Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

An Automated Linux Security Auditing and Hardening Framework

[1]Ahnaf Tahmid Hasan, [2]Md. Moinul Islam

[1,2]Department of CSE, Bangladesh University of Professionals (BUP)

ARTICLE INFO

Received: 25 Feb 2026

Accepted: 11 Apr 2026

Published: 20 Apr 2026

ABSTRACT

Linux systems are increasingly targeted by cyber-attacks due to misconfigurations, outdated packages, and exposed services. Traditional security assessment approaches are often manual, time-consuming, and fragmented across multiple tools. This paper presents a unified automated framework for Linux security auditing and hardening, integrating vulnerability assessment, CVE-based mapping using the National Vulnerability Database, automated remediation, patch management, network port scanning, and real-time security monitoring. The framework is implemented using a hybrid architecture combining Python, Bash scripting, and a Flask-based monitoring interface. It was evaluated across six widely used Linux distributions Ubuntu, Debian, CentOS, Fedora, Kali Linux, and Parrot OS using a systematic before-and-after hardening methodology. Experimental results demonstrate a 60 – 70% average reduction in detected vulnerabilities and misconfigurations, improved security scores, and minimal runtime overhead. Additionally, the framework introduces a closed-loop automated security pipeline that enables continuous assessment and remediation without manual intervention.These findings validate the framework's effectiveness, scalability, and portability, providing a practical solution for proactive Linux system security management. Future work will extend the framework to container and application-level security and incorporate machine learning-based anomaly detection.

Keywords: Linux Security, Automated Auditing, System Hardening, Vulnerability Assessment, CVE, Security Monitoring

I. Introduction

Linux operating systems have become a foundational technology for modern digital infrastructure. They are widely deployed in enterprise servers, virtualized platforms, and large-scale research environments. Their open-source nature, stability, and configurability make Linux an attractive choice for critical applications across sectors such as finance, healthcare, education, and government. However, as reliance on Linux increases, so does the need for robust and continuous security measures.

Linux systems are frequently targeted by cyber threats, including misconfigurations, unpatched vulnerabilities, privilege escalation exploits, and unauthorized access. Such threats can compromise system reliability, data confidentiality, and service availability. Traditional manual security auditing techniques are often slow, error-prone, and unable to adapt to the evolving nature of cyberattacks. Consequently, automated and scalable approaches for security auditing and system hardening are increasingly necessary. In Linux-specific contexts, centralized auditing frameworks provide enhanced system visibility and compliance tracking but typically lack real-time monitoring and automated remediation [1]. Existing security tools and methodologies address isolated aspects of Linux security, but no single solution covers auditing, monitoring, vulnerability assessment, and remediation

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

comprehensively. This has led to the adoption of integrated or multi-tool frameworks that combine complementary functions for more effective protection [3].

Prior research has emphasized structured and automated security architectures for improved vulnerability assessment and informed decision-making. Reference frameworks aggregate metrics from multiple system components, supporting centralized monitoring and automated security analysis[4]. Automation-based hardening strategies, such as integration with configuration management systems, have demonstrated improvements in cloud-based and virtualized Linux deployments; however, these approaches often fail to provide a unified, end-to-end solution for real-time monitoring, patch management, port scanning, and CVE-based vulnerability correlation [6].

The need for unified frameworks is particularly evident in large-scale, programmable infrastructures. Platforms like FABRIC operate as highly dynamic and distributed testbeds, supporting advanced networking and computing research. These environments demand continuous security visibility and rapid vulnerability mitigation to maintain operational reliability and system integrity [10]. Therefore, automated and integrated security solutions are essential for proactive auditing and adaptive hardening in such settings.

This work introduces an automated Linux security auditing and hardening framework inspired by Lynis, extended with integrated real-time monitoring, automated patch management, port scanning, and CVE-based vulnerability assessment.

