International Journal of Research in Engineering and Science (IJRES)
ISSN (Online): 2320-9364, ISSN (Print): 2320-9356
www.ijres.org Volume 13 Issue 7 ǁ July 2025 ǁ PP. 01-12

Adaptive Transformer-Based Log Embeddings for Real-Time Host-Based Intrusion Detection in Edge Computing Environments

Olubunmi.Famosinpe
McCombs School of Business
The University of Texas at Austin
0lubunmi.famosinpe@cnoinc.com

Abstract
Host-based intrusion detection systems (HIDS) assume a pivotal role in securing edge computing environments characterised by high-volume, heterogeneous, and latency-sensitive data streams. This paper introduces an innovative framework that utilises adaptive transformer-based log embeddings to detect intrusions in real-time on resource-constrained edge devices. The system harnesses discrete Fourier transform (DFT) processing, linear injector adaptation, and attention-based learning to convert raw log messages into significant embeddings. These embeddings are utilised by an inception-based classifier, which is fine-tuned using an oracle mechanism to improve detection accuracy across various attack scenarios. When evaluated on the Edge-IIoT dataset, the proposed approach exhibits high precision and recall while preserving low detection latency, thereby surpassing conventional HIDS baselines. The framework is designed to be lightweight, modular, and resilient to evolving log formats, providing an effective solution for scalable and adaptive intrusion detection within modern edge computing infrastructures.

Keywords: *Host-based intrusion detection, Transformer-based log embeddings, Edge computing security, Real-time anomaly detection, Adaptive deep learning, Log analytics*

Date of Submission: 25-06-2025 Date of acceptance: 05-07-2025

I. Introduction

Edge computing enhances computational capabilities by moving data storage and analysis to nearby servers, leveraging cloud computing for seamless service integration. However, the increase in attacks on systems using PCAP highlights the need for real-time, tunable models that accurately handle extensive streaming data. This also considers the cost and feasibility of the deep learning model in resource-constrained environments. Although declarative methods and separation of concerns have long been part of cyber-development, programmers often overlook non-functional aspects, idealising a self-sufficient engine. A more granular approach is necessary to facilitate the creation of a DLL ecosystem with flexible, composable components. The Host-based Intrusion Detection System (HIDS) is crucial to a system's security policy, as it incorporates diverse information sources, such as log files from high-performance computing environments. Previous extraction and tokenisation methods have marginalised context in native profiling and log tampering. Understanding multiple contexts allows exploration within source logs using aligned grammar, creating a benchmark for pattern refinement. Generative models must be viewed as adaptable algorithms for consistent production (Martins et al., 2022). This framework features a three-stage mapping for refining interpretative and assembly patterns. In pattern assembly, anchor types are expanded through distribution, directed by interpreters and producers aligned with dataset contexts. Pattern interpretation involves weighing the divergence of the contextual distributions of cleaned logs. The existing models lack self-adjusting scopes and encoding for extensive applications. Edge computing, with lower standards than cloud data centres, offers rapid pre-processing and inference, resource sensitivity, regulatory reachability, and compatibility with few-shot resistances (Efe & Abacı, 2022). To improve generalisation with self-adjusting scopes, quality mark propagation, and approximate sampling, expand HIDS's minimal pattern exteriors in limited contexts. Randomised tree encoders, with depths that sample paths, enhance accuracy without retraining on abstract input growths by recasting original distributions onto P-Scope instances (Nallakaruppan et al., 2024).

II. Background and Related Work

In edge computing, host-based intrusion detection systems (HIDSs) are increasingly deployed to protect machines by detecting attacks early, limiting damage and facilitating recovery. HIDSs analyse logs from operating systems and applications, like Linux and Windows. Traditional rule-based log analysis relies on domain

knowledge and may overlook previously unseen attacks, leading to unresolved cases. In contrast, deep learning methods capture unknown attack patterns but require large datasets for training, which are often lacking in the small-scale user environments typical of edge computing. To address these gaps, we propose an adaptive transformer-based log embedding for efficient and accurate intrusion detection in edge environments. Our architecture includes a log ingestion module, a log embedding extractor, and a log classification module (Chen et al., 2024). Security log files are collected and preprocessed from diverse user machines. We utilise transformers to create fixed-size embeddings of the transformed logs, taking into account a masked log due to variations in edge computing environments. A fine-tuned classification model ensures fast, accurate log classification at event collection. This study is the first to introduce log embedding extractors for adaptive HIDS in edge computing, revealing substantial performance differences among transformer-based models. Notably, significant improvements do not always correlate with better classification results on large log datasets. Extensive evaluations show that the adaptable embedding-extractor design and pretraining are key contributors to effectiveness (Afnan et al., 2023).

2.1. Overview of Intrusion Detection Systems

A fundamental requirement to guarantee security is to protect the network resources and data of IoT devices (Spadaccino & Cuomo, 2020). As a consequence, a growing interest in Intrusion Detection Systems (IDSs) has emerged to identify and prevent data theft and/or loss resulting from unauthorised access. An IDS is a software or hardware-based subsystem that identifies the abnormal use of an information system by analysing data flow. Several proposed techniques can be classified as either Host-based or Network-based solutions, based on the location of data gathering and processing. Host-based systems analyse log events from local operating systems, network services, and applications. More sophisticated host-based systems often duplicate input and maintain application-level information, while filtering out irrelevant data to produce a shorter, more heuristic output. Nonetheless, some systems analyse data exported from network-level packet sniffers. On the other hand, Network-based systems sniff and analyse packets on the network, potentially spotting malicious traffic between different network nodes (Alotaibi & Mishra, 2024). Although general-purpose, hybrid IDSs exist, many systems proposed in the literature are office or organisation-oriented, often built upon proprietary hardware and/or closed-source software. This means they cannot be applied in environments that require transparency, such as offices, homes, or public venues. Hence, this work proposes an IDS architecture that focuses on edge computing environment designs, where all computation is carried out on inexpensive, low-resource, lightweight PCs running FreeBSD, Linux, or Windows. In this context, IDSs deployed on testbeds gather host-level data through one or more agents that write events to a log file, which is shared with a separate analysis engine. With this, target information systems are decoupled from detection and data collection, making detections and data exports safer and easier. (Wang et al., 2023)

