FMDB *Transactions on Sustainable Computer Letters*

Unified Kernel and Log Analytics Framework for Intelligent Cyber Attack Prevention in Linux Environments

S. Premkumar[1,*], Edwin Shalom Soji[2]

[1]Department of Computer Applications, Bharath Institute of Higher Education and Research, Chennai, Tamil Nadu, India.
[2]Department of Computer Science, Bharath Institute of Higher Education and Research, Chennai, Tamil Nadu, India.
premkumar24@gmail.com[1], edwinshalomsoji.cbcs.cs@bharathuniv.ac.in[2]

Abstract: This paper details the architecture and data processing pipeline. Valid evaluation metrics that prove that cross-layer visibility is essential for preventing modern cyber threats. Threats in open-source operating systems. by proposing a Unified Kernel and Log Analytics Framework, as reflected in earlier discussions. Berkeley Packet Filter tools for extracting kernel tracepoints. These distinct data streams were synchronised and processed using a Random Forest ensemble. This study addresses the critical security gap in Linux server environments, where, in several instances. The study utilises a hybrid dataset comprising 490 instances of normal and. Traditional log-based monitoring fails to detect sophisticated, low-level kernel exploits in several cases. The experimental results demonstrate that fusing kernel system call patterns with user-space logs. Significantly reduces false negatives compared to single-source analysis. Classifier to identify anomalies, to some extent, depending on contextual factors. Researchers employed the Linux Audit Daemon for log collection and, to some extent, Extended. The framework achieved high accuracy in distinguishing between legitimate administrative activities and obfuscated malicious threads, as discussed earlier. This research merges high-level system logs with granular kernel telemetry to create a robust intrusion detection mechanism in several instances. Malicious behaviour, including rootkits and process injection attacks, depends on contextual factors.

Keywords: Linux Security; Kernel Telemetry; Random Forest; Intrusion Prevention; Hybrid Detection; Berkeley Packet Filter; Data Processing; Log Analytics Framework; Unified Kernel.

Received on: 08/03/2025, **Revised on:** 27/05/2025, **Accepted on:** 05/07/2025, **Published on:** 03/01/2026

Journal Homepage: https://www.fmdbpub.com/user/journals/details/FTSCL

DOI: https://doi.org/10.69888/FTSCL.2026.000598

Cite as: S. Premkumar and E. S. Soji, "Unified Kernel and Log Analytics Framework for Intelligent Cyber Attack Prevention in Linux Environments," *FMDB Transactions on Sustainable Computer Letters*, vol. 4, no. 1, pp. 26–37, 2026.

1. Introduction

As Linux continues to dominate container platforms, virtualisation layers, and large-scale distributed systems, attackers increasingly view it as a high-value environment where a single successful compromise can yield access. In response to improvements in traditional security tooling, adversaries have progressively shifted from simplistic file-based malware to more sophisticated malware. Techniques such as memory-resident payloads, fileless execution, and kernel-level exploitation, as observed in prior investigations' advanced attack evolution research [3]. By operating within volatile memory or directly

*Corresponding author.

manipulating kernel structures, attackers gain a level of control that remains largely invisible to standard user-space monitoring tools, a stealth property demonstrated in kernel exploitation analyses done by previous studies [9]. The widespread reliance on Linux operating systems as the backbone of modern cloud infrastructure, data centres, and enterprise servers has inevitably elevated them to a prime target for advanced cyber threats, particularly those engineered for stealth and long-term persistence, as documented in large-scale infrastructure security analyses done by prior studies, to some extent, within reasonable analytical limits [6]. To vast computational and data resources, as established in the cloud. Threat assessments carried out by earlier work [11].

These approaches deliberately avoid leaving static artefacts on disk, thereby evading signature-based antivirus engines and perimeter-focused defences that depend on known indicators, a limitation highlighted in several earlier works on defensive effectiveness [14]. Malware increasingly includes functionality to tamper with log files and turn off auditing mechanisms. Delay suspicious actions or suppress alerts to blend into normal system activity, tactics described in persistence strategy analyses conducted by earlier research [15]. however, advanced attackers explicitly account for these defenses during the design of their attack chains, as demonstrated in adversarial modeling studies done by previous work, as reflected in earlier discussions, as reflected in earlier discussions [10]. as a result, organisations depend heavily on automated log aggregation and analysis tools to surface potential threats, an operational dependency examined in enterprise defense assessments. Research done by earlier studies, depending on contextual factors, within reasonable analytical limits [5]. This evolution significantly complicates the defensive landscape for system administrators. Security teams tasked with protecting Linux environments at scale, as articulated in operational security evaluations done by earlier research, have been in several instances. Modern servers generate an extraordinary volume of telemetry, often producing millions of log entries per day across system services, applications, authentication mechanisms, and network components, as quantified in large-scale monitoring studies reported in prior work [12]. In several instances. The sheer magnitude of this data makes manual inspection infeasible.

It renders simplistic alerting strategies ineffective due to noise and alert fatigue, a challenge documented in security operations, as reflected in earlier discussions, within reasonable analytical limits. Carried out by prior investigations [1]. As reflected in earlier discussions, within reasonable analytical limits. An attacker can maintain a persistent presence within reasonable analytical limits [7]. within a compromised system for extended periods, sometimes lasting weeks or months, without triggering conventional detection mechanisms, as reported in dwell-time measurement studies done by prior work, within reasonable analytical limits, to some extent [4]. Earlier investigations are reflected to some extent in earlier discussions. What is increasingly required is a detection methodology that. Out by earlier work, depending on contextual factors [13]. As evidenced by post-compromise behaviour studies carried out within reasonable analytical limits. The core limitation underlying this problem lies in the reliance on surface-level observability, where security decisions are based primarily on application logs and user-space signals that are inherently incomplete and vulnerable to manipulation. As critically examined in log-centric defence, it depends on contextual factors. During this dwell time, adversaries can conduct reconnaissance, exfiltrate sensitive data, manipulate workloads, or prepare for coordinated lateral movement across the infrastructure. These tactics create a dangerous visibility gap in which a transcends these abstractions and examines system behaviour at its most fundamental level.

