Global Journal of Engineering and Technology Advances
eISSN: 2582-5003
Cross Ref DOI: 10.30574/gjeta
Journal homepage: https://gjeta.com/

(RESEARCH ARTICLE)

Check for updates

Development of an OSSEC-powered host-based intrusion detection system

Kikelomo Ibiwumi Okesola *, Zammy-Nora Chizaram Uzoma, Boluwatife Ibukun Rahmon and Oluwaseun Olufemi Taiwo

Department of Software Engineering, Babcock University, Ilishan-Remo, Ogun State, Nigeria.

Global Journal of Engineering and Technology Advances, 2026, 27(03), 197-208

Publication history: Received on 15 May 2026; revised on 21 June 2026; accepted on 23 June 2026

Article DOI: https://doi.org/10.30574/gjeta.2026.27.3.0161

Abstract

Current approach of host-based intrusion detection is often cumbersome because it is mainly executed via the command line and comes with a level of complication. Thus, this study attempts to develop the solution that will allow for effective, continuous host-based security monitoring through OSSEC, which is an open-source intrusion detection engine coupled with custom web interface. In the study, an Agile development approach employed. The back-end of the project was built with FastAPI framework and Python. While SQLite database was used for storing collected logs, and the Nginx is implemented as a web server. For the front-end of the system, HTML, CSS, and JavaScript was used to design the interface for alert visualization. OSSEC works as a host-based IDS solution by analysing system logs, file integrity checks, and policy violations. With the help of a purpose-designed web interface, the information is delivered to the administrator who can use it to react promptly to potential threats to the server. Hence, making intrusion detection system more usable.

Keyword: Intrusion Detection; Host-Based Intrusion; Process Monitoring; OSSEC; File Monitoring

1. Introduction

In the current times, cybersecurity has become highly important due to interconnectivity used to manage sensitive data and processes. Consequently, the security environment has changed because cyber criminals now use multiple techniques that are capable of bypassing the traditional security measures such as firewalls and antivirus software [1]. As a result, intrusion detection has become the most effective way of protecting cybersecurity through identifying all kinds of malicious behaviour that might slip through the initial defence system.

IDS solutions were designed to address the issue and allow detecting, analysing, and responding to various types of attacks, ranging from brute-force attacks to more advanced and persistent attacks. Initially, intrusion detection was based on network traffic; hence, leading to the creation of Network-based Intrusion Detection Systems (NIDS). However, the performance of NIDS was limited due to the increased number of TLS encryption and VPN tunnels [2].

To counter this change, intruders have had to resort to targeting endpoint systems, by using such attack methods as privilege escalation and process injection among others, all leaving little or no footprint at the network level. In turn, this scenario has contributed to the increasing significance of Host Based Intrusion Detection Systems (HIDS), which work directly on endpoints for intrusion detection [1]. As opposed to network IDS, host-based ID systems monitor events taking place on individual hosts, allowing them to detect such incidents as unauthorised changes to files, privilege escalations, covert processes, or log alterations [4].

The novelty of this project is the use of OSSEC, an open-source mature HIDS engine, along with the web-based user interface, that will present the administrator with consolidated real-time alerts. This way, the solution can offer the

* Corresponding author: Okesola, Kikelomo. I.

detection capabilities, as well as easy monitoring and analysis of attacks, ensuring affordable and scalable security for both colleges and SMEs, who cannot afford to purchase any expensive security products.

There are availability of host-based monitoring systems; however, their use becomes problematic due to two issues that are repeatedly raised. The first is that these systems produce a significant amount of unprocessed data and warnings which cannot be interpreted without specialized knowledge; hence, they cannot be used by organizations not having security experts on board. The other issue is that these systems are not easily integrated into user-friendly monitoring tools, resulting in delayed action against cyber threats.

This study aims to create a host-based intrusion detection system that makes endpoint security monitoring more accessible through an intuitive web interface.

2. Literature Review

Here, an overview of available literature related to the design and development of the HIDS that utilizes OSSEC as its backbone system are presented. This includes the history and classification of IDSs, host-based intrusion detection versus network-based intrusion detection systems, methods of detection, OSSEC architecture, and three major methods of detection.

2.1. Overview of Intrusion Detection Systems

An Intrusion Detection System (IDS) refers to a security tool utilized to observe any computer environment for any suspicious behavior or violation of policies and trigger alarms once the threat is detected [1]. As mentioned by Khraisat et al., IDS can be categorized along three main dimensions, namely, detection technique (signature based or anomaly based), location (NIDS or HIDS), and response mechanism (passive alarms or active actions) [5].

Despite being crucial components in the security of IT infrastructure, there are several issues reported regarding the efficiency of IDS. According to Ahmad et al., high levels of false positives affect the performance of IDS by causing alarm fatigue due to an excessive number of alerts [6]. Moreover, evasion mechanisms such as installing rootkits, tampering with logs, and fileless execution represent a challenge for IDS operating in hosts [7].

2.2. Host-Based Versus Network-Based IDS

NIDS work by collecting and analysing traffic flows within the network at critical junctures such as gateways and routers. However, according to Alazab et al., the extensive usage of TLS and virtual private network (VPN) tunnelling protocols has made most traffic in the enterprise network unreadable to deep packet inspection, thus eliminating the ability of NIDS to detect suspicious behaviour. Additionally, NIDS cannot monitor any activity that occurs within the host such as file editing, privilege elevation, or harmful process creation [1].