2.2. Techniques for Log Analysis in Edge Computing Architectures

Edge Computing (EC) enhances data processing by bringing the cloud closer to users through Edge Devices, empowering Access Domains (EDEADs) (Tian et al., 2019). Edge Computing Architecture (ECA) manages data source mapping and interactions, highlighting the integrity and confidentiality risks associated with these processes. Many approaches, however, introduce unexpected risks (Arshad et al., 2019). The Edge Computing Network (ECN) outlines EDEAD interactions, integrating Sensor-Empowering Edge Devices (SEDEs), Digital Collecting and Conservative Sharing Edge Devices (DCSSEDs), and Learning-Infering Edge Devices (LIEDs) to address semantic complexities. Proper processor balancing is crucial for data preprocessing tasks in edge computing (Duan et al., 2023). Edge Process Distribution (EPD) analyses architecture hierarchically, facilitating task allocation using EPN. EPD retrieves ECA and describes EC data relationships before allocation. Tasks triggering data sources create logical relationships, with interactions at the same EDEAD level described using Hybridity Constraint Satisfaction Proximity (HCSP) (Bing et al., 2023). Host-based intrusion detection systems (HIDS) utilise operating system (OS) audit trails to track security events. Traditional rule-based HIDS, relying on pattern matching, have limited capabilities. Enhanced logging and machine learning improve generalisation with labelled data. Earlier deep reconstruction models faced challenges with log timestamps and clustering overhead. A supervised feature extraction model struggled with an iterative, parser-dependent log format (Efe & Abacı, 2022). A self-supervised proximal future prediction learns embeddings from unstructured logs, reconciling various log encodings. A modular deep hierarchy improves adaptability through attention models. Pseudolabeling in semi-supervised log classification ensures the convergence of the log embedding model. A proposed pipeline integrates state-of-the-art pre-trained models (Lee et al., 2021). The framework's effectiveness was validated using popular datasets, with a focus on practicality, scalability, and compatibility. A memory-efficient transformer optimises cache policy for high-memory throughput, while foreground pruning enhances processing speed on longer logs. The discussed compression methods preserve log utility better than textual summarisation, suggesting future research directions (Mandal & Vipparthi, 2021).

2.4. Transformer Models in Machine Learning

Transformers, relying on Attention Mechanisms, have surpassed Recurrent Neural Networks (RNNs) in natural language processing (NLP) applications. Although RNNS can incorporate attention, standalone transformers significantly outperform them. Recent studies on log embedding using transformers include (Afnan et al., 2023), which introduced a dynamic transformer-based approach for processing high-speed system logs into event sequences. In the log embedding and alert prediction submodels, impact vectors from previously processed logs are combined with one-hot embeddings of new logs to capture temporal dependencies. The transformer model's core components are self-attention and multi-head attention mechanisms, which assess the relative importance of log message tokens at each time step, focusing on relevant tokens while filtering out the irrelevant ones. This focus is crucial for effective event detection. The automatic learning of attention weights from training data helps mitigate previous limitations related to domain-specific rules (Patwardhan et al., 2023). Additionally, another study applied self-attention models for encoding event/access logs as a supervised sequence-to-sequence task. Thanks to advancements in NLP, transformer models have been effectively utilised for log embedding. (Li & Fu, 2023) Fine-tuned pre-trained transformer models on CAN data to extract embeddings, exploring both generic pre-trained and auto-regressive models. This study also examines the extraction of embeddings for individual tokens, treating input CAN logs as token sequences, which allows for vector embeddings that capture contextual meanings. Consequently, generic pre-trained log models are used to encode individual tokens in CAN logs through their text-string formats.

III. Methodology

The number of internet-connected systems increases daily, posing a threat to system and data security. Edge computing, a subset of cloud computing, brings computation and storage closer to edge devices. With more interconnected devices, it becomes increasingly vulnerable to cyberattacks due to the increased number of access points. Currently, research lacks a real-time attack detection layer for monitoring insider attacks. Attackers can access hosts undetected, leading to the compromise of sensitive information. A Host-Based Intrusion Detection System (HIDS) employs detection components on the monitored host to identify possible intrusions, utilising classification techniques that include specification-based, stateful protocol analysis, knowledge-based, and anomaly-based intrusion detection. This paper proposes a new HIDS for edge computing using adaptive transformer-based text log embeddings. The method employs a convolution operation to extract contextual information from logs for log text embedding input sequences. A log embedding classifier is separately trained for each service to account for the varying log messages across services and time frames. Sparse aggregation of log embeddings is utilised for attack detection. The proposed model outperforms multiple baselines, achieving an area under the curve of over 6% with real-world datasets. Experiments on resource-constrained edge devices demonstrate average detection latency below 60 ms, enabling real-time inference. A new edge computing-based server-client architecture uses edge devices as a defence against security threats by monitoring and identifying anomalies in server and user activities with low latency. An innovative model utilises transformer-based pre-trained log embeddings for real-time and accurate attack detection. Only a small subset of log samples is used for training, conserving memory and computing resources, for each service class. A decoder-only transformer architecture processes parallel log messages while preserving sequence. Enhancing log message severity levels as a new input feature enables adaptive detection of attacks. A simple convolutional block core design yields a non-trainable, lightweight model to extract contextual information from raw logs. Lastly, the embedding model's output is aggregated sequentially to address differences in log messages across services and timeframes, utilising aggregate log embeddings for attack detection with an attention-based classifier.

3.1. Data Collection and Preprocessing

The dataset is collected to reflect a series of DoS attacks on a Random Forest-based network intrusion detection system, providing a simulated environment for port 80 on a router. This dataset consists of normal network behaviours and network attacks, as well as server and application logs. Each log entry includes a timestamp, a host name, a process name, a process ID (PID), a log level, and log content. The dataset provides general information on security events, which are collected from logs generated by the web server and host machines, including web requests, web accesses, and host activity logs. The access logs are filtered by IP address, resource, and time, comprising 20 types of regular log entries. In addition to standard labels, attack logs of 10 different types are included, including web injection attacks, webclient attack logs, card theft attack logs, crawler scan attack logs, remote procedure call attack logs, SSH brute-force attack logs, and other OS-level attacks. To preprocess the logs of diverse structures and formats, log entries in the dataset are converted to JSON format to standardise the structure. With the help of the regular expression module in Python, the conversion is completed by matching each type of log to the components defined. Moreover, timestamps, which were initially stored in the ‘log_time’ key-value pair, were parsed into a datetime format and then translated into seconds relative to the first timestamp stored in the ‘first_time’ variable. Afterwards, all values in the host name key, i.e., log host, were

corrected by resolving to IPs and standardising IPv4 addresses, which were initially in CIDR format and composed of '/' and '.' instead of traditional notation. As for the logs, which were already in JSON format, they still require basic filtering by removing incomplete logs before each preprocessing step. Additionally, filenames with the '.txt' extension must be converted to '.json' for consistency with the logs. Likewise, times were resolved to seconds in the same way. After preprocessing, information security event logs of three types from two sources were obtained and saved, serving as the input for training the log model.

