iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** III **Month of publication:** March 2026

DOI: https://doi.org/10.22214/ijraset.2026.78758

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

Encrypted Network Traffic Analysis for Real-Time Cybersecurity Threat Detection

Dr. D. Sreenivasulu[1], M. Virat Rana[2], D. Vinay Shyam[3], G. Nithin[4]
Dept. of CSE (Data Science), Institute of Aeronautical Engineering, Dundigal, Hyderabad

Abstract: ***This paper presents a real-time cybersecurity framework for Encrypted Network Traffic Analysis designed to detect malicious activity concealed inside encrypted communication channels. Modern enterprise networks rely heavily on encryption protocols such as TLS/SSL, HTTPS, QUIC, and VPN tunneling, rendering conventional deep-packet inspection methods ineffective. Threat actors now systematically abuse encryption to hide malware communications, data exfiltration pipelines, command-and-control (C2) channels, and advanced persistent threats. The proposed framework analyzes encrypted network flows without decrypting any payload, combining machine learning classification, unsupervised anomaly detection, TLS fingerprinting, behavioral flow modeling, and graph-based threat correlation to expose hidden malicious behavior. A real-time streaming pipeline, composite risk scoring engine, and explainable AI layer together deliver sub-300 ms threat detection with full decision transparency. Experimental evaluation demonstrates 94.2% classification accuracy, a ROC-AUC of 0.95, and the ability to process over 25,000 encrypted flows per second in a cloud-native Kubernetes environment. The solution is privacy-preserving, horizontally scalable, and designed for deployment in enterprise and government cybersecurity infrastructures.***

Keywords: Encrypted Traffic Analysis, TLS Fingerprinting, Cybersecurity, Machine Learning, Anomaly Detection, Behavioral Modeling, Network Security, Threat Detection

I. INTRODUCTION

Modern cybersecurity is undergoing a fundamental transformation driven by the near-universal adoption of encryption protocols across enterprise, government, and consumer networks. Standards such as TLS 1.2/1.3, HTTPS, QUIC, SSH, and encrypted VPN tunnels now protect the vast majority of network traffic worldwide.

While this shift is enormously beneficial for user privacy and data security, it has simultaneously created a critical and growing blind spot for security operations teams. Traditional inspection-based tools including next-generation firewalls, deep packet inspection appliances, and signature-based intrusion detection systems rely on reading packet payloads to identify threats. When those payloads are encrypted, these tools become effectively blind.

Cyber attackers have not missed this opportunity. They routinely route malware communications, lateral movement traffic, data theft operations, and remote command execution through encrypted channels precisely because they know that most security products cannot inspect them. Attackers go further still, actively mimicking the behavioral signatures of legitimate applications to avoid triggering any statistical anomaly detectors that might otherwise flag unusual traffic volumes or timing patterns. Advanced Persistent Threat (APT) groups in particular have developed highly refined techniques for operating entirely within encrypted channels, enabling them to maintain persistent access inside victim networks for months or years without detection.

Decrypting enterprise traffic at scale might seem like a straightforward remedy, but it introduces a cascade of problems. Privacy regulations such as GDPR, HIPAA, and local data protection laws in many jurisdictions restrict or prohibit the interception of encrypted communications. Beyond legal risk, the computational overhead of decrypting, inspecting, and re-encrypting traffic at line rate in high-throughput environments is prohibitive for most organizations. Architectural complexity, certificate management, and the elimination of end-to-end encryption guarantees further compound the challenges. These limitations have compelled the security research community to develop alternative detection strategies that do not require breaking encryption.

This paper introduces a dedicated Encrypted Network Traffic Analysis framework that operates exclusively on observable metadata packet sizes, inter-arrival times, TLS handshake parameters, flow behavioral signatures, and graph relationships between network entities to identify malicious activity without ever touching plaintext content. The system applies a layered detection architecture combining supervised machine learning, unsupervised anomaly detection, and graph-based threat correlation, all operating in real time on a streaming data pipeline. Its design prioritizes detection accuracy, processing speed, scalability, and full compliance with privacy and data protection requirements.

II. RELATED WORK

The challenge of analyzing encrypted network traffic for security purposes has attracted significant research attention over the past decade. Early work focused on traffic classification distinguishing different application types such as streaming video, web browsing, and file transfer using statistical features derived from flow metadata. While these approaches demonstrated that useful information survives encryption, their threat detection capabilities were limited to identifying broad traffic categories rather than flagging specific malicious behaviors [1][2].

