

LIGHT-HIDS: A Lightweight and Effective Machine Learning-Based Framework for Robust Host Intrusion Detection

Onat Gungor*, Ishaan Kale*, Jiasheng Zhou, and Tajana Rosing

Department of Computer Science and Engineering, University of California, San Diego

{ogungor, ikale, rjzhou, tajana}@ucsd.edu

Abstract—The expansion of edge computing has increased the attack surface, creating an urgent need for robust, real-time machine learning (ML)-based host intrusion detection systems (HIDS) that balance accuracy and efficiency. In such settings, inference latency poses a critical security risk, as delays may provide exploitable opportunities for attackers. However, many state-of-the-art ML-based HIDS solutions rely on computationally intensive architectures with high inference costs, limiting their practical deployment. This paper proposes LIGHT-HIDS, a lightweight machine learning framework that combines a compressed neural network feature extractor trained via Deep Support Vector Data Description (DeepSVDD) with an efficient novelty detection model. This hybrid approach enables the learning of compact, meaningful representations of normal system call behavior for accurate anomaly detection. Experimental results on multiple datasets demonstrate that LIGHT-HIDS consistently enhances detection accuracy while reducing inference time by up to $75\times$ compared to state-of-the-art methods. These findings highlight its effectiveness and scalability as a machine learning-based solution for real-time host intrusion detection.

I. INTRODUCTION

The increasing reliance on digital infrastructure has elevated cybersecurity to a critical priority across diverse sectors. In response to the rising complexity and frequency of cyber threats, machine learning has become an essential tool for developing intelligent and adaptive defense mechanisms [1]. A key application of machine learning within this domain is Host Intrusion Detection Systems (HIDS), where ML models are trained to analyze system-level data to identify potentially malicious activities [2]. Among available data sources, system call traces have proven particularly effective due to their fine-grained insight into program execution behavior [3]. Recent research has applied deep learning models, particularly those developed for natural language processing, to capture the sequential dependencies present in system call sequences [4]. Although these models offer strong detection performance, their high computational complexity and inference latency hinder practical deployment in real-time scenarios.

These challenges become even more pronounced in resource-constrained environments such as IoT gateways, embedded controllers, and other edge computing platforms. Devices in these settings often operate with limited memory,

compute power, and energy availability, making the deployment of conventional deep learning-based HIDS difficult without compromising responsiveness [5]. Moreover, edge devices frequently serve as the initial point of contact within a network, necessitating rapid, localized threat detection to reduce dependence on cloud infrastructure [6]. Due to the strict real-time requirements of edge platforms, high inference latency in host intrusion detection systems can result in significant delays when analyzing system calls prior to program execution, thereby hindering timely operation [7]. Additionally, during continuous monitoring, prolonged detection times increase the opportunity for attackers to exploit system vulnerabilities before anomalies are identified and addressed [2]. These considerations underscore the importance of lightweight machine learning approaches that balance inference speed, model compactness, and detection accuracy to meet the latency and efficiency demands of practical edge deployments. While TinyML techniques have shown potential for enabling efficient intrusion detection on constrained devices [8]–[10], existing research predominantly targets network telemetry rather than system call-based detection, highlighting a significant gap in securing edge-hosted systems.

To address these challenges, we propose LIGHT-HIDS (Figure 1), a lightweight host-based intrusion detection framework. Our method employs a hybrid anomaly detection pipeline where system call sequences are first processed by a compressed neural network feature extractor trained via Deep Support Vector Data Description. The resulting compact and discriminative feature embeddings are then provided to an efficient novelty detector that assigns anomaly scores used for classification. Experimental evaluation shows that LIGHT-HIDS reliably improves detection accuracy while achieving inference time reductions of up to $75\times$ relative to current state-of-the-art approaches. Unlike existing methods that are either too computationally intensive for practical edge deployment or too simplistic to effectively detect complex anomalies, LIGHT-HIDS achieves a balanced trade-off between efficiency and performance. To the best of our knowledge, this work represents one of the first lightweight and accurate HIDS designed with resource-constrained computing environments in mind, thereby addressing an important gap in edge security.

*Both authors contributed equally to this research.

