



iJRASET

International Journal For Research in
Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** X **Month of publication:** October 2026

DOI: <https://doi.org/10.22214/ijraset.2026.85111>

www.ijraset.com

Call: ☎ 08813907089

E-mail ID: ijraset@gmail.com

Context-Aware Network Behaviour Analyser

Prof. Vibha D. Gomase¹, Gopala Awadhi², Mohan R. Bhandekar³, Shrey Akulwar⁴, Jagruti Patle⁵, Khushi Sakhare⁶

Department of Computer Science and Engineering, Priyadarshini College of Engineering, Nagpur, Maharashtra, India

Abstract: *Conventional network-security tools such as firewalls and intrusion-detection systems examine packets, ports, protocols and signatures, but they rarely capture the application, process, user and operating conditions behind a connection. This gap makes it hard to separate legitimate activity from suspicious behaviour, especially when malicious traffic rides on commonly permitted ports or hides inside encrypted channels. This review paper surveys existing work in traffic classification, deep packet inspection, application-aware firewalls and machine-learning-based anomaly detection, and uses these findings to motivate a proposed Context-Aware Network Behaviour Analyser. The proposed system correlates live network flows with the originating process, the logged-in user, the time of activity and the host's recent behaviour, builds behavioural baselines for applications and users, and applies statistical and machine-learning techniques to flag meaningful deviations. A real-time dashboard is envisaged to present flows, process-to-flow associations, user activity and context-rich alerts to support faster investigation. The paper concludes by outlining an evaluation plan that compares the proposed analyser with a conventional rule-based monitor on detection rate and false-positive rate.*

Keywords: *Context-aware security, network behaviour analysis, process-to-flow attribution, anomaly detection, behavioural baselining.*

I. INTRODUCTION

Network monitoring has traditionally focused on packet- and flow-level details such as source and destination IP addresses, port numbers, protocols and known attack signatures. This approach works well against familiar threats, but it struggles when malicious activity resembles legitimate traffic, travels over encrypted channels, or originates from a trusted application that has been compromised. Looking at the flow alone rarely provides enough information to judge whether a communication is normal or suspicious.

In the context of this review, the term context refers to the additional information that surrounds a network connection: the process or application that created it, the active user, the time of communication, and the device's recent behaviour. Context makes it possible to distinguish between flows that look identical at the packet level but serve very different purposes — for instance, a browser fetching a legitimate update versus an unknown background process using the same port to exfiltrate data. Prior studies have shown that combining network traffic with host-level context and behavioural patterns improves anomaly detection and gives analysts better visibility into suspicious activity.

This paper reviews the major technical threads that feed into such a system — traffic classification, deep packet inspection, application-aware firewalls, and behavioural anomaly detection — and uses the review to propose a lightweight, host-level Context-Aware Network Behaviour Analyser. The proposed system correlates captured flows with process, user and temporal information, builds behavioural baselines, and raises context-rich alerts for unfamiliar processes, unexpected data transfers, or communication with previously unseen destinations.

The motivation for this work is not that any single existing technique is inadequate on its own, but that the techniques are rarely combined. A signature-based firewall and a machine-learning traffic classifier can each be individually accurate at the task they were designed for, yet a security analyst using both still has to manually connect "this flow looks statistically unusual" with "this flow came from this process, used by this user, at this time" before deciding whether to act. The review in Section II is therefore organised around identifying, for each technical thread, the specific piece of information it can reliably contribute to that manual correlation step, so that the proposed architecture in Section IV can automate the correlation itself rather than only adding another independent detector.

The remainder of the paper is organised as follows. Section II reviews related literature across four threads and summarises them in a comparison table. Section III states the problem this work addresses. Section IV describes the proposed methodology and system architecture, stage by stage. Section V discusses the expected outcomes and evaluation plan. Section VI discusses the limitations and implementation implications of the design, and Section VII concludes the paper.

II. LITERATURE SURVEY

A range of prior work informs the design of a context-aware network behaviour analyser, spanning traffic classification, deep packet inspection, application-aware firewalls, and machine-learning-based anomaly detection. This section reviews each thread in turn and draws out the specific ideas that carry over into the proposed system.

A. Traffic Classification and Deep Packet Inspection

Early traffic classification relied on port-based identification, which has become unreliable as applications increasingly use dynamic or non-standard ports to evade simple filtering. Deep Packet Inspection (DPI) improved accuracy by examining payload content, but it is computationally expensive at line rate and largely ineffective against encrypted traffic, which now makes up the majority of network flows on typical hosts and enterprise networks.