Machine learning classifiers including Random Forest, Support Vector Machines, and Gradient Boosting were subsequently applied to the encrypted traffic threat detection problem with encouraging results for known malware families. However, these supervised models depend heavily on labeled training data and consistently underperform against zero-day threats and the growing category of encrypted malware that deliberately mimics the statistical behavior of legitimate applications [3][5]. The vocabulary gap between training distributions and real-world attack evolution remains a fundamental challenge for purely supervised approaches.

TLS fingerprinting emerged as a complementary detection technique. The JA3 method generates a compact hash from TLS ClientHello parameters including cipher suites, extensions, elliptic curves, and compression methods. JA3S extends this to server responses. Research demonstrated that these fingerprints are surprisingly consistent for specific client implementations and correlate reliably with malware families [8][12]. However, fingerprint evasion has become standard practice among sophisticated threat actors, who randomize handshake parameters to defeat matching against known-bad signature databases.

Deep learning architectures including Convolutional Neural Networks, Long Short-Term Memory networks, and Variational Autoencoders have been explored for modeling the sequential structure of encrypted traffic flows. These models can capture temporal patterns invisible to feature-engineering approaches but require large labeled datasets and suffer from limited interpretability, making them difficult to deploy in operational security environments where analysts must understand and validate detection decisions [4][6]. Graph-based network analytics has more recently demonstrated value in detecting coordinated attack behavior by mapping entity relationships across the network [7]. The framework in this paper unifies all of these approaches into a single production-ready detection system.

III. METHODOLOGY

The proposed methodology is structured as a multi-stage, data-driven pipeline for analyzing encrypted network traffic without decryption. Each stage is designed to extract progressively richer behavioral signals from raw network observations, culminating in a real-time threat score and alert for every monitored flow.

A. Data Collection and Ingestion

Traffic data is gathered passively from strategic collection points across the enterprise network perimeter, including edge firewalls, cloud egress gateways, VPN concentrators, SD-WAN nodes, and internal network sensors. The collection layer captures only encrypted flows conforming to TLS, HTTPS, QUIC, SSH, DTLS, and related protocols. Raw packet capture is processed by Zeek and Suricata sensors that extract structured flow-level records without retaining payload content. These records are published in real time to an Apache Kafka message bus, providing a high-throughput, fault-tolerant ingestion channel for downstream processing. Supplementary training data is sourced from publicly available encrypted malware datasets, threat intelligence platform feeds, and controlled-environment C2 traffic simulations used to ensure broad coverage of malicious behavior patterns.

B. Data Pre-processing

1) Noise Filtering: Duplicate flows, corrupted packet sequences, and incomplete TLS handshakes are removed before any feature extraction occurs, preventing misleading signals from reaching the model layer.

2) Temporal Normalization: Timestamps, inter-arrival time series, and packet burst sequences are standardized to remove clock-drift artifacts and enable consistent comparison across sessions captured at different points in the network.

3) Flow Aggregation: Individual packet records are assembled into bidirectional flow objects representing complete sessions, providing the unit of analysis for both statistical and behavioral features.

4) TLS Metadata Extraction: The ClientHello and ServerHello handshake messages are parsed to extract cipher suite lists, TLS extension identifiers, Server Name Indication (SNI) values, negotiated TLS versions, certificate chain metadata, and JA3/JA3S fingerprint hashes.

5) Feature Encoding: Categorical fields including SNI domains, cipher suite names, and protocol identifiers are numerically encoded using a combination of label encoding and frequency-based embedding to prepare them for consumption by machine learning models.

6) Sequence Preparation: Ordered packet size sequences and inter-arrival timing vectors are extracted and padded to a fixed length for input to LSTM and GRU recurrent neural network models.

C. Feature Engineering

The feature engineering stage transforms raw flow records into a rich, discriminative representation capturing multiple dimensions of encrypted session behavior.

