

Cronjob Based Event Tracking and Access Control for Apache Webserver Intrusion Prevention

Paul Ndudiri Ohia, Emmanuel Waribo Oti

1, Department of Computer Science, University of Port Harcourt Choba, Nigeria

2, Department of Computer Science, University of Port Harcourt Choba, Nigeria

ABSTRACT

Apache webserver remain important platforms for deploying web applications, but their public-facing nature exposes them to repeated attacks such as SQL injection, brute-force attempts, proxy abuse, bot traffic, spam requests, and unauthorized execution of server-side scripts. This completed study integrated cronjob-based event tracking with access control for Apache webserver intrusion prevention. The developed system uses cronjob scheduling to activate log inspection, intrusion detection scripts, IP and proxy filtering, administrator notification, and access-control enforcement for protected resources. The work responded to the weakness of an existing web-based intrusion detection system that mainly recorded abnormal events without strong real-time notification, resource-aware scheduling, or direct restriction of suspicious requests. The completed system was tested against common webserver threat categories, including SQL injection, brute-force attempts, spam traffic, proxy activity, and bot access. The result showed that proxy traffic recorded the highest number of attempts, followed by spam and SQL injection. The implemented dashboard also provided login-based access control, detected-log reporting, alert display, visitor analytics, threat-location information, and administrator notification support. The study confirms that combining cronjob scheduling with access-control enforcement improves Apache intrusion prevention by enabling the server to observe, classify, restrict, record, and alert in a coordinated process.

Keywords:- Apache Webserver, Cronjob Scheduling, Access Control, Intrusion Prevention, Event Tracking, Server Logs

1. INTRODUCTION

Webserver application environments have become central to online banking, e-commerce, education, social networking, organizational portals, and cloud-based services. The same growth has increased the number of entry points available to attackers. A webserver is not only a platform that serves pages; it is also a gateway through which users, scripts, databases, and network services interact.

Apache webserver environments are commonly used because they are flexible, widely supported, and compatible with different operating systems. However, the source work shows that this popularity creates risk when administrators depend only on manual log review or delayed intrusion reports. Attacks such as SQL injection, brute-force login attempts, spam requests, proxy traffic, and bot activity may continue before an administrator notices the pattern.

This paper reports the completed cronjob security system in which access control was added as a direct part of Apache intrusion prevention. Access control determines who or what can reach protected resources after authentication and during request processing. In line with the OWASP 2021 emphasis on broken access control as a leading web application risk, the developed system treats access restriction as a practical partner to intrusion detection rather than a separate afterthought.

2. RELATED WORK

Recent web application security guidance shows that access control and logging are still major concerns. OWASP (2021a) ranked broken access control as the first category in

the OWASP Top 10 because many applications fail to enforce proper restrictions on users, roles, functions, and object-level resources. OWASP (2021b) also identified security logging and monitoring failures as a risk because attacks may persist when events are not captured, reviewed, and escalated quickly.

NIST's Zero Trust Architecture (Rose et al., 2020) also supports the need to make access decisions continuously rather than assuming that a request is trusted because it has reached an internal resource. Although zero trust is broader than Apache configuration, its principle is useful for webserver intrusion prevention: every request should be evaluated based on identity, source, resource sensitivity, and observed behaviour.

Intrusion detection research within 2020 and 2021 also emphasizes the value of log analysis, real-time traffic inspection, and behaviour monitoring. Kim et al. (2020) presented AI-IDS for real-time web intrusion detection, while Farooq et al. (2020) proposed webserver attack detection using machine learning and webserver logs. Zhong et al. (2021) further showed that security log analysis can support IDS warning generation. These studies support the need for systems that do not merely store logs but convert them into actionable security controls.

The main chapter work reviewed earlier intrusion detection systems that used signature matching, role-based checks, pattern matching, and log storage. The gap identified in the chapter is that the existing system detected intrusion attempts but did not provide strong immediate alerting, event-triggered scheduling, or access-control restriction for

suspicious requests. This paper therefore positions cronjob scheduling and access control as a combined improvement.

2.1 CHALLENGES OF THE EXISTING SYSTEM

The existing system reviewed in the main work provided intrusion detection for a webserver environment, but it had limitations that motivated the completed cronjob-based access-control system. These limitations are summarized below.