II. RELATED WORK

A. Host-based intrusion detection systems (HIDS)

Host-based intrusion detection systems (HIDS) are a fundamental component of modern cybersecurity architectures, designed to monitor internal system activity and detect signs of malicious behavior [11]. Among the various types of host-level data, system calls are especially valuable, as they mediate interactions between user-space applications and the operating system kernel [12]. This fine-grained visibility makes system calls well-suited for capturing the underlying behavioral patterns of executing programs. While traditional signature-based HIDS effectively detect known threats, they struggle to identify new attacks. Anomaly-based approaches address this limitation by leveraging machine learning models to characterize normal system behavior and detect deviations indicative of malicious activity [13]. Such techniques are particularly effective in identifying zero-day exploits and evolving threats.

Traditional anomaly-based HIDS methods, e.g., Isolation Forest [14], depend heavily on manual feature engineering and face difficulties capturing complex temporal dependencies in system call sequences [15]. Deep learning addresses these limitations by automatically learning hierarchical features and effectively modeling intricate sequential patterns. Recurrent architectures like Long Short-Term Memory (LSTM) networks have been foundational in this area, excelling at capturing long-range dependencies and detecting anomalies from system calls [16]. Expanding on this foundation, hybrid models that integrate Convolutional Neural Networks (CNNs) with RNNs enhance performance by leveraging CNNs to extract local patterns and RNNs to capture temporal context [17]. CNNs using dilated convolutions, such as those inspired by WaveNet, offer a parallelizable alternative that efficiently captures dependencies across varying temporal scales [4]. Transformer-based models have recently gained prominence in HIDS due to their self-attention mechanism, which simultaneously captures long-range relationships across entire sequences [18].

While Deep Learning (DL) methods offer powerful capabilities for modeling complex sequential patterns, they often come with substantial computational overhead that limits their suitability for real-time detection in resource-constrained environments. These models typically require large memory footprints and incur high inference latency due to their depth and complexity. Additionally, the sequential nature of system call data can lead to long input sequences, further increasing computation time and power consumption. Such constraints make it challenging to deploy conventional DL approaches directly on edge devices, highlighting the need for lightweight alternatives that can balance detection accuracy with efficiency.

B. Hybrid Anomaly Detection

Hybrid anomaly detection, which combines DL-based feature extraction with classical ML models, has been extensively studied [19]. In this paradigm, DNNs transform raw input into compact, discriminative representations, reducing the need for manual feature engineering. These features are then leveraged

by traditional algorithms, such as Isolation Forest or One-Class SVM, for anomaly detection. For example, autoencoders trained on normal data use reconstruction error or latent embeddings as input to classical detectors [20]. Another common approach employs CNNs followed by pooling layers to automatically extract hierarchical local patterns and distill salient features into compact representations suitable for anomaly detection [21]. We adopt this CNN-to-pooling strategy due to its demonstrated superior predictive performance.

Despite their success in general anomaly detection, these hybrid methods remain underexplored and insufficiently tailored to the unique challenges of host intrusion detection, such as the need for computational efficiency on resource-constrained devices, highlighting a critical gap that our work addresses. Building on this, we utilize DeepSVDD [22], which trains neural networks to map normal data into a compact hypersphere in latent space, concentrating normal samples near a learned center while pushing anomalies away. This approach facilitates learning compact representations of normal behavior that are highly effective for subsequent anomaly detection. These extracted features are then used to train a novelty detector, constituting a hybrid framework. To the best of our knowledge, we present the first efficient and effective hybrid solution specifically designed for HIDS.

III. LIGHT-HIDS FRAMEWORK

We propose LIGHT-HIDS, a lightweight host-based intrusion detection system designed for resource-constrained edge environments. As shown in Figure 1, LIGHT-HIDS adopts a hybrid anomaly detection pipeline that balances efficiency and detection performance. The framework is comprised of five principal modules: ① data preprocessing, ② feature extractor training, ③ novelty detector training, ④ threshold creation, and ⑤ anomaly classification. Integer-mapped system call sequences are first processed by a neural network-based feature extractor, and the resulting embeddings are fed into a novelty detector that outputs anomaly scores. These scores are then thresholded to classify sequences as benign or malicious. When applied in real-world scenarios, system call sequences can be tokenized and passed through this pipeline to generate a binary classification indicating the sequence's safety. Next, we describe the main modules of LIGHT-HIDS in detail.

