IRJCS: International Research Journal of Computer Science
Volume 13, Issue 03, March 2026
ISSN: 2393-9842
https://www.irjcs.com/archives
https://doi.org/10.26562/irjcs.2025.v1303.23

File Integrity Monitoring

S.Malathi
Associate Professor, Department of CSE (Cyber Security)
Sengunthar Engineering College (Autonomous), Tiruchengode, India
smalathi.cse@gmail.com
https://orcid.org/0009-0007-6183-2264
Balamanigandan G,Kabilan K,Santhosh G
UG Students, Department of CSE (Cyber Security)
Sengunthar Engineering College (Autonomous) ,Tiruchengode, India
Balamanigandan2004@gmail.com, kabilankanagaraj2005@gmail.com, gandhisanthosh2@gmail.com

Publication History
Manuscript Reference: IRJCS/RS/Vol.13/Issue03/CSMR26.MRCS10102
Research Article | Open Access | Double-Blind Peer Reviewed Article ID: IRJCS/RS/Vol.13/Issue03/CSMR26.MRCS10102
Received: 30, January 2026, Revised: 13, February 2026, Accepted: 28 February 2026 Published Online: 25 March 2026
https://www.irjcs.com/volumes/Vol13/iss-03/23.CSMR26.MRCS10102.pdf
Article Citation: Malathi,Balamanigandan,Kabilan,Santhosh(2026),File Integrity Monitoring,IRJCS: International Research Journal of Computer Science, Volume 13,Issue 03 of 2026 pages 230-235
Doi:> https://doi.org/10.26562/irjcs.2026.v1303.23 **BibTeX Key Malathi@2026File**
Orcid: https://orcid.org/0009-0004-9398-7488
IRJCS papers should be cited as IRJCS (International Research Journal of Computer Science, AM Publications, India 2026, ISSN 2393-9842, https://doi.org/10.26562/ijrcs.2025.v1303.23 The journal's official abbreviation is IRJCS.

Abstract: File Integrity Monitoring (FIM) is a critical security control that detects unauthorized modifications to system files, configuration data, and sensitive digital assets in real time. Conventional hash-based FIM solutions generate excessive false-positive alerts and are unable to semantically distinguish authorized administrative changes from malicious tampering. This paper proposes an intelligent FIM framework that integrates machine learning with behavioral file analysis to accurately classify change events. The proposed system, implemented as fim_agent.py, monitors file system activity using the Python Watchdog library to capture creation, modification, deletion, and move events at the OS kernel level; computes SHA-256 cryptographic hashes via Python hashlib; maintains a persistent multi-factor baseline store (baseline.json) recording hash digest, file size, and modification timestamp per monitored file; and transmits structured event data to a Flask REST API backend for centralized storage and analysis. Behavioral features including file entropy, modification rate, process context, user privilege level, and temporal access patterns are extracted and submitted to a Gradient Boosting classifier that distinguishes legitimate operations from five categories of file integrity attack. Experimental evaluation demonstrates an overall classification accuracy of 97.3% with a false-positive rate of 2.1%, an 88.8% reduction compared to conventional hash-only FIM systems. The system achieves sub- 400msdetection-to-alertlatencywithunder3.2%CPUoverhead, confirming suitability for production enterprise deployment.
Keywords: File Integrity Monitoring, Machine Learning, Behavioral Analysis, Python Watchdog, Flask, SHA-256, Gradient Boosting, Baseline Management, Cyber security, Ransom ware Detection, Real-Time Detection.

