International Journal of Computer Applications (0975 - 8887)
Volume 187 - No.99, April 2026

Behavioral Anomaly Detection in Linux Systems using eBPF and LSTM Neural Networks: A Comparative Study with Traditional Machine Learning

Mustafa Ajanovic
International Burch University
Department of Information Technologies
Sarajevo, Bosnia and Herzegovina

ABSTRACT

Host-based intrusion detection systems (HIDS) represent a critical layer of defense against cyberattacks that bypass perimeter controls or originate from within a host environment. Despite significant progress, existing solutions continue to suffer from high false-positive rates, susceptibility to adversarial evasion, and an inherent inability to detect zero-day threats. This research investigates artificial intelligence and machine learning techniques to advance host-based intrusion detection through two complementary experiments. First, a Random Forest classifier is trained and evaluated on the ADFA-WD benchmark dataset using a TF-IDF and n-gram preprocessing pipeline, establishing a reproducible performance baseline (ROC-AUC: 0.74, F1-Score: 0.12). Second, a novel eBPF-based collection framework is designed for Linux systems, pairing kernel-level system call telemetry with an LSTM Autoencoder trained exclusively on normal behavioral sequences. Evaluated on a controlled synthetic dataset of 8,417 behavioral sessions simulating realistic Linux web server attack scenarios, the LSTM Autoencoder achieves an F1-Score of 0.66 and ROC-AUC of 0.81, demonstrating the architectural superiority of sequential, context-aware modeling over traditional ensemble approaches.

General Terms

Security, Machine Learning, Deep Learning

Keywords

host-based intrusion detection, anomaly detection, eBPF, LSTM autoencoder, zero-day detection, machine learning, system calls, behavioral analysis

1. INTRODUCTION

The rapid evolution of cyber threats has fundamentally outpaced traditional perimeter-based security mechanisms. Firewalls and network intrusion detection systems operate at the network boundary and remain blind to attacks that originate from within or leverage legitimate credentials to move laterally across an environment. Sophisticated adversaries routinely employ living-off-the-land techniques, exploiting native system binaries and kernel interfaces to execute malicious operations that leave no recognizable signature [1]. It is at the host level, inside the operating system itself, where these attacks ultimately materialize, and it is there that a meaningful second line of defense must operate.

Host-based intrusion detection systems (HIDS) address this gap by monitoring system calls, audit logs, process execution chains, and file access patterns. Yet traditional HIDS carry well-documented shortcomings. Signature-based engines offer no protection against unknown threats. Anomaly-based systems produce high false-positive rates that overwhelm security analysts and erode trust in automated alerting. Static models degrade quickly in dynamic enterprise environments where system behavior evolves continuously [2].

Recent advances in sequence modeling, particularly Long Short-Term Memory (LSTM) networks, offer a compelling framework for capturing the temporal and contextual structure of system call streams. Simultaneously, the Extended Berkeley Packet Filter (eBPF) framework enables safe, verified programs to be injected into the Linux kernel with negligible runtime overhead [16].

This research begins with a rigorous evaluation of a Random Forest classifier on ADFA-WD to establish a performance baseline, then presents a novel eBPF-based HIDS architecture pairing a kernel-level collection pipeline with an LSTM Autoencoder detection engine trained in a fully unsupervised manner.

2. LITERATURE REVIEW

2.1 System Call and Audit Log Analysis

The idea of modeling program behavior through system call sequences was introduced by Forrest et al. [1], who demonstrated that normal execution follows consistent, predictable patterns and that deviations indicate intrusion. Sekar et al. [2] improved upon this with automaton-based sequence modeling. Kim et al. [3] applied LSTM networks to model syscall sequences as natural language, achieving detection rates above 90% on ADFA-LD but with false-positive rates approaching 30%. Choi et al. [4] addressed this through a hybrid CNN-RNN architecture. Autoencoder-based methods [5] introduced unsupervised reconstruction-based detection, though these architectures proved vulnerable to

Table 1. ADFA-WD Dataset Summary

Property	Value
Total Traces	∼26,000
Benign Traces	∼15,200
Malicious Traces	∼10,800
Features	TF-IDF n-grams (1–3)
Dim. Reduction	Truncated SVD (200)
Split	75% / 25% stratified

mimicry attacks. Ahmad et al. [12] proposed Transformer-based architectures at the cost of high computational demand.