A. Data Preprocessing

Prior to model training, the raw system call sequences were subjected to several preprocessing steps. We first extracted the system call traces, discarding auxiliary metadata such as process IDs and timestamps. Each unique system call name was then tokenized by mapping it to a distinct integer ID, producing a numerical representation suitable for neural network input. Our method follows an anomaly detection paradigm where models are trained solely on normal data to learn typical system behavior. Accordingly, the training and validation sets contain only normal sequences, while the test set includes both normal and anomalous samples to enable comprehensive evaluation of detection performance.

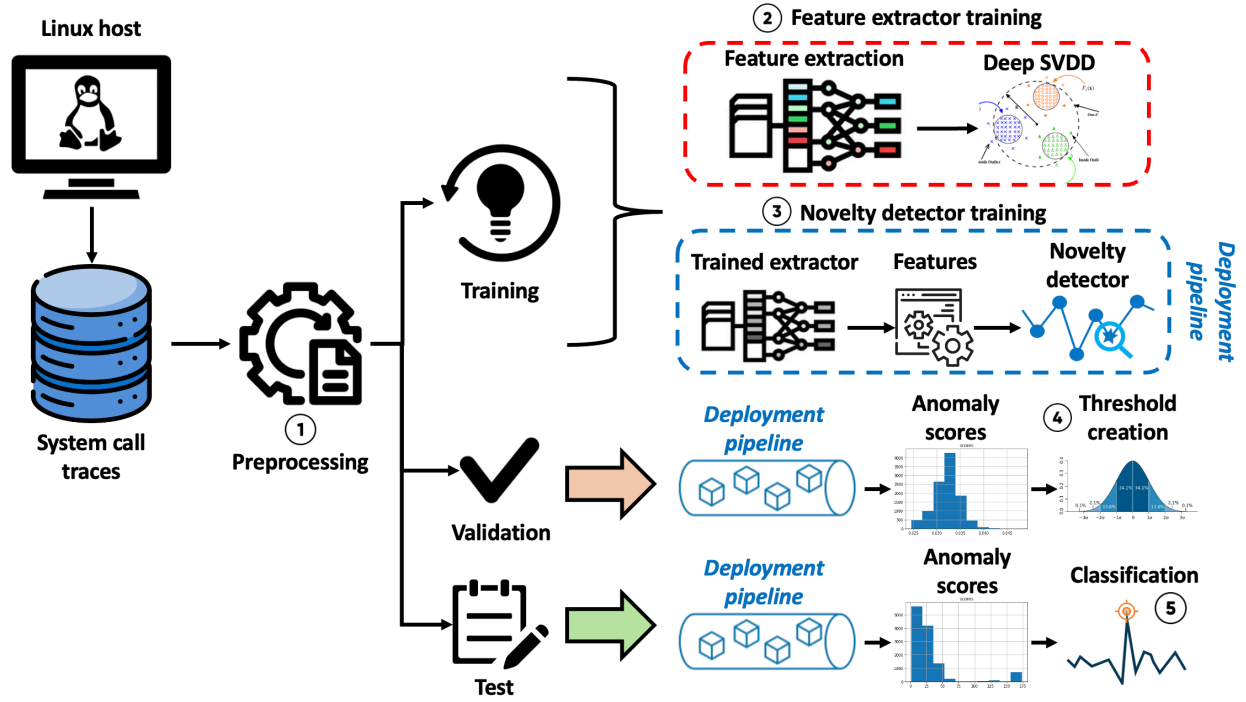


Fig. 1: Proposed Lightweight Machine Learning-Based Host Intrusion Detection System (LIGHT-HIDS)

B. Feature Extractor Training

To learn robust feature representations from system call sequences, we selected a neural network training framework specifically designed to optimize the feature extractor for anomaly detection. This motivated the adoption of Deep Support Vector Data Description (DeepSVDD), a deep learning-based anomaly detection framework [22]. DeepSVDD extends the classical SVDD approach by integrating a neural network. Within this framework, the neural network functions as a feature extractor that maps raw input data, such as sequences of system calls, into a lower-dimensional latent space where representations of normal data are enclosed within a compact hypersphere. The model jointly optimizes the radius of this hypersphere, minimizing its volume to tightly encapsulate the normal data, while simultaneously updating the parameters of the feature extractor through backpropagation. This joint optimization facilitates the learning of feature representations that are intrinsically suited for anomaly detection.