- I. Delayed administrator response: the existing system recorded intrusion attempts but did not provide a strong instant notification mechanism for proactive response at the time of occurrence.
- II. Continuous resource usage: security checking depended heavily on background log monitoring, which could increase server resource consumption in shared or quota-limited hosting environments.
- III. Weak prevention layer: the previous approach detected abnormal activity but did not sufficiently restrict suspicious IP addresses, proxy requests, bots, or unauthorized users from protected resources.
- IV. Limited access-control integration: role-based detection existed, but access control was not treated as a complete enforcement layer for authentication, authorization, and resource protection.
- V. Manual log interpretation: raw server logs were difficult for administrators to inspect quickly, especially during repeated attacks or high request volume.
- VI. Poor communication workflow: without SMS and email alerting, the administrator could miss events that required immediate action.

Existing Challenge	Improvement Introduced
Intrusion records without fast response	Email and SMS notification to alert the administrator.
High background monitoring load	Cronjob scheduling activates checks based on event or time.
Detection without enough restriction	Access control blocks or limits suspicious sources.
Manual log interpretation	GUI dashboard presents detected logs and analytics.
Proxy, bot, and spam traffic	IP/proxy filtering and ban logic reduce repeated access.

3. METHODOLOGY

A. Research Design

The study followed a design and development methodology. The source work analysed the Apache webserver intrusion detection process, identified weaknesses, and developed an improved system. The implemented system combines cronjob scheduling, IDS scripts, IP/proxy filtering, access control, graphical log presentation, and communication channels.

B. System Architecture

The implemented architecture begins when a user or automated agent sends a request to the Apache webserver application environment. The request is logged through the webserver's access and error logging mechanism. A cronjob event tracker activates security scripts based on scheduled timing or event patterns. The IDS script analyses the request, compares it with known suspicious patterns, and sends the result to the database and administrator dashboard.

Access control is placed before and after detection. Before a protected resource is served, the system checks whether the source, user role, and request behaviour are permitted. After suspicious behaviour is detected, the system can restrict the source, block proxy requests, deny unauthorized access, and preserve the log for review.

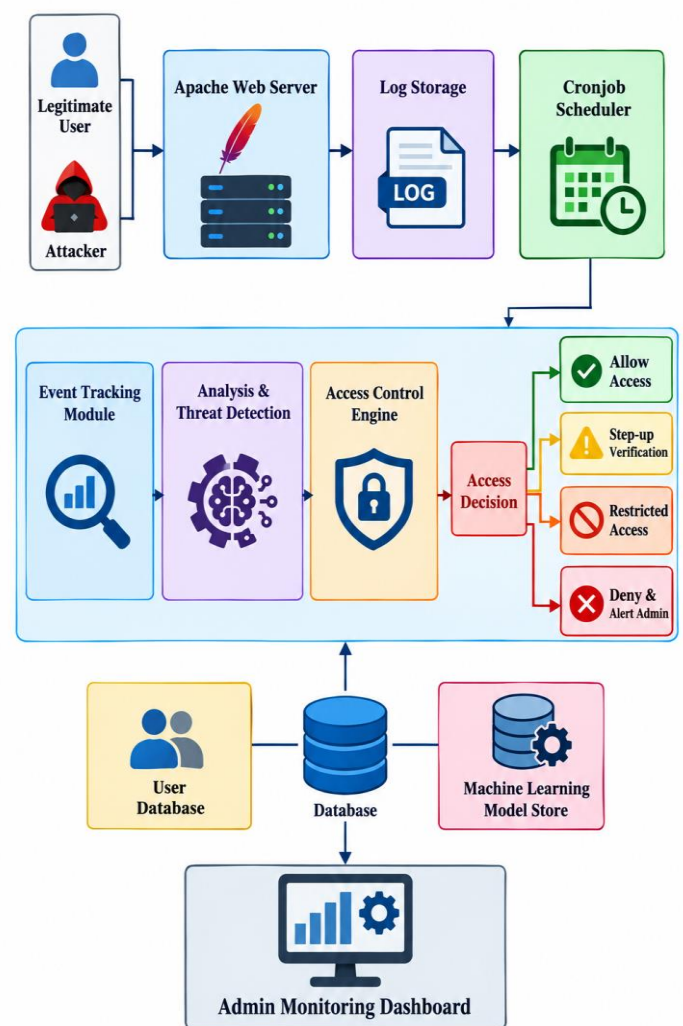


Figure 1: Proposed system architecture

The proposed architecture presents a Cronjob-based event tracking and access-control framework for Apache webserver intrusion prevention. Incoming requests from legitimate users or potential attackers are handled by the Apache web server, while generated access and event logs are stored for security analysis.