HIDS is installed on individual hosts and is immune to any traffic encryption and flow obfuscation. In their comparative assessment, Alghamdi et al. determined that HIDS reliably detects host-specific attacks, especially unauthorized file editing and privilege escalation that do not produce a network footprint. Martins et al. have supported the increasing significance of host-based intrusion detection by proving that most modern attacks on endpoints occur mainly from host-level actions and not from recognizable traffic flows [3, 4].

2.3. Detection Methodologies

One of the most common HIDS engines in use today is called OSSEC or Open-Source Security. The main features of OSSEC include file integrity checking, log analysis, rootkit detection, process monitoring, and real-time alerts. This system follows the server/agent approach to gathering data where security information collected from multiple agents is sent to a centralized OSSEC server for processing and correlation. OSSEC is developed by Trend Micro in collaboration with an active open-source community and works across numerous operating systems, such as Linux, Windows, Mac OS, and Solaris [9]. All data transmission between agents and the OSSEC server is done using AES encryption.

2.4. Overview of OSSEC

OSSEC (Open-Source Security), one of the most popular HIDS programs, performs functions such as file integrity checking, log analysis, rootkit detection, process monitoring, and real-time alerting using only one application. This program operates using a client-server approach in which the OSSEC server application analyses and processes the data gathered by light-weight clients installed on targeted systems. The OSSEC HIDS application is created by Trend Micro, Inc., and widely supported by the open-source development community. It supports operating systems like Linux,

Windows, Mac OS, and Solaris [9]. All the messages exchanged between the clients and the server use the AES encryption protocol.

2.5. Summary of Related Works

Table 1 Summary of Related Works

Author(s) & Year	Research Objective	Proposed Method	Result	Limitations
Park et al. (2021)	Evaluate a HIDS model using the LID-DS dataset	ML with host-based preprocessing and hyperparameter optimization	~6% improvement in accuracy, precision, and F1-score over baseline	Computationally intensive; limited for real-time resource-constrained deployments
Catherine et al. (2021)	Develop efficient HIDS using correlation-based partial decision tree	CPDT algorithm with feature selection on benchmark datasets	Detection accuracy of 99.95%, outperforming traditional DT methods	Primarily targets known attacks; limited capability against zero-day threats
Ou et al. (2021)	Improve HIDS detection via combined misuse and anomaly detection	Hybrid system: log analysis + back-propagation neural network	Significant improvements vs. single-method approaches	Requires extensive training data; high computational resources
Alghamdi et al. (2021)	Review IDS types, detection techniques, and architectural challenges	Systematic literature review of HIDS effectiveness	Confirmed HIDS superiority for detecting internal threats that evade NIDS	High false positive rates; no system implementation or experimental validation
Martins et al. (2022)	Review HIDS detection techniques, datasets, and evaluation metrics	Literature review: FIM, process monitoring, log analysis	Identified key detection mechanisms and evaluation metrics for HIDS	Many HIDS solutions lack real-world evaluation
Alazab et al. (2022)	Examine cybersecurity threat landscape and limits of perimeter-based defences	Analytical review of modern endpoint attack trends	Demonstrated that modern attacks increasingly target endpoints, reinforcing HIDS necessity	No specific IDS proposed or experimentally evaluated
Satilmis et al. (2024)	Systematically review HIDS published between 2020 and 2023	Review of signature-based and anomaly-based HIDS with ML	Revealed challenges: alert overload, limited explainability, poor usability	Purely a review; no implementation or experimental validation
Jiang et al. (2026)	Develop HIDS via audit log analysis with provenance graph learning	Provenance graphs from audit logs with NLP and graph convolutional networks	Improved accuracy across diverse attack scenarios including APTs	Significant computational complexity; requires extensive audit data

This can be seen from the literature review as shown in Table 1, since though there is sufficient evidence to prove the technical efficiency of host-based detection, there is not one particular piece of literature which gives an overall solution by incorporating OSSEC with its various capabilities of detecting into a customized web portal. This gap is what the current study aims to fill.

3. Methodology

3.1. System Architecture

The HIDS adopted a hybrid architectural style that combines Client-Server to enable communication between devices and the server; and OSSEC open-source engine using layered architecture; a logic of three layers – the monitored hosts layer, the detection/processing layer, and the storage/presentation layer depicted in Figure 1.

The first layer consists of OSSEC agents that monitor activity on the host. The agent gathers information about host activities in log form, file integrity through cryptographic hashing, process monitoring and other. The gathered information is transmitted to the OSSEC Manager in an encrypted manner (SSL/TLS). However, the second layer consists of the OSSEC manager software running on a dedicated server with Ubuntu 22.04 LTS OS and OSSEC version 3.8.0. OSSEC manager decrypts information coming from the monitored hosts layer through its own decryption library and processes using its hierarchical rules engine. Moreover, OSSEC Manager is backed by a FastAPI implementation of RESTful services used for user authentication, authorization, alert retrieval, and reporting. The last layer – the storage/presentation layer – consists of an SQLite database used to store collected alerts and logs, and the Nginx web server used to provide information to administrator through HTTPS.