3.2. Log Embedding Techniques

Log entries from various collector programs greatly differ in format. Even similar programs still produce diverse formats. Approaches that use grid or word-level tokenisation fail to capture semantic similarities, thereby diminishing the framework's effectiveness. This section introduces the first part of a two-part machine learning (ML)- based framework for classifying system logs, proposing Transformer-based log embeddings. Specifically, it outlines (i) preprocessing methods that convert log entries into a unique representation format with colour standardisation, and (ii) a Transformer-based architecture for high-quality embeddings with minimal information loss. The framework is evaluated on three varied open-source datasets to assess the generalisation capabilities of the proposed representations. An analysis of optimal hyperparameter choices for the embedding model is also provided (Guo et al., 2021). A general summary highlights three key aspects: a representation format that bridges semantic gaps and enhances log analysis robustness; the power of pre-trained encoders, such as Transformers, for effective embeddings; and the integration of a special token in the log event sequence to encode ordinal information. A unique training process is established for this hierarchical architecture. Ablation studies are conducted to evaluate the contribution of each technique. Logs are tokenised sequential data fed into a Transformer for embedding extraction, with various techniques enhancing performance, robustness, and interpretability (Le & Zhang, 2024). A Transformer-based APT detection framework utilising provenance graphs through a host-based approach is also proposed, focusing on learning log representations from extensive log files and detecting APT attack patterns. Log entries vary significantly in format and style of representation.

3.3. Transformer Architecture Adaptation

Log embedding methods based on transformer architectures have gained significant popularity due to their attention-based mechanisms, scalability, and high flexibility (Afnan et al., 2023). The transformer's extraordinary performance and transferability have prompted researchers to integrate its architecture into cybersecurity to address novel challenges (Li & Fu, 2023). The architecture of the transformer, which is primarily comprised of an encoder and a decoder, is initially crafted for translator-based systems. The neural network architecture contains a multi-head self-attention structure. Each attention head is presented in Equation (1). The three core components of the method are an encoder structure, which generates contextualised character representations; a multi-head self-attention (MHSA) mechanism, which captures intra-sequence dependencies with the self-attention module; and a position-wise feedforward network (FFN) consisting of two linear transformations with a ReLU activation in between (Huang et al., 2023). Given that the task is to classify the sequence of log events, only the encoder part of the entire transformer architecture is required for classifying system logs. Hence, the overall architecture is a transformer-based encoder (TBE) consisting of the embedding layer, the transformer encoder layer, and the classifier layer. Each log is separated from the entire sequence by the ENCODER and APPEND positions. The embedding layer, comprising a multi-head self-attention (MHSA) mechanism, generates a linguistic representation of the log events. The transformer encoder, comprising a stack of multi-head self-attention layers, enhances the context information of the embedding and establishes associations among the logs of the input sequence. The transformer encoder outputs the contextualised log representation, and generalised knowledge is contained in the formed graph. The classifier converts 1024-dimensional graph representations into a binary outcome, corresponding to a prediction of benign or malicious intent. (Nassiri & Akhloufi, 2023)

3.4. Real-Time Processing Framework

Processing log data to recognise attacks is crucial for rapid, real-time reactions. The computation expense depends on how efficiently models generate log representations. Large input logs increase processing costs and complexity. To manage this, logs are condensed by removing minor but essential details, creating interpretable event traces and log keys. Single log traces for detection use less context than entire logs, making them more efficient for mapping-based models. Post-processing accumulates overhead from narrowed node traces across computers, even those without valuable links. Passed-through logs can further condense data, yet require interpretation through natural language-based attacks. After conversion, steps are summarised by vectors from event traces, producing ten consecutive Trigrams from three-length sub-structures. These embedded vectors are aggregated via a dot-product attentional mechanism with residual links and layer normalisation. Real-time stages incorporate yearly or quarterly features to sum attention scores from ten or more Trigrams on an ordered basis.

Outputs from attention layers serve as inputs for the encoder stack, long past sequential steps. For the encoder stack, ordered Embedding Net outputs are computed with fixed-length traces of 31 characters, defined as stext. Each character is separated by 23 distinguishing values and indexed as one-hot embedded vectors of dimension 67. Overlooked indices become token representations for Sub-Word Layer Normalisation. A dropout distribution, disregarding 5% of characters, is applied after both operations, with weight parameters shared throughout the encoding process. After this, nine input logs are linked to a detection score indicating whether they are benign or malicious, with the representation dimension size adjusted to 512 via four-fold of the token sequential average on concatenated scores.

IV. Implementation

The implementation process of the proposed approaches involves embedding generation and one-class classification. The architecture is implemented using an API framework. The hyperparameters include an optimiser with a learning rate of 0.00005, a maximum of 30 pre-training epochs with a learning rate scheduler, and 30 epochs for one-class classification at a learning rate of 0.0001. K-fold cross-validation is applied with K = 2 for both embedding generation and one-class classification. The model weights for the second fold are saved for evaluation on the test dataset. To generate embeddings with the pre-trained model for one-class classification, the input representation consists of (token ID, event frequency) pairs from training data event labels based on the parsing algorithm. This representation is passed to the pre-trained model to produce log embeddings. In k-fold cross-validation, the model weights generated by the second fold are used to create embeddings for the test dataset. The generated embeddings for the training data train the OCC models. For both fits and predictions, the embeddings are reduced to match the corresponding OCC model's fitting step. During k-fold cross-validation, a counter tracks the number of folds that predict the classifier's outcomes. Therefore, the embedding dimension and bucket number for unused folds are set to null before generating predicted labels, and relevant files are produced. As only standard training samples are provided for one-class SVM and Isolation Forests, the embeddings and predicted labels for the attack class are removed before saving the file. Scripts for generating and saving predicted labels are executed after all proposed approaches are completed. For evaluation, ground truth labels and predictions of log entries are provided in two files. The first file contains columns titled “ID” and “target” aligned with predictions, where the ID column has integer log message IDS and the target column has “1” for attack logs and “-1” otherwise. The second file contains predictions in one column titled “target, " with “1” for attack logs and “-1” otherwise. The evaluation approach includes four metrics: True Positive Rate, False Positive Rate, F1 score, and Model Selection.