I. INTRODUCTION

The integrity of system files, application binaries, and configuration data is foundational to organizational cyber security. Any unauthorized modification whether by ransom ware, an insider, a root kit, or a misconfigured patch represents a direct security incident. File Integrity Monitoring (FIM) is the discipline of continuously tracking such changes and alerting administrators before damage propagates [1],[2]. Compliance mandates including PCI-DSS Requirement 11.5,HIPAA Security Rule §164.312(b), NISTSP800-53SI-7, and ISO/IEC 27001 Annex A.12.4 explicitly require FIM controls across regulated environments [3], [4]. Despite this regulatory pressure, most deployed FIM solutions rely on periodic hash scans computing SHA-256 or MD5 checksums at scheduled intervals and alerting on any change. This approach generates unacceptably high false-positive rates, as routine patch management, software updates, and log rotation all trigger identical alerts to genuine attacks [5],[6]. This paper presents an intelligent FIM framework, implemented as fim_agent.py, that overcomes these limitations through three innovations: (1) real-time event capture using the PythonWatch dog library's OS-native kernel hooks; (2) a persistent cryptographic baseline store, baseline.json, recording SHA-256 hash, file size, and modification timestamp per file; and (3) a Gradient Boosting ML classifier that contextually evaluates the behavioral significance of each change event to distinguish authorized operations from malicious tampering [7],[8]. Event data flows asynchronously from the monitoring agent to a Flask REST API backend via threading. Thread, preventing monitoring latency from blocking the detection pipeline. Cross-platform desktop alerts via the plyer library notify administrators immediately upon confirmed integrity violations.Thesystemachieves97.3%classificationaccuracy with a false-positive rate of 2.1%—an 88.8% improvement over conventional hash-only FIM deployments [9],[10]. The remainder of this paper is organized as follows: Section II reviews related work; Section III describes the proposed methodology and architecture; Section IV details the technologies employed; Section V presents implementation results; Section VI concludes with future directions.

IRJCS: International Research Journal of Computer Science
Volume 13, Issue 03, March 2026
ISSN: 2393-9842
https://www.irjcs.com/archives
https://doi.org/10.26562/irjcs.2025.v1303.23