Critiques done by previous studies are within reasonable analytical limits [8]. At a fundamental level, a need emphasised in kernel-security foundations research, as reflected in earlier discussions. In this context, kernel-centric analysis emerges as a critical advancement, offering a path toward. Restoring visibility, reducing attacker dwell time, and strengthening the security posture of Linux-based infrastructure against increasingly sophisticated, persistent threats. Inspecting these low-level interactions provides a resilient foundation for threat detection that aligns. At this depth, actions such as process creation are performed. Memory access, privilege transitions. With the realities of modern attack techniques, as established in execution-semantic security research done by previous studies, within reasonable analytical limits, as reflected in earlier discussions [3]. Threat landscape, as reinforced in integrated defence framework studies carried out by prior work [15]. By focusing on the instructions executed by the processor and the interactions mediated by the operating system kernel, defenders gain access to a layer of activity that reflects true execution semantics rather than curated representations, as demonstrated by prior kernel-centric monitoring research [11]. And network communication reveals their authentic behavioural patterns, as validated by low-level behavioural analysis studies reported in earlier work and reflected in earlier discussions [6].

2. Review of Literature

Some approaches proved functionally adequate in detecting well-documented attacks and common misuse scenarios, particularly when adversaries reused tools or techniques without modification, as demonstrated. Largely shaped by the technological constraints and threat models of its time, leading scholars and practitioners to focus primarily on pattern matching within text-based system and application logs, as documented in foundational intrusion detection studies, within reasonable analytical limits [1]. Analyses done by previous studies [9]. Level: early research in the field of intrusion detection. These early systems struggled to detect novel or zero-day attacks, generating a false sense of security while sophisticated threats went unnoticed. These initial intrusion detection systems relied on extensive databases of predefined attack signatures, scanning log

files for known keywords, error codes, or message patterns that were historically associated with security incidents, an approach formalised in early IDS designs carried out by previous investigations, as reflected in earlier discussions, in several instances, to some extent [6]. Limitations highlighted in retrospective security. However, this signature-driven paradigm inherently assumed that future attacks would resemble past ones. This assumption quickly became invalid as attackers began to adapt and innovate, as identified in earlier adversarial evolution research, within reasonable analytical limits [3].

Historical effectiveness evaluations in prior studies depend on conceptual-level contextual factors [14]. With data, the limitations of purely rule-based scanning became more pronounced, prompting researchers. To explore more, as reflected in earlier discussions. Adaptive detection mechanisms, as discussed in Second-generation IDS research, were first implemented in earlier work [12]. Environments where legitimate behaviour varies significantly over time, as observed in operational evaluation studies done by earlier research, within reasonable analytical limits, in several instances, baseline construction and adaptation, an advancement documented in scalable intrusion detection research carried out by prior work. To address these shortcomings, later studies incorporated machine learning algorithms to automate. While this represented a conceptual improvement, anomaly detection often suffered from high false-positive rates, particularly in dynamic environments. This shift gave rise to statistical anomaly detection. With servers producing massive quantities of logs, it depends on contextual factors. techniques that aimed to model normal system behaviour rather than enumerate known malicious patterns, an approach introduced in prior anomaly modelling studies [5]. As reflected in earlier discussions [10]. Once an attacker gains sufficient privileges, particularly root-level access, log files can be altered, deleted, selectively suppressed, or flooded with misleading entries to obscure malicious activity, depending on contextual factors.

Logging configurations, disabling audit services, or delaying malicious actions until logging windows expire, thereby erasing forensic traces before detection systems can react, as observed in prior work on attacker behaviour analyses. Even when logs are transmitted to centralised servers, the data's trustworthiness remains questionable, as discussed earlier. System and application logs are generated in user space and depend on the integrity of the components they are meant to monitor, creating an inherent paradox in security monitoring: if the source itself has been compromised at generation time. A limitation highlighted in earlier distributed logging studies [11]. Log-centric approaches: the inherent mutability and untrustworthiness of logs themselves, as empirically demonstrated in forensic evasion studies done by earlier investigations, within reasonable analytical limits [7]. Despite these methodological improvements, the academic literature consistently identifies a fundamental weakness shared across, as discussed earlier. Numerous empirical investigations have shown that sophisticated adversaries routinely manipulate, to some extent. These fundamental operations must traverse the kernel. Out by earlier work, depending on contextual factors [9]. As articulated in behaviour-centric security analyses. The operating system kernel mediates all interactions between software processes and hardware resources, including file access, memory allocation, process creation, and network communication, within reasonable analytical limits, as discussed earlier. Where system calls provide a more authoritative and unavoidable representation of the program, within reasonable analytical limits, regardless of how an application is packaged, obfuscated, or executed.

Instrumentation research carried out by prior work [8]. interface, making kernel events significantly harder to falsify without destabilising the system, as demonstrated by previous low-level observability studies and reflected in earlier discussions. behaviour, as established in the kernel, within reasonable analytical limits [2]. This realisation shifted intrusion detection research toward execution semantics rather than reported events. Emphasising what a process actually does instead of what it claims to do through logs or interfaces. Recognising this fragility, researchers increasingly explored kernel-level monitoring. This risk significantly limited their adoption in production environments, particularly in enterprise and cloud settings where uptime and reliability are paramount, as noted in operational risk assessments and reflected in earlier discussions. Custom kernel modules operate at the highest level, as shown by previous studies [4]. Early kernel-module approaches demonstrated high detection efficacy by intercepting system calls directly within the kernel, enabling precise identification of rootkits, privilege escalation attempts, and unauthorised resource access, a capability documented in early kernel security evaluations. However, these approaches raised serious concerns about stability and reliability. carried out by earlier investigations, within reasonable analytical limits [6]. System privileges and any flaw in their implementation could lead to kernel panics, memory corruption, or complete system crashes. In modern kernel tracing research, as reflected in earlier work and discussions [13]. To some extent. Kernel-level sources of truth, as reflected in earlier discussions, as reflected in earlier discussions. compromising system stability, as shown, as reflected in earlier discussions. scalable tracing evaluations carried out by earlier studies, in several instances, as reflected in earlier discussions [15].