Survey literature on network traffic classification points to a broader shift toward flow-statistical and behavioural features — packet-size distributions, inter-arrival times, byte-count ratios and flow duration — that remain observable and informative even when payloads themselves are opaque [4][6]. These statistical features are exactly the kind of signal a lightweight, host-level analyser can compute cheaply without deep payload inspection, and they form the numerical backbone of the flow records used later in this paper.

B. Machine-Learning-Based Traffic Classification

Building on flow-statistical features, machine-learning and deep-learning models have been shown to classify network traffic and identify applications using only flow-level statistics, without inspecting payload contents at all. Reported approaches achieve strong accuracy for distinguishing between application categories in real time, and some extend to identifying specific applications or activities within an encrypted session purely from timing and size patterns [5]. This line of work forms the basis of the behavioural-profiling component proposed later in this paper.

Rather than classifying traffic into predefined application categories, the proposed system adapts the same flow-statistical features to build a per-application, per-user baseline of what "normal" looks like, which is a subtly different but closely related learning problem.

C. Application-Aware and Host-Based Firewalls

Application-aware firewall research and several patented mechanisms describe techniques for associating a network socket with the process or application that owns it on a host, enabling policy enforcement at the application level rather than purely at the IP/port level [7][8]. This process-to-socket correlation is typically achieved by reading operating-system socket tables and matching local port and address tuples against the list of active processes.

Related work on application-context-aware firewalls for software-defined networks extends this idea further, enforcing context-sensitive policies dynamically across a network rather than only at a single host. These techniques directly motivate the process-to-flow mapping component described in Section IV, and confirm that the required OS-level information is both available and has already been used successfully for security policy enforcement.

D. Network Anomaly and Behaviour Analysis

Research on network tomography and traffic-anomaly mapping shows that statistical modelling of traffic volumes and flow patterns over time can reveal anomalies without any prior knowledge of specific attack signatures [9]. This supports an approach built on learned behavioural baselines rather than static, signature-based rule sets. Complementary work on context-aware intrusion detection using reinforcement learning frames anomaly response as a sequential decision problem, adapting detection thresholds as the environment changes rather than relying on fixed rules [1].

Explainable intrusion-detection surveys further argue that raising an alert is only half the job — analysts also need a human-readable explanation of why a flow was flagged, which motivates the "explainable alert generator" stage included in the architecture proposed in Section IV [2]. More recent self-supervised approaches to anomaly detection in encrypted traffic reinforce the idea that useful anomaly signals can be learned directly from flow statistics, without labelled attack data and without decrypting the traffic itself [3].

Table 1 summarises the four threads reviewed above and the specific idea each contributes to the proposed architecture.

Research thread	Representative work	Main strength	Limitation / gap
Traffic classification / DPI	[4], [6]	Works well on unencrypted payloads; flow-statistical features generalise to encrypted traffic	DPI is costly and blind to encrypted payloads
ML-based traffic classification	[5]	High accuracy from flow features alone, no payload needed	Classifies traffic type, not per-user/app normal behaviour
Application-aware firewalls	[7], [8]	Maps sockets to owning process/application reliably	Policy-focused; little behavioural learning over time
Network anomaly / tomography	[1], [2], [3], [9]	Detects unknown attack patterns via statistical deviation	Often lacks process/user attribution and plain-language explanation

E. Research Gap

Traffic classification, DPI, application-aware firewalls and anomaly detection have each been studied extensively and independently. Traffic-classification research optimises for identifying what an application is; application-aware firewalls optimise for identifying which process owns a connection; and anomaly-detection research optimises for flagging statistical deviation. However, most existing tools address only one of these layers in isolation, and few systems combine process-level attribution, user/temporal context and learned behavioural baselining into a single, lightweight, real-time analyser suitable for a single host or a small network segment. This review identifies that integration gap as the specific motivation for the system proposed in the remainder of this paper: rather than proposing a new classification algorithm or a new anomaly-detection model in isolation, the contribution is in combining process attribution, contextual enrichment and behavioural baselining into one coherent, explainable pipeline.