The novelty of this work lies in the design of a closed-loop automated security orchestration framework that integrates real-time monitoring, CVE-based vulnerability correlation, and automated remediation within a single pipeline, enabling continuous security enforcement without manual intervention. Unlike existing tools that operate independently or require manual intervention, the proposed system combines auditing, vulnerability assessment, CVE correlation, automated remediation, and real-time monitoring within a cohesive and continuously adaptive security lifecycle.

The contributions of this study are summarized as follows:

- Design of a unified and automated Linux security auditing and hardening framework integrating auditing, CVE-based vulnerability assessment, real-time monitoring, and automated remediation.
- Implementation of a closed-loop security architecture leveraging Python, Bash, and Flask, enabling continuous monitoring, vulnerability detection, and automated response.
- Experimental validation across multiple Linux distributions demonstrating significant vulnerability reduction, improved security posture, and minimal performance overhead.

The remainder of the paper is organized as follows. Section II reviews related work in Linux security auditing and hardening. Section III presents the architecture and implementation of the proposed framework. Section IV presents experimental setup and evaluation. Section V presents the results and discussions. Section VI concludes the study and outlines directions for future research.

II. Related Works

Work by [2] introduced an open-source security auditing tool for Unix and Linux systems, providing comprehensive evaluations of kernel parameters, installed packages, running services, and general system configurations. The tool generates detailed reports that identify potential misconfigurations and security weaknesses. While effective in establishing a baseline security posture, its functionality is limited to periodic assessments. The absence of real-time monitoring, automated patch management, port scanning, and CVE-based vulnerability correlation restricts its effectiveness in dynamic or large-scale environments where immediate action is required.

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Automated compliance frameworks based on SCAP standards have also been explored [5]. These frameworks evaluate Linux systems against predefined security policies and benchmarks, producing reports that indicate compliance levels. They have proven valuable in enterprise and research environments for enforcing security baselines and regulatory standards. However, SCAP-based frameworks are primarily designed for scheduled or periodic scans and do not provide continuous monitoring or automated remediation. Administrators must manually integrate the assessment results into operational workflows, increasing overhead and delaying responses to emerging vulnerabilities.Additional work has focused on tools like Lynis [7], which provide structured system auditing combined with guidance for hardening measures. These solutions examine system services, configuration files, and installed packages, generating actionable recommendations. Although Lynis improves security visibility and provides guidance for hardening, it is primarily intended for on-demand or periodic audits. It lacks integrated mechanisms for real-time resource monitoring, automated patching, or continuous vulnerability correlation, which are increasingly necessary for maintaining security in large, dynamic infrastructures.

OpenSCAP-based implementations [8] extend compliance assessment capabilities by incorporating automated evaluation of system configurations against defined policies. These approaches provide a systematic method to measure adherence to best-practice benchmarks. While highly effective in identifying configuration drift and compliance violations, OpenSCAP frameworks do not inherently provide continuous operational monitoring, port scanning, or automated remediation capabilities. Their focus remains on policy evaluation rather than real-time defense, limiting applicability in rapidly changing Linux environments.

File integrity monitoring solutions, such as AIDE [9], detect unauthorized modifications to critical system files by maintaining cryptographic checksums and performing integrity checks at scheduled intervals. These mechanisms are valuable for post-compromise analysis and forensic investigations. However, AIDE operates as a standalone tool, without integration into auditing, patch management, or vulnerability assessment workflows. Its isolated design restricts holistic and automated system hardening, especially in environments with multiple interdependent Linux hosts.

Standardized hardening guidelines, including those provided by DISA [11] and CIS [12], define prescriptive configurations for Linux systems. DISA's Security Technical Implementation Guides establish strict policies suitable for sensitive or government-operated environments, while CIS Benchmarks offer best-practice guidance across common Linux distributions. These frameworks improve overall security posture and provide widely accepted configuration baselines. Nevertheless, their manual implementation or reliance on external tools reduces operational efficiency. Without integration into automated or real-time monitoring frameworks, consistent application of these guidelines remains challenging, particularly in large-scale or frequently updated environments.