4.1. System Design

In recent years, concerns have risen about edge-based systems due to their importance in autonomous driving, VR gaming, and Smart City IoT. A host-based intrusion detection system (HIDS) is crucial for detecting cyberattacks early, including privilege escalation, credential theft, and lateral movement. Attackers often exploit system logs for reconnaissance, which contain unique features like host, executor, timestamp, and PPID. For HIDS, a sequence of logs is processed for detection, and Transformers leverage positional encodings to understand sequences. Log socialising systems are increasingly necessary due to the high volume of log storage processes, including cloud database scraping and real-time log analysis, as global attacks frequently target multiple systems. These systems require preprocessed logs, which are not platform-independent. Real-time HIDS needs performance optimisation for fast inference with large log sequences, updates for parallel computation, and improved memory allocation. While there has been work on optimising training models, comprehensive efforts to enhance real-time performance are lacking. ElogDex addresses this by extracting informative log sequences, removing redundant information, and maintaining valid data for future reference. It concurrently compresses log memory using logoids to offset time and data from log extraction, with a publicly shared reflective key for cross-platform analysis (Joraviya et al., 2024). The main challenges in log analysis are system efficiency and effectiveness. Overall memory usage impacts data handling and model deployment on low-memory devices. Most logging solutions fail to ensure a feasible, generic design across platforms that handles log diversity and do not address the need for large-scale datasets consisting of informative, long log sequences for real-time and cutting-edge Host-Based Intrusion Detection Systems (HIDS).

4.2. Implementation in Edge Environments and Integration with Security Tools

To deploy Log-HAT on an edge device, the minimum hardware and software configurations must be met. An edge device PC equipped with an Intel Core i5-5200U CPU, 8GB of RAM, and Windows 10, along with Python 3.10, is used. Log-HAT, developed with PyTorch, needs to be installed on the device. Raw system logs arrive in real-time; the first step is parsing them into predefined slots: timestamp, host, service, criticality, and message. Like training, log slots pass through encoding layers, generating log embeddings after the last layer. The final Log-HAT inference input is log embeddings concatenated with masks. Real-time inference uses Log-HAT

to flag abnormal logs as anomalous or mark them as usual (Afnan et al., 2023). Log-HAT focuses on internal HIDS detection, utilising pilfered Linux logs categorised into 11 types: authentication, sabotage, privilege escalation, and configuration policy changes. Attack detection occurs on a Linux honeypot running in a virtual machine (VM) with 1 CPU, 2 GB of RAM, and the Ubuntu operating system. Generated attack logs included login failures and unauthorised changes, captured on another machine with forwarding settings. Logs were pre-processed by removing extra fields, labelling, and pre-training validation. Raw logs were encoded for Log-HAT Training-A input. Finally, Log-HAT Test-B was run on the designed model with an invisible test dataset (S. Hameed et al., 2021). A key objective of this research was to design a log monitoring solution compatible with existing IT tools. The solution consists of two modules: log processing and monitoring. The former has been previously discussed, while the latter relates to the ELK stack. Log forwarding sends captured files to the application, eliminating the need for local storage. Logs are usually stored in text file formats, often with low signal-to-noise ratios. To enhance log details, two essential preprocessing steps were taken: filtering by key phrases and defining parsing functions to add log lines. The configuration file is customised for these processes. At the application side, two configurable windows visualise input and monitored logs in real-time. Log filtering is set to avoid screen overload, focusing on terms like 'failed login', 'SSHD', or 'successful login', which highlight brute-force attempts or abnormal access. SSHD is default-enabled in Linux, making these phrases widely monitored. Log lines containing these phrases automatically appear in the second monitoring window (Martini, 2023). Log highlighting differentiates parts of monitored logs by adding parsing functions. These functions can utilise regular expressions to match user-defined patterns, like highlighting IP addresses. Certain characters may be emboldened or colored red. Through filtering and parsing, clear information helps analysts save time during log monitoring.

V. Evaluation

Real-time Host Anomaly Detection is formulated as a sequence-based log representation learning problem. Log-based host anomaly detection is achieved through log representation learning and a downstream classification task. The task of log representation learning is considered a self-supervised sequence prediction task. Given an input sequence of tokenised logs, the objective is to predict the previous log representation based on future logs. To generate a representative log representation from logs, the input sequence is pre-trained separately on a set of log event sequences constructed at the event and token levels based on surrounding context. The self-supervised training procedure employs a hybrid masking strategy on both event and token-level logs, where the causal masking approach is applied to conceal the visibility of preceding logs after they are encoded. The training objective of log-based host anomaly detection involves a pre-training phase, a fine-tuning phase, and a downstream classification phase (Li et al., 2025). Both the input tokenised logs and the separated surroundings are then fed into the proposed Transformer-based framework to generate adaptive log event embeddings. The proposed pre-training phase, which leverages fine-grained sequence prediction tasks including masked log sequence prediction and masked log odds sequence prediction, enables the model to learn informative log embeddings in an unsupervised manner for diverse downstream classification tasks.

5.1. Performance Metrics

This study evaluates the performance of the proposed adaptive transformer-based log embeddings framework in two steps: (1) using the lightweight hybrid model for encoding and embedding log sequences from raw regular and OSDE logs and applying traditional ML and ANN methods to the resulting word embeddings for attack detection, and (2) employing the proposed adaptive transformer-based hybrid model for end-to-end encoding and attack detection of raw log sequences from high-volume input streams.

The following standard metrics are used to evaluate the proposed and baseline models in terms of their ability to detect attacks. Two metrics, the precision ratio P and the detection rate R, are used to evaluate the performance of intrusion detection. They are calculated as follows:

(1) $P=TP/(TP+FP)$.

(2) $R=TP/(TP+FN)$.

Where:

TP (True Positives): Detected attacks that occurred.

FP (False Positives): Detected attacks that did not occur.

FN (False Negatives): Attacks that did occur but were not detected.

TN (True Negatives): Non-attacks that did not occur.

The harmonic mean of the precision ratio P and the detection rate R is defined as the F1-score:

(3) $F1-score=2\times P\times R/(P+R)$.