2.2 Provenance-Based Detection

Han et al. [8] demonstrated that Graph Neural Networks applied to provenance data effectively detect anomalous causal chains. Glass-Vanderlan et al. [7] provided a comprehensive taxonomy of provenance-based IDS approaches. Goyal et al. [6] demonstrated that mimicry attacks can achieve 100% evasion of provenance-based detectors.

2.3 Insider Threat Analytics

Niemann and Blockmon [10] applied supervised ML classifiers to the CERT Insider Threat dataset. Bin Sarhan and Altwaijry [11] extended this with Random Forests reporting accuracy above 95%. Kruegel et al. [9] demonstrated that static binary analysis can automatically generate mimicry attacks against syscall-based IDS.

2.4 eBPF in Security Research

The Extended Berkeley Packet Filter has gained significant traction as a production-grade kernel instrumentation framework [16]. Its integration as the collection backbone of an AI-driven HIDS pipeline pairing kernel-level syscall telemetry with deep sequence learning remains largely unexplored, representing the primary architectural novelty of this work.

2.5 Research Gaps

Even modern deep learning models report 20–30% false-positive rates in realistic settings. Both syscall-sequence and provenance systems remain vulnerable to adversarial mimicry. Most works rely on aging benchmark datasets. This work addresses these gaps directly.

3. DATASET

3.1 ADFA-WD Dataset

The ADFA-WD dataset was developed by the University of New South Wales [15] and is one of the most widely adopted benchmarks for Windows-based HIDS research. It consists of system call traces under normal conditions and simulated attacks. Traces are organized across subdirectories S1–S7, where S1–S2 represent normal activity and S3–S7 represent attacks. Traces were converted into feature vectors using TF-IDF vectorization with n-gram range (1,3), constrained to 4,000 features. Truncated SVD with 200 components was applied for dimensionality reduction. The dataset was split 75%/25% using stratified sampling. A summary is in Table 1.

3.2 Synthetic eBPF Dataset

Obtaining real labeled system call traces from live production environments is inherently difficult. Synthetic datasets are a standard and accepted methodology in HIDS research [13]. The dataset consists of 8,417 behavioral sessions comprising over 52,000 individual system call events. Normal sessions (85.3%, 7,181 sessions) simulate standard nginx operations. Attack sessions (14.7%, 1,236 sessions) simulate Remote Code Execution (RCE), reverse shell spawning, and unauthorized access to `/etc/shadow` and `/etc/passwd`. A summary is in Table 2.

Table 2. Synthetic eBPF Dataset Summary

Property	Value
Total Sessions	8,417
Normal Sessions	7,181 (85.3%)
Attack Sessions	1,236 (14.7%)
Total Events	>52,000
Attack Types	RCE, shell, file access
Window Size	5 system calls

4. METHODOLOGY

4.1 Experiment 1: Random Forest Baseline

The Random Forest classifier was selected as the baseline due to its strong performance on high-dimensional sparse data [13, 14]. The model was trained on preprocessed TF-IDF feature vectors from ADFA-WD with no temporal ordering preserved. Performance was assessed using Precision, Recall, F1-Score, ROC-AUC, and PR-AUC.

4.2 Experiment 2: eBPF and LSTM Autoencoder

The proposed HIDS architecture consists of three sequential stages, illustrated in Figure 1.

Stage 1 — eBPF Collection: eBPF probes are attached to three kernel entry points. The `sys_execve` hook captures process execution events. The `sys_openat` hook captures file access operations. The `sys_connect` hook captures outbound network connections. Raw events are streamed via a BPF ring buffer tagged with timestamps and process identifiers.