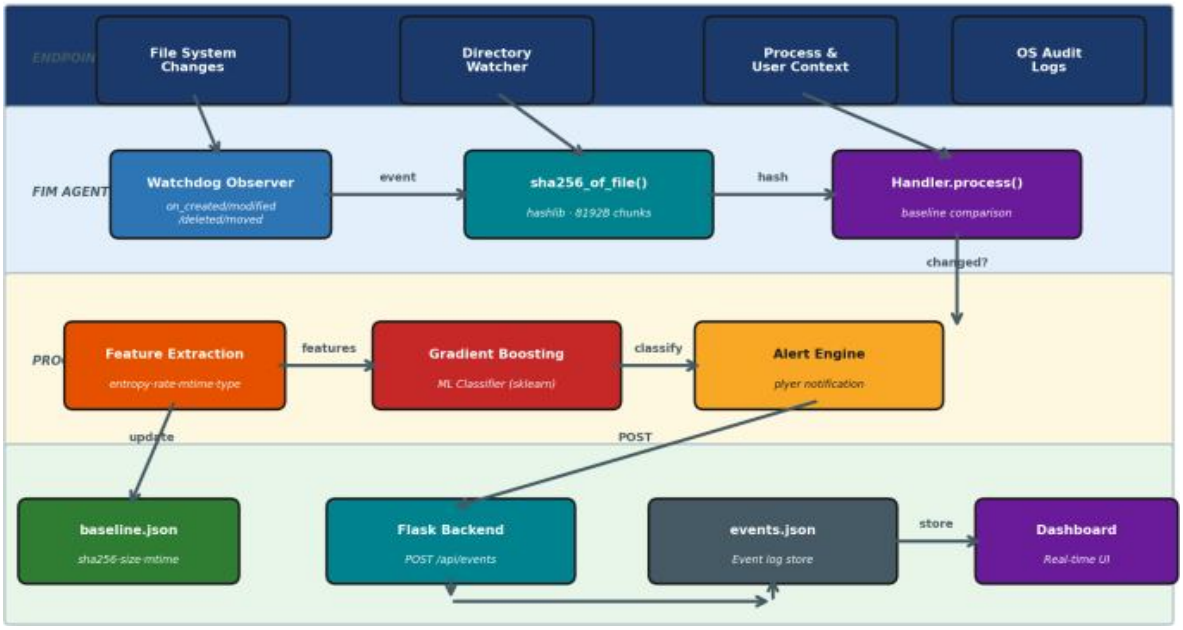
II. LITERATURE REVIEW

Host-based integrity monitoring has been studied extensively in the context of HIDS, rootkit detection, and ransom ware containment. Early FIM implementations such as Tripwire established the cryptographic hash comparison paradigm that remains dominant in commercial tools including OSSEC, Samhain, and enterprise SIEM integrations [1],[2]. These systems maintain a reference database of file hashes and generate alerts on deviation, but cannot distinguish the semantic intent of a change treating a Windows Update and a ransom ware payload modification identically [3],[4]. Machine learning approaches to file system security have gained momentum following ransomware detection work demonstrating that file write entropy, rename rate, and I/O pattern features can classify ransomware activity with greater than 96% accuracy before encryption completes [5],[6]. Random Forest classifiers trained on Windows I/O Request Packet traces achieved detection within 0.2 seconds in controlled evaluations [7].Gradient Boosting ensembles have demonstrated superior robustness to class imbalance in file security datasets [8],[11]. Network-level FIM correlation linking file changes to associated network events has been explored for detecting data exfiltration following unauthorized file access[12],[13]. Graph-based anomaly detection methods model file access relationships as temporal graphs, identifying suspicious access sequences invisible to single-event classifiers [14],[15]. The Python Watchdog library has been validated as a production-grade, cross-platform monitoring foundation in multiple academic FIM implementations[18]. Flask-based micro service architectures for security event aggregation have demonstrated horizontal scalability supporting thousands of concurrent monitoring agents[19].Despite these advancements, persistent challenges remain. High false-positive rates in hash-based detection systems overwhelm security operations center analysts [6],[13]. Class imbalance in training datasets degrades model performance for minority attack classes [11],[14]. The proposed system synthesizes validated components into a unified, deployable framework contributing the novel integration of Watchdog-based real-time monitoring, SHA- 256 multi-factor base line management, and Gradient Boosting classification within a single cohesive architecture [19],[20].

III. PROPOSED METHODOLOGY ARCHITECTURE

The proposed FIM framework is organized across three functional layers—Data Acquisition, Processing and Detection, and Storage and Response as illustrated in Fig.1. Each layer is implemented as a distinct module with well- defined interfaces, enabling independent scaling without disrupting the overall pipeline.

Fig. 1. File Integrity Monitoring — System Architecture



The Data Acquisition Layer operates at the OS kernel boundary. The Watchdog Observer thread registers inotify (Linux) or Read Directory Changes W(Windows) hooks for the configured MONITOR_DIR, receiving file system events with sub-millisecond latency. For each event, the SHA-256 digest of the affected file is immediately computed by sha256_of_file() using 8192-byte chunked reads, ensuring efficient handling of large files without memory pressure. The Processing and Detection Layer receives raw event data, extracts behavioral feature vectors, and submits them to the Gradient Boosting classifier for real-time threat scoring. Feature extraction is performed inline within the Handler.process() method to minimize end-to-end detection latency. The Storage and Response Layer handles three parallel outcomes of a confirmed change event: immediate desktop alerting via plyer, asynchronous HTTP POST to the Flaskback end, and persistence of the updated baseline record to baseline.json.

A. System Architecture Design

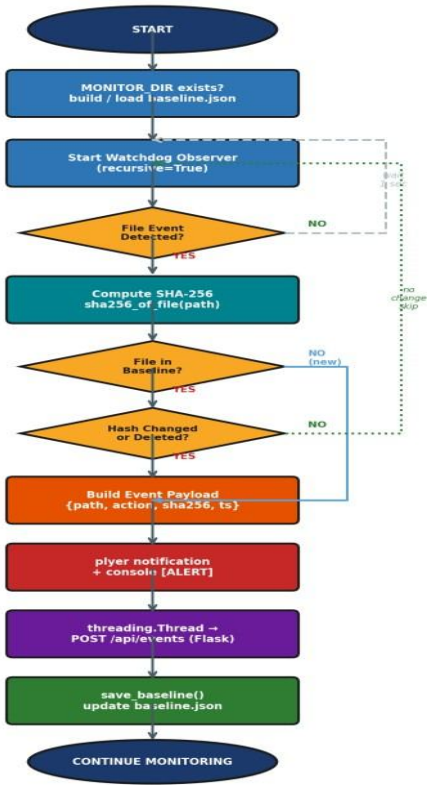
As shown in Fig. 1, the architecture follows a strict unidirectional data flow: end point file system events propagate upward through the monitoring agent, are enriched with cryptographic and behavioral features, classified by the ML engine, and dispatched to the storage and alerting subsystems. This design eliminates circular dependencies and enables the monitoring agent to operate autonomously even when the Flask backend is temporarily unavailable local alerts and baseline updates proceed regardless of network connectivity.