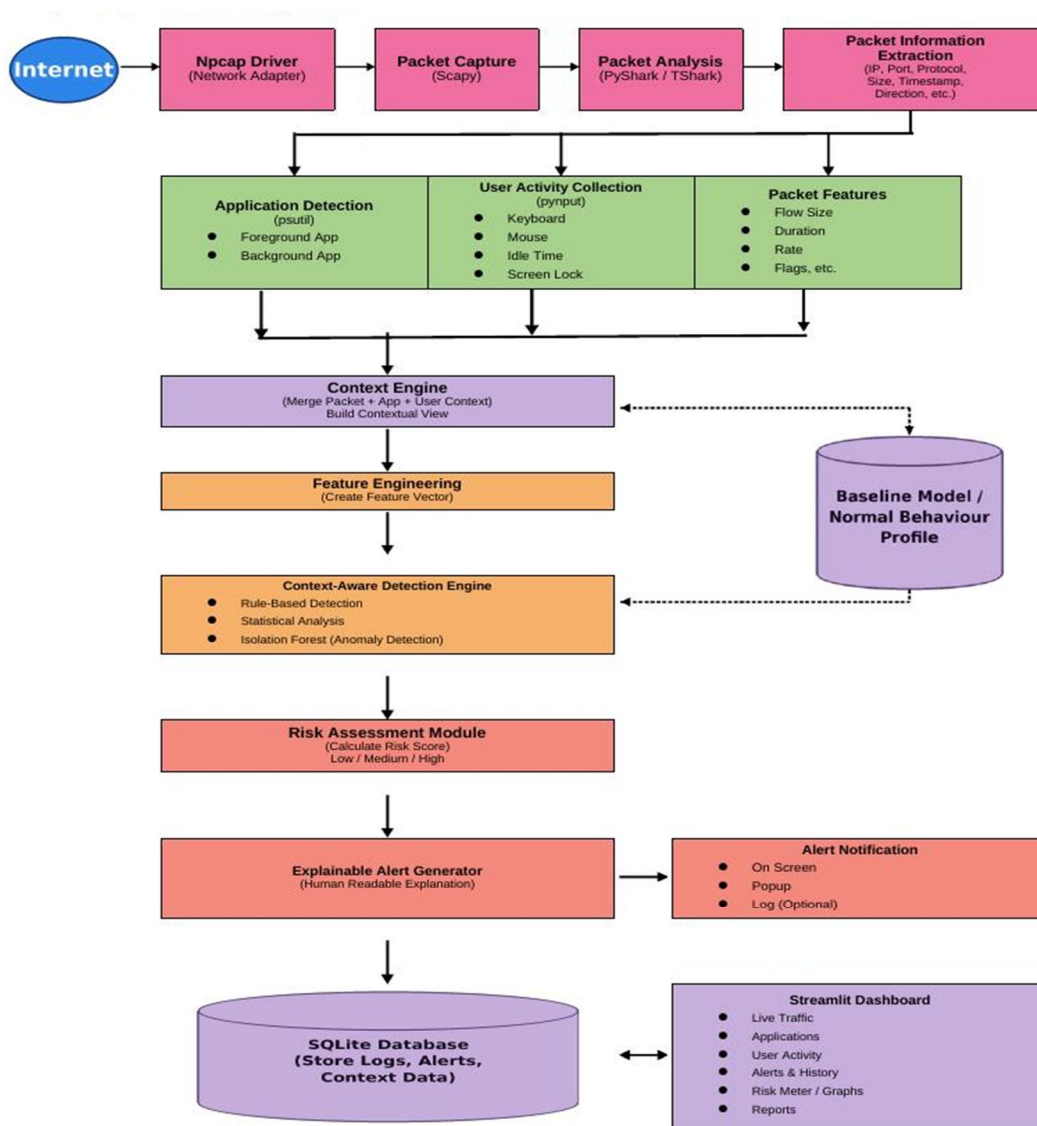
III. PROBLEM STATEMENT

Existing network-monitoring and firewall solutions mainly rely on fixed rules involving IP addresses, port numbers and protocols. These rules work for known and clearly defined threats, but they provide little information about the application or process responsible for a connection, or the circumstances under which the communication occurs. As a result, traffic from a malicious process can appear legitimate simply because it uses an open, commonly allowed port such as HTTPS port 443.

The absence of contextual information creates several practical challenges: malicious processes can hide inside permitted ports and encrypted traffic; analysts are flooded with low-value alerts because the monitor cannot separate unusual behaviour from routine activity; insider threats and compromised trusted applications can go undetected because their traffic does not match a known malicious signature; investigation is slower when the responsible process, user, device state and recent activity are not clearly identified; and static rules adapt poorly to changing application behaviour or previously unseen communication patterns. There is therefore a need for a monitoring approach that goes beyond packet- and flow-level inspection, correlates connections with process- and user-level information, learns normal behaviour, and raises context-rich alerts that reduce false positives and speed up investigation.