Our feature extractor architecture begins with a high-dimensional embedding space representing each system call in the input sequence. The feature space is progressively reduced along the sequence dimension using a convolutional neural network (CNN) composed of ReLU-activated one-dimensional convolutional (Conv1D) layers, followed by max pooling. After the sequence dimension is fully reduced, the feature extractor concludes with a fully connected layer. This layer projects the remaining filter outputs into the target feature space, generating the feature vectors that are subsequently input to the DeepSVDD objective. To facilitate deployment in resource-constrained environments, the feature extractor

is further optimized using TensorFlow Lite (TFLite) [23] compression with quantization, significantly reducing model size and improving inference efficiency.

C. Novelty Detector Training

After training the feature extractor using the DeepSVDD framework, we use it to generate latent representations for the training and validation sequences. These feature vectors are then provided to a lightweight, unsupervised anomaly detection model. Specifically, we adopt Isolation Forest [14], which serves as the novelty detector in our framework due to its efficiency and suitability for resource-constrained environments. To train the novelty detector, the original training sequences are first transformed into feature vectors using the trained feature extractor, and these vectors are then used to fit the novelty detection model. Once the novelty detector is trained on the feature vectors, it can assign an anomaly score to each new input vector. This establishes a “deployment pipeline”, wherein a sequence of integer-mapped system calls is processed through the trained feature extractor to generate a corresponding feature vector, which is then evaluated by the fitted novelty detector to produce an anomaly score. This end-to-end framework enables lightweight and efficient inference from raw system call sequences to anomaly detection outputs.

D. Threshold Creation

For evaluation, anomaly scores are first obtained by passing the validation sequences, which consist exclusively of normal data, through the proposed “deployment pipeline”. Based on the distribution of the resulting anomaly scores, we define a fixed threshold to convert the continuous scores

into binary anomaly labels, where a label of 0 denotes benign behavior and 1 indicates the presence of an anomaly. This binarization process is essential for downstream evaluation using classification-based metrics such as precision, recall, and F1-score. Specifically, the threshold is chosen as two standard deviations above the mean of the anomaly score distribution, since Isolation Forest assigns higher scores to inputs that are more likely to be anomalous. This method relies on the assumption that the score distribution of normal data is approximately Gaussian and is intended to capture statistical outliers that are indicative of abnormal behavior.

E. Classification via Threshold

The predetermined threshold is employed as a decision boundary to classify sequences within the unseen test set. Each test sequence is processed through the deployment pipeline to yield an anomaly score. Sequences exhibiting scores above the established threshold are classified as anomalous, whereas those with scores below the threshold are classified as benign. This thresholding mechanism provides a standardized procedure for converting continuous anomaly scores into binary classifications. Subsequently, the predicted labels are compared against ground truth annotations to rigorously assess the detection performance of the model.

IV. EXPERIMENTAL ANALYSIS

A. Experimental Setup

1) *Hardware*: Model training was conducted on a single NVIDIA GTX 2080 Ti GPU. For inference, we used CPU-only execution on an Intel Xeon Gold 6230 to simulate the lack of GPU acceleration typically encountered in resource-constrained edge devices. Additionally, we evaluate our models on the NVIDIA Jetson Orin NX with 16 GB of RAM [24], a representative platform for real-world edge deployments.