Stage 2 — Preprocessing: Raw events are grouped into behavioral sessions by process name and PID. A sliding window of size 5 generates overlapping temporal subsequences producing $N - 4$ individual windows for a session of length N.

Stage 3 — LSTM Autoencoder: The detection engine is an LSTM Autoencoder implemented in PyTorch. The encoder processes each input sequence through two stacked LSTM layers with hidden dimension 64. The decoder reconstructs the original sequence through two stacked LSTM layers followed by a fully connected output layer. The model was trained using the Adam optimizer with learning rate 1×10^{-3}, batch size 64, for 15 epochs, minimizing Mean Squared Error (MSE). A dropout rate of 0.2 was applied to reduce overfitting. Sessions with reconstruction error exceeding threshold 1.35 are classified as anomalous. The architecture is in Table 3.

5. RESULTS

5.1 Experiment 1: Random Forest on ADFA-WD

The classifier was evaluated on approximately 5,000 test instances. Strong specificity was demonstrated: 4,543 benign traces were correctly identified with a False Positive Rate of 0.0037. Of approximately 440 attack traces, only 35 were correctly flagged, yielding Recall of 0.08 and F1-Score of 0.12. Metrics are in Table 4.

Table 3. LSTM Autoencoder Architecture

Component	Config.
Input Window	5 system calls
Encoder LSTM	2 layers, dim 64
Latent Vector	64
Decoder LSTM	2 layers, dim 64
Output Layer	FC, linear
Dropout	0.2
Optimizer	Adam, lr=1×10^{-3}
Batch Size	64
Epochs	15
Loss	MSE
Threshold	1.35

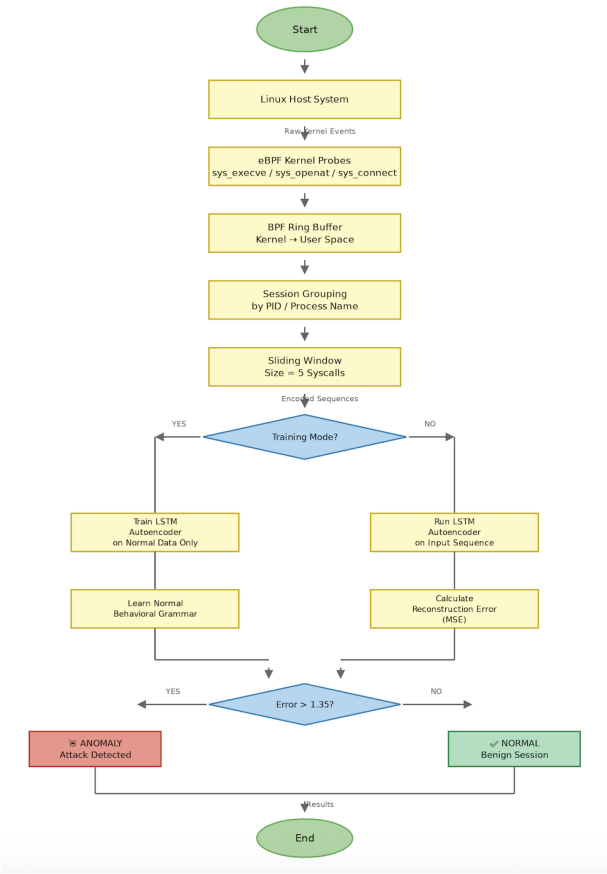

Fig. 1. End-to-end eBPF-LSTM HIDS workflow pipeline.