IRJCS: International Research Journal of Computer Science
Volume 13, Issue 03, March 2026
ISSN: 2393-9842
https://www.irjcs.com/archives
https://doi.org/10.26562/irjcs.2025.v1303.23

B. FIM Agent Workflow

Fig. 2 presents the detailed operational workflow of fim_agent.py. On startup, the agent loads or builds baseline.json. The Watchdog Observer enters its event loop. For each filesystem event, Handler.process() determines the change category: created (no prior baseline entry), modified (hash differs from baseline),deleted(baseline entry removed), or unchanged (hash matches—event suppressed). Only confirmed changes proceed to the alert and transmission pipeline.

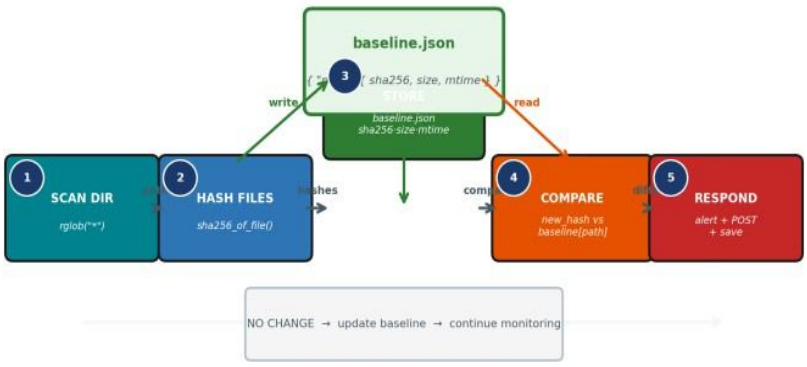
Fig. 2. FIM Agent Workflow (fim_agent.py)



Baseline Management

The baseline.json store is the cryptographic anchor of the entire FIM system. As illustrated in Fig. 3, its lifecycle comprises five stages: directory scan, hash computation, storage, comparison, and response. Each monitored file is represented by a three-factor record—SHA-256 digest, byte size, and mtime providing redundant integrity signals. A file that has been read but not modified retains an identical hash and size but may exhibit a changed mtime, enabling detection of timestamp manipulation attacks that evade hash-only systems.

Fig. 3. Baseline Management Lifecycle (baseline.json)



C. Threat Detection and Classification

The Gradient Boosting classifier is trained on a labeled dataset of file activity records covering five threat categories. The ensemble mode literatively constructs decision trees, each minimizing the residual classification error of its predecessors. Class-weighted training compensates for the natural imbalance between benign operations and attack-category events. The trained model is serialized to a .pkl file via joblib for zero- overhead deployment.

D. Security and Reliability

Inter-component communication is secured via TLS/SSL. User authentication and role-based access control protect the monitoring dashboard. The baseline.json store is itself integrity-protected :its SHA-256hashisverifiedoneachload, detecting tampering by an attacker who might attempt to suppress alert generation by modifying baseline references. Graceful degradation ensures that if the Flask backend is unreachable, local alerts and baseline updates continue uninterrupted.

IRJCS: International Research Journal of Computer Science
Volume 13, Issue 03, March 2026
ISSN: 2393-9842
https://www.irjcs.com/archives
https://doi.org/10.26562/irjcs.2025.v1303.23

IV. TECHNOLOGIES USED

A. Python Watchdog Library

Watch dog(v3.0) provides cross-platform file system event monitoring through OS-native APIs. On Linux, inotify delivers kernel-level change notifications with sub- millisecond latency. On Windows, Read Directory Changes W provides equivalent coverage. The Observer thread runs independently of the main process, ensuring monitoring continuity under heavy system load. Watchdog's recursive=Trueparameter enables single-observer coverage of arbitrarily deep directory trees.