2) *Datasets*: We evaluate our approach using two subsets from the Leipzig Intrusion Detection Data Set (LIDS) [25], a modern, system call-based dataset designed for anomaly-based HIDS research. The first subset corresponds to brute-force attacks, which constitute approximately 22% of attacks on edge gateways [26]. These attacks involve repeated, systematic attempts to guess login credentials, generating distinctive system call patterns due to frequent authentication failures. The second subset focuses on SQL injection attacks, a representative class of malware injection attacks that account for 26% of edge threats [25]. These attacks manipulate input to inject malicious SQL queries into applications, often causing harmful database operations and system behavior that can be captured at the system call level. Both subsets include labeled sequences of benign and malicious activity, facilitating comprehensive evaluation in an anomaly detection context. Together, these attack types cover a substantial portion of potential host-based threats, offering strong potential for generalizing HIDS performance to other attack categories.

3) *Evaluation Metrics*: Our evaluation focuses on two key aspects: detection performance and inference time.

- **Detection Performance**: Precision, recall, and F1-score to assess the model's ability to correctly identify malicious activity while minimizing false alarms.
- **Inference Time**: The elapsed time from inputting system call sequences to generating anomaly predictions, reflecting the model's computational overhead.

The primary goal is to show that our method maintains high detection accuracy while significantly lowering inference latency, enabling timely intrusion detection.

4) *Baselines*: We evaluate our approach against three categories of baselines: host-based intrusion detection (HIDS) methods, time-series anomaly detection models, and classical anomaly detection techniques.

- **HIDS methods**: We implemented several deep learning models commonly used in host-based intrusion detection, including WaveNet, LSTM, and CNNRNN hybrid, following the methodology of Ring et al. [4]. These models use language modeling techniques to predict the next system call in a sequence. For training, input-target pairs were constructed by shifting system call sequences by one position. During inference, anomaly scores were derived from the model's prediction certainty, calculated using the maximum softmax probabilities across the sequence.
- **Time-series anomaly detection**: We compare our approach against leading time-series anomaly detection models, including COUTA [27], TimesNet [28], and Deep SVDD [22]. These baselines were implemented using the DeepOD library [27], [29].
- **Classical anomaly detection**: We also evaluate traditional models, including Isolation Forest (IF) [14] and One-Class SVM (OCSVM) [30], which output anomaly scores analogous to those produced by our language modeling baselines. For input, we use average-pooled feature embeddings extracted from the LSTM model proposed in [4], resulting in fixed-length vectors matching the dimensionality of the system call embedding space.

B. Results

1) *Detection Performance*: Figure 2 presents a comparative analysis of the anomaly detection performance of our proposed method against state-of-the-art models on the brute-force (Figure 2a) and SQL injection (Figure 2b) datasets. In each subfigure, the x-axis represents the evaluation metrics (precision, recall, and F1 score) while the y-axis indicates the corresponding values. Different colors correspond to the various baseline methods, with LIGHT-HIDS highlighted in light green as the rightmost column. We observe that **LIGHT-HIDS achieves the highest F1 score across both datasets**, indicating robust performance in distinguishing intrusive activity from normal behavior relative to the baseline models. Although LIGHT-HIDS does not consistently outperform all baselines on every individual metric, such as recall on the SQL dataset, it maintains competitive performance by remaining