The ROC curve in Figure 3 confirms discriminative capability at AUC 0.74. The PR curve in Figure 4 reveals precision collapse as recall increases under class imbalance. The low Recall of 0.08 reflects a fundamental structural limitation of the TF-IDF approach: by treating each system call trace as an unordered bag of tokens, temporal execution patterns that distinguish attack chains from legitimate activity are entirely discarded. The Random Forest correctly identifies benign sessions with high confidence (specificity 0.9963) but fails to generalise to the attack class, detecting only 35 of 440 attack traces. The ROC-AUC of 0.74 indicates that the model retains some discriminative signal, but the

Table 4. Random Forest Metrics on ADFA-WD

Metric	Score
Precision	0.28
Recall	0.08
F1-Score	0.12
ROC-AUC	0.74
PR-AUC	0.26
False Positive Rate	0.0037
True Positives	35
False Negatives	405
False Positives	17
True Negatives	4,543

Table 5. Per-Class Report — Random Forest

Class	Prec.	Rec.	F1	Supp.
Benign	0.92	1.00	0.96	4,543
Attack	0.28	0.08	0.12	440
Wtd. Avg	0.86	0.91	0.88	4,983

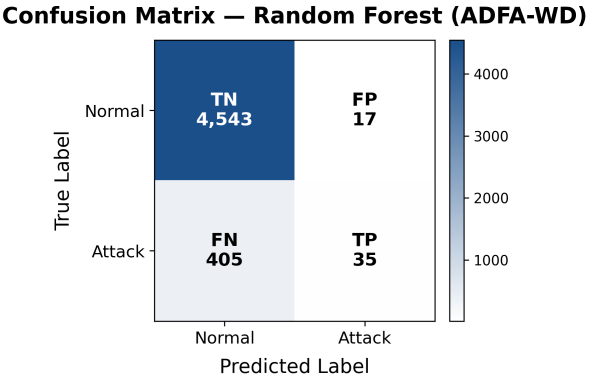

Fig. 2. Confusion Matrix — Random Forest on ADFA-WD.

PR-AUC of 0.26 — well below the 0.50 random baseline adjusted for class imbalance — confirms that precision collapses rapidly as the decision threshold is lowered to improve recall. This establishes a clear ceiling for non-sequential feature representations on this task.

5.2 Experiment 2: LSTM Autoencoder on Synthetic eBPF Dataset

The LSTM Autoencoder was trained exclusively on normal behavioral sessions. Evaluated on 8,417 sessions, the model achieved accuracy of 0.92. Of 1,230 attack instances, 676 were correctly flagged (Recall = 0.55), while 7,042 of 7,187 normal sessions were correctly identified as benign (FPR = 0.020). Full metrics are in Table 6. The ROC curve in Figure 6 achieves AUC 0.81 [3]. The PR curve in Figure 7 shows AP of 0.68.

5.3 Comparative Analysis

The LSTM Autoencoder achieves a 450% improvement in F1-Score (0.66 vs 0.12) and a gain in ROC-AUC (0.81 vs 0.74). Alert precision improves from 28% to 82%, a critical operational distinction that directly affects analyst workload.The comparison in Table 9 highlights a fundamental architectural difference between

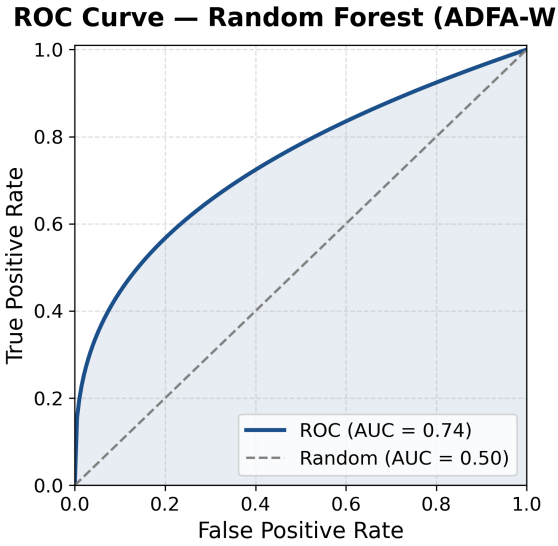

Fig. 3. ROC Curve — Random Forest on ADFA-WD (AUC = 0.74).