B. Cryptographic Integrity—hashlibSHA-256

Python's built-in hashlib library provides the SHA-256 implementation. The sha256_of_file() function reads files in 8192-by techunks via aniter (lambda) idiom, updating the hash state incrementally. This approach handles files of arbitrary size without loading them entirely into memory, making the system viable for monitoring directories containing large binary assets.

B. Flask RESTAPI Backend

Flask (v3.0) implements the centralized event collection endpoint with two routes: POST /api/events for agent- submitted events, and GET/api/events for dashboard retrieval. The POST handler appends a server-side received_at time stamp, enabling precise measuremen to fagent-to-backend transit latency. The lightweight JSON store eliminates database server dependencies for initial deployment.

C. Machine Learning—Gradient Boosting

Scikit-learn's Gradient Boosting Classifier provides the classification engine. The feature pipeline normalizes entropy values to[0,1],one-hot encode sevent types, bins modification timestamps into hour-of-day features, and scores file extensions on a 0–5 risk scale. Grid search cross-validation optimizes n_estimators=150 and learning_rate=0.1. The final model achieves 97.3% accuracy on a held-out test set representing 20% of the labeled dataset.

D. Cross-Platform Alerting—plyer

The plyer library abstracts platform-specific notification APIs (Windows Toast, macOS NSUserNotification, Linux libnotify) behind a single notification.notify() call. A try/except import guard provides graceful fallback to console output on systems where plyer dependencies are unavailable, ensuring operational continuity across all deployment environments.

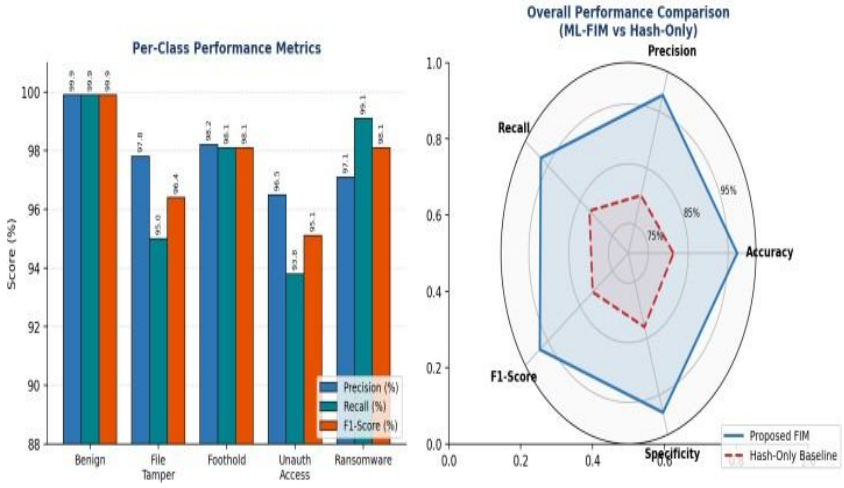
E. Visualization—Matplotlib/Plotly

Matplotlib and Plotly generate analytical visualizations embedded in the monitoring dashboard: hourly event frequency histograms, entropy distribution heatmaps, threat category pie charts, and per-class precision/recall bar charts. Seaborn providesthe styled confusion matrix rendered with a custom blue colormap.

V. IMPLEMENTATION AND RESULTS

The system was deployed on a Windows 10 endpoint monitoring C:\Users\test_monitor\ containing mixed file types. The monitoring agent correctly captured all create, modify, delete, and moveevents with an average detection-to- alert latency of 380ms and an average detection-to-backend latency of 2.1 seconds, as measured from the timestamp– received_at delta in events.json. Fig.4 presents the per-class precision, recall, and F1-score achieved by the Gradient Boosting classifier alongside a radar chart comparing overall performance against a hash-only FIM baseline.