Overall, existing studies demonstrate strengths in individual aspects of Linux security: auditing, compliance verification, file integrity monitoring, and prescriptive hardening. However, they generally operate as isolated solutions. Continuous real-time monitoring, automated patch management, port scanning, and CVE-based vulnerability correlation are rarely provided together in a unified framework. This limitation creates a need for an integrated system that consolidates auditing, compliance verification, and automated hardening, thereby reducing administrative overhead, improving proactive detection, and enhancing resilience in both enterprise and experimental Linux environments.

III. Proposed Methodology

Linux systems face frequent cyber threats, including misconfigurations, unpatched vulnerabilities, privilege escalation, and unauthorized access, which can compromise system reliability and data

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

confidentiality. Manual security audits are often slow and error-prone, making automated and scalable auditing and hardening solutions increasingly necessary.

This section presents the design and operational workflow of the Automated Linux Security Auditing and Hardening Framework, which provides a continuous, automated, and unified security solution for Linux-based systems. Unlike traditional tools that focus on periodic assessments, this framework integrates auditing, vulnerability assessment, automated hardening, real-time resource monitoring, port scanning, and CVE-based vulnerability correlation into a single modular pipeline, enabling proactive and scalable system security management.

The overall workflow is illustrated in Fig. 1, showing the sequential and iterative stages of system auditing and hardening.