IV. PROPOSED METHODOLOGY

Based on the gaps identified in Section II and III, this paper proposes a Context-Aware Network Behaviour Analyser built around six stages: traffic capture, process-to-flow mapping, context enrichment, behavioural baselining, anomaly detection, and alerting/visualisation. Figure 1 shows the proposed system architecture.



8

Fig. 1. Proposed system / block diagram of the Context-Aware Network Behaviour Analyser

A. Traffic Capture

Live packets are captured at the host's network interface using a packet-capture library such as libpcap, or Npcap on Windows, typically accessed through higher-level tools such as Scapy or PyShark/TShark. Captured packets are grouped into flows using the standard 5-tuple — source IP, destination IP, source port, destination port and protocol — producing a structured flow record rather than a raw packet stream. Each flow record additionally carries derived packet-information fields such as total size, direction and timestamp, which are extracted at capture time so that later stages do not need to re-parse raw packets.

B. Process-to-Flow Mapping

Each active flow is correlated with the local process that owns the corresponding socket using OS-level socket tables — the /proc filesystem on Linux, or equivalent APIs on Windows. Cross-platform libraries such as psutil provide this information, yielding a mapping of process name, PID and executable path for every observed connection. The mapping module also distinguishes between foreground and background applications, since a connection opened by a background process the user is not actively interacting with is, on its own, a mildly more suspicious signal than the same connection opened by the application currently in focus.

C. Context Enrichment

Each flow record is enriched with contextual attributes drawn from two additional sources: user-activity collection and packet-feature extraction. User-activity collection, implemented with a library such as pynput, tracks coarse signals like keyboard and mouse activity, idle time and screen-lock state, which together indicate whether a human was actively using the machine when a flow was observed. Packet-feature extraction summarises each flow's size, duration, rate and flags. Combining these two streams with the process mapping from stage B produces a single context-rich flow record that ties network activity to who was using the system, what they were doing, and when — rather than a bare packet trace.

D. Context Engine and Feature Engineering

A context engine merges the packet-level, application-level and user-level streams into one contextual view per flow. This merged view is then passed through a feature-engineering step that converts the raw contextual fields into a numerical feature vector suitable for both rule evaluation and machine-learning scoring — for example, encoding time-of-day cyclically, normalising byte counts, and flagging whether the destination has been seen before for this process/user pair.

E. Behavioural Baseline

For each application/user pair, the system builds a statistical baseline of normal behaviour over an observation period, covering typical destinations, data volumes, connection frequency and active hours. Simple statistics — minimum, maximum and average values computed over hourly, daily and weekday/weekend snapshots — form the starting point for these baselines, stored in a dedicated baseline-model store that both the rule engine and the anomaly-detection model consult during scoring.

F. Anomaly Detection and Risk Assessment

New flows are scored against the learned baseline using a combination of rule-based thresholds (for example, a new destination or an unusual hour) and an unsupervised machine-learning model such as Isolation Forest or a clustering-based outlier detector. This combination flags statistically significant deviations from established normal behaviour while still catching clear rule violations that a purely statistical model might miss. Scored flows are then passed to a risk-assessment module that converts the raw anomaly score into a simple Low/Medium/High risk label, which is easier for a non-specialist analyst to act on than a raw numerical score.

G. Explainable Alerting and Visualisation

Rather than emitting a bare alert, an explainable alert generator converts each flagged flow and its risk label into a human-readable explanation — for example, naming the process, the destination, and the specific rule or behavioural deviation that triggered the alert. Alerts are then delivered through on-screen notification, pop-ups, or an optional log, and are also written to persistent storage (proposed as a lightweight SQLite database) alongside the underlying flow, application and user context. A dashboard, proposed to be built with Streamlit, is intended to present live traffic, applications, user activity, alert history, a risk meter and simple summary reports, bringing process identity, user context and behavioural deviation into one view for faster, more informed triage than a raw packet-level alert would allow.