C. Cronjob Event Tracking

Cronjob scheduling was used because it is a standard server-side mechanism for running commands and scripts at defined times. In the completed system, cronjobs were configured to review logs, summarize incidents, initialize monitoring services, and trigger IDS scripts when relevant server events occurred. This reduced the need for the administrator to monitor logs manually at all times.

Cronjob Option	Security Use
@reboot	Initialize security services and verify that IDS scripts are active.
@hourly	Inspect recent access logs for repeated attack patterns.
@daily	Generate summary records for administrator review.
Event-triggered script	Run the IDS module when abnormal request behaviour is detected.

D. Access-Control Process

The access-control process uses authentication status, user role, client IP address, proxy status, request pattern, and requested resource sensitivity to decide whether a request should proceed. This supports OWASP's 2021 guidance on preventing broken access control by ensuring that users cannot reach resources or functions outside their permission level.

When a request fails the access-control rule, the system denies access, stores the event in the detection database, and can trigger the notification component. When a request passes the access-control check but later behaves suspiciously, the IDS module can still classify the event and recommend restriction.

E. Algorithm

Step 1: Load the Apache webserver application environment. Step 2: Capture the request event and write it to the appropriate log. Step 3: Activate the cronjob IDS process. Step 4: Check access-control rules for user role, IP address, proxy status, and protected resource. Step 5: Analyse the request using IDS scripts. Step 6: If the request is normal, allow processing; if suspicious, deny or restrict access. Step 7: Store the record in the database and notify the administrator through SMS or email.

4. RESULTS AND DISCUSSION

The source implementation tested the developed system against common webserver threats and recorded detected attempts. The result categories include SQL injection, brute force, spam, proxy activity, and bots. These categories are important because they represent both application-level attacks and automated access behaviours that can consume server resources or bypass normal user flows.

Detected Threat	Total Attempts	Access-Control Response
-----------------	----------------	-------------------------

SQL Injection	4	Reject suspicious input pattern and preserve log record.
Brute Force	1	Restrict repeated login attempts and alert administrator.
Spam	10	Limit source activity and flag repeated abnormal requests.
Proxy	18	Block or challenge proxy traffic before sensitive access.
Bots	2	Record automated behaviour and restrict repeated access.

Proxy traffic recorded the highest number of attempts, followed by spam and SQL injection. This supports the need for access control because intrusion detection alone may identify the source, but access control determines whether the source should continue reaching server resources.

The completed system also improved administrative visibility. Instead of reviewing raw text logs only, the administrator used the GUI dashboard to see detected logs, visitor analytics, attack categories, host information, and notification records. This made the system more practical for real webserver management.

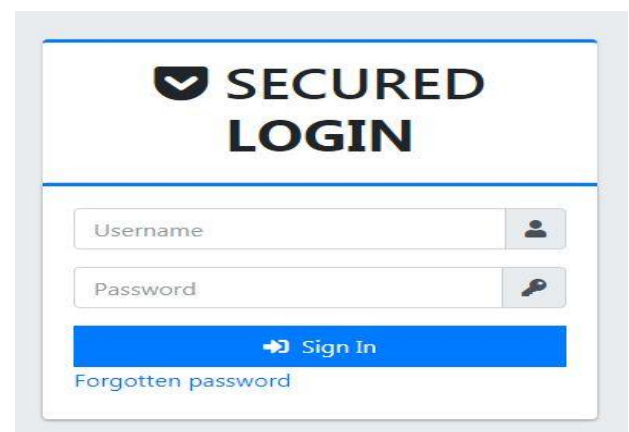


Figure 1: login access-control form.

1. Figure 1 shows the login access-control interface used to validate administrators or authorized users before they can reach protected monitoring functions. The screen demonstrates the first enforcement point of the system, where access is controlled before intrusion logs, notification records, and administrative functions are exposed.