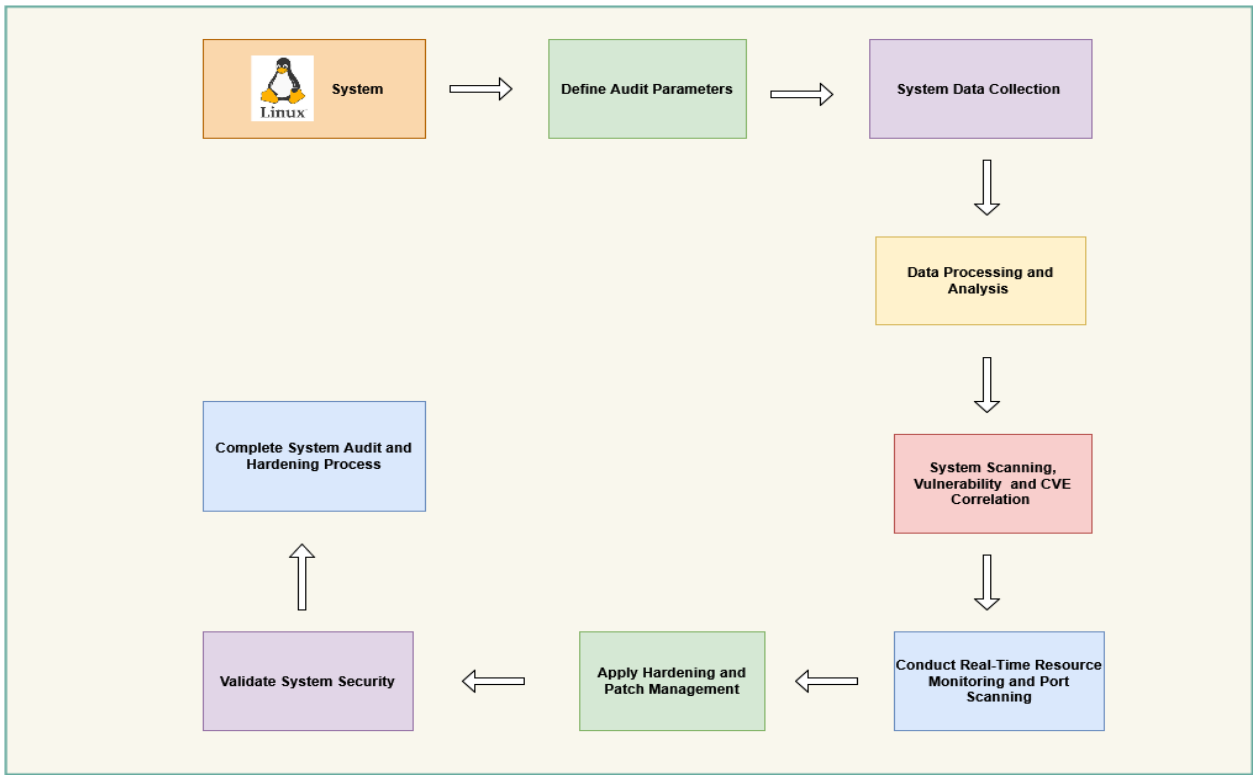

Fig. 1. Sequential and iterative stages of system auditing and hardening

A. Framework Overview

The framework is designed as a modular, closed-loop pipeline, allowing pre- and post-hardening evaluation. Each module operates independently, while results are aggregated through a centralized control layer, ensuring reproducibility, adaptability, and scalability across multiple Linux environments. The major phases include:

- Define Audit Parameters
- System Data Collection
- Data Processing and Analysis
- System Scanning, Vulnerability and CVE Correlation
- Conduct Real-Time Resource Monitoring and Port Scanning
- Apply Hardening and Patch Management

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

- Validate System Security and Report Generation

This architecture ensures measurable security improvements while maintaining system stability and compliance with best practices.

B. Define Audit Parameters

Audit parameters specify which Linux system components are evaluated according to security best practices and organizational policies. Evaluation includes user and group configurations, authentication and password policies, file and directory permissions, running services and daemons, kernel and network security parameters, and installed packages with their update status [13], [14]. These parameters guide subsequent data collection and assessment, ensuring adaptability to different environments.

C. System Data Collection

The framework collects baseline system information using automated scripts and native Linux utilities. Collected data includes OS and kernel version, active processes and services, network interfaces and open ports, installed software packages, system logs and configuration files, and resource usage statistics (CPU, memory, disk, network) [15]. This data provides a comprehensive view of the system's configuration, performance, and security posture, forming input for vulnerability assessment and compliance evaluation.

D. Data Processing and Analysis

Raw data is processed to extract actionable insights, identifying misconfigurations, deviations from security policies, and performance anomalies. These insights guide automated remediation [16].

E. Vulnerability Assessment and CVE Correlation

After system scanning, the framework performs vulnerability assessment by correlating installed packages and running services with CVE sources and the National Vulnerability Database [17]. It identifies known vulnerabilities, misconfigurations, privilege escalation risks, and exposed services.

Package information is extracted using apt, dpkg, and yum, then normalized and matched with NVD data using JSON feeds and API queries. Each identified vulnerability is mapped to corresponding CVE entries from the National Vulnerability Database and prioritized using CVSS scores to enable risk-based automated remediation.

High and critical vulnerabilities trigger immediate remediation, while lower-risk issues are scheduled for later updates. Pre- and post-hardening comparisons measure security improvement.This automated correlation significantly reduces manual effort and improves the accuracy of vulnerability identification.

F. Conduct Real-Time Resource Monitoring and Port Scanning

To enhance proactive security, the framework continuously monitors CPU, memory, disk, and network usage. It tracks active processes to detect anomalous system behavior and monitors open network ports to identify unauthorized or unexpected services [20]. This enables early detection of potential attacks and complements periodic audits, ensuring system resilience.

G. Apply Hardening and Patch Management

Based on assessment results, the framework executes automated hardening actions. These include disabling unnecessary services and open ports, enforcing strong password and authentication policies, applying kernel and network security configurations, updating outdated or vulnerable packages, and

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

securing SSH and remote access configurations [18], [19]. All actions are script-driven to ensure consistency, reduce human error, and maintain audit logs for traceability.

H. Validate System Security and Report Generation

After hardening, the framework validates that applied security measures are effective, previously detected vulnerabilities have been mitigated, and overall system stability and service availability are maintained. A comprehensive security report is generated, documenting pre- and post-hardening vulnerabilities, applied remediation actions, improvements in security score, and compliance status. This report provides administrators with actionable insights and supports ongoing monitoring, forming a closed-loop security lifecycle.