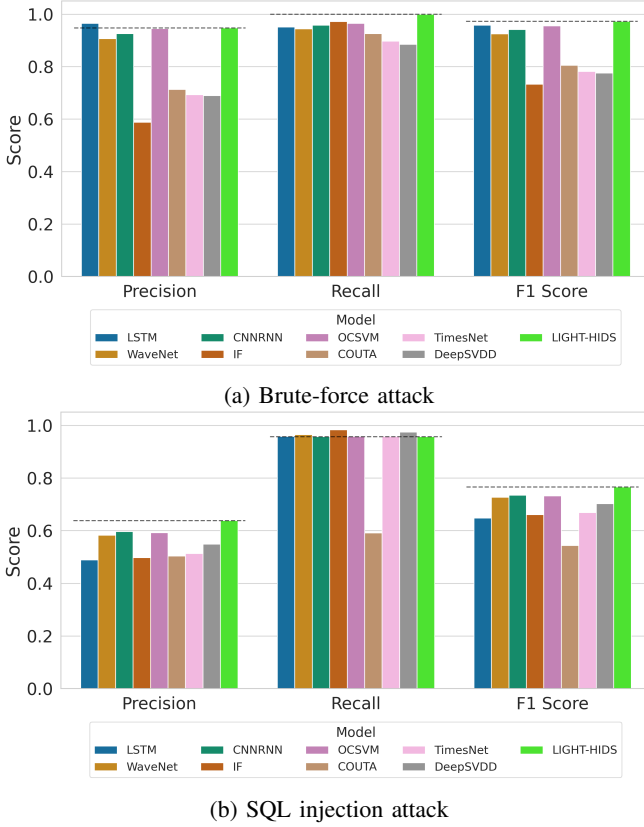


Fig. 2: Performance Comparison with State-of-the-Art

TABLE I: F1 Score Comparison by Baseline Category

Dataset	Category	Best Score	LIGHT-HIDS	↑ (%)
Brute Force	HIDS	0.958	0.973	1.57
	Time-Series	0.775		25.48
	Traditional	0.955		1.85
SQL Injection	HIDS	0.735	0.766	4.23
	Time-Series	0.702		9.03
	Traditional	0.732		4.59

within 0.05 of the top-performing models in these cases. Furthermore, methods specifically designed for HIDS exhibit better performance relative to other baseline categories.

To quantify improvements in F1 score, Table I reports LIGHT-HIDS's gains over the best-performing baseline within each category. For the brute-force dataset, the top models are LSTM (HIDS baselines), COUTA (time-series anomaly detection), and OCSVM (traditional anomaly detection), while for the SQL injection dataset, the leading models are CNNRNN, DeepSVDD, and OCSVM, respectively. LIGHT-HIDS demonstrates notable improvements on the brute-force dataset, ranging from 1.57% to 25.48%, with the largest gain observed over the time-series baseline. Similarly, on the SQL injection dataset, improvements range from 4.23% to 9.03%, again highlighting superiority over time-series methods. Table II further compares LIGHT-HIDS to the top three performing models per dataset, showing that HIDS-specific methods and

TABLE II: F1 Score Comparison over Best Baselines

Dataset	Model	F1 Score	LIGHT-HIDS	↑ (%)
Brute Force	LSTM	0.958	0.973	1.57
	OCSVM	0.955		1.85
	CNNRNN	0.942		3.29
SQL Injection	CNNRNN	0.735	0.766	4.23
	OCSVM	0.732		4.60
	WaveNet	0.726		5.42

OCSVM represent the strongest baselines.

When all baselines are taken into consideration, LIGHT-HIDS achieves up to **32.74%** improvement with an average gain of **16.24%** on the brute-force dataset, and up to **40.83%** improvement with an average gain of **14.11%** on the SQL injection dataset. These consistent gains across attack types and model categories underscore the robustness and effectiveness of LIGHT-HIDS in enhancing host-based intrusion detection.

2) *Inference Time*: Table III presents the inference-time speedup of LIGHT-HIDS relative to the best-performing baseline models on each dataset, as identified in Table II. Evaluations were conducted on both an Intel Xeon CPU and an NVIDIA Jetson Orin NX device. On the Brute Force dataset, LIGHT-HIDS achieves up to $33.77\times$ faster inference than CNNRNN on the CPU, while maintaining a superior detection accuracy (F1 score of 0.973 compared to 0.942). Similarly, for SQL Injection, LIGHT-HIDS delivers $75\times$ faster inference than CNNRNN, alongside a 4.23% improvement in F1 score. Comparable trends are observed on the Jetson, where LIGHT-HIDS significantly reduces inference latency relative to baseline models while sustaining competitive accuracy. While models such as LSTM and OCSVM demonstrate competitive F1 scores on Brute Force (0.958 and 0.955, respectively), LIGHT-HIDS achieves higher accuracy and 3.6 to $6.4\times$ faster inference on the CPU. For SQL Injection, LIGHT-HIDS outperforms OCSVM and WaveNet by 4.6% and 5.42% in F1, while offering nearly $10\times$ faster inference.