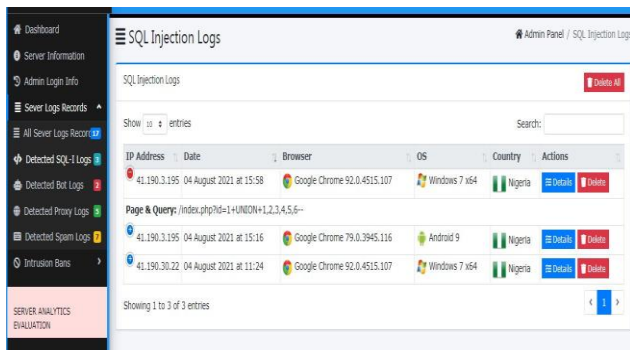


Figure 2: detected logs dashboard.

2. Figure 2 presents the detected logs dashboard generated by the completed system. The dashboard converts Apache log events and IDS outputs into readable records so that the administrator can review suspicious requests, attack categories, source information, and response status without manually searching raw server log files.

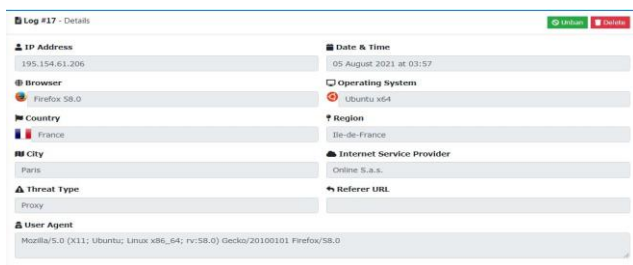


Figure 3: alert, analytics, threat-location.

3. Figure 3 shows the alert and analytics view, including notification output, visitor analytics, and threat-location information. This screen supports security decision-making by showing where suspicious activities are coming from, how they are grouped, and which events require administrator response.

5. SUMMARY

This paper presented the completed cronjob-based Apache webserver intrusion prevention system with access control as a core security function. The implemented work provided intrusion detection scripts, cronjob scheduling, IP/proxy filtering, GUI log reporting, communication alerts, and access-control enforcement. The completed study explains how these components work together to strengthen Apache webserver protection.

6. CONCLUSION

The study concluded that cronjob event tracking improved Apache webserver intrusion prevention when combined with access control. Cronjob scheduling reduced the burden of continuous manual monitoring, while access control restricted suspicious users, sources, and request patterns. Together, they provided a more complete approach than detection alone because the system observed, classified, restricted, recorded, and alerted.

7. RECOMMENDATIONS

- I. Webserver administrators should combine intrusion detection with access-control policies rather than relying only on log storage.
- II. Cronjob schedules should be tuned to the hosting environment so that security checks do not overload shared server resources.
- III. Access-control rules should cover IP address, proxy behaviour, user role, authentication state, and protected resource paths.
- IV. The dashboard should continue to use screenshot-supported reporting so administrators can interpret detected logs quickly during review and audit.
- V. Further work should test the model against larger datasets, more attack categories, and multiple webserver frameworks.

REFERENCES

- [1]. Bach-Nutman, M. (2020). Understanding the Top 10 OWASP vulnerabilities. arXiv preprint.
- [2]. Farooq, M. S., and Anjum, S. (2020). Web Server Attack Detection using Machine Learning. International Conference on Cyber Warfare and Security. <https://doi.org/10.1109/ICCWS48432.2020.9292393>
- [3]. Kim, A. C., Park, M., and Lee, D. H. (2020). AI-IDS: Application of Deep Learning to Real-Time Web Intrusion Detection. IEEE Access, 8, 70245-70261. <https://doi.org/10.1109/ACCESS.2020.2986882>
- [4]. OWASP Foundation. (2021a). A01:2021 - Broken Access Control. OWASP Top 10. https://owasp.org/Top10/A01_2021-Broken_Access_Control/
- [5]. OWASP Foundation. (2021b). A09:2021 - Security Logging and Monitoring Failures. OWASP Top 10. https://owasp.org/Top10/A09_2021-Security_Logging_and_Monitoring_Failures/
- [6]. OWASP Foundation. (2021c). OWASP Top 10:2021. <https://owasp.org/Top10/>
- [7]. Rose, S., Borchert, O., Mitchell, S., and Connelly, S. (2020). Zero Trust Architecture. NIST Special Publication 800-207. <https://doi.org/10.6028/NIST.SP.800-207>
- [8]. Zhong, W. (2021). A Security Log Analysis Scheme Using Deep Learning Algorithm for IDSs in Social Network. Security and Communication Networks, 2021, Article ID 5542543. <https://doi.org/10.1155/2021/5542543>