ENSC 835 Communication Network

Fall 2024

Final Project

Machine Learning-Based Attack Detection For Electric Vehicle Charging Infrastructure Security

URL: https://github.com/CrashedBboy/ML-NetworkAttack-Detection

Cheng-Lin Wu
Student ID: 301606107
cheng-lin_wu@sfu.ca
Team 1

Daegil Kang
Student ID: 301626179
dka160@sfu.ca
Team 1

Shiyu Zhang
Student ID: 301588601
sza212@sfu.ca
Team 1

December 31, 2024

Contents

Abstract

This project addresses the need for enhanced security within electric vehicle (EV) charging infrastructure, an area critical to sustainable energy transition yet vulnerable to cyber threats. With the rise of electric vehicle supply equipment (EVSE) networks, ensuring protection against potential cyberattacks, such as Distributed Denial of Service (DDoS) attacks, has become essential. This work utilizes the comprehensive Electric Vehicle Supply Equipment (EVSE) Dataset 2024, encompassing network traffic data, system performance metrics, and high-performance computing (HPC) statistics.
By leveraging advanced machine learning techniques, particularly transformer-based architectures, the project focuses on analyzing host-based data—HPC and kernel event logs—to develop a robust anomaly detection framework. Key contributions include employing transformers for sequence-based learning, processing high-granularity event sequences to detect irregular patterns, and implementing a preprocessing pipeline that ensures reliable data input for model training. The experimental setup and methodology outlined here offer insights into securing EVSE systems against large-scale, distributed cyber threats.

1 Introduction

Electric vehicles (EVs) are becoming an increasingly vital part of global efforts to reduce carbon emissions and transition toward sustainable energy [1]. As the adoption of EVs continues to grow, so does the need for robust, secure, and efficient EV charging infrastructure [2]. However, the rapid expansion of electric vehicle charging stations has also raised concerns about the potential for cyberattacks, which compromise the security and reliability of these facilities [3].
This project focuses on enhancing the security of EV charging stations by employing advanced anomaly detection techniques. Specifically, we aim to utilize a multi-dimensional dataset, the Electric Vehicle Supply Equipment (EVSE) Dataset 2024 from the Canadian Institute for Cybersecurity, which contains crucial data on network traffic and system performance from EVSE environments. This project aims to detect Distributed Denial of Service (DDoS) attacks by analyzing data from high-performance computing (HPC) and kernel events, utilizing machine learning models, specifically transformer-based architectures.
The report is organized as follows: Section 2 reviews related work on network anomaly detection, focusing on traffic-based and host-based approaches. Section 3 outlines the

data set used for this project, providing details on the 2024 electric vehicle supply equipment charger attack dataset of the Canadian Institute of Cybersecurity (CICVSE2024) [4] [5] and its components. Section 4 presents the methodology used, including the application of transformer architecture for detecting anomalies, particularly Distributed Denial of Service (DDoS) attacks, in the context of Electric Vehicle (EV) charging systems. Section 5 shows the results and analysis what we discovered through our experiments. Finally, Section 6 concludes the report.

2 Related work

Network anomaly detection is a critical aspect of cybersecurity, where machine learning techniques are increasingly used to identify unusual patterns indicative of potential threats. Existing approaches to data analysis in anomaly detection can generally be categorized into two primary classes: traffic-based features and host-based features (Host-based Intrusion Detection Systems, or HIDS). These classes differ in their focus, data collection points, and the types of threats they are adept at identifying.

Traffic-Based Features
Traffic-based analysis is one of the most widely used approaches in anomaly detection and focuses on monitoring and analyzing data packets as they traverse the network. Traffic-based features, such as packet size, protocol type, flow duration, and byte count, can reveal insights into traffic patterns and highlight irregularities (e.g., sudden surges in volume or atypical packet structures) indicative of potential security breaches.
Two well-known approaches utilizing traffic-based features for deep learning in network anomaly detection are the deep autoencoder-classifier model and the convolutional neural network (CNN) method for DDoS detection. The autoencoder-classifier model leverages unsupervised feature extraction from traffic flow characteristics like packet length and connection protocols, effectively distinguishing between normal and abnormal traffic patterns. This adaptability allows the model to detect sophisticated attack types, such as probes and DoS [6]. Alternatively, the CNN method focuses on DDoS detection by extracting packet- and flow-level features, capturing spatial dependencies in traffic patterns that reveal volumetric attacks [7]. By using flow characteristics like byte counts and packet intervals, this CNN-based approach excels in real-time, high-traffic environments, particularly within cloud networks.
The primary advantage of traffic-based approaches is their capability to detect a wide range of network-based threats, such as Distributed Denial of Service (DDoS) attacks

