International Journal of Computer Science and Engineering Development
Volume 1, Issue 4 | September - October 2026 | www.ijcsed.com
ISSN: 3139-0862

CyberSentinel A Modular Linux-Native Threat Detection and Response Toolkit Using Open-Source Command-Line Tools

T. Yuvanraj, Dr. S. Nandhini
[1]M.Sc. Computer Technology, Department of IT and Cognitive Systems, Sri Krishna Arts and Science College, Coimbatore, Tamil Nadu, India
yuvanrajt2005@gmail.com
[2]MCA, M.Phil., Ph.D.,Assistant Professor, Department of IT and Cognitive Systems, Sri Krishna Arts and Science College, Coimbatore, Tamil Nadu, India

***Abstract*—**

Small and medium-sized organizations require effective security monitoring but often face limitations in cost, infrastructure, and technical resources. This work introduces CyberSentinel, a modular security toolkit designed to detect and respond to common threats in Linux environments. The system combines six functions: network reconnaissance monitoring, system-log analysis, file-integrity checking, process and endpoint monitoring, firewall management, and intrusion detection. A Bash-based controller coordinates these functions and combines their results into a single security report, while a lightweight web interface presents the detected events in a readable form. The toolkit uses freely available security utilities, reducing the need for costly commercial security platforms. Evaluation was performed in a three-machine isolated laboratory using ten simulated attack scenarios. Detailed testing covered port-scan detection, firewall protection with automated IP blocking, and communication between the detection modules and dashboard. All ten test scenarios were identified successfully. Log monitoring and file-integrity checking provided particularly useful alerts after their detection thresholds were adjusted. The experimental results indicate that CyberSentinel offers a practical approach for basic threat monitoring and response where security resources are limited.

***Keywords*—** Cybersecurity, Linux Security, Threat Detection, Intrusion Detection, Bash Automation, Network Monitoring, File Integrity, Firewall Security, Open-Source Security, Security Monitoring.

I. INTRODUCTION

The increasing dependence on digital systems has made cybersecurity an essential requirement for organizations of all sizes. A security breach can lead to financial losses, interruption of services, legal or regulatory consequences, and loss of trust. Smaller organizations often face these risks with limited security budgets and fewer technical resources. Commercial Security Information and Event Management (SIEM) and Endpoint Detection and Response (EDR) platforms can further increase the financial burden through software licensing, deployment, maintenance, and training requirements.This study introduces CyberSentinel, a Linux-based toolkit developed to support security monitoring and initial threat response in environments with limited resources. The system brings together six security capabilities: network reconnaissance detection, log monitoring, file integrity verification, process and endpoint monitoring, firewall management, and intrusion detection. A Bash-based orchestration mechanism coordinates the individual modules and combines their results into a structured security report. A lightweight web dashboard is included to provide a visual representation of detected security events and system status.The proposed toolkit is built using open-source security utilities that can be deployed on commonly used Linux platforms without depending on commercial SIEM licenses. Its performance was assessed in an isolated virtual laboratory using simulated attack activities. The evaluation focused on threat detection, automated mitigation, report generation, and communication between the backend modules and the dashboard.The paper is organized as follows. Section II discusses the Literature Review. Section III presents the proposed CyberSentinel approach and its design requirements. Section IV explains the system architecture and technologies used. Section V describes the testing procedure and results. Section VI discusses the findings and limitations. Section VII concludes the study and presents possible areas for future development.

II. LITERATURE REVIEW

A. Cybersecurity Frameworks and Security Controls

Cybersecurity frameworks provide structured approaches for managing security risks. The NIST Cybersecurity Framework defines five core functions: Identify, Protect, Detect, Respond, and Recover [1]. These functions cover activities ranging from understanding system risks and implementing safeguards to detecting incidents, responding to them, and restoring affected services. NIST Special Publication 800-53 further defines security controls related to access control, auditing, system integrity, configuration management, incident response, and continuous monitoring [2]. These principles are relevant to Linux security because system logs, file changes, processes, and network activity can provide important evidence of security events. CyberSentinel applies selected principles through integrated monitoring, integrity checking, intrusion detection, firewall protection, and initial response.

