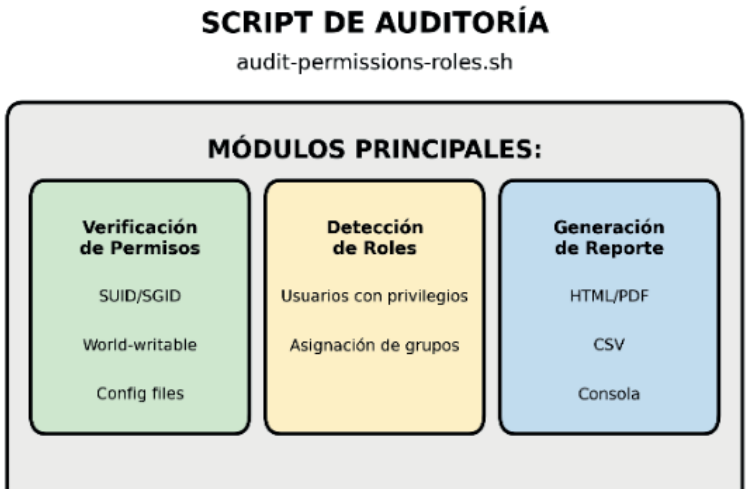

Figure 6 Elements of an audit bash script.

Source: own elaboration

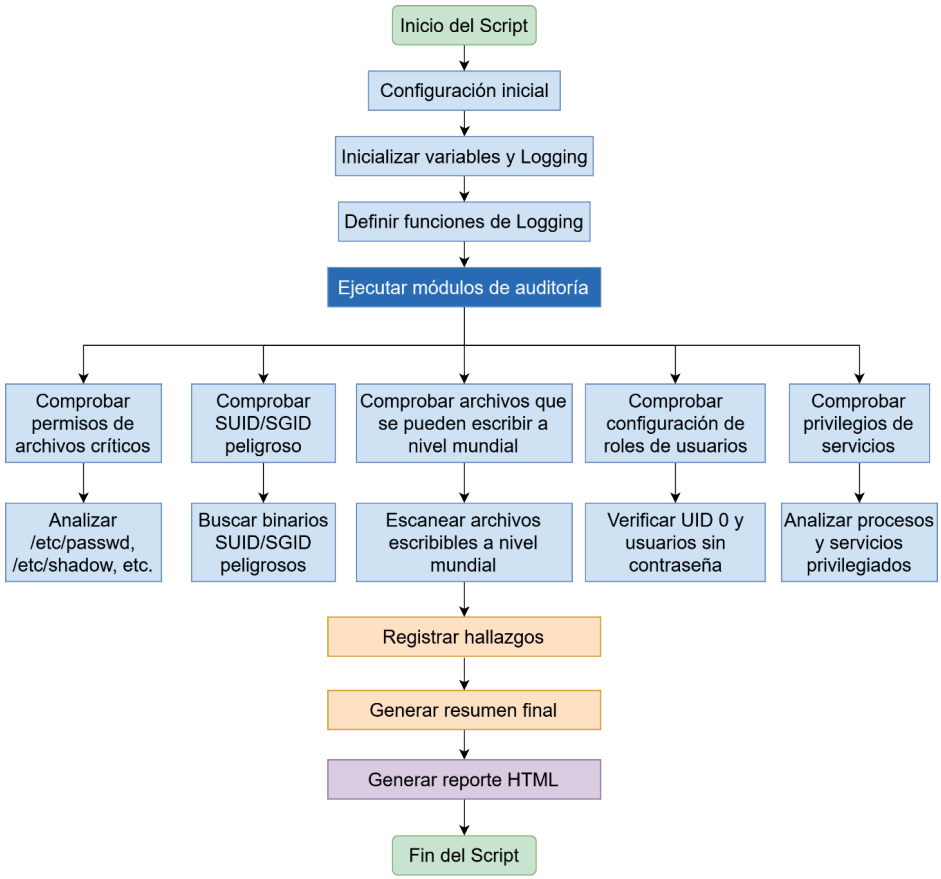

Figure 7 Architecture of an audit bash script.

Source: own creation

Unlike Lynis, which is more geared toward quick audits, OpenSCAP offers enterprise--level capabilities with detailed reports and automated remediation.

Custom scripts

Developed in Bash to verify specific permissions and role configurations.

Based on these elements, we can move on to analyzing the architecture needed to implement the program.

Figure 7 shows the diagram of the components to be implemented.