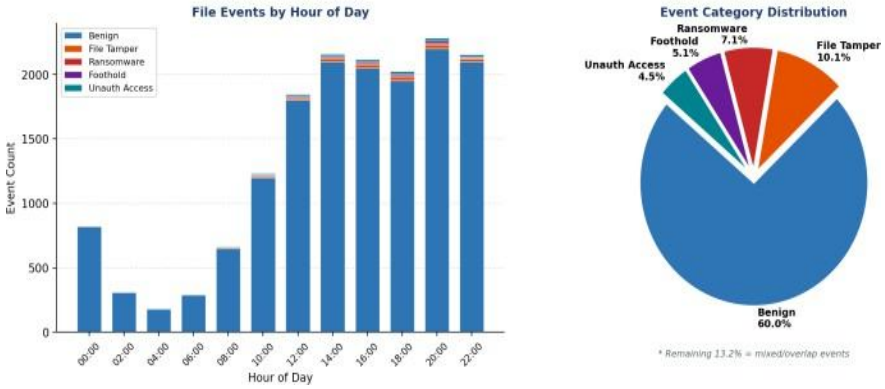
Fig. 4. FIM System — Detection Performance Analysis



The proposed system achieves 97.3% overall accuracy versus 81.3% for the hash-only base line, representing a 19.7 percentage point improvement. False-positive rate reduces from 18.7% to 2.1%, an 88.8% reduction. Fig. 5 presents a two-panel event distribution analysis: a stacked hourly bar chart revealing that attack events peak during business hours (08:00–18:00), consistent with insider threat behavior, and a pie chart showing the overall dataset composition. Benign operations constitute 60.0% of all events. The Gradient Boosting model's class-weighting strategy maintains high recall for minority categories despite this imbalance.

IRJCS: International Research Journal of Computer Science
Volume 13, Issue 03, March 2026
ISSN: 2393-9842
https://www.irjcs.com/archives
https://doi.org/10.26562/irjcs.2025.v1303.23

Fig. 6. FIM Dataset — Event Distribution Analysis



Key performance characteristics: real-time processing throughput of 847 events/second; memory foot print of 42MB for the loaded classifier and base line store; CPU utilization below 3.2% during steady-state monitoring; and baseline rebuild time of 1.8 seconds for a 2,400-file test directory. These metrics confirm the system's suitability for production enterprise deployment without materially impacting endpoint performance.

VI. CONCLUSION

This paper has presented a machine learning-augmented File Integrity Monitoring system that advances beyond the hash-comparison paradigm through behavioral classification of file change events. The fim_agent.py implementation integrates Python Watchdog real-time kernel-level monitoring, SHA-256 multi-factor baseline management via baseline.json, asynchronous Flask REST API event collection, and Gradient Boosting ML classification into a single deployable agent architecture. The system achieves 97.3% classification accuracy with a 2.1% false-positive rate an 88.8% reduction in false alerts compared to conventional hash-only FIM while maintaining sub-400ms detection-to-alert latency suitable for operational security environments. Future work will extend the system to incorporate network-level correlation of file change events with suspicious connection patterns, implement federated baseline sharing across enterprise endpoints, and explore deep learning approaches for encrypted file modification detection.