The Intel Xeon CPU achieves faster inference due to its higher processing power, whereas the NVIDIA Jetson Orin NX balances inference speed with energy efficiency, highlighting important trade-offs in deploying host intrusion detection systems on edge hardware. **These results show that LIGHT-HIDS delivers high detection accuracy with substantially reduced inference latency, making it well-suited for real-time host intrusion detection across both high-performance servers and resource-constrained edge devices.**

V. CONCLUSION

The rapid expansion of IoT and edge computing has introduced significant security challenges alongside new opportunities. Machine learning (ML)-based Host Intrusion Detection Systems (HIDS) are critical for accurately detecting complex attack patterns. However, most state-of-the-art ML-based solutions rely on computationally heavy models that hinder practical deployment in resource-constrained edge environments. This creates a pressing need for lightweight, efficient ML

TABLE III: Inference Time and Speedup of LIGHT-HIDS on Intel Xeon Gold 6230 (CPU-only) and NVIDIA Jetson Orin NX

Model	Brute Force				SQL Injection			
	Xeon (CPU)		Jetson		Xeon (CPU)		Jetson	
	Time (s)	Speedup	Time (s)	Speedup	Time (s)	Speedup	Time (s)	Speedup
LIGHT-HIDS	0.004	—	0.030	—	0.030	—	0.293	—
LSTM	0.016	3.61×	0.420	13.90×	0.113	3.76×	4.310	14.70×
WaveNet	0.073	16.57×	0.116	3.84×	0.295	9.83×	0.616	2.10×
CNNRNN	0.149	33.77×	1.011	33.47×	2.248	74.93×	10.939	37.31×
OCSVM	0.028	6.41×	0.056	1.87×	0.288	9.60×	0.494	1.68×

approaches that maintain high detection accuracy. To address these limitations, we present LIGHT-HIDS, an efficient host-based intrusion detection framework that employs a compressed neural network to extract compact and meaningful features from system call data. This feature extractor is paired with a lightweight novelty detection model, enabling accurate and fast anomaly detection. Our results show that LIGHT-HIDS improves detection accuracy while reducing inference time by up to 75× compared to state-of-the-art methods, demonstrating its potential as a scalable solution for real-time intrusion detection in resource-constrained environments.

ACKNOWLEDGMENTS

This work has been funded in part by NSF, with award numbers #1826967, #1911095, #2003279, #2052809, #2100237, #2112167, #2112665, and in part by PRISM and CoCoSys, centers in JUMP 2.0, an SRC program sponsored by DARPA.