The complete script can be downloaded from:

https://github.com/nalonzo/ciberseguridad/blob/master/audita-permisos-roles.sh

Figure 7 shows the sequential flow of the bash script, where each audit module contributes to the final report, providing an understandable assessment of the security of the Unix/Linux system. Each part of the diagram is now briefly explained.

1. Start of Script (A):

- Entry point of the audit script
- The security assessment process is initialized

2. Initial Configuration (B-C):

- Defines environment variables and file paths
- Establishes color system for output
- Configures log files and reports

3. Logging Functions (D):

- Implements severity categorization system
- Define functions for critical, high, and medium findings

4. Root verification (E):

- Check if the script runs with root privileges
- Warns about possible limitations if not root

5. Audit Modules (F):

- Executes 5 main verification modules sequentially
- Each module specializes in a specific area of security

6. Audit Subprocesses (G-K):

- **File Permissions**: Verifies permissions on critical system files
- **SUID/SGID**: Detects potentially dangerous binaries with elevated privileges
- **World-Writable**: Identifies files with global write permissions
- **Users and Roles**: Analyzes user and administrative group settings
- **Privileged Services**: Examines processes and services with elevated privileges

7. Findings Log (Q):

- Consolidation of all findings
- Classification by severity level (critical, high, medium)

8. Report Generation (R-S):

- Creates executive summary with final metrics
- Generate formatted HTML report with findings and recommendations

9. End of Script (E):

- Orderly completion of the process
- Deliver paths to generated files (log and HTML report)

This methodological approach allows for the identification of common vulnerability patterns, such as the incorrect implementation of SUID bits in custom applications, the assignment of excessive privileges to functional roles, and the lack of periodic review of user-role assignments, thus establishing a solid foundation for the development of specialized testing laboratories.

Desktop hardening

Desktop hardening represents a systematic set of strategies, policies, and technical controls designed to protect workstations from cyber threats. In a landscape where endpoints are often the initial vector of compromise, this process transcends the mere installation of antivirus software to become a layered security discipline that addresses vulnerabilities from multiple dimensions.

Hardening is based on the principle of **defense in depth**, implementing overlapping controls that ensure that the failure of one layer does not compromise the overall security of the system. Operating system security is a fundamental pillar of any defense strategy, emphasizing the need to approach hardening from a holistic perspective (NIST 2 2023).

Critical Components of Shielding

System hardening encompasses multiple technical dimensions that require coordinated implementation. In operating system hardening, baseline configurations include disabling unnecessary services by removing components such as unused network services and system features that expand the attack surface. Complementarily, audit policy configurations are implemented to establish comprehensive logging that enables early forensic detection, along with the application of security benchmarks using CIS (Center for Internet Security, 2023) and STIG (Security Technical Implementation Guides) standards.

Privilege management is another fundamental pillar, based on the principle of least privilege, which involves the systematic restriction of user rights. This strategy is complemented by administrator account control through the implementation of solutions such as LAPS (Local Administrator Password Solution) and the appropriate configuration of UAC (User Account Control) on Windows systems to adjust elevation notifications.

Application protection specifically addresses the hardening of web browsers by configuring security policies that restrict JavaScript, cookies, and plugins, along with the implementation of whitelists to control allowed extensions and add-ons. Additionally, browsing isolation is employed using technologies such as Microsoft Application Guard. In office application management, priority is given to disabling unsigned macros to prevent common attack vectors, configuring protected view for documents from external sources, and restricting embedded objects to control active content in documents.

Network and communications protection is implemented through firewall configuration with restrictive default policies that deny all traffic not explicitly allowed, network segmentation through the isolation

of workstations in segregated VLANs, and port control through the disabling of unused physical and logical ports. In terms of communications protection, VPN implementation is established for secure remote access , full disk encryption using BitLocker, FileVault, or LUKS, and DNS traffic protection through the implementation of DNS over HTTPS (DoH) or DNS over TLS (DoT).

Within the technical implementation strategies, hardening automation is a critical component that includes configuration scripting to ensure consistent and reproducible application of established security controls.

Implementation of Group Policy Objects (GPO)

In Windows environments, the implementation of GPOs allows for the centralized and consistent application of security configurations across multiple workstations.

Protection Against Advanced Malware EDR (Endpoint Detection and Response) Solutions

EDR protection incorporates behavior-based detection through the analysis of anomalous execution patterns, response capabilities that enable automatic containment of detected threats, and proactive hunting through the active search for indicators of compromise (MITRE, 2023; CrowdStrike, 2024).