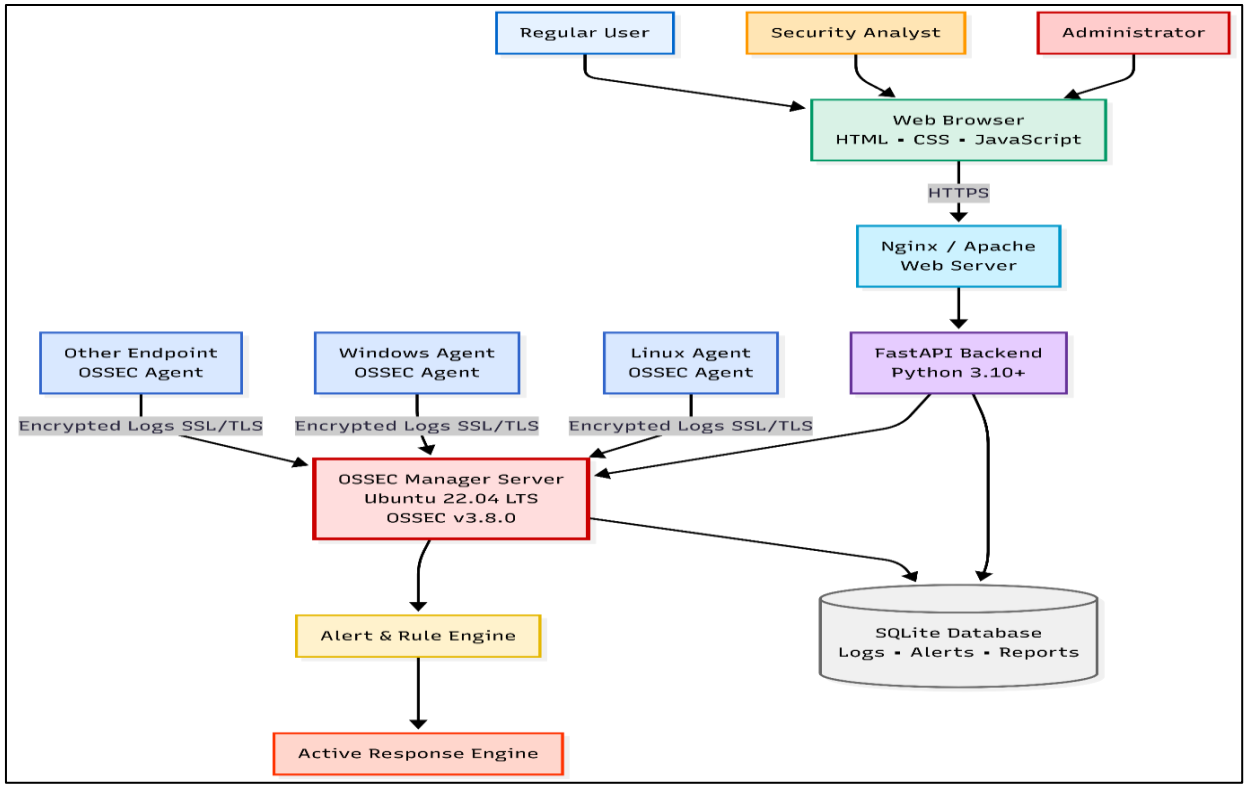

Figure 1 Client-Server Architecture Diagram

3.2. System Model

A system model is depicted using Use Case Diagram, Activity Diagram, and System Flowchart (Figures 2,3, and 4). The Use Case Diagram includes the following human actors: Administrator, Security Analyst, and Regular User; in addition to the OSSEC Agent which is considered as a system actor. Main use cases in the diagram include Monitor File Integrity, Analysing System Logs, Monitoring Running Processes, Generating Security Alerts, Showing Alert Dashboard, Filtering/Searching Alerts, Showing Forensic Logs, and Generating Reports (Figures 2).

On the other hand, the Activity Diagram (Figures 3) models the process from detecting events at a monitored endpoint until showing them in the web dashboard. The activity starts when an agent detects an event on a host, sends encrypted data to the OSSEC Manager, the manager analyses the detected event based on the rule library, creates an alert record in case of matching rules and writes it to the database where the FastAPI polling will fetch the new entry to update the alert dashboard. In the System Flowchart (Figures 4)., one can understand the complete process from collecting the

event to displaying it as an alert in the web UI, and it acts in a loop fashion throughout the registered agents at the same time.