REFERENCES

- [1] A. Kabbani, M. Kabbani, and F. Kabbani, “A survey on edge computing (ec) security challenges: Classification, threats, and mitigation strategies,” *Electronics*, vol. 14, no. 4, p. 175, 2025.
- [2] A. Singh *et al.*, “Real time intrusion detection in edge computing using machine learning techniques,” *Turkish Journal of Engineering*, vol. 9, no. 2, pp. 385–393, 2025.
- [3] M. Banach *et al.*, “Design of a distributed hids for iot backbone components,” in *2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR)*, pp. 81–88, 2020.
- [4] J. H. Ring, C. M. Van Oort, S. Durst, V. White, J. P. Near, and C. Skalka, “Methods for host-based intrusion detection with deep learning,” *Digital Threats: Research and Practice*, vol. 2, no. 4, pp. 26:1–26:29, 2021.
- [5] P. Joshi *et al.*, “Enabling all in-edge deep learning: A literature review,” *IEEE Access*, vol. 11, pp. 3431–3460, 2023.
- [6] T. Zhukabayeva *et al.*, “Cybersecurity solutions for industrial internet of things—edge computing integration: Challenges, threats, and future directions,” *Sensors*, vol. 25, no. 1, p. 213, 2025.
- [7] H. Chen, S. Chen, W. Li, W. Yang, and Y. Feng, “Impact analysis of inference time attack of perception sensors on autonomous vehicles,” *arXiv preprint arXiv:2505.03850*, 2025.
- [8] H. Im and S. Lee, “Tinyml-based intrusion detection system for in-vehicle network using convolutional neural network on embedded devices,” *IEEE Embedded Systems Letters*, 2024.
- [9] T. Saranya, D. Jeyamala, S. Indra Priyadarshini, *et al.*, “A secure framework for miot: Tinyml-powered emergency alerts and intrusion detection for secure real-time monitoring,” in *2024 8th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*, pp. 13–21, IEEE, 2024.
- [10] P. Fusco *et al.*, “Tinyml-based intrusion detection system for handling class imbalance in iot-edge domain using siamese neural network on mcu,” in *International Conference on Advanced Information Networking and Applications*, pp. 389–402, Springer, 2025.
- [11] H. Satilmiş *et al.*, “A systematic literature review on host-based intrusion detection systems,” *Ieee Access*, vol. 12, pp. 27237–27266, 2024.
- [12] M. Liu, Z. Xue, X. Xu, C. Zhong, and J. Chen, “Host-based intrusion detection system with system calls: Review and future trends,” *ACM computing surveys (CSUR)*, vol. 51, no. 5, pp. 1–36, 2018.
- [13] I. Martins *et al.*, “Host-based ids: A review and open issues of an anomaly detection system in iot,” *Future Generation Computer Systems*, vol. 133, pp. 95–113, 2022.
- [14] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation forest,” in *2008 eighth IEEE international conference on data mining*, pp. 413–422, IEEE, 2008.
- [15] A. L. Buczak and E. Guven, “A survey of data mining and machine learning methods for cyber security intrusion detection,” *IEEE Communications surveys & tutorials*, vol. 18, no. 2, pp. 1153–1176, 2015.
- [16] G. Kim *et al.*, “Lstm-based system-call language modeling and robust ensemble method for designing host-based intrusion detection systems,” *arXiv preprint arXiv:1611.01726*, 2016.
- [17] A. Chawla, B. Lee, S. Fallon, and P. Jacob, “Host based intrusion detection system with combined cnn/rnn model,” in *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 149–158, Springer, 2018.
- [18] P. Kunwar, K. Aryal, M. Gupta, M. Abdelsalam, and E. Bertino, “SoK: Leveraging transformers for malware analysis,” *IEEE Transactions on Dependable and Secure Computing*, 2025.
- [19] R. Chalapathy, A. Santarcangelo, and S. Chawla, “Deep learning for anomaly detection: A survey,” *ACM Computing Surveys (CSUR)*, vol. 52, no. 6, pp. 1–35, 2019.
- [20] R. Liu and K. Gryllias, “A deep support vector data description method for anomaly detection in helicopters,” *Mechanical Systems and Signal Processing*, vol. 155, p. 107611, 2021.
- [21] O. Al-Qatf and T. Al-Qatf, “Anomaly detection using convolutional neural networks (cnn),” *International Journal of Advanced Computer Technology*, vol. 13, no. 1, 2024.
- [22] L. Ruff *et al.*, “Deep one-class classification,” in *International Conference on Machine Learning (ICML)*, pp. 4393–4402, PMLR, 2018.
- [23] “Tensorflow lite.” <https://www.tensorflow.org/lite>. Accessed: 2025-07-09.
- [24] NVIDIA Corporation, “Jetson orin nx module.” <https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/>, 2024. Accessed: 2025-07-15.
- [25] M. Grimmer, T. Kaelble, F. Nirsberger, E. Schulze, T. Rucks, J. Hoffmann, and E. Rahm, “Dataset report: Lid-ds 2021,” *Critical Information Infrastructures Security*, pp. 63–73, 2023.
- [26] Y. Xiao, Y. Jia, C. Liu, X. Cheng, J. Yu, and W. Lv, “Edge computing security: State of the art and challenges,” *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1608–1631, 2019.
- [27] H. Xu, Y. Wang, S. Jian, Q. Liao, Y. Wang, and G. Pang, “Calibrated one-class classification for unsupervised time series anomaly detection,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 11, pp. 5723–5736, 2024.
- [28] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long, “Timesnet: Temporal 2d-variation modeling for general time series analysis,” *arXiv preprint arXiv:2210.02186*, 2022.
- [29] H. Xu, G. Pang, Y. Wang, and Y. Wang, “Deep isolation forest for anomaly detection,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 12, pp. 12591–12604, 2023.
- [30] A. Bounsiar and M. G. Madden, “One-class support vector machines revisited,” in *2014 International Conference on Information Science & Applications (ICISA)*, pp. 1–4, IEEE, 2014.