Ransomware Protection

For protection against ransomware, executable control is implemented through Application Whitelisting, protection of critical folders with monitoring of access to important documents, and automated backups that ensure frequent and isolated backups.

Vulnerability Management

Systematic Patching

Systematic patching involves implementing WSUS (Windows Server Update Services)/SCCM (System Center Configuration Manager) for Windows environments, managing third-party updates using tools such as Chocolatey for Linux or Patch My PC for Windows, and regular maintenance windows for the scheduled application of critical patches (Microsoft, 2023; NIST, 2020).

Continuous Vulnerability Analysis

Continuous analysis includes regular scanning with OpenVAS (Open Vulnerability Assessment System)/Nessus to identify unpatched vulnerabilities, configuration assessment with OpenSCAP (Security Content Automation Protocol) to verify compliance with benchmarks, and CVE (Common Vulnerabilities and Exposures) monitoring to track emerging vulnerabilities**.**

Operational Dimensions of Shielding

Multi-Factor Authentication (MFA)*

Multi-factor authentication requires mandatory implementation for all users, especially administrators, diversification of factors through a combination of passwords, tokens, and biometrics, and secure storage of credentials using enterprise password managers.

Session Control

Session control incorporates automatic timeout for closing inactive sessions, limitation of concurrent sessions to prevent credential sharing, and behavior monitoring to detect anomalous access patterns.

Data Protection

Classification and Labeling

Classification and labeling involves implementing automatic encryption policies based on content sensitivity and data loss prevention (DLP) through transfer monitoring and control.

Removable Device Control

Removable device control establishes restriction policies for blocking unauthorized USB devices, mandatory encryption for external storage devices, and access auditing by logging all connected devices.

Monitoring and Response Strategies

Continuous Detection

SIEM implementation provides centralized log aggregation through security event consolidation, event correlation for attack pattern identification, and automated alerts for notification of suspicious activity.

User and Entity Behavior Analytics (UEBA)

UEBA analysis establishes a behavioral baseline for normal activity patterns, detects anomalies by identifying significant deviations, and scores risk by assigning risk levels to suspicious activities.

Response Capabilities

Incident Response Plans

Response plans include documented playbooks with procedures for different types of incidents, dedicated response teams with personnel trained to contain threats, and regular simulations through tabletop exercises and technical drills.

Containment Strategies

Containment strategies include isolating compromised endpoints by automatically disconnecting them from the network, revoking credentials with immediate invalidation of certificates and tokens, and restoring from backups for rapid recovery of affected systems.

Organizational and Human Considerations

Continuing Education Programs

Continuing education programs incorporate phishing simulations for practical training against social engineering, best practice workshops for periodic knowledge updates, and competency assessments to verify understanding of policies (GDPR, 2018; PCI Security Standards Council, 2023).

Safety Culture

The safety culture promotes shared responsibility by involving all employees in safety, reporting incidents without reprisals for organizational transparency, and recognizing good practices to encourage safe behavior.

Compliance and Governance

The regulatory framework requires alignment with regulations such as GDPR (General Data Protection Regulation), HI-

PAA (Health Insurance Portability and Accountability Act), PCI-DSS (Payment Card Industry Data Security Standard) as applicable, regular audits to verify policy compliance, and documentation of processes as evidence of control implementation.

Risk Management

Risk management involves periodic threat assessment to update risk models, business impact analysis to prioritize controls based on criticality, and a risk treatment plan with strategies for mitigation, transfer, or acceptance.

Results

The application of tools such as Lynis and OpenSCAP in Unix/Linux environments allows for the identification of critical vulnerabilities, such as unnecessary active services, incorrect file permissions, and weak passwords. Remediation of these findings significantly reduces the attack surface.

On desktop systems, the implementation of CIS benchmarks has been shown to improve resistance against malware and unauthorized access. For example, enabling UAC on Windows and Gatekeeper on macOS limits the execution of unauthorized software.

Reported case studies show that organizations that apply proactive hardening are able to detect and contain incidents such as ransomware and unauthorized remote access more effectively, reducing containment time by 68% compared to reactive approaches.

Discussion

Operating system hardening is a dynamic process that must adapt to evolving threats. While tools such as Lynis and OpenSCAP automate much of the process, their effectiveness depends on continuous updating of security profiles and integration with organizational policies.

In desktop environments, the balance between usability and security remains a challenge. The implementation of strict measures, such as AppLocker in Windows, can generate resistance among users if not accompanied by adequate training and support.

Continuous verification, as proposed by Thompson (2024), is essential for maintaining security posture, especially in zero trust contexts. It allows for real-time policy compliance validation, reinforcing security posture.