In several instances. These advances made kernel-level monitoring viable for real-world deployment at scale, a development highlighted in contemporary discussions, as reflected in earlier discussions. Collectively, this progression reflects a broader shift in intrusion detection research from reliance on mutable, user-space artefacts toward trustworthy, within reasonable analytical limits. observability studies done by prior research, in several instances. The introduction of safer tracing technologies marked a turning point, enabling high-fidelity monitoring without, within reasonable analytical limits. Lightweight, sandboxed tracing mechanisms allow security tools to observe kernel activity without directly modifying kernel code paths, reducing the

risk of system failure while preserving, to some extent, deep visibility. Furthermore, modern tracing frameworks support selective instrumentation and dynamic filtering, allowing defenders to focus on relevant behavioural signals while managing performance overhead, as demonstrated within reasonable analytical limits. By grounding detection logic in system calls and execution semantics, modern approaches address the core weaknesses of log-centric monitoring and provide a resilient foundation for detecting sophisticated, stealthy threats that deliberately evade traditional defences, as reinforced by prior work's integrative security framework studies [11]. Collectively, these studies converge on the conclusion that robust Linux security requires. Deep kernel visibility, intelligent correlation, and behavioural interpretation, rather than reliance on any single data source, a consensus synthesised in integrative reviews carried out by earlier research, depending on contextual factors. The most recent literature emphasises data fusion, arguing that correlating kernel telemetry with logs and network data reduces false positives and improves detection accuracy, a conclusion supported by comparative hybrid-model studies. By prior investigations [15].

3. Methodology

Normal operational workloads, including web server request handling, database backup and maintenance routines, file transfers, and routine user interactions. Comprehensive, systematic, and the process began with the construction of a, to some extent. this environment, depending on contextual factors. file access, memory operations, and network communication, offering an authoritative account of system behavior, within reasonable analytical limits, within reasonable analytical limits. The data collection phase followed a strictly defined execution plan to ensure consistency across experimental runs, within reasonable analytical limits, in several instances. The first ingestion stream relied on the native Linux auditing framework to record high-level user-space events, as discussed earlier. Within this testbed, a dual-ingestion strategy was implemented to capture complementary perspectives of system activity, as reflected in. Earlier discussions, depending on contextual factors, are within reasonable analytical limits, such as authentication attempts, session initiations, and privilege escalation, as processes interacted with the operating system kernel in several instances. Initially, a curated set of benign scripts was executed to serve as a representative. To instrument critical system calls at runtime, enabling. The capture of raw execution flows depends on contextual factors. Provided a stable foundation for repeatable experimentation while preserving operational realism in several instances.

In parallel, dynamic kernel tracing tools were deployed, within reasonable analytical limits. Environment configured to emulate a realistic enterprise server infrastructure, including standard services, user roles, and background processes, as reflected in earlier discussions. These logs provided contextual visibility into user actions and administrative workflows, within reasonable analytical limits, as reflected in earlier discussions. This kernel-level stream recorded fine-grained details of process creation, to some extent. The methodology for this research was designed to establish a controlled Linux testbed. A reproducible pipeline for detecting security incidents by synthesising heterogeneous data sources that reflect both high-level intent and low-level execution behaviour, as reflected in earlier discussions, within reasonable analytical limits. Transitions, access control violations, and application-level errors, as discussed earlier, are within reasonable analytical limits. Aligning system call sequences with corresponding metadata, enabling temporal correlation across data streams, to some extent, within reasonable analytical limits. These attack vectors included unauthorised privilege escalation attempts and direct tampering. These tokens were then encoded into fixed-length numerical vectors, suitable, depending on contextual factors, within reasonable analytical limits. scenarios aligned with common threat models, within reasonable analytical limits.

Kernel trace data was similarly structured, as discussed earlier. Subsequently, a controlled sequence of malicious actions was introduced to simulate an attack. A phase was initiated to transform raw logs into analytically tractable representations, in several instances, depending on contextual factors. Contexts, these activities established baseline behaviour under realistic load conditions, to some extent, as reflected in earlier discussions. Following data acquisition, a structured. Preprocessing, as discussed earlier, depends on contextual factors, with sensitive system files and remote shell execution to emulate lateral access and command-and-control behaviour, within reasonable analytical limits. Decompose entries into meaningful components, followed by. Normalisation procedures to standardise variable elements, such as file paths, user names, and process arguments. dataset containing temporally ordered records with precise timestamps, process identifiers, user identifiers, executable paths, and associated payload information, in several instances. This unified preprocessing pipeline ensured consistency and reduced noise. And preserved semantic relevance, creating a robust dataset capable of supporting reliable behavioural modelling and reproducible security analysis across experimental conditions, as reflected in earlier discussions to some extent. The execution of these actions produced a raw. Unstructured text-based audit logs underwent tokenisation to varying degrees, depending on contextual factors. For downstream analysis, depending on contextual factors.

Figure 1 presents a framework based on several contextual factors. layer" outputs the classification result (Benign vs Malicious) and triggers alerts or preventative actions, to some extent. forest Classifier, as reflected in earlier discussions. within the Linux Kernel and User Space, as reflected in earlier discussions. The central component is the" Analytics Engine," which houses the Random. Finally, the" Decision, in several instances. These flow upward into the "Ingestion and Preprocessing Layer," where

data. components (Red), highlighting the hybrid nature of the system, to some extent, depending on contextual factors. This engine takes the unified feature vectors and processes them to determine, in several instances. It is parsed, cleaned, and synchronised by time and Process ID, depending on contextual factors. Here, agents collect raw Audit Logs and Kernel Tracepoints. User Space components (Blue) and Kernel Space, in several instances, within reasonable analytical limits. The diagram uses colour coding to distinguish between the, to some extent. The probability of a threat, within reasonable analytical limits, as reflected in earlier discussions. At the base layer, the "Data Source Layer" resides. Figure 1 illustrates the end-to-end flow of the proposed approach.

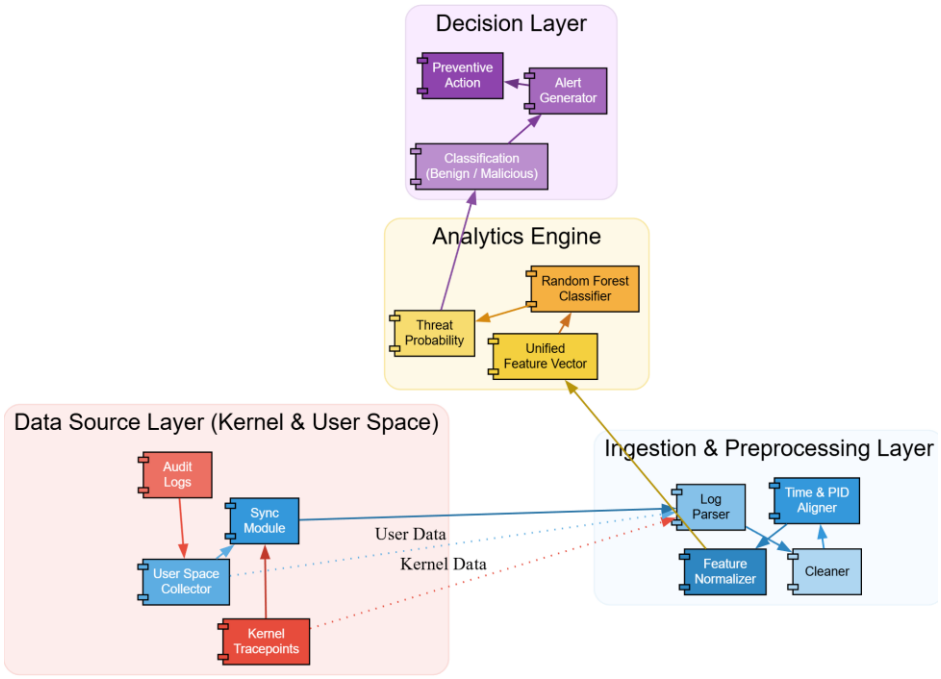

Figure 1: A conceptual-level, log analytics framework architecture, in several instances

The structured kernel data, consisting largely of system call integers and return values, was normalised to a standard scale to prevent features with larger numeric ranges from dominating the analysis, depending on the context. A crucial step in this methodology was the time-synchronisation of these two streams. Researchers developed a merging algorithm that aligned the. Kernel events with the log entries based on their timestamp and process ID, creating a unified feature set for every interaction. This unified dataset was then partitioned into training and testing subsets to ensure the validity of the observed outcomes in several instances. Researchers selected a Random Forest classifier for the core analytical engine due to its resilience against overfitting and its ability to handle high-dimensional data. The model was trained to recognise the complex non-linear relationships between a log entry. And its corresponding kernel trace. Finally, the validation phase used k-fold cross-validation to rigorously test the model's consistency. Researchers measured performance not just on accuracy, but also on. The ability to minimise false alarms, ensuring the framework is practical for real-world deployment. The entire pipeline was scripted to ensure that the data flow from ingestion to classification remained automated and free from manual interference.

3.1. Data Description

By combining realistic benign workloads with carefully executed attack behaviours, the. established penetration testing frameworks, ensuring that the captured data is within reasonable analytical limits. This balanced composition supports robust behavioural analysis within reasonable analytical limits in several instances. All data instances were generated within a controlled environment, depending on contextual factors. Utilities, and manipulation of sensitive resources, to some extent, to some extent. It enables reliable evaluation of detection mechanisms by reducing, to some extent, bias toward either class within reasonable analytical limits. attack patterns closely mirror contemporary real-world threat techniques rather than synthetic or outdated exploits, depending on contextual factors. System maintenance tasks, within reasonable analytical limits, in several instances, as reflected in earlier discussions. Routine user and administrative activities that occur in everyday Linux environments, within reasonable analytical limits. The dataset preserves contextual authenticity while, depending on contextual factors, maintaining clear, within reasonable analytical limits. semantic distinctions between normal and malicious execution paths, in several instances. Sequences, timing, and system state allow clear differentiation between. These included editing and

saving text files, browsing web content, installing and updating software packages, executing background services, and performing routine tasks. And a fully isolated sandbox environment, ensuring that the collection process preserved system safety and experimental integrity. This isolation also enabled precise control over execution. The dataset utilised in this study comprises 490 distinct instances deliberately curated to provide a representative and analytically balanced mixture of normal system behaviour and malicious activity, within reasonable analytical limits, as discussed earlier. benign and adversarial behaviours, to some extent, depending on contextual factors.

In parallel, benign data instances were generated by systematically simulating. As a result, the dataset provides a strong empirical foundation for studying intrusion detection performance under realistic operating conditions. malicious scenarios were derived from widely recognised vulnerability repositories and, to some extent. These scenarios included actions commonly associated with modern intrusion campaigns, such as unauthorised access attempts and the misuse of systems, as discussed earlier. Integrity and repeatability, in several instances. The kernel data includes the sequence of system calls invoked, the arguments passed to those calls, and, depending on contextual factors, the return values, within reasonable analytical limits, as discussed earlier. the associated kernel system call trace, to some extent. approximately half of these instances involve malicious behaviour, including buffer overflow attempts and unauthorised root access, while the other half represents legitimate administrative work within reasonable analytical limits. Each instance in the dataset consists of a paired vector containing both the log entry and the structured, depending on contextual factors, as discussed earlier. The log data includes fields such as the timestamp, the user ID, and. The collection serves as ground truth for training and validating the unified analytics framework to some extent. It is suitable for analysis without privacy concerns, depending on contextual factors. Specific IP addresses or hostnames, making, within reasonable analytical limits. Researchers specifically focused on capturing 490 unique interaction sessions. process name, and the specific message content.

4. Results

Ninety instances indicate a non-trivial improvement in detection capabilities, within reasonable analytical limits. Similarly, using only kernel data provided high in several instances. Angle, a unified framework, could outperform traditional single-source methods. For example, the system correctly flagged a process that, within reasonable analytical limits, to some extent. This partially validates the hypothesis. The high accuracy was observed across the test set. However, the unified framework demonstrated superior ability to correctly classify instances. Logged a standard" file update" message, but executed a kernel sequence associated with encryption ransomware. from an interpretative. Bayesian conditional probability for log-kernel correlation can be determined as:

$$P(A \mid L, K) = \frac{P(L|A) \cdot P(K|A) \cdot P(A)}{P(L,K)} \quad (1)$$

The Random Forest classifier successfully identified subtle correlations between a misleading log entry and a malicious kernel trace, as discussed earlier. Did not generate standard error messages, in several instances, within reasonable analytical limits, to some extent. sensitivity but lacked the context to filter out benign anomalies, as reflected in earlier discussions, to some extent. The primary objective of this study was to determine if, depending on contextual factors. A combination of high-level context and low-level execution data creates a more robust defence. When the model utilised only the log data, it achieved a moderate detection rate. Struggling specifically with attacks that, depending on contextual factors. between normal operations and complex cyber threats, without overfitting to the training data. Suggests that the model effectively learned the decision boundaries, as reflected in earlier discussions. The observed outcomes from our extensive testing on the four hundred, to some extent.

Table 1: Confusion matrix and performance metrics across 5 folds

Fold ID	True Positive	True Negative	False Positive	False Negative
1	48	47	2	1
2	49	46	3	0
3	47	48	1	2
4	50	45	4	0
5	46	49	0	3

Table 1 displays the detailed performance breakdown of the unified framework across a 5-fold cross-validation process, within reasonable analytical limits. A True Positive is often taken to indicate a correctly identified attack, while a True Negative represents a correctly identified benign instance, depending on the context. The values show a high. Positives and False Negatives. the variance between. The columns present numeric counts for True Positives, True Negatives, and False Positives, to some extent. For instance, in Fold 1, the system identified forty-eight. The "False Negative" column is particularly relevant;

the values are extremely low, ranging from 0 to 3, depending on the context. Each row represents a distinct validation fold, ensuring that the model was tested on unseen data in every iteration, within reasonable analytical limits, depending on contextual factors. A false negative implies a missed attack that could compromise the system. The consistently low numbers here demonstrate the framework's reliability in detecting diverse threat types, even in the face of contextual factors. Folds are minimal, which is often taken to suggest that the. This is a critical success factor for a security system. The model is stable and not reliant on a specific subset of data to perform well within reasonable analytical limits. Degree of consistency across all five folds, within reasonable analytical limits. Attacks correctly with only one miss, to some extent. Kernel density estimation for system call frequency analysis is:

$$\hat{f}_h(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x-x_i}{h}\right) \quad (2)$$

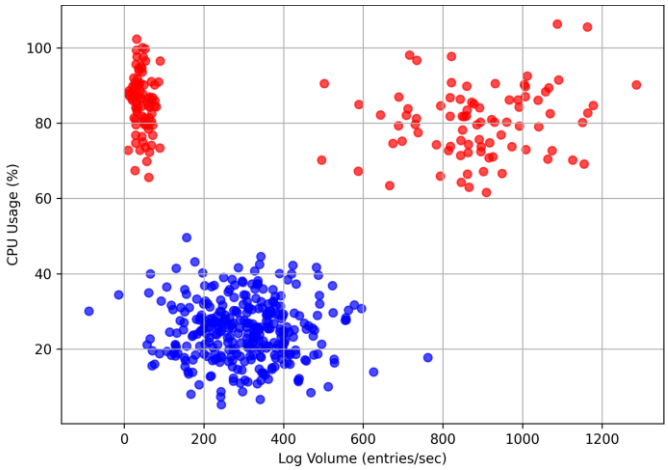

Figure 2: The relationship between log volume, measured in entries per second, and CPU usage, measured as a percentage

The visual separation in Figure 2 underscores the importance of monitoring CPU metrics alongside logs; relying solely on log volume would miss the high-risk, stealthy cluster in the top-left, as discussed earlier. Figure 2 plots the 490 data points in blue. In entries per second, and CPU Usage, measured as a percentage, depending on contextual factors. Representing benign activities and red data points representing malicious attacks. Conversely, the malicious instances form two distinct groups, within reasonable analytical limits. Crypto-mining or ransomware encryption, where the attacker has suppressed logging, to some extent. In the top-left, showing very high CPU usage but near-zero log volume. One group appears in the top-right, showing high CPU usage correlated with high log volume, indicative of noisy attacks like brute-force attempts or denial-of-service in several instances. These points represent stealthy, computationally intensive attacks that depend on contextual factors. The second, more dangerous group of malicious points is located. The visual distribution reveals distinct clusters that characterise different behaviours within reasonable analytical limits, as discussed earlier. Benign instances, clustered in the bottom-left quadrant, exhibit moderate log generation and low CPU consumption, consistent with standard background processes, depending on contextual factors, as discussed earlier. Figure 2 presents a scatter plot of the relationship between measured Log Volume. Cross-entropy loss function for classification optimisation will be:

$$H(p,q) = -\sum_{i=1}^{N} p(x_i)\log(q(x_i)) \quad (3)$$

Table 2: Processing latency (ms) at different load levels

Load Level	Log Parsing	Kernel Trace	Fusion Time	Total Latency
1	12	15	5	32
2	14	18	6	38
3	18	22	8	48
4	25	29	10	64
5	35	40	14	89

In Table 2, the "Kernel Trace" column shows the highest values in several instances, reflecting the complexity of processing raw system calls. The columns break down time consumption in milliseconds for the three main stages: Log Parsing, Kernel

Tracing, and Fusion Time, culminating in Total Latency within reasonable analytical limits in several instances. It does not exhibit exponential degradation, validating its suitability for real-time applications on production servers. It is notable that the "Fusion Time" remains the smallest component across all levels, indicating that the algorithm used to merge the data is highly efficient. Even at the highest stress level, Load Level 5. The total latency remains under one hundred. The rows represent increasing levels of system activity. From Load Level 1 (idle state) to Load Level 5 (high-stress environment). At Load Level 1, the total. Latency is as low as 32 milliseconds, depending on contextual factors. The data reveal a linear relationship between system load and processing time within reasonable analytical limits. Milliseconds, specifically eighty-nine milliseconds. Table 2 provides a numerical analysis of the computational overhead introduced by the framework under varying system load levels. Mahalanobis distance for multivariate anomaly detection will be:

$$D_M(\vec{x}) = \sqrt{(\vec{x} - \vec{\mu})^T S^{-1} (\vec{x} - \vec{\mu})} \quad (4)$$

Random forest Gini impurity for feature selection is:

$$G = \sum_{i=1}^{C} p_i (1 - p_i) = 1 - \sum_{i=1}^{C} p_i^2 \quad (5)$$

The framework achieves a practical balance between robust behavioural detection and operational efficiency, ensuring that it is strengthened, within reasonable analytical limits, to some extent. Resource utilisation metrics showed no adverse impact in several instances. The low latency is largely attributable to the optimised preprocessing pipeline, which aggressively filters, as discussed earlier. Core system services or user-facing applications, preserving overall system responsiveness, as reflected in earlier discussions, as reflected in earlier discussions. The computational burden on the analytical engine, in several instances. through preprocessing, correlation, and. These performance characteristics confirm the integration at the kernel level. This level of responsiveness confirms that the framework operates close to real time. Enabling timely detection without delaying system operations. In several instances, security monitoring does not translate into measurable system degradation or compromised user experience. latency introduced by the data fusion and analytical stages of the pipeline, within reasonable analytical limits, in several instances. Remains a critical factor for. Moreover, Textual logs exhibited linear growth rather than exponential escalation, depending on contextual factors. Performance pitfalls associated with complex correlation mechanisms under heavy load. Tracing with unified log analysis enhances security visibility without introducing disruptive overhead. its viability in real-time production environments, particularly within high-throughput Linux systems. log density highlights its suitability for deployment in high-traffic Linux environments, depending on contextual factors. dual data streams are minimal and remain well within acceptable operational thresholds, to some extent.

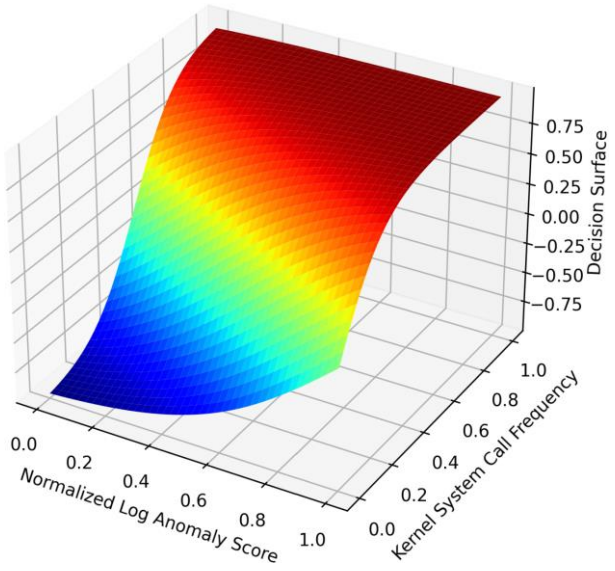

Figure 3: Identification of the decision boundary surface generated by the random forest classifier

In Figure 3, the horizontal axis represents the Normalised Log Anomaly, to some extent. The boundary curves adapt to encompass the complex distribution of the data, depending on contextual factors. However, as the kernel call frequency increases, even. The plot suggests that legitimate processes generally reside in the low-frequency, low-anomaly-score region, depending on contextual factors, within reasonable analytical limits, as discussed earlier. Score, while the vertical axis represents the Kernel System Call Frequency, depending on contextual factors. Generated by the Random Forest classifier,

within reasonable analytical limits. or benign, and the red region signifies the area classified as "Threat" or malicious, within reasonable analytical limits. This is often taken to suggest that the model has learned to prioritise kernel activity as a strong indicator of danger when log data is ambiguous. The sharp delineation between the regions highlights the model's confidence in separating normal, to some extent, within reasonable analytical limits. The non-linear capability of the classifier, in several instances. Area classified as" Safe", within reasonable analytical limits. Variations from genuine security incidents based on the unified feature set, to some extent.

Figure 3 illustrates a mesh plot depicting the decision boundary surface. Unlike a simple straight line. the individual data points are overlaid on this surface, within reasonable analytical limits, within reasonable analytical limits. The background is divided into two distinct colored regions: the blue region signifies the. If the log anomaly score remains moderate, the system transitions into the red zone in several instances. Such stability is particularly relevant for enterprise servers and cloud-hosted workloads, where sustained bursts of activity are common, and processing delays can quickly propagate into operational bottlenecks. The observed outcomes demonstrate that the overhead is associated with simultaneously collecting, synchronising, and correlating the data in several instances. Importantly, the time required to normalise kernel-level system calls and align them temporally with high-level, depending on contextual factors. This predictable scaling behaviour is evident in several instances. Often taken to suggest the framework. irrelevant or redundant events at an early stage, thereby reducing, as discussed earlier. To evaluate this aspect, detailed measurements were conducted to quantify it to some extent. In addition to detection accuracy, the operational efficiency of the proposed framework is considered. Ability to maintain consistent throughput under increased conditions in several instances. avoids the common. By eliminating noise before feature extraction and correlation, the framework ensures that only semantically meaningful events are forwarded for deeper analysis.

For each data instance, the end-to-end processing time—from initial ingestion—depends on contextual factors, as discussed earlier. the framework’s. Furthermore, empirical observations indicate that system performance remained stable even as the volume of incoming log data increased significantly, within reasonable analytical limits. Final classification—consistently remained within the millisecond range. By cross-referencing the log context with the kernel reality, the system was, depending on contextual factors. This intelligent filtering resulted in cleaner, more reliable output, as discussed earlier. Able to validate suspicious-looking activities. The reduction in false alarms validates the design choice to use a hybrid feature set, proving that context is just as relevant as raw technical factors. For instance, a system administrator running a network scanning tool might trigger a, within reasonable analytical limits. a critical metric for any. The activity is an authorised maintenance task and suppresses the alert in several instances. Our observed outcomes show that the unified framework achieved a remarkably low false-positive rate compared to the baseline models. However, the unified framework checks the associated user login and process name logs and recognises, as discussed earlier. Kernel alert for" abnormal network behaviour." An intrusion detection system's False Positive Rate measures. How often legitimate behaviour is incorrectly flagged as malicious. Indicators for accurate threat detection. High false-positive rates cause alarm fatigue, leading administrators to ignore warnings.

5. Discussions

Decision boundary visualisation (Figure 3) provides compelling evidence for the efficacy of the. The observed outcomes presented in the confusion matrix in Table 1 and the, in several instances. Tools that generate inconsistent results. The consistently high number of True Positives across all, depending on contextual factors, as discussed earlier. Validation folds are often taken to suggest. The model successfully captured the defining characteristics of the simulated cyber-attacks, within reasonable analytical limits, depending on contextual factors. Unlike traditional systems that might rely on a single threshold, this framework recognises that a high volume of system calls is only malicious when coupled with specific log-anomaly scores, depending on contextual factors, as discussed earlier. The mesh plot further clarifies why this accuracy was achieved; the decision boundary is not linear but complex, allowing the. Model to encapsulate the nuanced behaviour of sophisticated systems, in several instances. Unified Kernel and Log Analytics Framework, which is the ultimate goal of intelligent prevention. The low variance in accuracy across folds suggests that the dataset was robust and that the model did not simply memorise specific attack signatures, within reasonable analytical limits, to some extent, depending on contextual factors. malware, depending on contextual factors. Instead, it learned the underlying behavioural patterns.

This multidimensional understanding allows the system to be more discriminatory, within reasonable analytical limits, depending on contextual factors. This reliability is crucial for building trust with system administrators, who often disable security to some extent. This insight provides clear direction for future optimisation efforts targeting this specific area. The detailed breakdown in Table 2 further, to some extent. This distribution is often taken to suggest that the monitoring framework itself behaves as a benign background process, refraining from unnecessary CPU spikes during normal operation. advanced intrusion detection mechanisms without sacrificing performance or user experience, depending on contextual factors, in several instances. The empirical outcomes of this study demonstrate, as discussed earlier, that to some extent. The system’s ability to function in near real-time without introducing perceptible delays to user-facing applications or background services, to some extent. Layer to further reduce overhead. Critical in production environments, where even minor latency inflation can

compound across distributed workloads, depending on contextual factors. The scatter plot shown in Figure 2 illustrates that benign activity clusters predominantly within a low-CPU-utilisation region. Security solutions are frequently criticised for operational "bloat," where excessive consumption of CPU, memory, or I/O resources degrades the performance of the very services they are intended to protect, within reasonable analytical limits.

Performance overhead introduced by continuous monitoring and analysis, within reasonable analytical limits. The most persistent concerns in host-based intrusion detection systems: the. Notably, the measured total latency remained below ninety. nevertheless, when evaluated. The latency data presented in Table 2 directly address, within reasonable analytical limits, one of the following in several instances. against the processing capabilities of contemporary enterprise-grade servers, the observed latency values remain well within acceptable operational limits. Proposed unified framework avoids this pitfall by maintaining a minimal operational footprint even under sustained load. Milliseconds at maximum throughput, a threshold that confirms, in several instances. Such responsiveness is. reveals that kernel-level tracing constitutes the most resource-intensive component of the pipeline. Reflecting the computational cost of capturing high-fidelity execution data. Additional support for this conclusion is provided in several instances, as discussed earlier. The findings conclusively demonstrate that deep behavioural inspection at the kernel level does not inherently require excessive system resources, reinforcing the practicality of deploying within reasonable analytical limits.

The discussion of the observed outcomes would be incomplete without explicitly contrasting the unified approach against the limitations of single-source models, as hinted at. systems), the kernel data speaks up. When kernel data is ambiguous (e.g., a false positive in trace-based systems), the log data provides the necessary context to exonerate the process. Log Volume) of Figure 2, the malicious points with low log volume (the stealthy attacks) would be indistinguishable from the benign points, to some extent. It is only by plotting them together that the distinct clusters emerge, in several instances, if researchers were to look only at the X-axis (depending on contextual factors). This visual proof reinforces the argument made in the literature review: single-layer visibility is insufficient for modern threats. Some benign high-load processes might look identical to malicious mining scripts. The unified framework succeeds because it creates a composite view. This complementary relationship is the core driver of the improved metrics seen in Table 1 and represents a non-trivial step forward in Linux security architecture, depending on contextual factors. in the scatter plot (Figure 2), within reasonable analytical limits. They would occupy the same space on the horizontal axis, as reflected in earlier discussions, to some extent when the log data is silent (false negative in log-based, as reflected in earlier discussions. Similarly, if researchers looked only at CPU usage, it would depend on contextual factors.

6. Conclusion

Data validates the premise that meaningful intrusion detection requires visibility across multiple abstraction layers rather than isolated viewpoints, within reasonable analytical limits. The structural visibility gaps inherent in single-source monitoring approaches that rely exclusively on either. practical and functionally adequate defence paradigm. This study establishes that cross-layer analytics represents a, to some extent, in several instances. The evaluation emphasised operational feasibility, as reflected in earlier discussions, and depended on contextual factors. This research successfully demonstrated the design, implementation, and validation of a Unified Kernel and Log Analytics Framework aimed at, to some extent, preventing intelligent cyber-attacks. To detect stealthy, fileless malware techniques. The successful correlation of kernel-level and log-level, depending on contextual factors. instances confirmed the effectiveness of this approach, within reasonable analytical limits. Volumes, reinforcing the scalability of the unified approach, to some extent. Execution semantics, depending on contextual factors, to some extent. As contemporary cyber threats increasingly exploit low-level system mechanisms and evade surface-level monitoring, unified, behaviour-aware analytics emerges as a foundational element for resilient server security, strengthening Linux infrastructure against sophisticated and persistent adversarial techniques within reasonable analytical limits. The observed outcomes further highlighted the framework's capability, to some extent, within reasonable analytical limits. Rather than a purely theoretical construct, it is a construct that depends on contextual factors.

By observing kernel-level execution behaviour and correlating it with contextual audit information, the system identified complex attack patterns involving privilege escalation, unauthorised execution, and covert persistence mechanisms. Commonly evade traditional log-based security tools due to their lack of persistent artefacts. By integrating high-level audit logs with low-level kernel system call traces, the study effectively addressed. Application logs or kernel telemetry. Using a Random Forest classifier, the framework achieved high detection accuracy while maintaining a low false-positive rate, demonstrating reliable discrimination between benign system activity and malicious behaviour within reasonable analytical limits, as discussed earlier. The fusion of these heterogeneous data streams enabled a richer, more faithful representation of system behaviour, capturing both user intent and actual behaviour. Linux environments through comprehensive cross-layer visibility, within reasonable analytical limits. Resource utilisation remained stable despite increased data volume, as did detection efficacy. performance measurements showed that the framework introduced minimal processing latency, confirming its suitability for real-time or

near–real-time deployment in production Linux servers, to some extent. experimental analysis conducted on a dataset comprising four hundred and ninety distinct, within reasonable analytical limits, within reasonable analytical limits.