I. Implementation

The framework is implemented using a hybrid architecture combining Python, Bash scripting, and a Flask-based web interface. Python acts as the core orchestration layer, managing auditing, vulnerability analysis, automation, and report generation. Bash scripts handle low-level system operations such as data collection, port scanning, service monitoring, and patch management.

CVE correlation is performed using data from the National Vulnerability Database via JSON feeds and REST APIs, where installed packages are matched against known vulnerabilities and prioritized using CVSS scores.

A Flask-based dashboard provides real-time visualization of system status, including resource usage, detected vulnerabilities, and remediation actions. The framework operates as a standalone host-based system and can be extended to distributed environments using SSH-based execution for scalability.

After completing all security operations, the framework provides a unified execution flow where Python orchestrates high-level decision-making, while Bash scripts handle system-level enforcement. The integrated Flask dashboard ensures continuous visibility of system state, enabling real-time security awareness and simplified administrative control. The modular design of the framework allows easy integration with additional security tools and supports scalability for enterprise-level deployments.The experimental configurations and vulnerability scenarios were applied consistently across all test environments to ensure reproducibility and fairness in evaluation.

IV. Experimental Setup and Evaluation

This section describes the experimental environment and evaluation methodology employed to assess the effectiveness, portability, and performance of the proposed automated Linux security auditing and hardening framework.

A. Experimental Environment

To evaluate the framework under heterogeneous operating conditions, experiments were conducted on six widely used Linux distributions:

- Ubuntu
- Debian
- CentOS
- Fedora
- Kali Linux
- Parrot Security OS

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Each operating system was deployed on a virtualized platform with identical baseline configurations to ensure consistency and fairness in evaluation. The virtual machines were configured with 2 vCPUs, 4–8 GB of RAM, 40–50 GB of storage, and an x86_64 architecture.

Default operating system installations were initially used to reflect standard deployment scenarios. Subsequently, common security weaknesses including outdated software packages, unnecessary running services, weak authentication policies, and exposed network ports were deliberately introduced in a controlled manner to emulate realistic insecure system environments typically observed in production systems.

B. Evaluation

The evaluation followed a systematic before-and-after hardening approach to quantify the security improvements achieved by the proposed framework. Initially, baseline security audits were conducted on each Linux distribution, and all identified vulnerabilities, configuration weaknesses, and exposed network services were documented. The proposed framework was then executed using its default auditing and hardening policies, applying automated remediation, patch updates, service hardening, and configuration enforcement. Post-hardening security audits were subsequently performed using the same assessment criteria to ensure consistency. The effectiveness of the framework was evaluated using several metrics, including the total number of detected vulnerabilities and misconfigurations, the reduction in exposed network ports, improvements in the overall system security score, and runtime performance as well as resource overhead.To ensure reliability, each experiment was conducted three times under identical conditions, and the reported results represent average values. The variation across runs remained within ±2%, indicating consistent framework performance. All experiments were conducted in an isolated virtual environment to ensure reproducibility and eliminate external interference.Additionally, baseline scans using existing tools such as Lynis were conducted for reference. The proposed framework demonstrated improved remediation coverage and faster vulnerability resolution due to its automated hardening capabilities.

V. Results and Discussions

A. Security Improvement Analysis

The experimental results demonstrate that the proposed framework significantly improves the security posture of Linux systems across all tested distributions. Prior to hardening, a high number of vulnerabilities and insecure configurations were detected, particularly in security-oriented distributions such as Kali Linux and Parrot OS, which expose multiple services by default.