and port scanning, in real-time. Since this data is aggregated at the network level, it can cover multiple devices and nodes, making it effective in detecting large-scale attacks. However, these methods may struggle to detect insider threats or attacks originating from compromised internal devices that do not generate unusual traffic patterns. Additionally, traffic-based analysis can be resource-intensive, requiring high data storage and processing capabilities, especially in high-traffic networks.

Host-Based Features

Host-based approaches, also known as Host-based Intrusion Detection Systems (HIDS), focus on individual devices or hosts within the network. By collecting data directly from endpoints (e.g., server logs, application logs, and system performance metrics), HIDS can identify anomalies based on host behavior rather than just traffic patterns. Commonly used host-based features include CPU usage, memory consumption, login attempts, and application activity. Byrnes, Jeffrey, et al. utilizes a method for detecting intrusions by analyzing sequences of system calls as host events.

This approach leverages machine learning to learn normal patterns of system calls on a host device, effectively building a model that recognizes typical behaviors. When a new sequence of system calls deviates significantly from these learned patterns, it is flagged as a potential anomaly or intrusion. By focusing on the sequential structure and frequency of system calls, the system captures fine-grained host activity, allowing it to detect both known and previously unseen intrusions. This implementation demonstrates enhanced accuracy and responsiveness, making it well-suited for real-time monitoring and anomaly detection on individual hosts [8].

Host-based approaches excel in detecting threats within the network, such as insider attacks or malware activity, by focusing on system-specific data often invisible in traffic-based methods. HIDS can offer a fine-grained view of individual device behavior, making it ideal for detecting anomalies in critical systems.

However, these methods can be difficult to scale across large networks due to the necessity of deploying and maintaining monitoring agents on each host. Additionally, host-based approaches may suffer from privacy concerns, as they involve close monitoring of endpoint activities.

3 Dataset

The CIC EV Charger Attack Dataset 2024 (CICEVSE2024)[4][5] is a foundational dataset designed to support advanced research on electric vehicle supply equipment (EVSE) security. It provides a detailed examination of both benign and adversarial conditions involving EV chargers, capturing intricate data across multiple dimensions. This dataset is instrumental in understanding how different attack scenarios impact EV chargers, making it an invaluable resource for security assessment and resilience testing in EVSE environments. The synthetic data was collected using a lab setup, as illustrated in Figure 1, which involved a Raspberry Pi to monitor EVSE-B under both normal and attack scenarios.