1) Statistical Flow Features: Per-flow statistics including packet size mean, median, maximum, minimum, standard deviation, and variance; inter-arrival time distributions at multiple percentiles; traffic directionality ratios measuring inbound versus outbound byte volume; flow duration and total byte count; and packet burstiness coefficients that quantify irregular transmission patterns associated with covert channel behavior.
2) TLS Behavioral Features: TLS version and cipher suite entropy measures; deviation of observed JA3 fingerprints from application-specific baseline profiles; SNI domain reputation scores derived from threat intelligence lookups; certificate validity period anomalies; and detection of version downgrade negotiation attempts indicating potential man-in-the-middle activity.
3) Temporal Sequence Features: Fixed-length packet size and timing sequences capturing the micro-behavioral rhythm of each session, which differs characteristically between human-generated interactive traffic, automated application traffic, and malware beaconing or exfiltration patterns.

D. Model Development and Training

The detection layer deploys both supervised and unsupervised models in parallel, exploiting their complementary strengths. Supervised classifiers learn from labeled examples of benign and malicious encrypted flows, building explicit decision boundaries for known attack patterns. Fully connected deep neural networks process the engineered feature vectors, while LSTM networks consume the sequential packet representations. These models are trained on curated datasets combining real enterprise traffic, public benchmark datasets, and synthetically generated attack traffic, with care taken to prevent class imbalance from degrading minority-class recall.

Unsupervised anomaly detection models run in parallel to identify threats that fall outside the training distribution entirely. Autoencoder neural networks learn a compressed representation of normal traffic behavior; sessions that cannot be accurately reconstructed produce elevated error scores flagged as anomalous. Isolation Forest models identify statistical outliers by measuring how quickly individual flows can be isolated from the broader population through random feature partitioning. One-Class SVM and K-Means clustering provide additional independent anomaly signals. Together, these models ensure coverage of both known malware families and novel zero-day attack behaviors.

E. Anomaly Detection and Scoring

1) Autoencoder Reconstruction Error: The reconstruction loss for each session is compared against a threshold learned from normal traffic. Sessions exceeding the threshold receive a proportional anomaly contribution to their composite risk score.
2) Isolation Forest Anomaly Score: Each flow receives a normalized outlier score reflecting how statistically isolated it is from the majority of observed sessions, with lower isolation depth indicating greater anomaly.
3) TLS Fingerprint Deviation: Observed JA3 hashes are checked against per-application behavioral baselines and global malware fingerprint databases, flagging sessions where the fingerprint is inconsistent with the claimed application or matches a known threat actor toolkit.
4) Traffic Entropy Analysis: Byte-level and timing-level entropy calculations detect structured low-entropy data patterns characteristic of compressed or pre-encrypted exfiltration payloads masquerading as random-looking traffic.
5) Beaconing Pattern Detection: Spectral analysis of inter-session timing intervals identifies the highly regular clock-like communication rhythm that distinguishes malware C2 check-in traffic from human-driven session initiation patterns.

F. Real-Time Detection Pipeline

The end-to-end detection pipeline is built on a stream processing architecture designed for sub-second latency at enterprise scale. Encrypted flow metadata is ingested from the Kafka bus by Apache Flink stream processing jobs that perform real-time feature extraction and model inference in parallel. Each arriving flow passes through the feature extraction stage, supervised classifier, and anomaly detection ensemble simultaneously, with their individual outputs aggregated by a risk scoring engine into a single composite threat score.

Scores above configurable thresholds automatically generate structured alerts categorized by threat type and severity level. The alert stream feeds both the analyst dashboard and downstream SIEM and SOAR integrations for automated playbook execution. The entire pipeline is containerized and deployed on Kubernetes, enabling horizontal autoscaling in response to traffic volume fluctuations.

Fig. 1. System Pipeline and Flow Classification Diagram

G. Continuous Retraining and Adaptation

The threat landscape evolves continuously, and a static model trained once at deployment will inevitably degrade as attacker techniques change. The framework addresses this through a structured continuous learning pipeline. Security analysts review and label prioritized alerts generated by the detection engine, creating a stream of fresh, real-world training examples that reflect current attacker behavior patterns. Automated model drift monitors track key performance indicators including false positive rates, detection rates for known threat categories, and anomaly score distribution statistics. When drift is detected beyond configured thresholds, retraining jobs are triggered automatically using accumulated labeled data, with new model versions validated against held-out test sets before promotion to production.

Adversarial training techniques are incorporated into each retraining cycle, exposing models to synthetic evasion attempts including fingerprint randomization, traffic timing jitter, and payload size padding, to improve their resilience against deliberate model evasion. Feature importance analysis guides ongoing optimization of the feature engineering pipeline, ensuring that the most discriminative signals receive appropriate weighting as the pattern of network activity evolves.