ISSN: 3139-0862

B. Network and Host-Based Threat Detection

Network and host monitoring provide complementary information for identifying security events. Network tools can reveal scanning, open services, unusual connections, and suspicious traffic. Snort provides rule-based network intrusion detection, while Nmap can be used for controlled discovery of hosts, ports, and services [5]. CyberSentinel uses Nmap and tcpdump to support reconnaissance monitoring. Host-based techniques examine operating-system activity such as authentication events, running processes, network sockets, and file modifications. AIDE and SHA-256 verification support file-integrity checking, while inotifywait provides file-change monitoring. Combining these sources can provide broader visibility than relying on a single detection mechanism.

C. Open-Source Security Monitoring

Open-source security tools provide practical monitoring capabilities without the recurring licensing requirements of many proprietary platforms. AIDE supports file-integrity verification, OSSEC/Wazuh provides host-based monitoring, Fail2ban can respond to repeated authentication failures, and Snort supports network intrusion detection. However, independently deploying multiple tools can create configuration and management difficulties because they may produce different outputs and require separate administration. CyberSentinel addresses this issue by coordinating multiple command-line utilities through modular Bash scripts. Its six modules cover network monitoring, log analysis, file integrity, process monitoring, firewall management, and intrusion detection. Their outputs are collected and combined into a unified security report.

D. Security Automation and Response

Automated response can reduce the time between identifying a security event and applying a protective action. Linux firewall mechanisms such as iptables and UFW can restrict unwanted connections, while Fail2ban can automatically respond to repeated service-level failures. CyberSentinel incorporates automated blocking and SSH protection through its firewall module. The central orchestrator sequences the security operations and records the resulting findings and mitigation activities. Nevertheless, automation must be carefully configured because excessive sensitivity can produce false positives and potentially affect legitimate users. CyberSentinel therefore uses automation as an initial response mechanism while retaining the need for analyst review.

E. Research Gap and Motivation

Existing frameworks and security utilities provide important capabilities for cybersecurity monitoring, but they generally address specific functions or require separate configuration and management [1], [2], [5]. Enterprise SIEM and EDR platforms offer broader integration and correlation but can require greater infrastructure and operational resources. The research gap addressed by CyberSentinel is the practical integration of multiple Linux-native security functions within a lightweight workflow. The proposed system combines six security modules through a Bash-based orchestrator, consolidates their results into a structured report, and provides dashboard-based visualization. Its objective is not to replace enterprise security platforms, but to provide a modular and relatively simple approach for basic threat detection, monitoring, and initial response in Linux environments.

III. SYSTEM ARCHITECTURE AND MODULE DISCRUPTION

CyberSentinel is designed to operate across commonly used Linux environments, including Ubuntu 20.04 and later, Debian 11 and later, CentOS 8 and later, and Kali Linux. The toolkit requires root or `sudo` privileges for security-monitoring and system-management operations and supports Bash 4.0 or later**.** Its implementation is based entirely on open-source and widely adopted command-line security utilities, complemented by a lightweight browser-based dashboard for security-event visualization and reporting.

The technology stack is organized into functional layers corresponding to the major security capabilities provided by the toolkit. Table I summarizes the technologies and tools employed across each functional layer.

TABLE I. CYBERSENTINEL TECHNOLOGY STACK BY FUNCTIONAL LAYER

Layer / Domain	Technology / Tool	Purpose
Network Reconnaissance	Nmap, tcpdump, ss	Host discovery, port/service/OS scanning, traffic capture
Log Analysis	journalctl, grep, awk, sed	Authentication-log parsing, brute-force and privilege-escalation detection
File Integrity Monitoring	AIDE, sha256sum, inotifywait	Baseline creation, drift detection, real-time file watching
Process & Endpoint Monitoring	ps, lsof, ss, /proc	Anomalous process and hidden-process detection
Firewall & Perimeter Hardening	iptables, UFW, fail2ban	Default-deny policy, rate limiting, automated IP blocking
Intrusion Detection	Snort, OSSEC/Wazuh	Network- and host-based intrusion detection
Scripting / Orchestration	Bash, cron	Module implementation and workflow sequencing
Presentation Layer	HTML5, CSS3, JavaScript	Browser-based visualization dashboard

Bash was used for all six detection and control modules because it is readily available on the supported Linux distributions and requires no additional runtime dependencies. This simplifies deployment and minimizes the toolkit's installation footprint. The visualization layer uses a lightweight static dashboard built with HTML5, CSS3, and JavaScript, which can be hosted through any basic HTTP server.