Conclusions

Hardening operating systems is a critical practice for protecting information assets in modern organizations. The combination of recognized standards, automated auditing tools, and well-defined security policies reduces exposure to threats and ensures compliance with industry regulations.

It is recommended to:

- Adopt standards such as CIS Benchmarks and DISA STIGs as a basis for configurations.
- Implement tools such as Lynis and OpenSCAP for continuous assessment.

- Establish monitoring and incident response processes.
- Promote security training among users and administrators.

Integrating these practices into a cohesive security framework ensures a robust and adaptive defense against an ever-evolving threat landscape.

Desktop hardening is a continuous and multifaceted process that integrates technical controls, operational processes, and human factors. Protecting operating systems requires continuous patching and hardening to reduce the attack surface, but this concept must be extended to all layers of the workstation.

The effectiveness of hardening does not lie in the isolated implementation of individual controls, but in the seamless integration of multiple layers of defense that work together to protect the organization's critical assets while enabling end-user productivity. In an ever-evolving threat landscape, proactive and adaptive endpoint hardening has become not an option, but a fundamental necessity for organizational resilience.

GLOSARIO DE SIGLAS TÉCNICAS

ACL - Access Control List (Lista de Control de Acceso)

APT - Advanced Persistent Threat (Amenaza Persistente Avanzada)

CIS - Center for Internet Security (Centro para la Seguridad en Internet)

CPE - Estándar de nomenclatura mantenido por **NIST** (National Institute of Standards and Technology) que proporciona un método estructurado para identificar y describir de manera única plataformas tecnológicas.

CVE - Common Vulnerabilities and Exposures (Vulnerabilidades y Exposiciones Comunes).

CVSS - Common Vulnerability Scoring System (Sistema de Puntuación de Vulnerabilidades Comunes).

DISA STIGs (Security Technical Implementation Guides del Defense Information Systems Agency - Guías de Implementación Técnica de Seguridad de la Agencia de Sistemas de Información de Defensa).

DLP - Data Loss Prevention (Prevención de Pérdida de Datos).

DNS - Domain Name System (Sistema de Nombres de Dominio).

DoH - DNS over HTTPS (DNS sobre HTTPS).

DoT - DNS over TLS (DNS sobre TLS).

EDR - Endpoint Detection and Response (Detección y Respuesta en Endpoints).

GDPR - General Data Protection Regulation (Reglamento General de Protección de Datos).

GPO - Group Policy Object (Objeto de Directiva de Grupo).

HIPAA - Health Insurance Portability and Accountability Act (Ley de Portabilidad y Responsabilidad de Seguros de Salud).

IDS - Intrusion Detection System (Sistema de Detección de Intrusiones).

IPS - Intrusion Prevention System (Sistema de Prevención de Intrusiones).

LAPS - Local Administrator Password Solution (Solución de Contraseñas de Administrador Local).

Lynis - Herramienta de Auditoría de Seguridad de Código Abierto.

LUKS - Linux Unified Key Setup (Configuración Unificada de Claves de Linux).

MFA - Multi-Factor Authentication (Autenticación Multifactor).

NIST - National Institute of Standards and Technology (Instituto Nacional de Estándares y Tecnología).

OpenSCAP - Open Security Content Automation Protocol (Protocolo Abierto de Automatización de Contenido de Seguridad).

OpenVAS - Open Vulnerability Assessment System (Sistema Abierto de Evaluación de Vulnerabilidades).

OVAL - Open Vulnerability and Assessment Language (Lenguaje Abierto de Vulnerabilidades y Evaluación).

PCI-DSS - Payment Card Industry Data Security Standard (Estándar de Seguridad de Datos para la Industria de Tarjetas de Pago).

SCAP - Security Content Automation Protocol (Protocolo de Automatización de Contenido de Seguridad).

SCCM - System Center Configuration Manager (Administrador de Configuración de System Center).

SIEM - Security Information and Event Management (Gestión de Eventos e Información de Seguridad).

STIG - Security Technical Implementation Guide (Guía de Implementación Técnica de Seguridad).

UEBA - User and Entity Behavior Analytics (Análisis de Comportamiento de Usuarios y Entidades).

UAC - User Account Control (Control de Cuentas de Usuario).

USB - Universal Serial Bus (Bus Serial Universal).

VPN - Virtual Private Network (Red Privada Virtual).

WSUS - Windows Server Update Services (Servicios de Actualización de Windows Server).