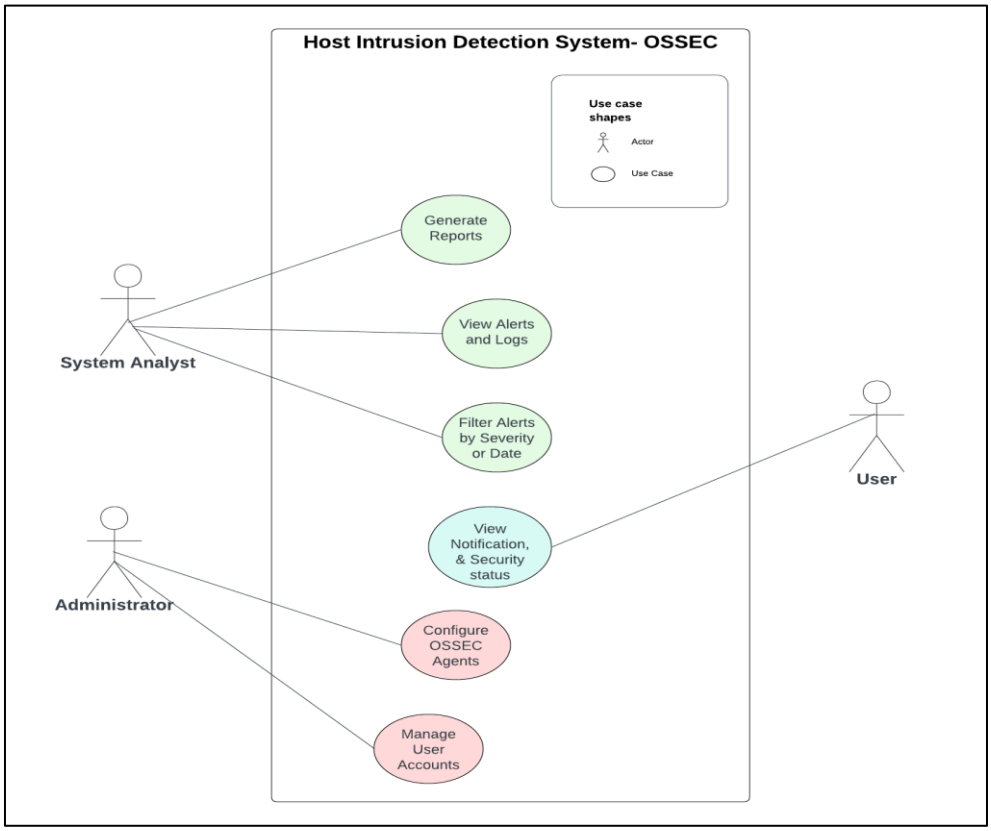

Figure 2 Use Case Diagram

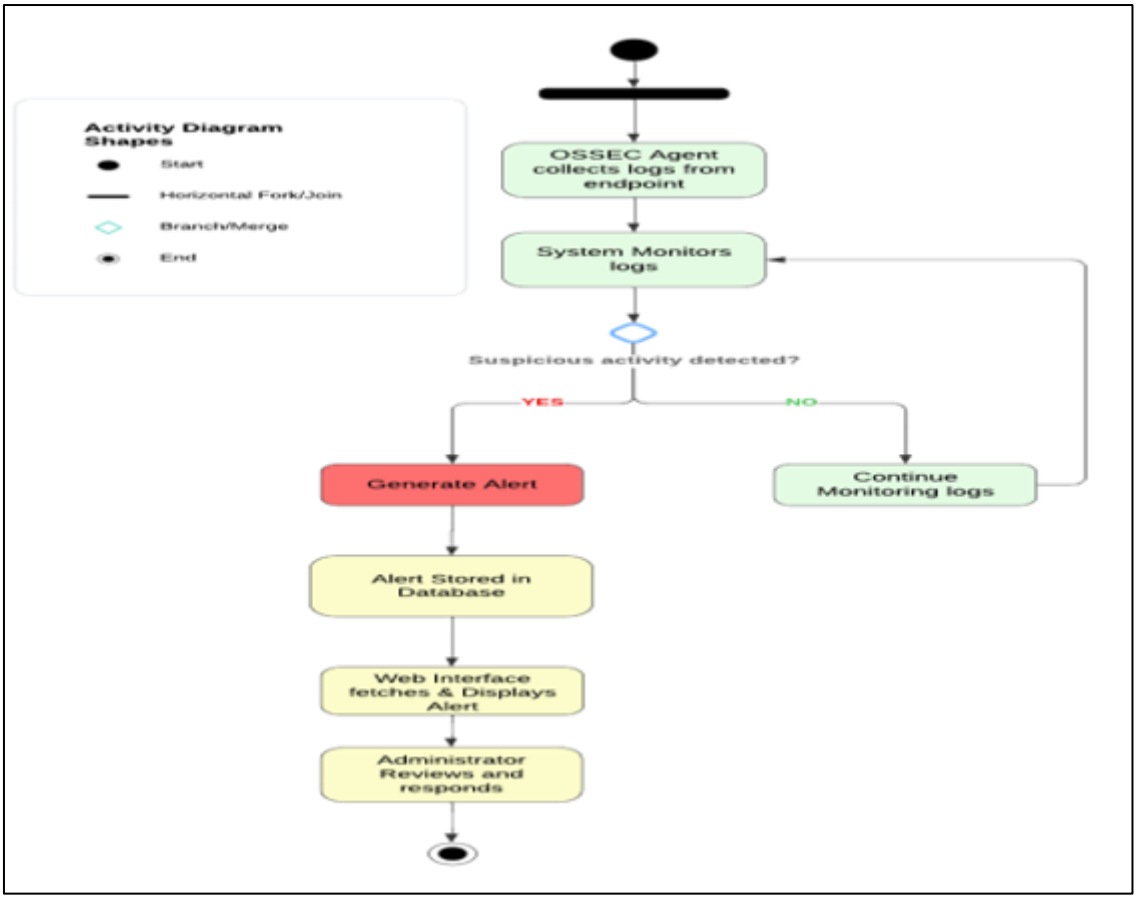

Figure 3 Activity Diagram

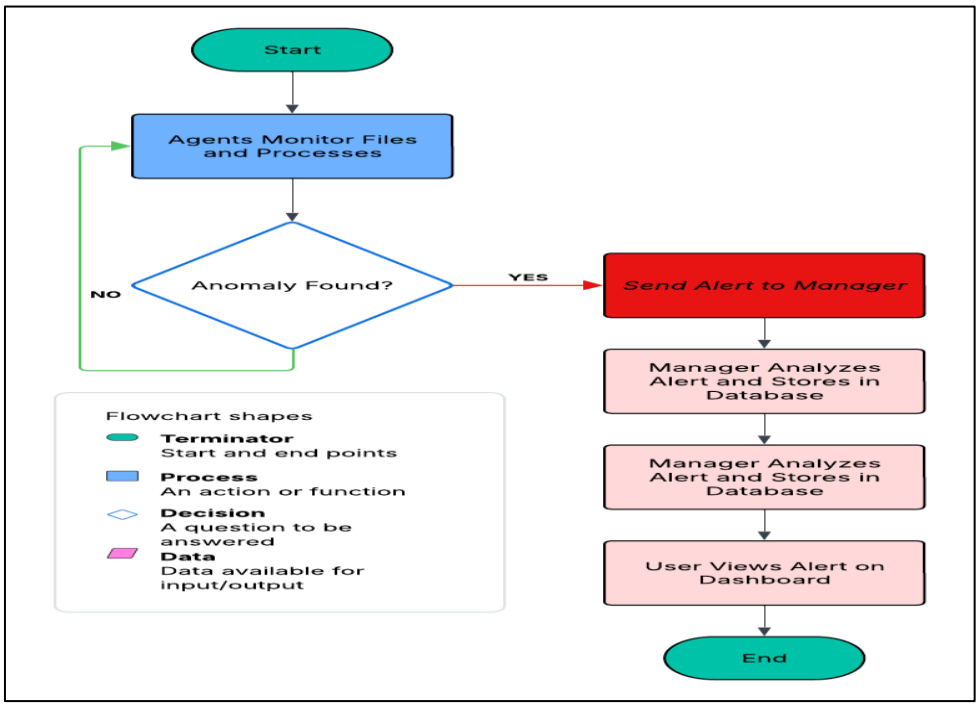

Figure 4 System Flowchart

3.3. System Requirements

System requirements are organised into functional requirements Table 2, which specify the operations and capabilities the system must provide, and non-functional requirements Table 3, which specify performance, security, and usability characteristics.

Table 2 Functional Requirements

ID	Functional Requirement
FR1	Monitor critical system files and directories using OSSEC's FIM module and detect any unauthorised modification, deletion, or creation.
FR2	Continuously monitor active system processes and identify suspicious execution based on predefined OSSEC rules and signatures.
FR3	Collect and analyse operating system and application logs to detect known attack signatures, abnormal access attempts, and policy violations.
FR4	Generate real-time alerts when suspicious activities are detected by the file integrity, process monitoring, or log analysis modules.
FR5	Securely store all generated alerts, event logs, and historical data in a centralised SQLite database to support incident analysis.
FR6	Provide a web-based dashboard that displays alerts, system events, and summarised security information in a user-friendly format.
FR7	Allow authorised users to log in using valid credentials before accessing any monitoring or analysis features.
FR8	Allow users to filter and search alerts based on severity level, rule ID, date range, and affected host.

Table 3 Non-Functional Requirements

ID	Non-Functional Requirement
NFR1	Process and display alerts in real time with minimal latency, ensuring security events are visible to administrators promptly upon detection.
NFR2	Accommodate monitoring of multiple endpoints simultaneously without any degradation in detection or interface performance.