Attack detection performance is evaluated by five metrics: True Negative (TN), True Positive (TP), False Negative (FN), and False Positive (FP) rates, which measure correct, missed, and incorrectly labelled predictions. The True Negative Rate (TNR) measures the ratio of correctly labelled benign logs, while the False Positive Rate (FPR)

measures incorrectly labelled benign logs. TP indicates the percentage of detected attack logs, FN measures the number of missed detections, and the False Discovery Rate (FDR) indicates false positives among optimistic predictions. Reliability is assessed through FDR and TNR; higher TNR or lower FDR indicates a more reliable model. Besides accuracy and detection performance, we evaluate time overhead, defined as the duration for making attack detection predictions. In the first experimental step, text-based log embedding approaches are excluded due to slow keyword extraction, and the recent model cannot be evaluated without pre-trained checkpoints. All models are implemented in PyTorch and trained on a computer equipped with an Intel Core i9-9900KF processor, 32 GB of RAM, and an NVIDIA GeForce RTX 3090 GPU, with all benchmarked results obtained under the same conditions.

5.2. Experimental Setup

This section describes the experimental setup, including the datasets and parameters. The experiments utilise the Edge-IIoT dataset, generating log messages for both attack scenarios and typical scenarios. Attack and regular messages are classified into clusters using DBSCAN, with normal messages labelled as benign (0) and attacks as positive classes. Table 4 presents the sample types and their corresponding percentages in each class. Additionally, APC groups SOTA log embedders that extract embeddings from the log samples. The log embeddings are uniformly processed through a discrete Fourier transform (DFT) method, producing magnitude and phase outputs. These embeddings are then made adaptive in a linear fashion using injector mechanisms and are passed through attention layers before output classification. The inception model is trained on a labelled log message subset to tune hyperparameters and improve attack identification, utilising an Oracle mechanism to select samples with different attacks (Amine Ferrag et al., 2022).

5.3. Results and Discussion

The real-time host-based intrusion detection model (RT-HIDS) is developed to automatically classify log messages as usual or suspicious, adapting to online settings. Considering the complex and diverse nature of real-world log events and their evolving characteristics, log embeddings are derived from deep learning log encoders to capture comprehensive information. Existing log encoders usually utilise recurrent modules to transform raw logs into dense embeddings. However, the inherent auto-regressive masking scheme neglects some token context during computation. To enhance representation ability, a hierarchical transformer-based log embedding decoder (HT-LED), a novel universal log encoder that can examine both short-term and long-term relationships, is proposed. This helps extract historical information for a better understanding of contexts. Using the transformer architecture, multi-level context queries are designed to attend to the hierarchical token representations learned from encoding logs. The self-attention pooled one-hot mask embeddings thereof indicate the awareness level of attention context. Adapting to edge environments for efficient model deployment, a model pruning tuning pipeline is proposed to enhance inference performance with minimal accuracy drop. Acknowledging that the model is highly desirable for classifying logs in real-world environments with dynamic log formats, a unified log format generator based on abstract syntax tree (AST) is proposed to preprocess a variety of log types into a common standard. A framework for real-time detection based on online model adaptation, along with a local auto-log formatter utilising a customised language model pre-trained on various log data, is also proposed to achieve log moderation in edge environments with limited resources. The proposed log embedding ensemble would leverage log encoders and share weight information. To accommodate dynamic changes, it can divide and dynamically remove log encoders, utilising knowledge transfer and cross-peer weight regulation mechanisms. The comprehensive experiments conducted on the public and self-collected datasets have demonstrated the effectiveness of all proposed methods. As one of the model ensembles, DeepLog—a unified log encoder matrix composed of various log transformer model instances—is implemented. It shows promising results than existing log encoders on both accuracy and efficiency aspects (Afnan et al., 2023).

VI. Case Studies

Log files are crucial for understanding incidents or attacks on computing systems. They contain records of system activities, including performance, user requests, access control changes, and security violations. Analysing logs is essential for detecting anomalous activities that breach a system's security policy. Host-based intrusion detection systems analyse logs to identify attacks and respond by logging events, sending alerts, or aborting sessions. However, log-centric data presents challenges for detection due to its high dimensionality and class imbalance, resulting in many benign logs that degrade performance. Despite advanced log mining techniques, extracting useful features from flexible log formats remains a challenging task, making accurate log classification difficult. Cellular operators widely adopt intelligent network equipment, leaving it vulnerable to attacks such as denial-of-service and eavesdropping. A host-based intrusion detection system can fuse multi-dimensional logs in edge computing environments, focusing on managing continuous log streams for quick detection. A patch-based neural network is proposed to model logs and structured data in real-time. An

incremental training mechanism based on transfer learning enables the model to learn from new logs as needed. Detection performance is evaluated on real-world cellular logs in a central monitoring platform, showing high accuracy, speed, and robustness to noise and unexpected log classes.

6.1. Application in Smart Homes

With a significant increase in Internet of Things (IoT) devices in smart homes globally, attention to improving their security is critical. Built-in security is often inadequate due to manufacturers' lack of experience. Millions of smart homes have devices with serious security flaws (Rahman Shahid et al., 2019). Default passwords are often left unchanged after installation, and most vendors fail to patch their product software for newly discovered vulnerabilities. Many devices remain vulnerable to remote access attacks for years, potentially exposing sources for DDoS attacks to IoT malware. Even newer device versions with enhanced security can still be vulnerable to new attacks. A robust and configurable Intrusion Detection System (IDS) that adapts to evolving threats is essential. The solution must detect and remediate, such as blocking malicious communications. While Medium Access Control (MAC) Address Filtering is standard, determined attackers can spoof MAC addresses. Anomaly detection methods monitor expected device behaviour but may produce false positives. The proposed two-layer detection system identifies and halts anomalous communications between IoT devices and the Internet through the home router. Anomaly detection should be implemented on the device, while detection and remediation occur at the router. This method enables the effective detection of new attacks. IoT devices perform specific tasks, making their behaviour predictable when they function correctly. With proper thresholds, anomaly detection is advantageous here, as it is trained on legitimate device communications, since most attacks result in anomalous behaviour. Adversaries often struggle to simulate legitimate behaviour.