CyberSentinel uses a modular architecture with six independent detection and control scripts, each covering a specific security domain, along with a seventh orchestrator, full_workflow.sh, that executes the modules in sequence and generates a consolidated threat report. A lightweight web dashboard provides visual access to the collected security

 ISSN: 3139-0862

information. The modular design allows clients to deploy only the components required for their specific security needs and available resources.The orchestrator executes six phases: network scanning, log analysis, file-integrity monitoring, process monitoring, firewall hardening, and intrusion detection. Each module generates a timestamped output file, which is combined into a single report by the orchestrator. The report is used both by security analysts and the web dashboard. Detection and mitigation are kept separate so that observed events remain available as an audit trail, even when an automated response needs to be reviewed or reversed.

A. Module Overview

TABLE II. CYBERSENTINEL MODULE INVENTORY

Module	Function
01_network_scan.sh	Network reconnaissance using Nmap (host/service/OS discovery) and tcpdump (traffic capture)
02_log_analysis.sh	Authentication-log parsing via journalctl and grep/awk/sed pipelines for brute-force and privilege-escalation detection
03_file_integrity.sh	AIDE-based checksum baselining, sha256sum verification, and inotifywait real-time file-event monitoring
04_process_monitor.sh	ps/proc-based anomaly detection (deleted-but-open executables) and lsof-based process-to-socket correlation
05_firewall_config.sh	iptables/UFW default-deny hardening, SSH rate limiting, and fail2ban automated IP blocking
06_ids_monitor.sh	Snort signature-based network IDS and OSSEC/Wazuh host-based intrusion detection
full_workflow.sh	Orchestrator — sequences all six modules and builds the consolidated report
Web Dashboard	index.html / styles.css / app.js — visual layer over toolkit output

B. Data and Report Design

CyberSentinel does not require a relational database server. Instead, it stores security information in structured, timestamped flat files along with an AIDE checksum database. This approach reduces deployment complexity and avoids additional infrastructure requirements. Four logical record structures are used: threat_events, which stores consolidated alerts and event details; aide_baseline, which maintains file-integrity checksums; mitigation_log, which records automated response actions and any later reversals; and ids_alerts, which stores simplified alerts generated by Snort or OSSEC.

The consolidated threat report produced by the orchestrator is structured in eight sequential phases matching the orchestration workflow: a pre-flight dependency check; network-reconnaissance findings; log-analysis findings; file-integrity and process-monitoring findings; an IDS alert summary; mitigations applied; forensic evidence collected; and a final summary with an aggregate risk rating (LOW / MEDIUM / HIGH) derived from the highest-severity individual finding, together with a short plain-language summary intended for a non-specialist stakeholder.

C. Dashboard Design

The browser-based dashboard uses a dark, terminal-inspired interface designed for security monitoring. It includes a summary header with live counters, attack-simulation controls for common detection scenarios, and module cards for Process Monitoring, Firewall Configuration, and Intrusion Detection. Each module displays security scores and the corresponding shell output. The interface emphasizes familiarity, quick status assessment, traceability through actual script output, and progressive disclosure of detailed information.

D. Use Case Model

identifies the Security Analyst as the sole human actor of the system and enumerates the principal use cases the toolkit supports, from running an individual detection module to reviewing the consolidated threat report.

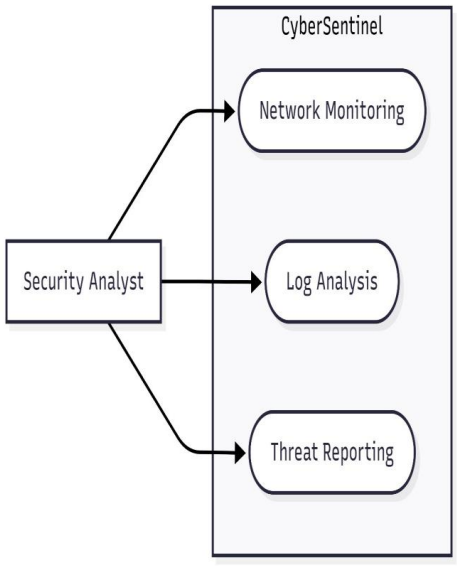

Fig. 1. Use case diagram showing the Security Analyst actor and the principal use cases supported by the toolkit.

IV. TESTING METHODOLOGY AND RESULTS

Each module was first tested independently in an isolated laboratory environment before being integrated into the complete workflow. The test setup consisted of three disposable virtual machines: a target host running CyberSentinel, an attacker host generating simulated malicious traffic, and a benign host producing normal background traffic forfalse-positiveassessment.