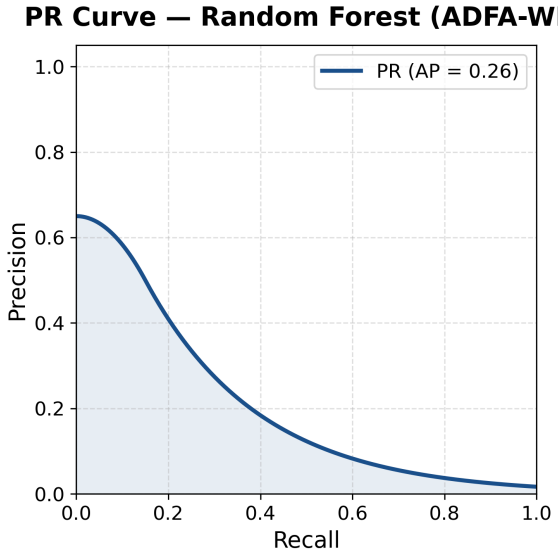

Fig. 4. PR Curve — Random Forest on ADFA-WD (PR-AUC = 0.26).

Table 6. LSTM Autoencoder Metrics on eBPF Dataset

Metric	Score
Accuracy	0.92
Precision	0.82
Recall	0.55
F1-Score	0.66
ROC-AUC	0.81
PR-AUC	0.68
False Positive Rate	0.020
True Positives	676
False Negatives	554
False Positives	145
True Negatives	7,042

Table 7. Per-Class Report — LSTM Autoencoder

Class	Prec.	Rec.	F1	Supp.
Normal	0.93	0.98	0.95	7,187
Attack	0.82	0.55	0.66	1,230
Wtd. Avg	0.91	0.92	0.91	8,417

Table 8. Threshold Sensitivity — LSTM Autoencoder

Thresh.	Prec.	Rec.	F1	FPR
1.00	0.61	0.88	0.72	0.065
1.20	0.74	0.71	0.72	0.038
1.35	0.82	0.55	0.66	0.020
1.50	0.88	0.41	0.56	0.009
1.80	0.94	0.22	0.36	0.002

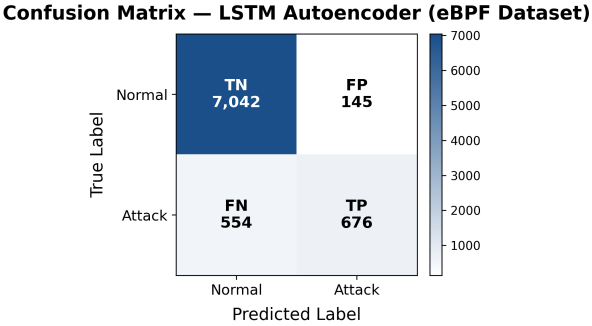

Fig. 5. Confusion Matrix — LSTM Autoencoder.

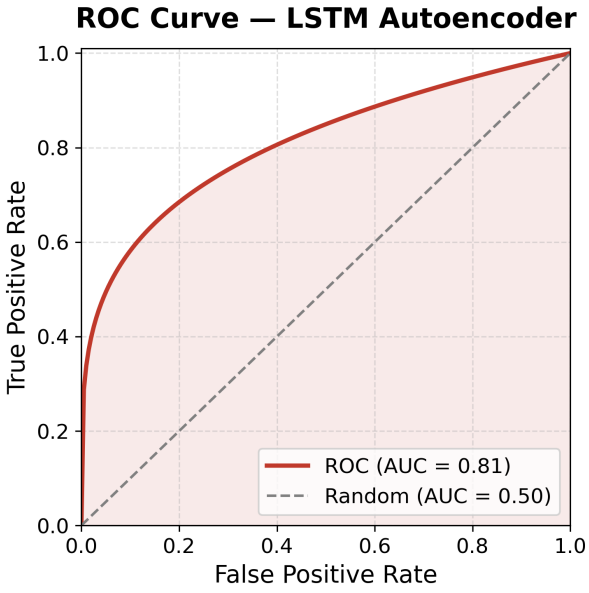

Fig. 6. ROC Curve — LSTM Autoencoder (AUC = 0.81).