6.2. Applications in Industrial IoT and Healthcare Systems

With the rise of IoT devices, attention has shifted to advanced threat modelling for security in large-scale IoT applications (Arshad et al., 2019). Previously, the focus was on new attack types for IoT and IIoT devices. As IIoT systems proliferate, supporting intelligent transportation and oil exploration, understanding evolving detection challenges becomes essential. We must reassess the threat landscape related to IIoT applications, with a focus on previously unexplored attack vectors. Updating attack parameters in our model is crucial for addressing threat evolution. The dynamic nature of edge node deployments introduces numerous variables into attack models, even as the overall architecture remains constant. Changes occur in edge devices, necessitating a qualitative attack modelling framework using DREAD metric aggregation (Savic et al., 2021). We revise previous assumptions for the edge computing context, particularly in the white-box model, where adversaries have limited knowledge of the architecture. New detection challenges are reframed based on specific deployment locations, attack types, and modelling metrics for assessing attack effectiveness. The Internet and mobile technologies have made healthcare a significant target for cybercrime. Although electronic health record (EHR) systems and Internet of Medical Things (IoMT) devices enhance healthcare services, they also raise security concerns about the protection of sensitive medical information. Personal health records are particularly vulnerable, often targeted by DoS, data injection, and replay attacks. Between 2017 and 2019, cyberattacks on healthcare systems increased by 350%, and attempts to access health data rose by 900% in 2019. Despite the numerous intrusion detection systems (IDSs) available for healthcare, evolving hacking techniques and the use of diverse IP addresses pose significant challenges (Akshay Kumaar et al., 2022). The effectiveness of IDS is crucial for ensuring patient safety and privacy in the Internet of Things (IoT). Balancing low complexity with high robustness restricts analytical improvements, and learning- based methods often face challenges with false positives. Edge-based IDS using neural models can capitalise on hardware capabilities. A boundary for hardware capability was established to enable dynamic cross-checking between model metrics and security enhancements, thereby protecting models against more sophisticated adversaries (S. Hameed et al., 2021). An Enhanced Intrusion Detection System (EIIDS) was created from these models, outperforming other solutions in healthcare IoT. To mitigate false positives and robustness issues in learning-based methods, multi-layered verification systems have been introduced to dynamically adapt to device capabilities.

VII. Challenges and Limitations

Real-time host-based intrusion detection is crucial in edge computing environments. Traditional methods utilise rules and heuristics provided by security analysts, but they often fail to identify new attack patterns. Machine-learning intrusion detection offers promise for detecting unknown attacks; however, it often misses semantic information and struggles with high-dimensional data in raw logs. Recently, transformer-based language models have gained traction in detecting attacks. However, these methods create unified embeddings for either structured or textual logs, limiting their application. Additionally, manually tuning hyperparameters is time-consuming and requires significant domain knowledge. A novel approach, LogEmb, utilises a transformer-based log embedding framework with an adaptive architecture to process structured and unstructured logs from various

systems. The approach is evaluated using threat intelligence datasets that contain complex cyber-attack logs from Linux software services. Extensive experiments show that LogEmb outperforms state-of-the-art methods that only handle either structured or unstructured logs, achieving sub-second real-time processing speeds. Security remains vital in networking and information systems, as attacks can compromise the integrity of users, data, and systems, potentially leading to significant consequences. Log-based security monitoring is essential for defence. Reliable intrusion detection, informed by past experiences and forensic evidence, can effectively identify unauthorised access attempts. This Disappearing Computation design integrates host-based approaches, observation-only models, deep learning, and predictive log processing to enhance real-time security operations. Specifically, a hybrid model combining transformers and recurrent neural networks is proposed to enrich a data-centric approach with automation and prediction capabilities. It transforms time-stamped log messages into fixed-sized embeddings for use in standard stance inference models. Experiments with real-world datasets demonstrate that the proposed designs yield more compact models and improved predictive efficiency over common requests. At the same time, transformer architectures excel at modelling time dependencies among log sequences (Afnan et al., 2023).

7.1. Scalability Issues

The proposed method is formulated as a promising solution for log embedding and presented to address the log embedding problem in real-time LIDs. Evaluated on the Linux Audit Dataset, the proposed method demonstrates robust performance across various settings. However, it is not without limitations. It remains a significant obstacle to scaling to analyse the real-time high-volume logs. The reasons include the fact that there are many fine-tuned hyperparameters for Transformer models, including optimisation hyperparameters, encoder depth, width, and number of heads, as well as batch size and token length. It cannot scale it to work with an enormous number of hyperparameters. If multiple Transformer models are trained simultaneously in a distributed manner, for inference, two encoders from different models need to analyse the same incoming token, and the module storing +1 million parameters has to be utilised per second. It is infeasible. Training a multi-head multi-model transformer directly might be inefficient. To address these issues, the following directions are proposed: ° Efficient improvement on token-stream embeddings: While the log-inserted time is usually regarded as of secondary significance, it is crucial in many LID tasks. Precisely modelling it would be beneficial. Meanwhile, the text representations in the other logs would also need to be effectively captured (Afnan et al., 2023). ° Bed constituent architecture (multi-encoder based): A multi-head transformer-based decision-level fused LID is attractive. The most used LID is the modelling of ENCODED-Timestamp. Each candidate model can be placed as an individual branch to process the same token stream. Decision-level fusion would be implemented on their outputs aggressively. Exchanging information between these encoders is also a good topic. The same token would be tokenised/viewed and incorporated into two different encoders, providing a good opportunity to learn the non-trivial crosstalk representations between discriminative features separately captured by the two encoders.

7.2. Data Privacy Concerns

The proliferation of smart devices and their connected sensors has made it easier than ever before to collect data on human activity in the physical world. Devices such as computer webcams or smartphones can integrate audio/video-enabled sensors to glean insight into individuals' behaviour. Organisations in industries as diverse as finance, retail, insurance, and healthcare derive valuable business insights through the analysis of speech and video data. However, as data processing shifts from general-purpose cloud servers to local edge devices, concerns arise regarding the collection and storage of potentially sensitive, paralinguistic information (Aloufi et al., 2020). In this work, a strategy is presented to preserve individuals' paralinguistic privacy, relying on unsupervised deep learning (DR) for speech recognition systems that leverage low-dimensional representations. The goal is to demonstrate that by leveraging DR and quantisation models, trustworthy and privacy-aware edge-based processing is feasible for a wide range of intelligent systems, including speech-activity detection, ASR, speaker verification, emotion recognition, and more. A novel approach is also presented that exploits the sensitivity of ASR models' classification head to input representations, converting the mostly resilient DR vectors to indistinguishable vectors for the ASR model. Real-time voice communications often include non-linguistic information, such as speaker characteristics, emotion, and stress, which may be private or confidential in nature. Hence, before sending raw audio or speech data, it is necessary to detect sensitive information in real-time to protect users' privacy. Inspired by the success of recent works in privacy protection at the edge, a privacy-preserving data routing method is proposed, which passes the mixture of original audio data onto the edge device to reject sensitive data. However, this may restrict the use of unwanted speech types in some practical applications; for example, filtering out all speech-containing audio data may pose challenges in protecting vocal passwords or bank card releases.