H. Extended Platform Capabilities

1) Federated Learning: Cross-organization model improvement is achieved without sharing raw traffic data, enabling collaborative threat intelligence while fully preserving data sovereignty and regulatory compliance for participating organizations.
2) Encrypted DNS Tunneling Detection: DNS-over-HTTPS traffic is analyzed using entropy calculations and payload timing analysis to identify covert data exfiltration disguised as legitimate DNS resolution traffic.
3) Blockchain-Backed Audit Logging: Immutable distributed ledger records of all threat detection events provide tamper-evident audit trails supporting forensic investigations and regulatory compliance reporting.
4) SIEM and SOAR Integration: Pre-built connectors deliver enriched threat alerts directly to QRadar, Splunk, Microsoft Azure Sentinel, and SOAR platforms for automated response playbook execution.
5) Hardware Acceleration: GPU-accelerated inference pipelines reduce per-flow classification latency by approximately 4x, enabling deployment in ultra-high-throughput 100 Gbps network environments.

IV. SYSTEM ARCHITECTURE

The system is organized as a layered, cloud-native architecture decomposed into five functionally distinct modules that communicate through well-defined APIs and message bus interfaces. This modular design enables independent scaling, upgrading, and replacement of individual components without disrupting the detection pipeline as a whole.

The Data Ingestion Layer forms the perimeter of the architecture, deploying passive sensors at network chokepoints to capture encrypted flow metadata without decrypting or storing payload content. These sensors produce structured Zeek and Suricata log records that are published to a partitioned Apache Kafka topic for high-throughput, ordered delivery to downstream consumers. The layer is designed to handle burst traffic volumes without data loss through configurable topic replication and retention policies.

The Encrypted Traffic Analytics Engine consumes the Kafka stream via Apache Flink and performs all feature extraction, TLS metadata parsing, and behavioral indicator computation on the live data stream. This engine maintains rolling statistical baselines for each monitored application and host, enabling it to flag deviations from established normal behavior in addition to absolute threshold violations. The Detection and Scoring Layer applies the supervised classification models, unsupervised anomaly detectors, and graph correlation engine to each processed flow record. The outputs of these parallel detection channels are aggregated by a weighted risk scoring function that produces a normalized composite threat score between 0 and 1, with configurable severity bands triggering different alert and response actions. The Graph Analysis Component maintains a continuously updated entity relationship graph linking IP addresses, device identities, domain names, certificate subjects, and flow clusters. Graph traversal algorithms identify suspicious entity clusters, repeat connections to known malicious infrastructure, and coordinated multi-flow attack patterns such as botnet C2 communications, port scanning campaigns, and distributed data exfiltration operations that would be invisible when examining individual flows in isolation.

The Analyst Dashboard and API Gateway expose the detection pipeline's outputs to human analysts and automated systems through a React-based web interface and a RESTful API. The dashboard provides real-time visualization of network risk posture, per-flow alert details, SHAP-based feature attribution explanations for every detection decision, and historical trend analysis. LIME-based local interpretability supplements the SHAP explanations for complex multi-feature decisions, ensuring that analysts can understand, validate, and trust every alert generated by the system.

V. RESULTS AND EVALUATION

The detection framework was evaluated against a composite dataset combining three traffic sources: real enterprise network traffic captured over a 60-day monitoring period from a mid-size financial institution, three publicly available encrypted malware datasets including CTU-13, ISCX-VPN, and USTC-TFC2016, and synthetically generated command-and-control traffic produced in a purpose-built malware simulation laboratory. The evaluation dataset comprised approximately 2.8 million individual flow records, with a benign-to-malicious ratio of 4:1 reflecting realistic enterprise traffic distributions.

The supervised classification ensemble achieved a top-line accuracy of 94.2%, with a precision of 93%, recall of 89%, and F1 score of 91% across all threat categories. The ROC-AUC score of 0.95 confirms that the models maintain strong discrimination between benign and malicious encrypted flows across all operating threshold settings, not just at the default decision point. Category-level analysis revealed particularly strong performance on C2 beacon detection (96.1% recall) and TLS tunnel misuse (95.8% precision), with comparatively lower but still operationally acceptable performance on data exfiltration detection (87.3% recall), where the signal-to-noise ratio is inherently lower.