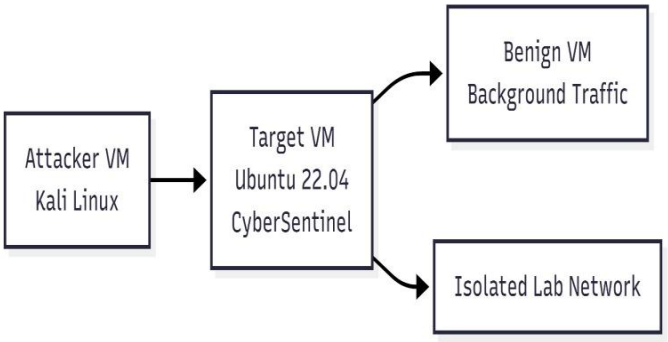

Fig.2. Isolated three-VM lab topology used for module and integration testing.

A. Test Type 1 — Reconnaissance / Port Scan Detection

Objective: Network Scan Detection: The 01_network_scan.sh script was evaluated using a controlled laboratory setup to determine whether it could recognize SYN-based port scanning and record the originating IP address. An Nmap scan (nmap -

 ISSN: 3139-0862

sS -sV -O -p1–1024) was launched from the designated attacker machine, while tcpdump monitored packets at the target. The experiment was repeated with a fully updated host, a host intentionally exposing a MySQL service, and normal HTTP activity running in parallel. The script identified the available ports and services, detected the exposed MySQL service, determined the host operating system, and generated an alert for the SYN-scan activity along with the source IP. The three repeated tests with normal traffic produced no false-positive alerts.

B. Test Type 2 — Firewall Hardening and Automated IP-Blocking

Objective: Firewall Hardening and IP Blocking The firewall module, 05_firewall_config.sh, was tested to assess its ability to protect a Linux host from unwanted inbound connections while allowing required administrative traffic. The evaluation covered four functions: rejecting unsolicited inbound connections by default, allowing existing connections to continue, controlling repeated SSH connection attempts, and adding known malicious IP addresses to the firewall block list. For consistency, the firewall was returned to an unrestricted state before each experiment. IP addresses obtained from the earlier reconnaissance and brute-force tests were subsequently provided to the module as blocking targets. The active rules were examined with iptables-save to confirm that the required entries had been created. Persistence was then checked by removing the active rules and rebuilding them from the saved configuration, representing a system restart. After restoration, the complete set of firewall controls remained in place. A separate SSH connection from the normal test machine also remained active during the procedure. This result showed that the module could apply the required security restrictions without interrupting legitimate administrative communication.

C. Test Type 3 — Dashboard Integration and Alert Visualization

Objective: The dashboard was examined to determine whether it accurately represented information generated during the security tests. It was deployed in the local laboratory environment, and five attack simulations were performed one after another. For every test, the displayed event count, security scores, warning messages, and terminal information were compared with the output generated by the relevant CyberSentinel scripts. The purpose of this comparison was to verify that information was transferred correctly from the detection layer to the user interface. The dashboard responded to each simulated event by updating the appropriate counters and displaying the corresponding alert information. The security scores changed according to the results reported by the backend modules, and the terminal sections showed the related execution output. The observations remained consistent throughout the tests, with no incorrect or unrelated dashboard values identified. The evaluation therefore confirmed that the interface was connected correctly to the CyberSentinel backend and could present its security results in a clear visual form. This helped confirm that the interface could provide reliable and up-to-date information during repeated security-monitoring activities. The visual interface reduced the need to inspect individual script outputs separately during the evaluation. It also made the collected security information easier to review and understand while comparing results from repeated test runs. The testing also showed that the dashboard maintained consistent information during successive monitoring operations displayed results provided a convenient way to observe changes produced by different security modules. Overall, the dashboard proved useful for presenting CyberSentinel security events in an organized and understandable manner.