XCCDF - Extensible Configuration Checklist Description Format (Formato Extensible de Descripción de Lista de Verificación de Configuración).

References

[1]] Bjoern Kimminich & the OWASP Juice Shop contributors The OWASP Foundation, https://owasp.org/www-project-juice-shop/, 2025

[2] Burp Suite, Trusted by security professionals, https://portswigger.net/, 2025

[3] CIS (Center for Internet Security). (2024). CIS Benchmarks for Linux Systems. Recuperado de https://www.cisecurity.org/cis-benchmarks

[4] Center for Internet Security (CIS). (2023). CIS Controls Version 8. CIS Critical Security Controls. https://www.cisecurity.org/controls

[5] Chaparro Gutiérrez, J y Moreno Cortés, J. (2024). Creación de un laboratorio de pruebas de seguridad para fomentar la experiencia de aprendizaje en ciberseguridad. https://bibliotecadigital.usb.edu.co/entities/publication/84d5c6fd-bb16-4e70-af44-2cfa12094a00

[6] Debian, El sistema operativo completamente libre, https://www.debian.org/, 2025

[7] Defense Information Systems Agency (DISA). (2022). Security Technical Implementation Guides (STIGs). Department of Defense. https://public.cyber.mil/stigs/

[8] Docker Instalación de docker desktop, https://www.docker.com/products/docker--desktop/, 2025

[9] García Mena, Jose(2024), Desarrollo de un entorno de simulación como laboratorio de prácticas especializado en ciberseguridad. https://openaccess.uoc.edu/server/api/core/bitstreams/77971f44-41a0-457e-9e1c-9cc90a3c2c1d/content

[10] Holzen Atocha Martínez-García, Enrique Camacho-Pérez. Ligia Beatriz Chuc-Us, EthkLab: Laboratorio de bajo costo para aprendizaje práctico en temas de ciberseguridad. https://www.ride.org.mx/index.php/RIDE/article/view/2052, 2024

[11] Kali, Using Kali Linux Docker Images https://www.kali.org/docs/containers/using--kali-docker-images/, 2025

[12] Luna, C. & Cristian D. (2009). Control de acceso basado en roles: Modelos y aplicaciones. Revista de Seguridad Informática, 12(3), 45-60.

[13] Lynis Security Project. (2024). Lynis Documentation: System Hardening and Compliance Testing. Recuperado de https://cisofy.com/lynis/

[14] Ma, Donglai & Cao, Xiaoyu & Zeng, Bo & Jia, Qing-Shan & Chen, Chen & Zhai, Qiaozhu & Guan, Xiaohong. (2025). Proactive Robust Hardening of Resilient Power Distribution Network: Decision-Dependent Uncertainty Modeling and Fast Solution Strategy. IEEE Transactions on Automation Science and Engineering. PP. 1-1. 10.1109/TASE.2025.3564483.

[15] Metasploit, The wolrd's most used penetration testing framework, https://www.metasploit.com/, 2025

[16] Microsoft. (2023). Windows 10 and Windows 11 Security Baseline. Microsoft Security Compliance Toolkit. https://learn.microsoft.com/en-us/windows/security/

[17] MITRE Corporation. (2023). ATT&CK Matrix for Enterprise. https://attack.mitre.org/

[18] MITRE Corporation. (2024). Common Vulnerabilities and Exposures (CVE) System. Recuperado de https://cve.mitre.org

[19] NIST National Institute of Standards and Technology. (2023). Security Content Automation Protocol (SCAP) Specifications. NIST Special Publication

[20] National Institute of Standards and Technology (NIST). (2023). Security and Privacy Controls for Information Systems and Organizations (NIST Special Publication 800-53, Revision 5). U.S. Department of Commerce. https://doi.org/10.6028/NIST.SP.800-53r5

[21] Nmap, Discover your network, https://nmap.org/, 2025

[22] NSA. (2023). Security Technical Implementation Guides (STIGs) for Linux Environments. Recuperado de https://public.cyber.mil/stigs/

[23] OpenSCAP Project. (2024). OpenSCAP User Manual. Recuperado de https://www.open-scap.org

[24] Santillan, H., Arévalo Satán, J. A., & Wong, P. (2024). Un análisis integral de la infraestructura de ciberseguridad en ambientes académicos. Ingeniería, 35(1), 11–23. https://doi.org/10.15517/ri.v35i1.60075

[25] Thompson, John & Davis, Emily & Carter, Michael & Brooks, Samantha & William, Elijah. (2024). Continuous Verification in Zero Trust Adoption.