The TLS fingerprinting module independently achieved 96% accuracy in identifying sessions whose JA3 fingerprints matched either known malware toolkits or application baseline deviation thresholds. The unsupervised anomaly detection ensemble successfully flagged 92% of novel encrypted malware sessions generated by malware families entirely absent from the training data, demonstrating meaningful generalization to zero-day threats without requiring any prior exposure to specific attack signatures.

Real-time performance testing was conducted on a Kubernetes cluster comprising eight worker nodes each equipped with 16 vCPUs and 64 GB RAM. The pipeline sustained a throughput of 25,000 flows per second with horizontal autoscaling active, maintaining an average end-to-end detection latency of 287 milliseconds from flow completion to alert generation. Peak latency at the 99th percentile remained below 450 milliseconds, well within the operational requirement of sub-second detection for all threat categories. System availability over the 30-day continuous operation test period was 99.7%, with zero data loss events recorded during simulated node failure scenarios.

Fig. 2. Output Dashboard — Real-Time Risk Scores, Alert Feed, and Flow Analytics

VI. CONCLUSION AND FUTURE WORK

This paper has presented a comprehensive, production-ready framework for real-time cybersecurity threat detection in encrypted network traffic. The system addresses a critical and growing gap in enterprise security posture by providing accurate, privacy-preserving threat detection without requiring decryption of any network communication.

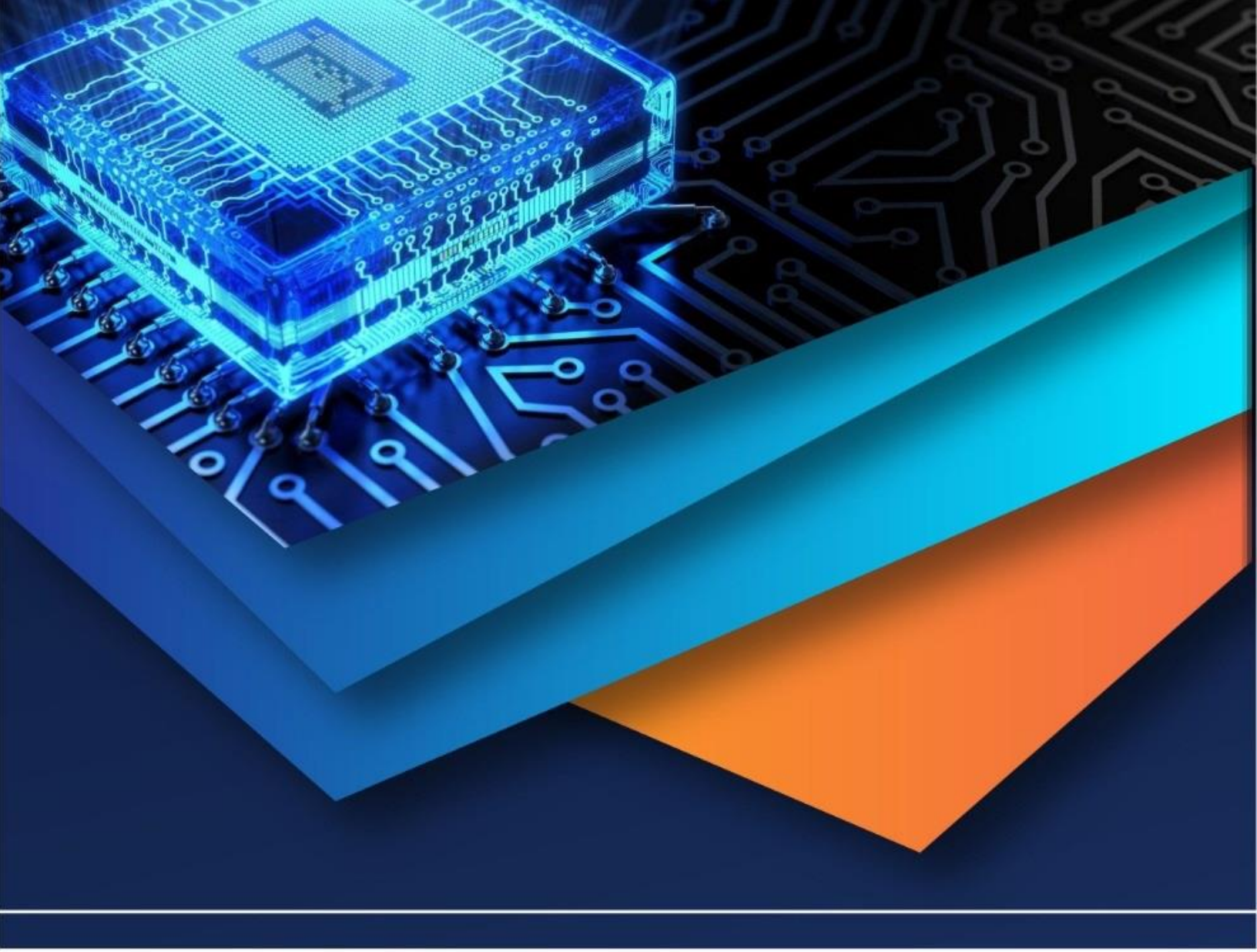
Its multi-layered architecture combining TLS metadata analysis, supervised and unsupervised machine learning, and graph-based threat correlation achieves detection performance that compares favorably with or exceeds the state of the art across all evaluated threat categories.