NFR3	Encrypt all data transmission between agents, the OSSEC Manager, and the web interface using SSL/TLS.
NFR4	Ensure the web interface is simple and intuitive enough for users with minimal technical expertise.
NFR5	Run continuously with critical processes configured to restart automatically following any unexpected termination.
NFR6	Support easy addition of new OSSEC detection rules without requiring modification of the core system architecture.

3.4. Development Approach

The design of the system was done under the principles of Agile software development. This is an incremental and iterative process which focuses on providing working solutions in small iterations and constant feedback. The use of agile was necessary since it incorporated multiple layers of OSSEC which needed their own separate configuration, testing, and tweaking. It enabled us to first configure and test OSSEC before connecting it with the database layer, then to test the database layer before connecting the entire solution to a web interface.

3.5. Technologies Used

This system employs contemporary open-source software technologies. OSSEC HIDS version 3.8.0 acts as the alert engine. For OSSEC Manager and web stack, we use Ubuntu Server 22.04 LTS as the operating system environment. For

the back-end API, Python 3.10 programming language along with the FastAPI framework handles alert data retrieval, password hashing and authentication with bcrypt, and user roles authorization via JWT tokens. SQLite 3.39 acts as the database engine in this case. Nginx 1.24 acts as the web server and proxy, with SSL/TLS encryption implemented by configuring it as the SSL termination point. The front-end web interface uses modern web programming languages including HTML5, CSS3, Bootstrap 5.3, and JavaScript which is responsible for implementing filters and polling mechanism to retrieve data from the FastAPI back-end API layer.