7.3. Model Interpretability

Trustworthy inference plays a critical role in the practical deployment of deep learning models for intrusion detection. LogShield adopts a straightforward yet practical approach, providing comprehensive insight into the training process and enhancing model interpretability. Two distinct, reliable inferences are ensured: tracking resource usage to build an audit trail for tampering detection and comparing host computing resources on standard metrics to identify non-supported consumptions. It can be seen that LogShield has a high activation in frames 242 and 243, indicating the model's prediction of suspiciousness in those two frames. For the corresponding time point, the parent of the suspicious process is logged and displayed in the bottom input of the visualisation. Furthermore, the mechanisms used to search for these inputs are highlighted in the corresponding upper half of the visual. From the above figures, it can be inferred that the intrusion detection system detects the emergence of the process recursively. To compare the accumulated resource usage, LogShield inputs resource usage data of a host and logs the inference results when the computation resource usage is recorded every second. When suspicion peaks, it alerts us to two uses involving parent and child processes. To investigate how the detected processes lead to an attack, the redundant computations on the CPU are tracked, generating an audit trail that records the emergence and behaviours of both.

VIII. Enhancing Log Analysis and Integrating AI Technology: A Study on Federated Learning

Log Analysis categorises messages to identify incidents, using Log-MLM and Holistic Log Analysis to predict log lines. Node entities represent functionalities, and edge entities illustrate interfaces, clarifying user behaviour during server requests. Analysed edge logs include complete log IDs or hash values from various digests. After creating the provenance graph, traces can be generated in two ways for Wrapper inputs. For Holistic Log Analysis, initial logs consist of ICULog class sequences, while log analysis uses API call logs with related function details. Invoked APIs cannot be logged directly, but they can be inferred from hash values. Log length impacts performance; tasks such as Log Classification and Log Clustering reduce candidate logs, despite challenges in Log-MLM. Longer log segments or repeated analyses may reveal incomplete evidence. As log volumes rise, numerous edge cases emerge from extended log windows or analyses. Enhancing initial logs with incident evidence can lead to redundancy, and Log Code Generation may cause arbitrary repetitions, necessitating management in Dockerized environments. AI drives advancements, reshaping life. The evolution of AI has prompted research into AI-enabled IoT networks, generating vast amounts of data that provide valuable insights into IoT operations. This data-centric AI field faces challenges, including security, privacy, resource management, robustness, and analytics. Recent advancements support the use of distributed processing to address learning challenges from device networks (Aneja et al., 2021). Predictions showed connected devices would surpass the global population by 2020. Basic security measures are often ineffective for IoT, as devices frequently operate over insecure radio waves. Despite the potential of a data-centric approach, it remains underexplored; reliance on collected data can lead to a loss of historical insight, affecting accuracy. Traditional consensus methods work in centralised learning but require data migration, raising security concerns. There is growing interest in distributed learning processes that utilise local device logs within edge or fog architectures. However, conventional methods rely on long-term states that may not adapt well to minor changes in the time series.

The abundance of IoT devices requires edge computing to mitigate latency and bandwidth bottlenecks. One inherent risk of edge computing is the sensitivity of data, which is often personal and reveals intimate information about users. To address this problem, federated learning is a proper and rapidly rising discipline. The process of ML training is heavily parallelised in FL, with initial parameters being sent to devices, trained using privacy-preserving operations, and updated parameters subsequently being sent back. In this manner, instead of sending the data with privacy issues to the cloud, only the updated parameters or weights are sent. As a result, the user information remains local. The NF and GAN concepts are used in (Belenguer et al., 2022) to identify cyberattacks while learning in a privacy-preserving manner. A federated learning system comprising three parties — namely, the server, an edge server, and clients — is proposed to accommodate a multi-tenant federation with multiple clients within the same organisation. Security concerns are discussed when implementing FL, including poisoning and model inversion attacks, with several countermeasures and defence strategies presented. State-of-the-art publications on FL for NIDS are analysed from a dataflow perspective, with possible industries and research directions further indicated.

With the prevalence of IoT devices, edge computing is becoming a standard architecture for meeting the demand for low-latency and bandwidth-efficient services. Security and safety concerns arise regarding malicious attacks on these inexpensive and controlled devices, as well as the potential unauthorised access to private information. Various kinds of anomalous behaviours are investigated for intrusion detection systems. However, traditional centralised data collection architectures often cannot work, requiring new distributed detection systems, such as federated learning (FL), as a solution. However, FL has not been extensively explored for edge HIIDS yet. In this work, the framework, challenges, and applications of federated learning (FL) for high-impact industrial

digital systems (HIIDS) in edge computing are introduced, along with a recap of prior works, a proposed framework, and its limitations.

IX. Conclusion

This paper presents an adaptive transformer-based engine for embedding hierarchical logs, enabling real-time security monitoring and host-based intrusion detection in cloud, edge, and fog computing environments. We estimate the spatiotemporal and spectral distribution of log data. Logs from geo-distributed hosts contain valuable information for cybersecurity and are generated as services perform operations, such as accessing files and executing transactions. The hierarchical nature of logs prompted the creation of a multi-scale transformer-based recurrent neural network that embeds each log in varying contextual window lengths. This utilises a visual attention mechanism that considers log hierarchies and contextual windows to preserve hierarchical semantics (Afnan et al., 2023). The Log-E engine captures changes in log characteristics, such as hierarchical access and selection of contextual window lengths, enabling efficient log embeddings in edge-cloud environments for detecting various APT attacks swiftly. This is achieved using the adaptive transformer-based Log-E engine, a multi-scale spatiotemporal attention architecture (Over-L), and advanced fine-tuning methods to ensure accurate detection (Parker et al., 2019). Attention is focused on embedding diverse log data hierarchically, extracting log embeddings on resource-constrained edge devices, and ensuring fault tolerance of plug-and-play devices. This work proposes stateful log data analysis against APTs, employing an unsupervised learning approach to analyse log generation patterns and build baseline models, aiming to aid in reports of zero-day attacks shared by security agencies.