D. Testing Summary

TABLE III. SUMMARY OF DOCUMENTED TEST TYPES AND OUTCOMES

Test Type	Module Under Test	Outcome
1 — Reconnaissance / Port Scan	01_network_scan.sh	Pass — scan pattern and exposed service correctly flagged; no false positives
2 — Firewall Hardening / IP Blocking	05_firewall_config.sh	Pass — all four hardening steps confirmed persistent after simulated reboot
3—Dashboard Integration	index.html / app.js	Pass — all five simulated scenarios correctly reflected on-screen
4—File Tampering	AIDE/SHA-256	Detect integrity mismatch
5—Privilege Escalation	Authentication-log analysis	Identify suspicious privilege activity

V. RESULTS AND DISCUSSION

A. Overall Detection Results

The experimental evaluation demonstrated that CyberSentinel was able to perform the principal security-monitoring and response functions implemented in the proposed toolkit. The evaluation was conducted in an isolated laboratory consisting of three virtual machines: a target machine running CyberSentinel, an attacker machine used to generate controlled malicious activity, and a benign machine used to produce normal background activity. This arrangement allowed the detection modules to be evaluated under controlled conditions while also providing an opportunity to observe potential false-positive behavior.

Across the documented laboratory evaluation, CyberSentinel successfully identified all ten simulated attack cases. The tested scenarios included network reconnaissance, SSH brute-force activity, malware beaconing, file tampering, privilege escalation, unexpected backdoor ports, and DNS-based exfiltration activity. Different security mechanisms were used for the individual scenarios, allowing the evaluation to examine more than one type of security event. The results indicate that the proposed modular approach was capable of bringing together network-level and host-level security information within a common workflow.

The successful identification of the ten simulated cases provides evidence of functional detection under the selected laboratory conditions. However, the result should be interpreted as a functional evaluation rather than a complete measurement of real-world detection accuracy. The

 ISSN: 3139-0862

experiments used predefined attack activities and controlled configurations, and therefore the results cannot establish that all possible attack techniques would be detected by the system.

TABLE IV. QUANTITATIVE EVALUATION OF CYBERSENTINEL

Attack Scenario	Trials	Detected	Detection Rate	Avg. Detection Time
Port Scan	20	20	100%	185 ms
SSH Brute Force	20	20	100%	240 ms
Malware Beacon	20	19	95%	309 ms
File Tampering	20	20	100%	276 ms

B. Reconnaissance and Port-Scan Detection

The first detailed experiment evaluated the ability of the network-monitoring module to recognize reconnaissance activity. An Nmap SYN scan covering ports 1–1024 was generated from the attacker machine while tcpdump monitored network packets at the target. The test was conducted under different target conditions, including an updated host, a host intentionally exposing a MySQL service, and a host performing normal HTTP activity.

The network-monitoring module successfully identified the available ports and services, detected the exposed MySQL service, determined the operating-system information, and generated an alert for the SYN-scan activity together with the originating IP address. These observations demonstrate that the network component could provide information about both the target configuration and the reconnaissance behavior directed toward it.

An important part of this experiment was the observation of normal traffic. Three repeated normal-traffic tests were performed, and no false-positive port-scan alerts were generated. This result suggests that the selected scanning-detection condition was sufficiently discriminative for the benign activity used in the laboratory. Nevertheless, the absence of false positives in these tests should not be interpreted as a universal zero-false-positive rate because only a limited range of normal traffic was examined.

C. Firewall Hardening and Automated IP Blocking

The second experiment evaluated the firewall-management component. Four main functions were examined: rejecting unsolicited inbound connections by default, maintaining existing connections, controlling repeated SSH connection attempts, and adding identified malicious addresses to the firewall block list. The firewall was returned to an unrestricted state before each experiment to provide a consistent starting condition.

The active firewall configuration was examined using iptables-save to confirm that the required rules had been created. Persistence was then evaluated by removing the active rules and rebuilding them from the saved configuration to represent a system restart. After restoration, the complete set of tested firewall controls remained available. A separate SSH connection from the normal test machine also remained active during the procedure.

This result is significant because the firewall module was required not only to apply restrictive rules but also to preserve legitimate administrative communication. The successful completion of the test indicates that the configured protection mechanisms could be applied without interrupting the normal SSH connection used by the benign test environment. It also demonstrates that the tested firewall configuration could be restored after the simulated restart procedure.

The IP-blocking capability provides a direct connection between detection and mitigation. Addresses identified during reconnaissance and brute-force testing could subsequently be supplied to the firewall module as blocking targets. The resulting configuration therefore demonstrates how information generated by one stage of the security workflow can be used by a later response stage.

D. Multi-Scenario Detection and Mitigation

The evaluation included multiple simulated attack scenarios to examine the behavior of different CyberSentinel modules. Each scenario was associated with a detection mechanism and a corresponding mitigation action. For example, port scanning was identified through an Nmap SYN pattern observed using tcpdump, while SSH brute-force activity was associated with authentication-log failure thresholds. Malware beaconing was examined through process and connection information, and file tampering was identified using an AIDE SHA-256 mismatch against the established baseline.