V. EXPECTED OUTCOMES AND EVALUATION

The proposed analyser is expected to be evaluated on a test dataset that combines normal usage traffic with injected anomalous scenarios, such as simulated data exfiltration or an unfamiliar process making outbound network calls. Performance will be measured using detection rate and false-positive rate, and the results will be compared against a baseline rule-only monitor that uses the same flow capture stage but skips process, user and behavioural context. This comparison is intended to show how much value the behavioural and process-aware components add over conventional signature- and rule-based detection, rather than only demonstrating that the prototype runs.

- 1) A live dashboard displaying flows, process-to-flow associations, user activity, behavioural trends and anomaly alerts.
- 2) Context-rich alerts for events such as an unfamiliar process initiating an outbound connection, an application transferring an unusual amount of data, activity at unexpected hours, or communication with previously unseen destinations.
- 3) A measurable reduction in false-positive alerts compared with a conventional rule-based monitor, owing to the added process-, user- and behaviour-level context.
- 4) A working prototype suitable for a single host or a small network segment, demonstrating that context-aware analysis is practical without heavyweight infrastructure.

- 5) A comparative evaluation report summarising detection rate, false-positive rate and qualitative alert usefulness for the proposed analyser versus the rule-only baseline.

VI. DISCUSSION

The literature reviewed in Section II shows that the individual building blocks required for a context-aware analyser — flow-statistical classification, process-to-socket correlation, and behavioural anomaly modelling — are each well established on their own. The contribution of the design proposed here is architectural: it places these blocks in a single pipeline so that a security analyst sees one explained alert per anomalous flow, carrying process identity, user context and a plain-language reason, instead of three separate outputs from three separate tools that the analyst would otherwise have to correlate manually.

Two limitations are worth noting for future work. First, behavioural baselines need an initial observation period before they become reliable, during which the system may fall back mostly on rule-based detection; the length of this warm-up period and its effect on early false-positive rates is an open question that the planned evaluation in Section V is designed to quantify. Second, host-level user-activity signals such as keyboard and mouse events raise privacy considerations that any deployment of this system would need to handle carefully, for example by storing only coarse activity indicators rather than raw input events, and by keeping all context data local to the host rather than transmitting it elsewhere.

From an implementation standpoint, the modular pipeline in Section IV also has a practical benefit: each stage — capture, process mapping, context enrichment, baselining, detection, and alerting — can be developed and tested independently before integration. This lowers the risk of the project compared with attempting to train a single end-to-end model directly on raw packets, since each intermediate output (flow records, process mappings, context-enriched records, baselines) is inspectable on its own and can be validated against the literature reviewed in Section II. It also means that individual components, such as the anomaly-detection model, could later be swapped for a more sophisticated alternative without redesigning the rest of the system.