4. Graphical View of System Implementation

4.1. Home Page

The Home Page serves as the first screen of the system where there is a brief explanation about the OSSEC-based Host Intrusion Detection System (Figure 5). This section features three main functionalities, which are File Integrity Monitoring, Process Monitoring, and Log Analysis, shown on individual cards. There is a card called the Intelligent Alert System, which shows current alert notifications sorted according to their levels, proving that the system is working actively on monitoring the host. Finally, the footer has a link to the Administrator Access that leads to the Login Page.

Figure 5 Home Page

4.2. Registration and Login

The Request Access page (Figure 6) manages the process of creating an account. It requires a user email address, password, and confirmation password, then forwards these credentials for approval by the administrator; this means that an account will be activated only after it passes through the administrator's review. The Login page verifies whether the user can gain entry into the site through their email and password (Figure 7).

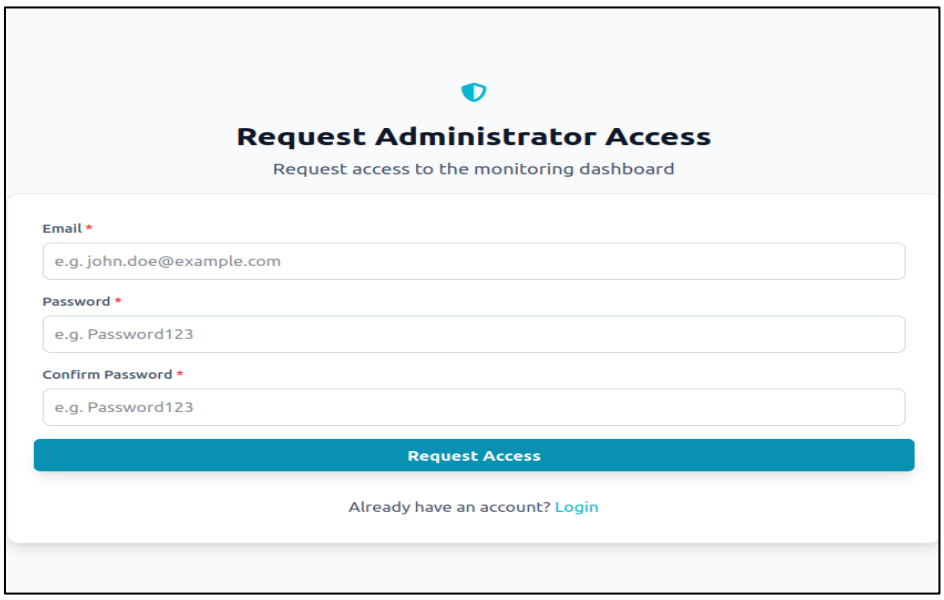

Figure 6 Registration Page

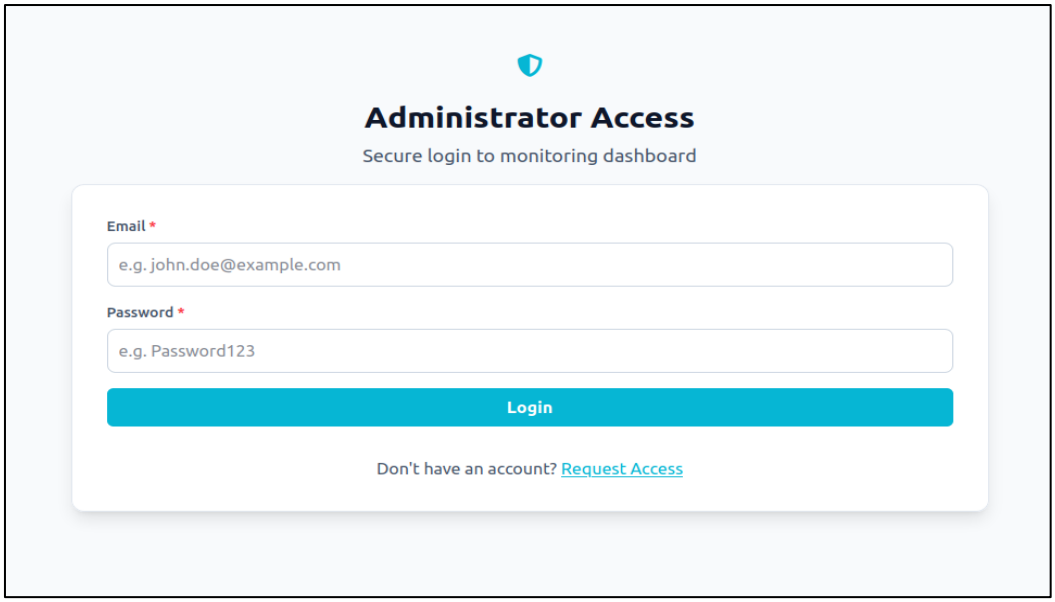

Figure 7 Login Page

4.3. Security Dashboard

Security Dashboard is the main dashboard of the system used for monitoring purposes (Figure 8.) In particular, a recent alerts panel presents information regarding description of an alert, hostname, IP address, user, rule ID, its severity, and timestamp. Information on CPU usage, memory usage, disk usage, network usage, and system uptime is presented in a system resources panel. In addition, an OSSEC Services panel provides information on status of such OSSEC components as detection engine, log collector, monitor daemon, remote daemon, syscheck daemon, and mail daemon.