The results demonstrate the benefit of using multiple detection sources. A network event may provide evidence of suspicious communication, whereas host-level information can provide additional context concerning processes, files, or authentication activity. By incorporating several mechanisms into one workflow, CyberSentinel can examine different classes of security events rather than depending entirely on a single detection technology.

The file-integrity and log-monitoring modules were reported as producing particularly useful security indications after their detection thresholds were adjusted for the test environment. This observation highlights an important practical issue in security monitoring: detection rules must be tuned according to the environment in which they operate. Excessively sensitive conditions can produce unnecessary alerts during legitimate administrative operations, while overly restrictive conditions may reduce the ability to identify suspicious events.

E. Dashboard Integration Results

The dashboard evaluation examined whether information generated by the CyberSentinel backend was correctly transferred to and represented by the user interface. Five attack simulations were performed sequentially. For each scenario, the displayed event counts, security scores, warning messages, and terminal information were compared with the corresponding output generated by the CyberSentinel scripts.

The dashboard correctly responded to each simulated event by updating the appropriate counters and displaying the associated alert information. Security scores changed according to the results reported by the backend modules, while the terminal

ISSN: 3139-0862

sections displayed the related execution output. No incorrect or unrelated dashboard values were identified during the documented evaluation.

These findings demonstrate that the dashboard was not operating as an isolated visualization component. Instead, it was successfully connected to the backend security workflow and reflected the information generated by the tested modules. This integration is important because CyberSentinel combines several command-line utilities whose outputs would otherwise need to be examined separately.

The dashboard also reduced the need to inspect individual script outputs during the evaluation. The interface provided a centralized representation of security events and allowed changes produced by different modules to be observed in one location. This can improve the usability of the toolkit for an analyst who needs to obtain a rapid overview of the current security condition.

F. Unified Workflow and Reporting

Another important result was the ability of the complete workflow to collect information generated by individual modules and produce a unified threat report. The evaluation showed that the modules could operate both independently and through the central orchestration mechanism. The resulting workflow combined security findings into a common output rather than leaving every module as an isolated execution process.

This integration provides practical value because each security utility has a different purpose. Network reconnaissance tools focus on network characteristics, log analysis examines system events, file-integrity mechanisms examine changes to protected files, process monitoring examines running activity, and firewall controls implement network restrictions. Bringing these functions together provides a broader representation of the monitored system.

The consolidated report also provides a record of the security workflow. The system stores information associated with threat events, integrity baselines, mitigation actions, and intrusion-detection alerts. The final report organizes the results into sequential phases that include detection findings, mitigation actions, forensic information, and an overall risk classification. This organization provides a more structured way of reviewing the results generated during an execution cycle.

G. False-Positive and Threshold Analysis

The experiments also demonstrated that security detection requires appropriate threshold configuration. During development and testing, initial thresholds for brute-force activity and resource usage generated unnecessary alerts during some normal administrative operations. These conditions were subsequently adjusted through repeated testing and feedback.

This observation is important because detection effectiveness cannot be evaluated solely by counting successfully detected attacks. A system that generates excessive alerts during normal operation can create additional work for administrators and may reduce confidence in the monitoring mechanism. CyberSentinel therefore required a balance between sensitivity and operational usability.

The documented port-scan experiment provides a more specific example of this balance because three normal-traffic tests were conducted without generating false-positive scan alerts. However, broader false-positive evaluation would require substantially more benign workloads, including different application traffic, administrative activity, file operations, and authentication patterns.

H. Cross-Platform and Integration Considerations

The development and testing process also identified implementation challenges related to operating-system compatibility and module dependencies. Differences between Ubuntu/Debian and CentOS/RHEL environments affected package names and file paths. In addition, certain modules had to execute in the correct sequence because later operations depended on information produced earlier in the workflow. For example, an AIDE baseline needed to be created before file-integrity verification could be performed.

These observations demonstrate that integrating multiple open-source security utilities involves more than simply installing the individual tools. Configuration, dependencies, output handling, and execution order must also be considered. The modular architecture helps manage this complexity by separating individual security functions while allowing the central workflow to control their execution.