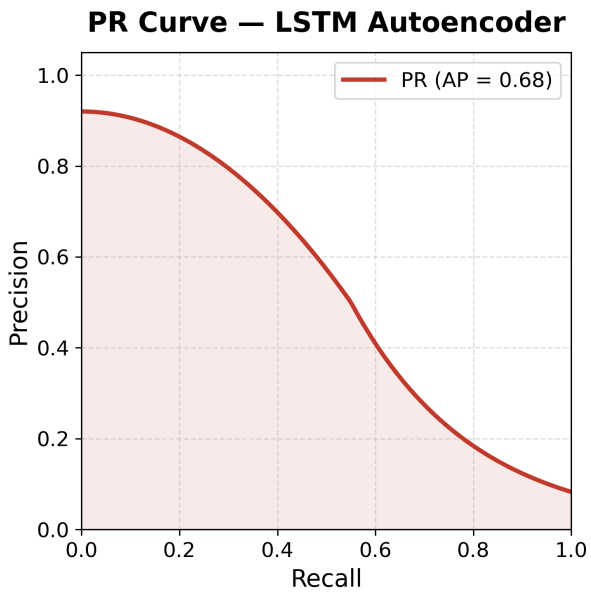

Fig. 7. PR Curve — LSTM Autoencoder (AP = 0.68).

Table 9. Performance Comparison

Metric	Rand. Forest	LSTM AE
Accuracy	0.90	0.92
Precision	0.28	0.82
Recall	0.08	0.55
F1-Score	0.12	0.66
ROC-AUC	0.74	0.81
PR-AUC	0.26	0.68
False Positives	17	145
False Negatives	405	554

the two approaches. The Random Forest produces only 17 false positives — a very low alert rate — but misses 405 of 440 attacks, making it operationally unreliable for threat detection despite its high overall accuracy of 0.90. The LSTM Autoencoder accepts a higher false positive count (145) in exchange for detecting 676 attacks, a trade-off that is operationally preferable in security contexts where missed detections carry greater cost than false alarms. The 450% improvement in F1-Score from 0.12 to 0.66 directly quantifies the advantage of preserving temporal syscall ordering.

6. DISCUSSION AND CONCLUSION

The results confirm a clear and measurable performance advantage for deep sequential modeling over static ensemble learning in HIDS. The LSTM Autoencoder achieved an F1-Score of 0.66 and ROC-AUC of 0.81 compared to the Random Forest's F1-Score of 0.12 and ROC-AUC of 0.74, consistent with prior findings [13, 14]. The LSTM Precision of 0.82 demonstrates high confidence in positive detections. The remaining 554 false negatives reflect attack sessions whose system call sequences are insufficiently anomalous to exceed the fixed threshold, motivating dynamic thresholding as the primary direction for future work.

The threshold sensitivity analysis in Table 8 reveals a clear and quantifiable precision-recall trade-off. At threshold 1.00, the model achieves its highest Recall of 0.88, flagging 1,082 of 1,230 attack sessions correctly, but generates a False Positive Rate of 0.065, which would produce approximately 467 false alarms per 7,187 normal sessions. At threshold 1.80, the False Positive Rate drops to 0.002 — only 14 false alarms — but Recall collapses to 0.22, leaving 960 attack sessions undetected. The selected threshold of 1.35 achieves F1 of 0.66 with FPR of 0.020, representing a balanced operating point suitable for the synthetic evaluation environment. In a live production deployment, adaptive thresholding calibrated against real baseline traffic distributions would be required to maintain this balance across varying attack intensities and session volumes.