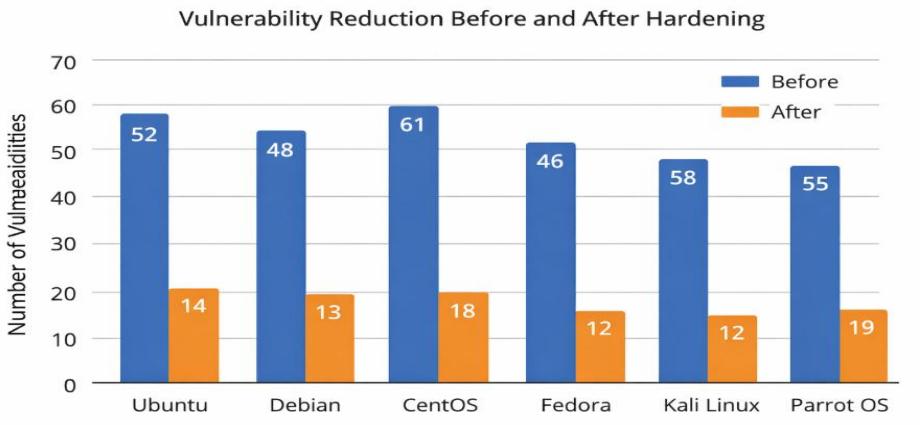

Fig. 2. Vulnerability Reduction Before and After Hardening

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Fig. 2 illustrates the reduction in vulnerabilities across all tested Linux distributions. A significant decrease is observed after applying the proposed framework, with Fedora and Ubuntu showing the highest improvements. Security-focused distributions such as Kali Linux and Parrot OS initially exhibited higher vulnerability counts due to default enabled services, but achieved notable reductions after hardening.

Fedora, Ubuntu, Debian, and CentOS demonstrated the most significant enhancements. Although security-oriented distributions like Kali Linux and Parrot OS initially showed elevated vulnerability levels because of default enabled services, they made substantial progress in reducing these vulnerabilities following hardening procedures.

Table I. Security Comparison of Before and After Hardening

Distribution	Before Hardening	After Hardening	Port Reduction	Security Improvement
Ubuntu	52	14	6 → 2	+68%
Debian	48	13	5 → 2	+66%
CentOS	61	18	7 → 3	+63%
Fedora	46	12	5 → 1	+70%
Kali Linux	58	21	9 → 4	+59%
Parrot OS	55	19	8 → 3	+61%

Overall, the framework achieved an average 60–70% reduction in detected vulnerabilities and misconfigurations across all evaluated distributions.Compared to baseline auditing tools, the proposed framework provides not only detection but also automated remediation, resulting in a more significant reduction in system vulnerabilities

B. Comparison with Existing Tools

To highlight the advantages of the proposed framework, a feature-based comparison with commonly used Linux security tools is presented.

Table II. Feature Comparison with Existing Tools

Feature	Lynis	OpenSCAP	AIDE	Proposed Framework
Automated Auditing	Yes	Yes	Yes	Yes
CVE Mapping	No	Yes	No	Yes

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Automated Hardening	No	No	No	Yes
Real-time Monitoring	No	No	No	Yes
Port Scanning	No	No	No	Yes
Unified Reporting	No	No	No	Yes

Unlike Lynis and OpenSCAP, which focus on periodic auditing and compliance, the proposed framework enables a continuous automated security lifecycle. These tools also require manual remediation and lack real-time monitoring, while AIDE operates independently without integration into vulnerability workflows.

In contrast, the proposed framework integrates auditing, CVE-based vulnerability assessment with the National Vulnerability Database, real-time monitoring, and automated hardening in a unified pipeline, reducing response time and administrative overhead.This demonstrates the advantage of an integrated approach over isolated security tools in dynamic environments.

C. Performance and Overhead Analysis

The runtime impact of the proposed framework was evaluated to ensure its suitability for production environments.

Table III. Runtime Overhead After Hardening

Distribution	CPU Overhead	Memory Overhead	Service Disruption
Ubuntu	3.1%	4.0%	None
Debian	3.3%	4.2%	None
CentOS	4.0%	4.8%	Minor
Fedora	3.0%	3.9%	None
Kali Linux	4.5%	5.1%	Minor
Parrot OS	4.2%	4.9%	Minor

The observed overhead remained within acceptable limits, confirming that the framework enhances system security without significantly impacting performance or availability.