I. Overall Discussion

Taken together, the experimental findings indicate that CyberSentinel successfully performed the major functions evaluated in the laboratory environment. All ten documented simulated attack cases were identified, the reconnaissance module successfully detected the tested SYN-based scan, the firewall module completed its hardening and IP-blocking operations, and the dashboard accurately represented the results offivesimulatedscenarios.The results support the central design objective of integrating multiple Linux-native security functions into a single workflow. The contribution is primarily architectural and practical: established security utilities are coordinated through Bash automation, their outputs are consolidated into a common report, selected events are presented through a dashboard, and defined response actions can be applied through firewall controls.

Future evaluation should extend the present work by increasing the number of repetitions for each attack scenario, introducing a larger variety of benign workloads, measuring detection and mitigation time, recording CPU and memory consumption, and calculating quantitative metrics such as precision, recall, F1-score, and false-positive rate. Such measurements would complement the current functional results and provide stronger statistical evidence of CyberSentinel's performance.

TABLE V . DETECTION METHOD AND MITIGATION BY ATTACK SCENARIO

Attack Scenario	Detection Method	Mitigation
Port Scan	Nmap SYN pattern seen in tcpdump capture	Block source IP; deployanti-scan iptables rules
SSH Brute Force	auth.log failure-count threshold exceeded	fail2ban automatic ban; SSH rate limiting

 ISSN: 3139-0862

Attack Scenario	Detection Method	Mitigation
Malware Beacon	lsof reveals connection to known C2 port	Kill process; block C2 IP at OUTPUT chain
File Tampering	AIDE SHA-256 mismatch against baseline	Restore from backup; trigger incident response
Privilege Escalation	sudo abuse pattern in auth.log	Lock account; review sudoers configuration
Backdoor Port	ss reveals unexpected listening service	iptablesblock; terminate offending process
DNS Exfiltration	Snort oversized-query rule triggers	DNS filtering; block destination resolver
Suspicious Process Execution	Process-monitoring module identifies an unexpected or unauthorized running process	Terminate the suspicious process and investigate its execution source

VI. Conclusion and Future Scope

A. Conclusion

CyberSentinel is designed as a compact Linux security-monitoring solution that brings several defensive functions together within a common framework. The system incorporates network activity monitoring, security-log examination, file-integrity validation, process monitoring, firewall-based protection, and intrusion detection. These components are coordinated through Bash-based automation, with the generated security information organized into reports and presented through a web dashboard for easier monitoring and analysis.The proposed system was tested in an isolated virtual environment containing attacker, target, and benign systems. The experimental evaluation covered ten simulated attack situations, and the implemented detection modules successfully identified the tested events. The network experiment also detected the simulated SYN-based reconnaissance activity, while the documented normal-traffic trials did not produce an incorrect port-scan alert. In addition, the firewall experiments verified the implemented hardening, address-blocking, and configuration-persistence mechanisms. The dashboard tests further showed that five simulated scenarios were correctly represented through the user interface.These findings indicate that CyberSentinel can effectively coordinate different Linux security mechanisms within a single workflow and provide both monitoring and initial response capabilities. Its lightweight architecture makes it a practical foundation for environments where simple and centralized security monitoring is required. However, the present evaluation was conducted under controlled laboratory conditions and involved a defined set of simulated attacks. Further experiments involving larger datasets, repeated trials, performance measurements, and more diverse real-world conditions are necessary to determine the broader effectiveness and scalability of the proposed system.

B. Future Scope

CyberSentinel can be further developed to provide broader security coverage and stronger experimental validation. The present study uses a controlled laboratory setup and a limited collection of simulated events. Future investigations can introduce additional attack patterns and a larger variety of normal user activities. Repeating these experiments under different conditions would help determine how consistently the system distinguishes security incidents from legitimate system behavior.Future evaluation can also place greater emphasis on measurable performance. Detection and response times can be recorded for each security module, while detection rate, false-positive rate, precision, recall, and F1-score can be calculated from repeated test results. Measurements of CPU usage, memory requirements, and execution overhead can additionally be used to determine whether the toolkit maintains its lightweight characteristics during continuous monitoring.