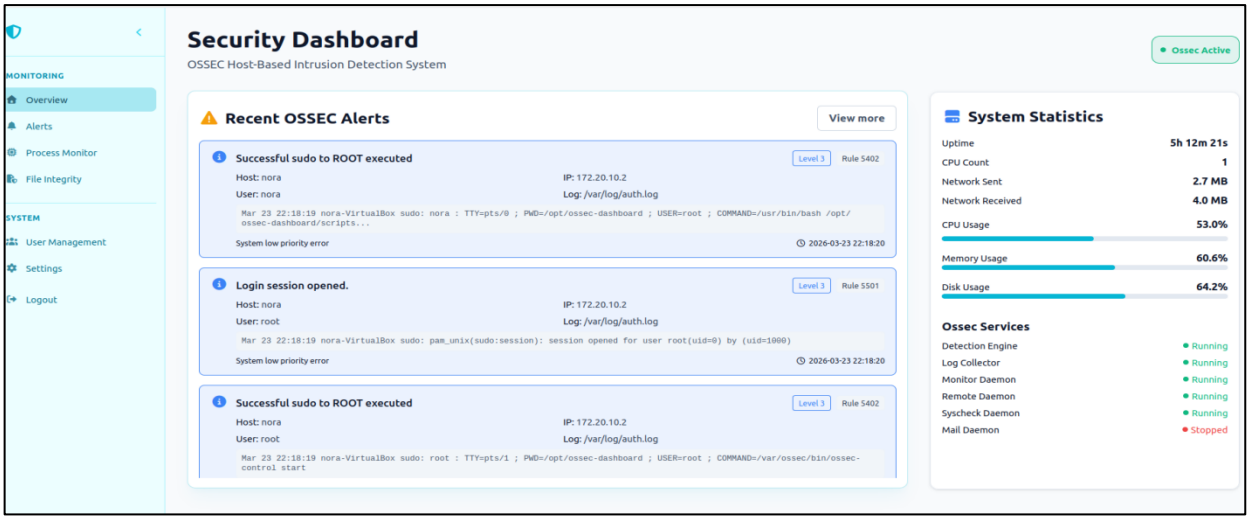

Figure 8 Security Dashboard

4.4. Alert Management

Alert Management is a page used to gain access to alerts generated by OSSEC system. Here, one can filter alerts through a search box using description, hostname, rule ID, or user parameters. In addition, alerts can be filtered according to their severity – informational, moderate, high, and critical. The table of alerts shows important metadata for each one.

4.5. Process Monitoring

This page presents all currently running processes of the monitored computer. There is a field for filtering processes by name, command, username, and PID, as well as a field for filtering by statuses. The processes' table includes name, process ID, parent ID, whitelist statuses, username, CPU load, memory consumption, and age of processes.

4.6. File Integrity Monitoring

This page provides information on all files under monitoring with the use of the syscheck module of OSSEC. There is a field for narrowing down files by paths, and there is a field for narrowing down files with certain integrity statuses. The table includes paths to files, verification statuses, size, permission mode, modification date, and current and baseline hashes.

4.7. User Management

User Management page allows administrators to manage user accounts' parameters. This page has a field for searching for user account by username, the field for selecting status, and a directory where all users that have been registered are listed for approval, disabling, and changing their roles. The OSSEC Configuration page allows administrators to set parameters of OSSEC monitoring through the web interface.

5. System Testing

System testing analysed the HIDS as a complete system under artificial intrusion conditions. The black box testing was done to test the system behaviour in relation to its input and output without analyzing the system code. The essential activities such as user login, access authorization, alert notification and visualization, process monitoring, and file integrity checking were all covered during the testing process (Table 4 to Table 8).

5.1. Authentication Tests

Table 4 Blackbox Test — Authentication Login

S/N	Test Case Description	Result
1	Login with valid registered email and correct password — session created and user redirected to dashboard.	Pass
2	Login with correct email but wrong password — access denied and error message displayed.	Pass
3	Login with email not registered in the database — system rejected credentials and access denied.	Pass

5.2. Access Request Tests

Table 5 Blackbox Test — Request Access

S/N	Test Case Description	Result
1	Submit access request with valid email and password — account created with pending unapproved status in database.	Pass
2	Attempt login with credentials of an unapproved account — system rejected login attempt and denied access.	Pass
3	Administrator approves pending account — account activated and user able to log in successfully.	Pass

5.3. Alert Detection Tests

Table 6 Blackbox Test — Alerts Route

S/N	Test Case Description	Result
1	Modify files in /etc directory — alert generated displaying affected file path and hash mismatch.	Pass
2	Log in using root account — high-privilege login alert produced with correct rule ID and log source.	Pass
3	Simulate repeated failed login attempts — brute force alert triggered and displayed with correct severity level.	Pass
4	Filter alerts by critical severity — only critical alerts returned in the display.	Pass

5.4. Process Monitoring Tests

Table 7 Blackbox Test — Processes Route

S/N	Test Case Description	Result
1	Execute a process outside the expected whitelist baseline — process flagged and displayed with associated metadata.	Pass
2	Filter processes by suspicious status — only processes matching the selected status returned and displayed.	Pass