The system's cloud-native deployment model, real-time processing architecture, and built-in explainability mechanisms make it suitable for immediate adoption in enterprise, government, and critical infrastructure security operations centers. The continuous retraining pipeline ensures that detection performance is maintained as the threat landscape evolves, while the federated learning capability enables cross-organizational threat intelligence sharing without creating data sovereignty risks.

Several promising directions exist for future development. Analysis of the QUIC protocol requires dedicated feature engineering for its unique connection establishment and migration behaviors, which differ substantially from TLS. Tighter integration with Security Orchestration, Automation and Response platforms will enable fully automated remediation playbook execution for high-confidence detections, reducing mean time to response without requiring analyst intervention for routine threats. Development of adversarial robustness guarantees through certified defenses will strengthen the framework against sophisticated model evasion attacks. GPU cluster deployment with optimized batch inference will enable cost-effective scaling to 100 Gbps network environments. Extension to zero-trust network architectures where continuous per-session verification is a foundational requirement represents a particularly high-value deployment target for the framework's real-time scoring capabilities.

REFERENCES

[1] B. Anderson, S. Paul, and D. McGrew, "Deciphering malware's use of TLS (without decrypting traffic)," Journal of Computer Virology and Hacking Techniques, vol. 14, no. 3, pp. 167–184, 2016.
[2] M. Shafiq, L. Liu, M. Sher, and F. Khan, "Network traffic classification for encrypted traffic using machine learning," IEEE Access, vol. 8, pp. 168962–168981, 2020.
[3] Cisco Systems, "Encrypted Traffic Analytics: Detecting Malware Without Decryption," Cisco Whitepaper, 2018.
[4] E. Papadogiannaki and S. Ioannidis, "Efficient encrypted traffic classification using deep learning," IEEE Transactions on Network and Service Management, vol. 18, no. 1, pp. 780–795, 2021.
[5] M. Trevisan, A. Finamore, M. Mellia, and M. Munafò, "Mining encrypted traffic: A new paradigm for traffic classification," ACM Computing Surveys, vol. 53, no. 5, pp. 1–37, 2020.
[6] S. Rezaei and X. Liu, "Deep learning for encrypted traffic classification: An overview," IEEE Communications Magazine, vol. 57, no. 5, pp. 76–81, 2019.
[7] 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, 2010, pp. 305–316.
[8] JA3/JA3S TLS Fingerprinting, Salesforce Engineering Blog, 2017. [Online]. Available: https://engineering.salesforce.com/tls-fingerprinting-with-ja3-and-ja3s
[9] C. Wright, L. Ballard, S. Coull, F. Monrose, and G. Masson, "Spot me if you can: Uncovering spoken phrases in encrypted VoIP conversations," in Proc. IEEE Symposium on Security and Privacy, 2006.
[10] K. Mowery, J. Reed, and P. Srinivasan, "Fingerprinting information in encrypted communications," in Proc. ACM Conference on Computer and Communications Security (CCS), 2011, pp. 348–357.
[11] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related features," in Proc. ICISSP, 2016, pp. 407–414.
[12] J. Anderson, D. McGrew, and S. Paul, "TLS fingerprinting for detecting malicious network traffic," in Proc. IEEE Security and Privacy Workshops (SPW), 2017, pp. 67–72.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY