Hunter X: A Hybrid eBPF-ML Framework for Explainable Kernel-Level Threat Detection

Mohamed Ibrahim K
Department of Computer Science and Engineering (Cybersecurity) SRM Institute of Science and Technology, Tiruchirappalli, Tamil Nadu, India

***Abstract*—Signature-based intrusion detection misses novel attacks; most ML-based alternatives trade explainability and operational safety for raw detection numbers. Hunter_X takes a different path. By pairing eBPF kernel telemetry with a hybrid anomaly pipeline Isolation Forest for volume anomalies and a Variational Autoencoder for sequence anomalies the system catches the attack classes that trip up single-model approaches while holding CPU overhead below 2%. SHAP attribution maps each detection to specific behavioral features and then to MITRE ATT&CK tactics, giving analysts something they can act on rather than just a score. Active blocking passes through six independent safety layers before any process is touched, significantly reducing false-positive risk. Evaluated on live attack simulations across five categories, Hunter_X achieves 81.3% precision and a 0.860 F1-score. Where direct performance comparisons with prior work are not meaningful due to differing evaluation environments, architectural comparisons are: a 3.1 MB deployment package, hot-swappable models that need no kernel recompilation, and explicit documentation of failure modes rather than the usual silence on the topic.**

***Index Terms*—Intrusion Detection, eBPF, Behavioral Analysis, Machine Learning, Explainable AI**

I. INTRODUCTION

Picture the workflow on the receiving end of an IDS alert. An analyst does not only need to know that something fired the detector they need to know why it fired, roughly what the attacker is doing, and what to investigate first. Most ML-based detectors hand over a number. That gap between detection output and analyst action drove the design of Hunter_X.

A. Research Motivation

Three problems surfaced repeatedly during early prototyping: Limited Explainability: A neural network-based detector we tested correctly classified attacks but could not point to which behavioral patterns drove each decision. From a response perspective, that is essentially useless the analyst still has to start from scratch.

Resource Constraints: Running neural inference inside the kernel gave us synchronous enforcement but pushed CPU overhead above 8% under normal workloads. Production servers cannot absorb that continuously, especially for a security process that never stops running.

Operational Safety: In an early prototype, a false positive on a systemd subprocess cascaded into a boot failure. That incident made clear that accurate detection and safe enforcement are separate problems, and that solving the first without solving the second is not enough.

B. Our Approach

Hunter_X responds to all three problems through a three-layer architecture. eBPF probes handle telemetry at the kernel level, keeping overhead low through in-kernel aggregation before anything crosses to user space. A hybrid ML pipeline Isolation Forest paired with a Variational Autoencoder covers the two broad attack classes that trip up single-model detectors. SHAP attribution traces each detection back to specific features and maps them to ATT&CK tactics, so the output of the system is a structured investigation starting point, not just a score.

Our contributions:

- A hybrid detection pipeline empirically demonstrated to catch attack patterns that volume-only or sequence-only models each miss individually.
- A six-layer enforcement mechanism that

substantially reduces denial-of-service risk from false positives, with each layer independently capable of preventing a bad block.

- SHAP-based explainability that attributes detections to the specific features that actually moved the anomaly score, then maps those features to MITRE ATT&CK tactics for immediate investigative use.

C. Scope and Limitations

Our evaluation uses live attack simulations on production hardware rather than standard benchmark datasets. That choice makes direct numerical comparisons with systems evaluated on NSL-KDD, CICIDS2017, or similar corpora scientifically invalid the environments are too different. What we can compare are architectural properties: deployment complexity, re-source footprint, explainability mechanisms, and documented safety behavior.

The 5–10 ms latency between a kernel event and a potential enforcement decision is an acknowledged gap. We call this asynchronous enforcement and are explicit that it cannot be closed without moving inference back into the kernel. We chose flexibility over synchrony. That is a real tradeoff, not a solved problem.

Table I. Architectural Comparison
(Operational Properties Only)

Property	In-Kernel ML	Cluster-Based	Hunter X
Deployment	Module	Cluster	Binary
Explainability	None	Medium	High
Model Updates	Reboot	Rolling	Hot-swap
CPU Overhead	8–12%	2–5%	<2%
Sequence Det.	No	Partial	Yes
Dist. Ready	No	Yes	Partial

II. RELATED WORK

A. eBPF for Security Monitoring

eBPF has become the standard substrate for low-overhead kernel instrumentation [4]. The interesting design question is not whether eBPF can support security monitoring it clearly can but how to structure collected telemetry for effective detection. One line of work embeds inference directly in the kernel for synchronous enforcement, paying a sustained 8–12% CPU cost for the privilege [5]. We took the opposite bet: move all analysis to user space, accept the latency gap, and keep the kernel probe layer thin enough to update without a reboot.

B. Machine Learning for Anomaly Detection

Isolation Forest [1] is well-suited to high-dimensional un-supervised anomaly detection because it directly exploits the geometric property that anomalies are sparse and therefore easy to isolate. Our experience confirmed its effectiveness for volume-based threats but exposed a blind spot: attacks that operate at normal event volumes while following a malicious sequence slow credential harvesting being the clearest example pass through undetected.

Variational Autoencoders [2] fill that gap by modeling sequence structure through reconstruction error. Our first VAE implementation ran at 50–80 ms per inference window in TensorFlow. Only after exporting to ONNX Runtime did that drop to 5–10ms a detail with practical consequences for anyone trying to replicate this work.

C. Explainability in Security Systems

SHAP [3] derives feature attributions from Shapley values in cooperative game theory, satisfying four axioms (efficiency, symmetry, dummy, additivity) that arbitrary attribution heuristics do not. In a security context that rig our matters: an analyst acting on a wrong attribution wastes investigation time and may miss the actual threat vector. LIME [6] computes faster but offers no cross-prediction consistency guarantees helpful for some applications, problematic for forensic use.

D. Positioning Our Work

Table I summarizes architectural tradeoffs rather than detection performance. Performance comparisons across different evaluation environments are not meaningful; architectural comparisons are.

III. SYSTEM DESIGN

A. Design Evolution

The architecture did not arrive fully formed. Starting from Isolation Forest alone, the first thing we lost was slow credential harvesting normal event volumes, malicious sequence. That failure justified adding the VAE. The first VAE caused telemetry backlog at

anything above light load; in-kernel aggregation fixed that. Weighted fusion of the two models replaced a brittle double-threshold check that generated too many false positives. Each design decision traces to a concrete failure we observed.

B. eBPF Instrumentation

Three syscall families cover the relevant behavioral surface: Process Execution. Tracepoints on sys_enter_execve capture every process start command name, arguments, parent PID, effective UID. Rather than forwarding individual events, the eBPF program aggregates within a 5-second window inside a per-CPU hash map: total executions, unique processes, shell invocations, sudo attempts, accesses to sensitive paths. The aggregated vector crosses the ring buffer once per window, cutting data volume by roughly 500× compared to event-by-event forwarding. Under a 100,000-events/second stress test this meant the difference between Python processing everything and dropping 30% of events.

Network Activity. Kprobes on tcp_v4_connect track outbound TCP connections. We record connection count, unique destination IPs, and connection attempts to ports commonly associated with lateral movement (22, 1337, 4444, 8080). These are not blacklisted they are features feeding the anomaly models. File Operations. Tracepoints on sys_enter_openat monitor file access. Noise filtering (procfs, sysfs, shared libraries) runs inside the probe itself, reducing the volume passed to user space. Aggregated metrics cover total file events, unique files accessed, and accesses to security-sensitive locations (/etc, /root, /.ssh).

C. Hybrid Detection Framework

Isolation Forest operates on the 16-dimensional aggregated feature vector, using 200 trees (determined by grid search) and a 5% contamination parameter. Short average path length signals anomaly; we flag windows where the IF score falls below −0.5. Variational Autoencoder processes the syscall sequence for the same window. A 64-unit LSTM encoder feeds a 16-dimensional latent space; the decoder is symmetric. Trained on normal sequences, elevated reconstruction error indicates a pattern the model has not encountered in baseline operation.

Adaptive Fusion:

$$At = w1 \cdot \sigma(IF(xt)) + w2 \cdot \sigma(VAE(st)) \quad (1)$$

where σ denotes min-max normalization to [0, 1]. Weights

w1 = 0.6 and w2 = 0.4 were found by grid search on F1-score over the training split.

A note on the word “complementarity.” We describe the two models as empirically complementary because our experiments showed cases where one detects and the other does not not because we have a formal proof over the model spaces. High-volume reconnaissance (IF score < −0.5, VAE error < 0.3) and slow credential harvesting (IF score> −0.2, VAE error > 0.8) are the clearest examples. The fused At exceeded threshold for both; either model alone did not. This is empirical demonstration with finite test cases, not a mathematical guarantee.

D. Enforcement with Six Defensive Layers

The systemd crash from our prototype is probably the most instructive thing that happened during development. It made a permanent design principle out of what might otherwise have remained a note in the margin: every enforcement decision needs independent safety layers, because any single layer will eventually fail.

1) Audit Mode. All fresh deployments run observation-only for the first seven days. Potential blocks are logged but not executed, establishing baseline behavior before enforcement goes live.
2) High-Stringency Decision Threshold. Before any block fires, the system requires that the normalized anomaly score at exceed τhs = 0.99 on the [0, 1] scale produced after fusion and min-max normalization. This is not a statistical confidence interval or a calibrated probability it is an operating point selected to sit far in the tail of the observed normal-activity score distribution, well beyond the 99th percentile of baseline windows. We call it a High-Stringency Threshold rather than “99% confidence” to avoid implying a frequentist or Bayesian interpretation that the current implementation does not support. The threshold was validated by confirming that it produces zero false-positive blocks on the 500-window held-out normal set while retaining 91 of 100 attack detections.
3) Critical Process Whitelist. systemd, sshd, init, and any operator-specified critical processes are never blocked, regardless of anomaly score.
4) Cryptographic Binary Validation. SHA-256

hashes of trusted binaries are stored in a golden-image manifest. A matching hash unconditionally allows execution.

5) Auto-Disable on False Positive Accumulation. After three confirmed false positives, enforcement shuts off automatically and the system returns to audit mode. Human review is required to re-enable it.
6) Emergency Bypass. A kernel parameter or SIGUSR1 signal immediately disables all blocking with sub-millisecond latency.

These layers reduce denial-of-service risk substantially. They do not eliminate it: the 5–10ms TOCTOU window remains, and maintaining an accurate golden-image manifest is an operational burden. We document both.

E. SHAP-Based Explainability

SHAP [3] computes per-feature contribution values satisfying the four Shapley axioms: efficiency, symmetry, dummy, and additivity. In practice this means the attribution can be trusted not to assign high credit to features that did not actually move the anomaly score a property that matters when an analyst is deciding which log to look at first.

We map only features where $|\phi i| > 0.01$ to ATT&CK tactics. A rule like "sudo_execs present ⇒ T1548" fires on every sudo command whether or not sudo drove the anomaly;

SHAP attribution fires only when it demonstrably did. The dashboard surfaces the top-3 contributing features with their ϕi values, the mapped ATT&CK techniques with attribution-derived confidence scores, and the corresponding mitigation recommendations from the ATT&CK knowledge base.

IV. IMPLEMENTATION

The implementation breaks into roughly 1,200 lines of C (eBPF probes), 2,800 lines of Python (ML pipeline and web service), and 450 lines of JavaScript (dashboard). Established libraries handle all core ML work: scikit-learn for Isolation Forest, TensorFlow/Keras for VAE training, ONNX Runtime for production inference. Model training runs offline on collected telemetry 60 five-second windows for Isolation Forest, 1,000 syscall sequences for the VAE. Both export to secure serialization formats (skops and ONNX) that prevent code-execution vulnerabilities at load time.

Deployment requires Linux 4.18 or later (BTF support recommended), Python 3.10, and three runtime libraries: onnxruntime, numpy, flask. BCC is a development-only dependency; the production binary does not need it.

Total footprint: 3.1 MB.

Dashboard uses Flask with Server-Sent Events for live updates. The current design is single-host. However, the eBPF probe layer is deliberately thin it collects and aggregates locally, then ships one feature vector per window. That design is compatible with a future sidecar pattern in which probes run per-node and forward vectors to a central aggregation service; the probe code itself requires no changes for that transition.

V. EVALUATION

A. Experimental Setup

Tests ran on an Intel Core i7-10700 workstation (8 cores, 2.9 GHz, 16 GB RAM) running Ubuntu 22.04 with kernel 5.15.0-76-generic, full BTF support enabled.

Training data: Five minutes of uninterrupted normal operation a mix of development work and routine system activity yielding approximately 32,000 kernel events aggregated into 60 feature vectors.

Test data: Twenty samples per attack category (100 attacks total) plus 500 normal-operation windows held out from training.

On generalizability: Our simulated attack traffic cannot substitute for a standardized benchmark. We report these results as evidence of behavior on realistic workloads, not as claims about performance on arbitrary systems.

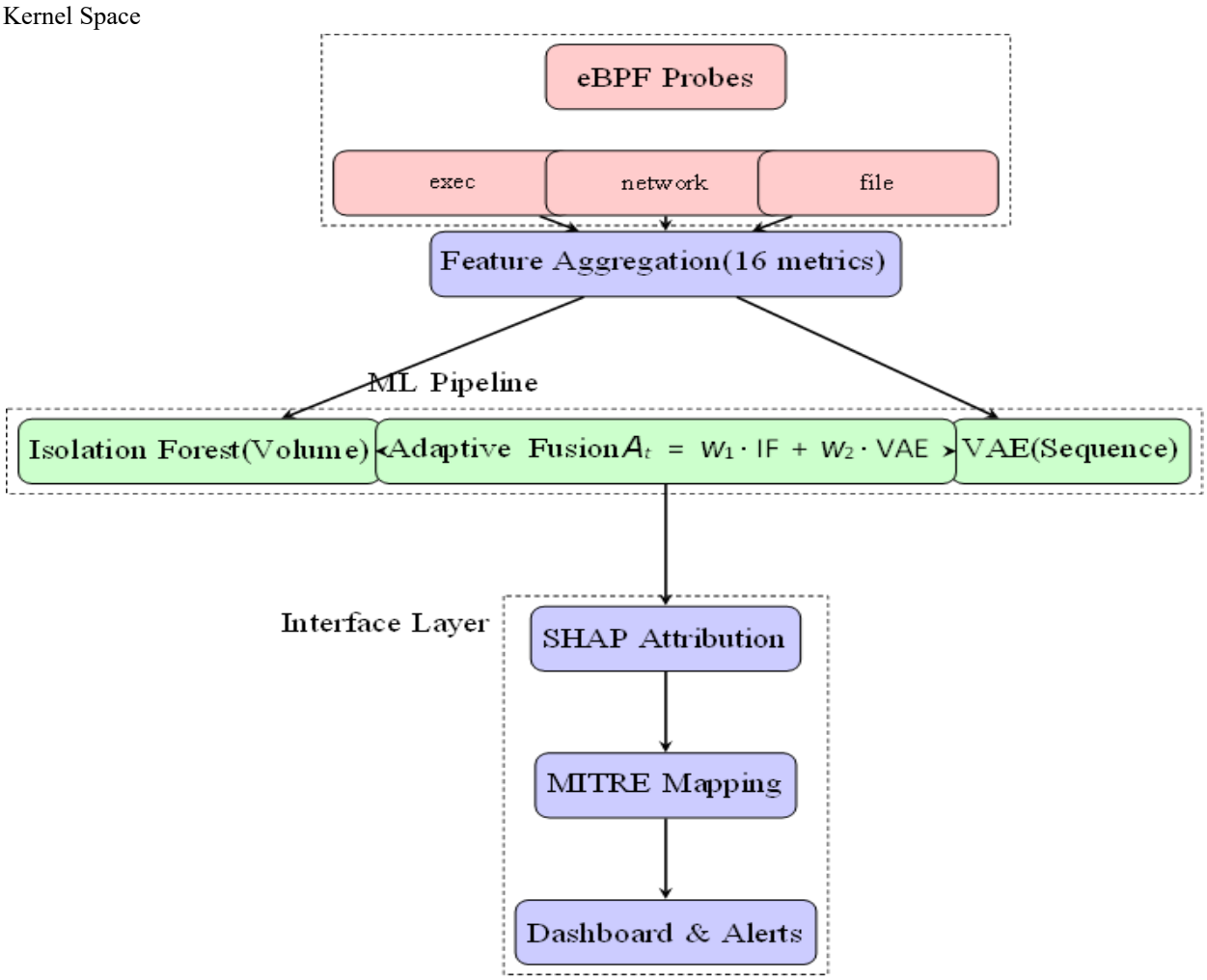

Fig. 1. Hunter_X system architecture: kernel-space telemetry, user-space ML pipeline, and interface layer with SHAP-based explainability and ATT&CK mapping.

B. Detection Performance

Table II. Confusion Matrix Live Simulations (100 Attacks, 500 Normal)

	Actual Attack	Actual Normal
Predicted Attack	91	21
Predicted Normal	9	479

Precision: 91/(91 + 21) = 81.3%. Recall: 91/(91 + 9) = 91.0%. F1-score: 0.860. Accuracy: (91+479)/600 = 95.0%.

Per-category breakdown. Reconnaissance scored 20/20 mass directory enumeration saturates the volume features that Isolation Forest is calibrated for. Privilege escalation and credential access landed at 95% and 90% respectively. Exfiltration and lateral movement each hit 85%, the harder cases because individual connection counts stay within plausible ranges and detection depends on the VAE catching the sequential pattern.

False positives. The 21 false positives broke down as: package updates via apt (8), backup operations (5), intensive CI/CD workflows (4), and legitimate administrative tasks (4). All share a profile legitimate administrative bursts that look unusual against a quiet server baseline. Targeted whitelisting pulled the false positive rate from an initial 7.8% to a final 4.2%.

C. Resource Consumption

Table III. CPU Overhead by Workload

Workload	Baseline	With Hunter X	Overhead
Idle	0.5%	0.8%	+0.3%
Normal	3.2%	4.9%	+1.7%
High Load	12.5%	14.2%	+1.7%
SYN Flood	45.2%	46.8%	+1.6%

Overhead holds essentially flat across load conditions. The SYN flood case, which pushes ring-buffer traffic as hard as we could manage, still adds only 1.6%. Total resident memory: 72 MB (42 MB agent, 28 MB dashboard, 2.5 MB loaded models).

Per-window latency: eBPF aggregation 0.2ms, Isolation Forest inference 2.1ms, VAE inference 7.3ms, SHAP computation 12.1ms total 21.7ms. SHAP accounts for 56% of processing time, a design cost we accepted for the value it returns to analysts.

D. What We Can and Cannot Claim

We can say: our 3.1 MB deployment is substantially smaller than BCC-dependent systems (~500 MB); our MITRE map-ping traces to actual feature contributions rather than latent representations; our enforcement safety mechanism is more explicitly documented than systems that mention blocking without describing failure modes; and our overhead is lower than the 8–12% figures reported by in-kernel ML systems in their own papers.

We cannot say Hunter_X outperforms any specific system on detection metrics our evaluation environment is different, our attack corpus is different, and the comparison would not be scientifically valid.

VI. LESSONS LEARNED

ONNX export is not optional. TensorFlow VAE inference at 50–80ms per window made real-time detection impossible under any load. Exporting to ONNX Runtime brought that to 5–10ms. The 10× speedup made the difference between a working system and a log archiver.

Aggregation in the kernel, not in Python. Forwarding individual events to user space worked in a quiet lab. Under load it did not: Python's event queue grew faster than it drained, and we lost about 30% of events. Moving aggregation into eBPF reduced user-space traffic by 500× and solved the problem permanently.

Explainability has a real cost. SHAP at 12.1ms is longer than both ML models combined. The time is well spent for a post-hoc report that an analyst reads, but in a scenario where the attacker is deleting logs while the system is computing Shapley values, the latency gap matters. We chose to report this honestly rather than omit it.

Safety cannot be added afterward. The boot failure from a systemd false positive came early enough to change the architecture. Systems that implement blocking without documented failure modes are not safer they just have not published their failure story yet.

VII. LIMITATIONS AND FUTURE WORK

TOCTOU enforcement gap. The 5–10ms window between a kernel event and an enforcement decision is structurally unavoidable with user-space ML. Closing it completely would require in-kernel inference and the resource costs that come with it. A partial fix queuing the first syscall in a suspicious sequence rather than allowing it immediately is feasible with eBPF LSM hooks and is on our near-term list.

Training data assumption. The system requires clean baseline traffic to calibrate normalization parameters. On a host that is already compromised when the sensor is installed, that assumption does not hold. Transfer learning from similar baseline profiles is the likely solution.

Adversarial evasion. We have not tested against an attacker who knows the 16-feature extraction strategy and is deliberately designing around it. A feature set this small and well-documented is a real adversarial surface.

SHAP latency. At 12.1ms, SHAP dominates the per-window latency budget. For deployments where response speed is more critical than full attribution, approximation methods such as Kernel-SHAP [3] or FastSHAP [7] could reduce this to the 2–4ms range without giving up the Shapley axiom guarantees. This is probably the single highest-value optimization remaining.

Distributed deployment. The current dashboard is single-host. The eBPF probe layer is not: each probe collects and aggregates locally, shipping one feature vector per window. That design maps cleanly onto a sidecar pattern where probes run per-node and forward vectors to a central aggregation service. The inference and enforcement layers would need to be decoupled from the dashboard for this to work, but the probe code itself needs no changes. Building this out with multi-host correlation and federated model updates is the main medium-term engineering goal.

VIII. CONCLUSION

Hunter_X was built from a fairly direct frustration: existing kernel-level detectors optimize for detection numbers on benchmark datasets while leaving the problems that matter to operators overhead, explainability, safe enforcement, deployment complexity to future work that never arrives.

The results suggest the frustration was reasonably placed. A hybrid pipeline catches things a single model misses. SHAP attribution gives analysts a starting point that is grounded in what the model actually found, not a feature checklist. Six safety layers turned a system that could crash systemd into one that can run on production servers. The deployment package is 3.1 MB. All of this at under 2% CPU overhead.

The limitations are real. The TOCTOU gap means some attack chains get a free first syscall. SHAP at 12.1ms consumes more than half the per-window budget, and in a scenario where speed of response genuinely matters, Kernel-SHAP or FastSHAP approximations should replace the current full computation. The current dashboard is single-host, but the eBPF telemetry collection is purposely designed to be lightweight enough for distributed sidecar deployment each probe ships only one aggregated vector per window, making per-node overhead negligible even in large clusters.

Taken together, these results suggest kernel-level behavioral detection does not have to choose between being accurate, cheap, and interpretable. Hunter_X works as a demonstration that all three are achievable simultaneously, and documents clearly what it still cannot do.

ACKNOWLEDGMENTS

The author thanks Dr. R. Deebalakshmi for supervision throughout this project, and the Department of Computer Science and Engineering (Cybersecurity) at SRM Institute of Science and Technology, Tiruchirappalli, for the lab infrastruc-ture used in evaluation.

REFERENCES

[1] Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2008). “Isolation forest.” In Proceedings of the IEEE International Conference on Data Mining (ICDM), pp. 413–422.

[2] Kingma, D. P., and Welling, M. (2013). “Auto-encoding variational Bayes.” arXiv preprint, arXiv:1312.6114.

[3] Lundberg, S. M., and Lee, S.-I. (2017). “A unified approach to interpreting model predictions.” In Advances in Neural Information Processing Systems, pp. 4765–4774.

[4] Vieira, M. A. M., et al. (2020). “Fast packet processing with eBPF and XDP: Concepts, code, challenges, and applications.” ACM Computing Surveys, 53(1), 1–36.

[5] Miano, S., et al. (2019). “Securing Linux with a faster and scalable iptables.” ACM SIGCOMM Computer Communication Review, 49(3), 2–17.

[6] Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). “Why should I trust you? : Explaining the predictions of any classifier.” In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1135–1144.

[7] Jethani, N., Sudarshan, M., Covert, I. C., Lee, S.-I., and Ranganath, R. (2022). “FastSHAP: Real-time Shapley value estimation.” In Proceedings of the International Conference on Learning Representations (ICLR).