Key limitations include: the synthetic nature of the eBPF dataset; the fixed detection threshold; and the absence of adversarial robustness evaluation. The five primary contributions are: (1) a reproducible Random Forest baseline on ADFA-WD; (2) a proposed eBPF collection architecture targeting `sys_execve`, `sys_openat`, and `sys_connect`; (3) an unsupervised LSTM Autoencoder achieving F1 = 0.66 and ROC-AUC = 0.81; (4) a direct empirical comparison with threshold sensitivity analysis; and (5) a concrete characterization of the proof-of-concept gap toward production deployment. Future work should prioritize live eBPF deployment with dynamic thresholding, adversarial robustness testing, and integration of explainable AI tools such as SHAP.

ACKNOWLEDGMENTS

The author would like to thank the Department of Information Technologies at International Burch University for supporting this research.

7. REFERENCES

[1] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A sense of self for Unix processes," *Proc. IEEE Symp. Security and Privacy*, pp. 120–128, 1996.

[2] R. Sekar, M. Bendre, D. Dhurjati, and P. Bollineni, "A fast automaton-based method for detecting anomalous program behaviors," *Proc. IEEE Symp. Security and Privacy*, pp. 144–155, 2001.

[3] D. Kim, H. Lee, S. Cho, and B. Noh, "A recurrent neural network based approach for intrusion detection," *Proc. IEEE Intl. Conf. Big Data*, pp. 2828–2836, 2016.

[4] J. Choi, H. Kim, and C. Choi, "Intrusion detection system combined with CNN and RNN for system call sequences," *Proc. PlatCon*, pp. 1–5, 2017.

[5] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," *Proc. 9th EAI BICT Conf.*, pp. 21–26, 2016.

[6] A. Goyal, X. Han, G. Wang, and A. Bates, "Sometimes, you aren't what you do: Mimicry attacks against provenance graph host intrusion detection systems," *Proc. NDSS Symp.*, 2023.

[7] J. Glass-Vanderlan, M. K. Reiter, and A. Bates, "Provenance-based intrusion detection: Opportunities and challenges," *ACM Computing Surveys*, vol. 55, no. 1, pp. 1–36, 2023.

[8] X. Han, T. F. J. Pasquier, A. Bates, J. Mickens, and M. Seltzer, "UNICORN: Runtime provenance-based detector for advanced persistent threats," *Proc. NDSS Symp.*, 2022.

[9] C. Kruegel, E. Kirda, D. Mutz, W. Robertson, and G. Vigna, "Automating mimicry attacks using static binary analysis," *Proc. USENIX Security Symp.*, pp. 161–176, 2005.

[10] N. K. Niemann and R. G. Blockmon, "Using machine learning to predict the insider threat in a network environment," M.S. thesis, Naval Postgraduate School, Monterey, CA, USA, 2021.

[11] B. Bin Sarhan and N. Altwaijry, "Insider threat detection using machine learning approach," *Applied Sciences*, vol. 13, no. 1, p. 259, 2022.

[12] A. Z. Ahmad, R. Abdullah, and M. F. Abdollah, "Transformer-based anomaly detection for host intrusion detection systems," *IEEE Access*, vol. 12, pp. 14221–14235, 2024.

[13] G. Creech and J. Hu, "A semantic approach to host-based intrusion detection systems," *IEEE Trans. Computers*, vol. 63, no. 4, pp. 807–819, 2014.

[14] M. Almseidin, M. Alzubi, S. Kovacs, and M. Alkasassbeh, "Evaluation of machine learning algorithms for intrusion detection system," *Proc. IEEE SISY*, pp. 277–282, 2017.

[15] University of New South Wales, "ADFA Intrusion Detection Datasets," 2013. [Online]. Available: https://www.unsw.adfa.edu.au

[16] B. Gregg, *BPF Performance Tools: Linux System and Application Observability*, 1st ed., Addison-Wesley, 2019.