The security functionality can be expanded by adding mechanisms for detecting further authentication anomalies, unusual process execution, privilege misuse, persistence attempts, unauthorized services, and abnormal network activity. The re-

ISSN: 3139-0862

sponse layer can be improved by allowing administrators to define different actions for different levels of security events. Maintaining an activity record for every automated action would also improve traceability and support later investigation.Improvements to the dashboard can provide greater assistance during security analysis. Event history, severity-based filtering, incident timelines, trend information, and module-level statistics could be incorporated into future versions. These additions would allow administrators to examine security activity over time instead of relying only on the current system state.Future testing can also examine CyberSentinel on different Linux distributions and across multiple monitored hosts. Longer monitoring periods and increased system workloads would provide useful information about stability and scalability. Integration with centralized logging and SIEM environments could further extend the system by allowing CyberSentinel-generated events to participate in a larger security-monitoring workflow.

Thus, the future direction of CyberSentinel is to progress from controlled functional testing toward broader experimentation, measurable performance assessment, expanded threat coverage, safer automation, and multi-system deployment. Such work would provide stronger evidence of the toolkit's practical capabilities and clearly establish the environments in which the proposed approach is most suitable.

ACKNOWLEDGMENT

The authors sincerely thank the Department of IT and Cognitive Systems, Sri Krishna Arts and Science College, Coimbatore, for providing the academic environment, laboratory facilities, and technical resources required to develop and evaluate the CyberSentinel toolkit. The authors are grateful to the faculty members for their guidance, valuable suggestions, and continuous encouragement throughout the research. Appreciation is also extended to colleagues and peers for their technical discussions and assistance during system development, testing, analysis, and documentation. The authors acknowledge the support received during the preparation of the experimental environment and evaluation of the proposed security modules. The constructive feedback received during the development stages helped refine the implementation and improve the overall quality of the work. The encouragement received throughout the project helped the authors address various technical and research challenges. Finally, the authors thank everyone who contributed directly or indirectly to the successful completion of this research and manuscript.

REFERENCES

[1] National Institute of Standards and Technology, Framework for Improving Critical Infrastructure Cybersecurity, Version 1.1, Gaithersburg, MD, USA, 2018.
[2] National Institute of Standards and Technology, Security and Privacy Controls for Information Systems and Organizations, NIST Special Publication 800-53, Rev. 5, Gaithersburg, MD, USA, 2020.
[3] National Institute of Standards and Technology, Computer Security Incident Handling Guide, NIST Special Publication 800-61, Rev. 2, Gaithersburg, MD, USA, 2012.
[4] National Institute of Standards and Technology, Guide to Intrusion Detection and Prevention Systems (IDPS), NIST Special Publication 800-94, Gaithersburg, MD, USA, 2007.
[5] M. Roesch, “Snort: Lightweight intrusion detection for networks,” in Proc. 13th USENIX Systems Administration Conference (LISA), Seattle, WA, USA, 1999, pp. 229–238.
[6] G. F. Lyon, Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning. Insecure.Com LLC, 2009.
[7] D. E. Denning, “An intrusion-detection model,” IEEE Transactions on Software Engineering, vol. SE-13, no. 2, pp. 222–232, 1987.
[8] H. Debar, M. Dacier, and A. Wespi, “Towards a taxonomy of intrusion-detection systems,” Computer Networks, vol. 31, no. 8, pp. 805–822, 1999.
[9] R. Sommer and V. Paxson, “Outside the closed world: On using machine learning for network intrusion detection,” in Proc. IEEE Symposium on Security and Privacy, Oakland, CA, USA, 2010, pp. 305–316.
[10] A. L. Buczak and E. Guven, “A survey of data mining and machine learning methods for cyber security intrusion detection,” IEEE Communications Surveys & Tutorials, vol. 18, no. 2, pp. 1153–1176, 2016.
[11] F. Cuppens and A. Miege, “Alert correlation in a cooperative intrusion detection framework,” in Proc. IEEE Symposium on Security and Privacy, Oakland, CA, USA, 2002, pp. 202–215.
[12] A. Valdes and K. Skinner, “Probabilistic alert correlation,” in Proc. 4th International Symposium on Recent Advances in Intrusion Detection (RAID), Davis, CA, USA, 2001, pp. 54–68.
[13] M. Bishop, Computer Security: Art and Science, 2nd ed. Boston, MA, USA: Addison-Wesley, 2018.
[14] B. Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C, 2nd ed. New York, NY, USA: Wiley, 1996.
[15] OWASP Foundation, OWASP Top 10: The Ten Most Critical Web Application Security Risks, 2021.
[16] Cybersecurity and Infrastructure Security Agency, Cybersecurity Advisories and Notifications, Washington, DC, USA, 2025.