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

VI. Conclusions

.Linux systems are becoming more frequent targets of cyberattacks, mainly due to issues like misconfigurations, outdated software packages, and exposed services. This paper introduces an Automated Linux Security Auditing and Hardening Framework designed to overcome the shortcomings of traditional manual methods and isolated security tools. The framework combines multiple features, including security auditing, vulnerability assessment based on CVEs, automated system hardening, patch management, real-time monitoring, and network port scanning, to deliver a more complete and proactive security approach.

Experimental results conducted on six Linux distributions Ubuntu, Debian, CentOS, Fedora, Kali Linux, and Parrot OS showed a significant reduction in vulnerabilities, improved overall security scores, and only minimal impact on system performance. These findings demonstrate the framework's effectiveness, scalability, and adaptability for enterprise environments as well as large-scale experimental setups. Future improvements will aim to expand support for application-level and container security, along with the integration of machine learning based anomaly detection techniques.

References:

[1] Kumar, S. and Singh, R. (2025). Framework for Security Auditing in Linux. SSRN. Available at: https://ssrn.com/

[2] Balakrishnan, G. (2023). Lynis – Open Source Security Auditing & Pentesting Tool. GBHackers. Available at:https://gbhackers.com/

[3] Rodak, M. (2022). OpenSCAP Report: A Tool for Visualizing Security Compliance. Thesis, Brno University ofTechnology. Available at: https://www.vut.cz/

[4] Ahmed, M., Naqvi, S.A., and Josephs, M. (2016). A Reference Architecture for Security Metric Aggregation and Decision Making. Journal of Information Security and Applications, 29, 1–12.

[5] Caraballo-Vega, J. Automated SCAP Security Compliance. NASA NCCS. Available at: https://www.nccs.nasa.gov/

[6] Singh, A. (2024). Comprehensive VM Hardening Guide using OpenSCAP. Medium. Available at: https://medium.com/

[7] Lynis – Security auditing tool for Unix/Linux systems. https://cisofy.com/lynis/

[8] OpenSCAP – Open Source Security Compliance Solution. https://www.open-scap.org/

[9] AIDE – Advanced Intrusion Detection Environment. https://aide.github.io/

[10] Baldin, Ilya, Anita Nikolich, James Griffioen, Indermohan Inder S. Monga, Kuang-Ching Wang, Tom Lehman, and Paul Ruth, (2019), FABRIC: A National-Scale Programmable Experimental Network Infrastructure. https://fabric-testbed.net/

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

[12] Center for Internet Security (CIS). CIS Benchmarks. https://www.cisecurity.org/cis-benchmarks/

[13] A. T.Deshmukh and P.N. Mahalle, Enhancing Security in Linux OS, Int. J. Comput. Appl., vol.117,no.12,pp.34–37,2015,doi:10.5120/20609-3239.

[14] N. Arora, T. Bhosale, V. Sharma, and J. Supe, Linux Hardening, Int. J. Recent Innov. Trends Comput.Commun., vol. 2, no. May, pp. 1019–1022, 2014.

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

[15] M. Anisetti, C. A. Ardagna, E. Damiani, and F.Gaudenzi, A Security Benchmark for OpenStack,IEEE International Conference on Cloud Computing,CLOUD, vol. 2017-June. pp. 294–301, 2017, doi:10.1109/CLOUD.2017.45.

[16] S. Rahalkar, Network vulnerability assessment :identify security loopholes in your network's infrastructure. Packt Publishing Ltd, 2018, doi:August 31, 2018.

[17] A. K. Nepal, Linux Server & Hardening Security,no.August,pp.0–65,2014,doi:

10.13140/2.1.5079.2329.

[18] Y. Bhardwaj, Server Hardening: Securing Unix-like workstations, Int. J. Comput. Sci. Eng.,vol. 3, no. 3, pp. 24–32, 2015.

[19] M. U. Aksu, E. Altuncu, and K. Bicakci, A First Look at the Usability of OpenVAS Vulnerability Scanner, Workshop on Usable Security (USEC)2019. 2019,doi:10.14722/usec.2019.23026.

[20] S. Rahalkar, OpenVAS, in Quick Start Guide to Penetration Testing, Springer, 2019, pp. 47–71.