5.5. File Integrity Monitoring Tests

Table 8 Blackbox Test — Files Route

S/N	Test Case Description	Result
1	Modify a file in the /etc monitored directory — file flagged with hash mismatch and tamper alert displayed.	Pass
2	Filter monitored files by tampered status — only affected files returned and displayed correctly.	Pass

6. Conclusion

This study developed and deployed an intrusion detection system that runs on OSSEC software but uses a bespoke web interface. The system is capable of detecting any unauthorized changes made to files, any unusual activity associated with processes running on the system, and even log-related attacks. The use of the signature-based technique was found to be suitable for this kind of environment, as it provided clear and precise information regarding potential risks in real-time. With respect to its architecture, this solution was based on a three-layer client-server model, which resulted in negligible overhead for end devices.

There are certain limitations to this solution, which include reliance only on the signature-based technique, thus failing to identify zero-day attacks. Also, the use of SQLite database is likely to become a limitation in a high-throughput scenario where numerous devices need to communicate with the server simultaneously. A future enhancement to this system may incorporate integration of an anomaly-based or machine-learning-based detection technique alongside the current one, possible integration of the current system with a SIEM platform for cross-host event correlation, and replacing SQLite database with PostgreSQL or Elasticsearch.

Compliance with ethical standards

Disclosure of conflict of interest

The authors declare that they have no conflict of interest regarding the publication of this paper.

References

[1] Alazab, M., Rm, S., Parimala, M., Maddikunta, P. K. R., Gadekallu, T. R., & Pham, Q. V. (2022). Multidirectional LSTM-based network intrusion detection system. IEEE Access, 10, 102481–102490. https://doi.org/10.1109/ACCESS.2022.3207047

[2] Aldribi, A., Traore, I., Moa, B., & Nwamuo, O. (2020). Hypervisor-based cloud intrusion detection through online multivariate statistical change tracking. Computers & Security, 88, 101619. https://doi.org/10.1016/j.cose.2019.101619

[3] Alghamdi, R., Bellaiche, M., & Bhattacharya, S. (2021). An adaptive intrusion detection and prevention system for Internet of Things. International Journal of Wireless Information Networks, 28(4), 460–480. https://doi.org/10.1007/s10776-021-00537-3

[4] Martins, I., Resende, J. S., Sousa, P. R., Silva, S., Antunes, L., & Gama, J. (2022). Host-based IDS: A review and open issues of an anomaly detection system in IoT. Future Generation Computer Systems, 133, 95–113. https://doi.org/10.1016/j.future.2022.03.001

[5] Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2021). Survey of intrusion detection systems: Techniques, datasets and challenges. Cybersecurity, 2(1), 20. https://doi.org/10.1186/s42400-019-0038-7

[6] Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., & Ahmad, F. (2021). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. Transactions on Emerging Telecommunications Technologies, 32(1), e4150. https://doi.org/10.1002/ett.4150

[7] Abdullahi, M., Baashar, Y., Alhussian, H., Alwadain, A., Aziz, N., Capretz, L. F., & Abdulkadir, S. J. (2022). Detecting cybersecurity attacks in internet of things using artificial intelligence methods. Electronics, 11(2), 198. https://doi.org/10.3390/electronics11020198

[8] Sarker, I. H., Furhad, M. H., & Nowrozy, R. (2021). AI-driven cybersecurity: An overview, security intelligence modelling and research directions. SN Computer Science, 2(3), 173. https://doi.org/10.1007/s42979-021-00557-0

[9] Faker, O., & Dogdu, E. (2020). Intrusion detection using big data and deep learning techniques. In Proceedings of the 2020 ACM Southeast Conference (pp. 86–93). ACM. https://doi.org/10.1145/3374135.3385284

[10] Ozkan-Okay, M., Samet, R., & Aslan, O. (2021). A comprehensive systematic literature review of intrusion detection systems. IEEE Access, 9, 157727–157763. https://doi.org/10.1109/ACCESS.2021.3129424

[11] Satilmis, H., Demirci, M., & Ozkan-Okay, M. (2024). A systematic literature review of host-based intrusion detection systems. Computers & Security, 138, 103670. https://doi.org/10.1016/j.cose.2023.103670

[12] Jiang, W., Li, Z., Luo, Y., Huang, C., & Zhang, Y. (2026). HIDS based on audit log analysis with structural provenance graph learning. Expert Systems with Applications, 245, 123021. https://doi.org/10.1016/j.eswa.2025.123021

[13] Park, D., Kim, S., Kwon, H., Shin, D., & Shin, D. (2021). Host-based intrusion detection model using Siamese network. IEEE Access, 9, 76614–76623.

[14] Jiang, J., Chu, H., & Tian, D. (2026). A novel host-based intrusion detection approach leveraging audit logs. Future Generation Computer Systems, Vol. 174.

[15] Satılmış, H., Akleylek, S., & Yüce Tok, Z. (2024). A systematic literature review on host-based intrusion detection systems. IEEE Access. DOI: 10.1109/ACCESS.2024.3367004

[16] Alghamdi, R., & Bellaiche, M. (2022). Evaluation and selection models for ensemble intrusion detection systems in IoT. IoT, 3(2), 285–314. https://doi.org/10.3390/iot3020017