VII. CONCLUSION

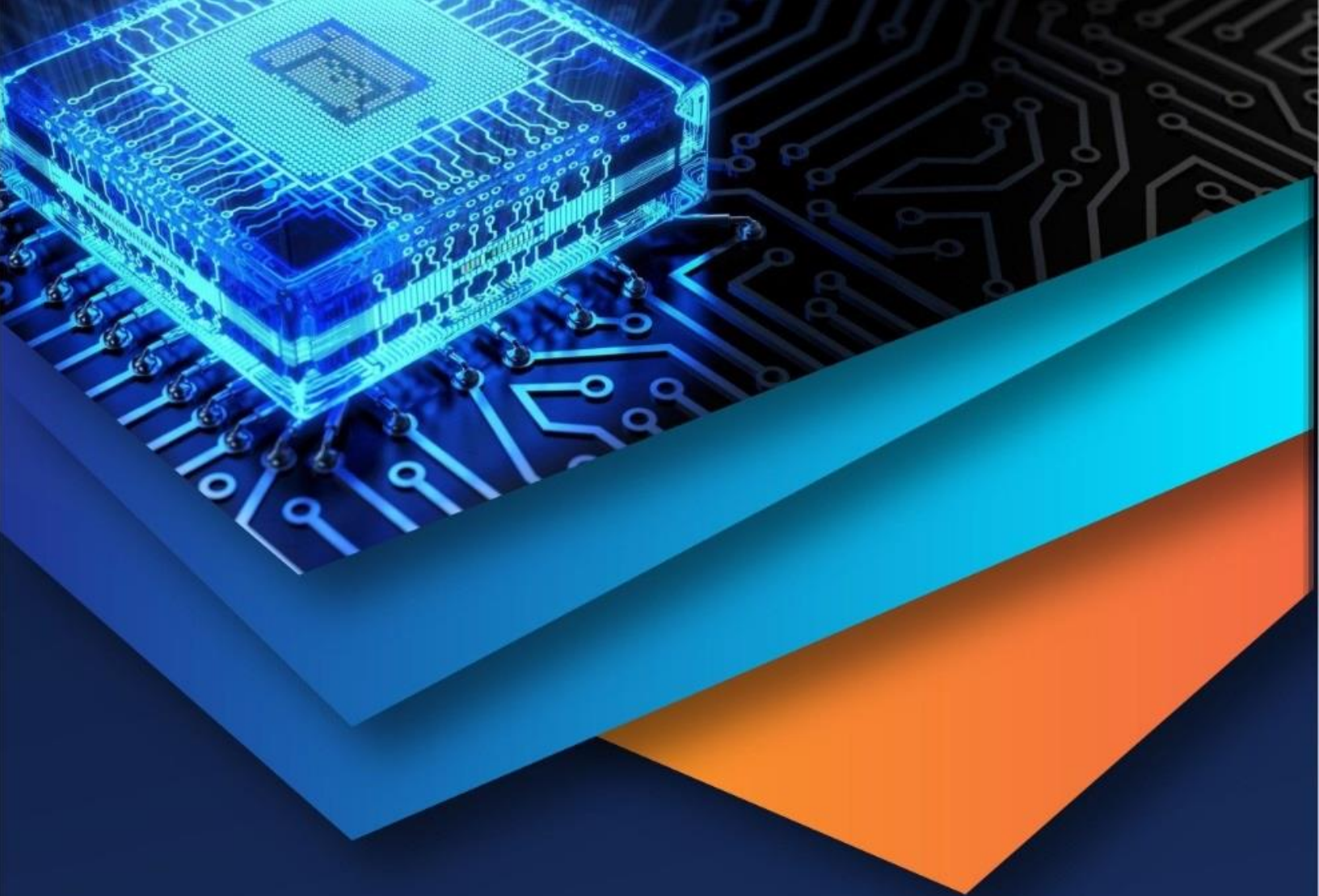
This review has surveyed traffic classification, deep packet inspection, application-aware firewalls and behavioural anomaly detection, and used the survey to identify a common limitation: most existing tools address only one of these layers in isolation, leaving a gap in systems that combine process-level attribution, user/temporal context and learned behavioural baselining. The proposed Context-Aware Network Behaviour Analyser addresses this gap by combining flow capture, process attribution, contextual enrichment and behavioural anomaly detection into a single, lightweight system. By adding application-, user- and behaviour-level visibility to conventional traffic analysis, the proposed system aims to complement existing network-security mechanisms, reduce false positives, and provide more meaningful information to security analysts. Future work will focus on building and evaluating the prototype described in Section IV against the plan outlined in Section V.

VIII. ACKNOWLEDGMENT

The authors thank their project guide, Prof. Vibha Gomase, Department of Computer Science and Engineering, Priyadarshini College of Engineering, for guidance throughout the review and formulation of the proposed methodology, and gratefully acknowledge the department for the resources and support extended during the preparation of this paper.

REFERENCES

- [1] K. Sethi, R. Kumar, N. Prajapati, and P. Bera, "A context-aware robust intrusion detection system: a reinforcement learning-based approach," *International Journal of Information Security*, vol. 19, no. 6, 2020.
- [2] S. Neupane, J. Ables, W. Anderson et al., "Explainable intrusion detection systems (X-IDS): a survey," *IEEE Access*, vol. 10, 2022.
- [3] "Anomaly detection in encrypted network traffic using self-supervised learning," *Scientific Reports*, vol. 15, 2025.
- [4] "Network Traffic Classification: Techniques, Datasets, and Challenges," Higher Education Press / ScienceDirect.
- [5] "Deep Packet: A Novel Approach for Encrypted Traffic Classification Using Deep Learning," *arXiv*.
- [6] "A Survey on Deep Packet Inspection," *ResearchGate*.
- [7] "Application-Aware Firewall Mechanism for Software Defined Networks," *Journal of Network and Systems Management*, Springer.
- [8] "Application Firewall," *HandWiki*.
- [9] "Estimating Traffic and Anomaly Maps via Network Tomography," *arXiv*.
- [10] "Network Traffic Classification for Data Fusion: A Survey," *ScienceDirect*.



10.22214/IJRASET



45.98



IMPACT FACTOR:
7.129



IMPACT FACTOR:
7.429



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089  (24*7 Support on Whatsapp)