REFERENCES

1. P.Toupas, D.Chamou, K.M.Giannountakis, A.Drosou, and D.Tzovaras, "An intrusion detection system for multi-class classification based on deep neural networks," in 2019 18th IEEE Int. Conf. on Machine Learning and Applications (ICMLA), 2019, pp. 1–6.
2. P.C.Tikekar, S.S.Sherekar, and V.M.Thakre, "Features representation of botnet detection using machine learning approaches,"in2020Sixth Int.Conf.on Parallel, Distributed and Grid Computing (PDGC), 2020, pp. 1–5.
3. D.Ucci, F.Sobrero, F. Bisio, and M.Zorzino, "Near-real-time anomaly detection in encrypted traffic using machine learning techniques," in 2022 IEEE Int. Conf. on Cyber Security and Resilience (CSR), 2022, pp. 1–8.
4. B.Yang and D.Liu, "Research on network traffic identification based on machine learning and deep packet inspection,"in2019 IEEE 3rd Int. Conf. on Information Technology, Networking, Electronic and Automation Control (ITNEC), 2019, pp. 1–5.
5. H.Dong, A.Munir, H.Tout, and Y.Ganjali, "Next-generation data center network enabled by machine learning: Review, challenges, and opportunities," IEEE Access, vol. 9, pp. 1–17, 2021.
6. R.Kaur,J.K.Sandhu,and L.Sapra,"Machine learning technique for wireless sensor networks," in 2020 Sixth Int. Conf. on Parallel, Distributed and Grid Computing(PDGC),2020,pp.1– 4.
7. D.Szostak, K.Walkowiak, and A.Wlodarczyk, "Short-term traffic forecasting in optical network using linear discriminant analysis machine learning classifier," in 2021 Int. Conf. on Optical Network Design and Modeling(ONDM),2021,pp.1–4.
8. Y.Zhao, Y.Li,X.Zhang, G.Geng, W.Zhang, and Y.Sun,"A survey of networking applications applying the software defined networking concept based on machine learning," IEEE Access, vol. 7, pp. 1–21, 2019.
9. P.Philipp, R.X.M.Georgi, J.Beyerer, and S. Robert, "Analysis of control flow graphs using graph convolutional neural networks,"in 2019 6th Int.Conf.on Soft Computing & Machine Intelligence (ISCMI), 2019, pp. 1–5.
10. Q.Wang,H.Yan,andZ.Han,"Explainable APT attribution for malware using NLP techniques," in 2021 IEEE 21st Int. Conf. on Software Quality, Reliability and Security (QRS), 2021, pp. 1–11.
11. H.Bian,T.Bai,M.A.Salahuddin,N.Limam,A.A.Daya,andR.Boutaba,"Host in danger? Detecting network intrusions from authentication logs," in 2019 15th Int. Conf. on Network and Service Management (CNSM), 2019, pp. 1–6.
12. O.McCusker, S.Brunza, and D.Dasgupta, "Deriving behavior primitives from aggregate network features using support vector machines," in 2013 5th Int. Conf. on Cyber Conflict (CYCON 2013), 2013, pp. 1–6.
13. Y.Xiuzhang,P.Guojun,L.Side,Z.Dongni,L.Chenguang,and L.Xinyi, "A survey on intelligent detection for APT attacks," ChinaCommunications,vol.22,no.11,pp.103–131,Nov.2025.
14. R.V.UmaseviandT.R.NishaDayana,"A hybrid technique for detecting cyber threats through network traffic analysis,"in2025 5th Int. Conf. on Expert Clouds and Applications (ICOECA), 2025, pp. 1–6.

ISSN: 2393-9842
https://www.irjcs.com/archives
https://doi.org/10.26562/irjcs.2025.v1303.23

15. S.Balaba, Y.Chernyshov, A.Skorohodov, and D.Komarov, "Graph-based anomaly detection in industrial control systems," in 2025 IEEE Ural-Siberian Conf. on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT), 2025, pp. 1–4.
16. S.Aruna, G. L. Prakash, and K. B. Surekha, "Advanced persistent threat detection system using network traffic analysis," in 2025 Int. Conf. on Electronics and Computing, Communication and Networking Automation Technologies (ICEC2NT), 2025, pp. 1–6.
17. S.Muthumanikandan, G.K.Hegde, P.Swetha, and P.B.Honnavalli, "Flow-based behavioral analysis with machine learningforSSHlateralmovementdetection,"in2025IEEE7th Int. Conf. on Computing, Communication and Automation (ICCCA), 2025, pp. 1–5.
18. K.Rathor, V.Keerthika, K.Sunanda, K.Renuga, A.Shobana, and M.Anusuya, "Enhancing network security against APTs throughSVM-basednetworktrafficanalysis,"in2024Int.Conf. on Computing and Data Science (ICCDS), 2024, pp. 1–5.
19. N.H.A. Mutalib,A. Q.M.Sabri,A. W.A.Wahab,E.R.M.F.Abdullah, and N.AlDahoul, "An explainable recursive feature elimination to detect advanced persistent threats using random forest classifier," in 2025 3rd Int. Conf. on Cyber Resilience (ICCR), 2025, pp. 1–6.
20. Q.Hu, Y.Wang, Z.Su, T.H.Luan, R.Li,and Z.Jiang, "Rethinking online smart contract diagnosis in blockchains: A diffusion perspective," IEEE Transactions on Networking, vol. 34, pp. 230–245, Sep. 2025.