References:

[1]. Martins, I., Resende, J. S., Sousa, P. R., Silva, S., Antunes, L., & Gama, J. (2022). Host-based IDS: A review and open issues of an anomaly detection system in IoT. Future Generation Computer Systems, 133, 95-113. github.io

[2]. Efe, A. & Abacı, N. (2022). Comparison of host-based intrusion detection systems and network-based intrusion detection systems. Celal Bayar University Journal of Science. dergipark.org.tr

[3]. Nallakaruppan, M. K., Somayaji, S. R. K., Fuladi, S., Benedetto, F., Ulaganathan, S. K., & Yenduri, G. (2024). Enhancing the security of host-based intrusion detection systems for the Internet of Things. IEEE Access, 12, 31788-31797. ieee.org

[4]. Chen, Z., Simsek, M., Kantarci, B., Bagheri, M., & Djukic, P. (2024). Machine learning-enabled hybrid intrusion detection system with host data transformation and an advanced two-stage classifier. Computer Networks. sciencedirect.com

[5]. Afnan, S., Sadia, M., Iqbal, S., & Iqbal, A. (2023). LogShield: A Transformer-based APT Detection System Leveraging Self-Attention. [PDF]

[6]. Spadaccino, P. & Cuomo, F. (2020). Intrusion Detection Systems for IoT: opportunities and challenges offered by Edge Computing and Machine Learning. [PDF]

[7]. Alotaibi, F. A., & Mishra, S. (2024). Cyber Security Intrusion Detection and Bot Data Collection using Deep Learning in the IoT. International Journal of Advanced Computer Science & Applications, 15(3). [HTML]

[8]. Wang, P., Lu, J., Wu, Y., & Zhang, J. (2023). "A bundle of serendipity": A crowdsourcing program for picture book information organisation oriented toward preschool children. Library & Information Science Research. [HTML]

[9]. Tian, Z., Shi, W., Wang, Y., Zhu, C., Du, X., Su, S., Sun, Y., & Guizani, N. (2019). Real Time Lateral Movement Detection based on Evidence Reasoning Network for Edge Computing Environment. [PDF]

[10]. Arshad, J., Ajmal Azad, M., Mahmoud Abdeltaif, M., & Salah, K. (2019). An intrusion detection framework for energy-constrained IoT devices. [PDF]

[11]. Duan, T., Zha, J., Lin, N., Wang, Z., Tan, C., & Zhou, Y. (2023). The Rise of Metal Halide Perovskite Memristors for Edge Computing. Device. sciencedirect.com

[12]. Bing, Z., Wang, X., Dong, Z., Dong, L., & He, T. (2023). A novel edge computing architecture for an intelligent coal mining system. Wireless Networks. [HTML]

[13]. Lee, J. D., Lei, Q., Saunshi, N., & Zhuo, J. (2021). Predicting what you already know helps: Provable self-supervised learning. Advances in Neural Information Processing Systems, 34, 309–323. neurips.cc

[14]. Mandal, M., & Vipparthi, S. K. (2021). An empirical review of deep learning frameworks for change detection: Model design, experimental frameworks, challenges and research needs. IEEE Transactions on Intelligent Transportation Systems, 23(7), 6101–6122. [PDF]

[15]. Patwardhan, N., Marrone, S., & Sansone, C. (2023). Transformers in the real world: A survey on NLP applications. Information. mdpi.com

[16]. Li, X. & Fu, H. (2023). SecureBERT and LLAMA 2 Empowered Control Area Network Intrusion Detection and Classification. [PDF]

[17]. Guo, H., Lin, X., Yang, J., Zhuang, Y., Bai, J., Zheng, T., ... & Li, Z. (2021). Translog: A unified transformer-based framework for log anomaly detection. arXiv preprint arXiv:2201.00016. [PDF]

[18]. Le, V. H. & Zhang, H. (2024). PreLog: A Pre-trained Model for Log Analytics. Proceedings of the ACM on Management of Data. researchgate.net

[19]. Huang, S., Liu, Y., Fung, C., Wang, H., Yang, H., & Luan, Z. (2023). Improving log-based anomaly detection by pre-training hierarchical transformers. IEEE Transactions on Computers, 72(9), 2656-2667. [HTML]

[20]. Nassiri, K. & Akhloufi, M. (2023). Transformer models are used for text-based question-answering systems. Applied Intelligence. [HTML]

[21]. Joraviya, N., Gohil, B. N., & Rao, U. P. (2024). DL-HIDS: deep learning-based host intrusion detection system using system calls-to-image for a containerised cloud environment. The Journal of Supercomputing. [HTML]

[22]. S. Hameed, S., Selamat, A., Abdul Latiff, L., A. Razak, S., Krejcar, O., Fujita, H., Nazir Ahmad Sharif, M., & Omatu, S. (2021). A Hybrid Lightweight System for Early Attack Detection in the IoMT Fog. ncbi.nlm.nih.gov

[23]. Martini, F. K. A. (2023). Hypercheck: Developing a Reminder and Data Logging System for Hypertension Patients. diva-portal.org

[24]. Li, J., Zhang, R., & Liu, J. (2025, April). MM-LogVec: System Log Anomaly Detection Method Based on Multimodal Representation Learning. In ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 1–5). IEEE. [HTML]
[25]. Amine Ferrag, M., Friha, O., Hamouda, D., Maglaras, L., & Janicke, H. (2022). Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications for Centralised and Federated Learning. [PDF]
[26]. Rahman Shahid, M., Blanc, G., Zhang, Z., & Debar, H. (2019). Anomalous Communications Detection in IoT Networks Using Sparse Autoencoders. [PDF]
[27]. Savic, M., Lukic, M., Danilovic, D., Bodroski, Z., Bajovic, D., Mezei, I., Vukobratovic, D., Skrbic, S., & Jakovetic, D. (2021). Deep Learning Anomaly Detection for Cellular IoT with Applications in Smart Logistics. [PDF]
[28]. Akshay Kumaar, M., Samiayya, D., M. Durai Raj Vincent, P., Srinivasan, K., Chang, C. Y., & Ganesh, H. (2022). A Hybrid Framework for Intrusion Detection in Healthcare Systems Using Deep Learning. ncbi.nlm.nih.gov
[29]. Aloufi, R., Haddadi, H., & Boyle, D. (2020). Paralinguistic Privacy Protection at the Edge. [PDF]
[30]. E. Verma, M., & A. Bridges, R. (2018). Defining a Metric Space of Host Logs and Operational Use Cases. [PDF]
[31]. Aneja, S., Ang Xuan En, M., & Aneja, N. (2021). Collaborative adversary nodes learning from the logs of IoT devices in an IoT network. [PDF]
[32]. Belenguer, A., A. Pascual, J., & Navaridas, J. (2022). GowFed - A novel Federated Network Intrusion Detection System. [PDF]
[33]. Parker, L., D. Yoo, P., Asyhari, T., Chermak, L., Jhi, Y., & Taha, K. (2019). DEMISe: interpretable deep extraction and mutual information selection techniques for IoT intrusion detection. [PDF]