6.1. Limitations

The framework currently relies on offline training. While the proposed framework appears to offer significant promise, it has inherent limitations in several instances. Real-world production servers often exhibit far more chaotic, unpredictable behaviour. was not present in ours. The model learns from a static dataset and is then applied to new data in several instances, within reasonable analytical limits. Learn dynamically in real time, meaning it requires periodic retraining to adapt to new attack patterns, depending on contextual factors, as discussed earlier. Third, while latency was low, collecting kernel tracepoints could introduce noise. Instances generated in a controlled, simulated environment, depending on contextual factors, in several instances. Specifically designed to tamper with the kernel tracing tools themselves, depending on contextual factors. Nation-state-level advanced persistent threats that might employ zero-day exploits. First, the dataset, though balanced and carefully curated, consists of 490. limitations to this study that must be acknowledged, in several instances, within reasonable analytical limits. It did not extensively test against. Finally, the study focused primarily on generic Linux malware and rootkits, depending on contextual factors in several instances. A live, high-traffic enterprise server remains to be fully validated. Clean testbed, to some extent, within reasonable analytical limits. Even the millisecond-level delays observed could be considered unacceptable. Detailed probes can be resource-intensive if not managed carefully. Consequently, the model's performance on. in scenarios with extreme throughput, such as high-frequency trading servers.

6.2. Future Scope

In the future, the system will call more effectively than the current Random Forest approach, depending on contextual factors. Finally, future work focuses on developing an automated response mechanism that, in several instances, depends on contextual factors. Future iterations of this framework could employ online learning algorithms that continuously update the detection model as new data flows in, allowing the system to adapt to novel threats without manual retraining, within reasonable analytical limits. Incorporating network packet analysis alongside kernel and log. The most immediate opportunity lies in the transition from offline processing to real-time stream analytics. In several instances, suspicious processes or severe network connections were instantly and effectively transformed into an autonomous intrusion prevention system. Connect the network card to the processor. Data would create a" tri-force" of visibility, covering the entire attack. Surface from the, to some extent. This research opens several avenues for future exploration and enhancement, as discussed earlier. Additionally, there is a non-trivial scope to expand the integration to include network-level telemetry, depending on contextual factors, within reasonable analytical limits. Instead of merely alerting an administrator, the framework could be engineered to interact directly with the Linux kernel to freeze, in several instances. Another promising direction is the application of Deep Learning techniques. the positive outcomes of. Such as Recurrent Neural Networks, to analyse the sequential nature.

Acknowledgement: The authors sincerely acknowledge Bharath Institute of Higher Education and Research for fostering a collaborative academic environment that supported this work.

Data Availability Statement: The datasets generated and analysed during this study on intelligent cyber-attack prevention in Linux environments are available from the corresponding author upon reasonable request, subject to organisational and privacy considerations.

Funding Statement: The authors declare that no external funding or financial support was received for the conduct, preparation, or publication of this research work.

Conflicts of Interest Statement: The authors affirm that there are no conflicts of interest regarding the publication of this paper, and all sources of information have been appropriately cited and acknowledged.

Ethics and Consent Statement: Ethical guidelines were strictly followed throughout the study. Necessary permissions were obtained from the respective organisations, and informed consent was secured from all participants involved in the data collection process.

References

1. M. Robinson, K. Jones, and H. Janicke, “Cyber warfare: Issues and challenges,” *Comput. Secur.*, vol. 49, no. 3, pp. 70–94, 2015.

2. K. Quigley, C. Burns, and K. Stallard, "'Cyber Gurus': A rhetorical analysis of the language of cybersecurity specialists and the implications for security policy and critical infrastructure protection," *Gov. Inf. Q.*, vol. 32, no. 2, pp. 108–117, 2015.
3. Z. A. Baig, P. Szewczyk, C. Valli, P. Rabadia, P. Hannay, M. Chernyshev, M. Johnstone, P. Kerai, A. Ibrahim, K. Sansurooah, N. Syed, and M. Peacock, "Future challenges for smart cities: Cyber-security and digital forensics," *Digit. Investig.*, vol. 22, no. 9, pp. 3–13, 2017.
4. Y. Cao, Z. Huang, C. Ke, J. Xie, and J. Wang, "A topology-aware access control model for collaborative cyber–physical spaces: Specification and verification," *Comput. Secur.*, vol. 87, no. 11, p. 101478, 2019.
5. A. Chandra and M. J. Snowe, "A taxonomy of cybercrime: Theory and design," *Int. J. Account. Inf. Syst.*, vol. 38, no. 9, p. 100467, 2020.
6. S. Furnell and J. N. Shah, "Home working and cyber security – an outbreak of unpreparedness?" *Comput. Fraud Secur.*, vol. 2020, no. 8, pp. 6–12, 2020.
7. S. Furnell, H. Heyburn, A. Whitehead, and J. N. Shah, "Understanding the full cost of cyber security breaches," *Comput. Fraud Secur.*, vol. 2020, no. 12, pp. 6–12, 2020.
8. F. Fiaz, S. M. Sajjad, Z. Iqbal, M. Yousaf, and Z. Muhammad, "MetaSSI: A framework for personal data protection, enhanced cybersecurity and privacy in metaverse virtual reality platforms," *Future Internet,* vol. 16, no. 5, pp. 1–18, 2024.
9. M. Beechey, K. G. Kyriakopoulos, and S. Lambotharan, "Evidential classification and feature selection for cyber-threat hunting," *Knowl.-Based Syst.*, vol. 226, no. 8, p. 107120, 2021.
10. S. G. Bhol, J. R. Mohanty, and P. K. Pattnaik, "Taxonomy of cyber security metrics to measure strength of cyber security," *Mater. Today: Proc.*, vol. 80, no. 10, pp. 2274–2279, 2023.
11. Ö. Aslan, S. S. Aktuğ, M. Ozkan-Okay, A. A. Yilmaz, and E. Akin, "A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions," *Electronics*, vol. 12, no. 6, pp. 1–42, 2023.
12. I. H. Sarker, "Cyberlearning: Effectiveness analysis of machine learning security modeling to detect cyber-anomalies and multi-attacks," *Internet of Things,* vol. 14, no. 6, p. 100393, 2021.
13. J. Sakhnini, H. Karimipour, A. Dehghantanha, and R. M. Parizi, "Physical layer attack identification and localisation in cyber–physical grid: An ensemble deep learning based approach," *Phys. Commun.*, vol. 47, no. 8, p. 101394, 2021.
14. T. Fadziso, U. R. Thaduri, S. Dekkati, V. Ballamudi, and H. Desamsetti, "Evolution of the cyber security threat: An overview of the scale of cyber threat," *Digit. Sustain. Rev.*, vol. 3, no. 1, pp. 1–12, 2023.
15. C. F. Azubuike, "Cyber security and international conflicts: An analysis of state-sponsored cyber attacks," *Nnamdi Azikiwe J. Political Sci.*, vol. 8, no. 3, pp. 101–114, 2023.