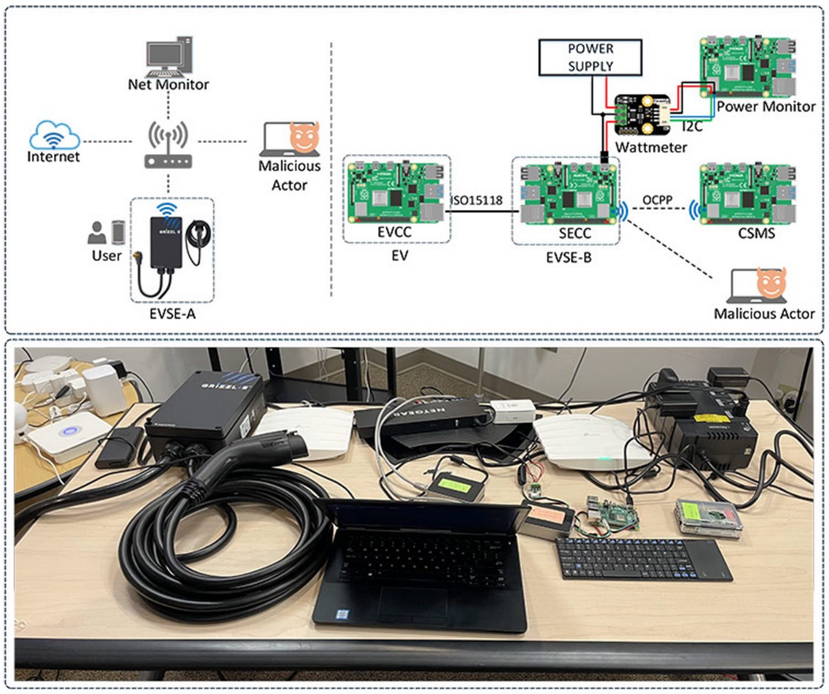


Figure 1: Lab Setup for data collection, EVSE Dataset 2024, Datasets, Research, Canadian Institute for Cybersecurity [5]

The dataset is organized into three core subdirectories, each covering distinct types of data relevant to EV charger operations:

1. Network Traffic: This directory contains raw packet capture (pcap) files and pre-processed CSV files for two distinct types of EV chargers, EVSE-A and EVSE-B. The pcap files enable detailed packet-level analysis of network communication, which is critical for identifying abnormal traffic patterns, intrusion attempts, and other network-based attacks. The CSV files provide extracted features that facilitate statistical analysis and machine learning applications [5].

2. Host Events: Focusing on EVSE-B, this subdirectory holds comprehensive records of host-based events, particularly Hardware Performance Counters (HPC) and Kernel Events. These metrics, captured under both normal and attack scenarios, give insight into the internal processes and performance of the EV charger hardware. HPC data captures low-level operations of the system, while Kernel Events reveal system calls and kernel-level actions, allowing researchers to monitor performance changes under stress and detect potential exploits targeting system resources or privileges [5].

3. Power Consumption: This component of the dataset includes detailed power consumption data for EVSE-B, covering both benign and compromised states. It reveals how different attack scenarios affect power usage, providing a direct view of the physical impact of malicious activities. This subdirectory is essential for profiling the operational stability and efficiency of EV chargers in hostile environments [5].

This project will place a strong emphasis on analyzing the host-based data from the Host Events subdirectory, specifically leveraging Hardware Performance Counters (HPC) and Kernel Events to detect anomalies in EVSE-B under both benign and attack conditions.

The HPC data offer a detailed view of low-level hardware performance, which can provide early indicators of abnormal activity. Key HPC metrics encompass several crucial indicators of system behavior, especially in detecting anomalous activity. Instructions represent the number of executed commands, and unusual spikes or drops in this metric may indicate malicious activity. Cache misses, specifically $L1d\_cache\_wr$ (Level 1 data cache writes) and $L2d\_cache\_rd$ (Level 2 data cache reads), serve as indicators of inefficient memory access or bottlenecks, potentially triggered by an attack. Additionally, CPU migrations and $dTLB-store-misses$, which track data TLB write misses, tend to increase under intense computational loads or system exploitation. Memory access, including both Mem_access_rd (reads) and Mem_access_wr (writes), further aids in identifying unauthorized access patterns by reflecting irregular data handling behaviors. These HPC events capture detailed aspects of the CPU and memory behavior, allowing us to detect deviations from typical patterns under attack scenarios.

Similarly, Kernel Events offer key insights into system-level operations that can help identify suspicious behaviors. System calls, such as *Syscalls_sys_enter_close* and *Syscalls_sys_enter_read*, log entry points for various system-level actions, where anomalies may indicate attempts to interact maliciously with files or processes. Scheduler activities, including *Sched_migrate_task* and *Sched_switch*, monitor task migrations and context switches, potentially flagging abnormal process behavior under malicious influence. Networking events, such as *Net_dev_xmit* (network transmission) and *Qdisc_dequeue* (dequeue events), reveal shifts in network operations, like unusual traffic patterns generated during an attack. Additionally, memory and interrupt management events, including kmem kfree (memory freeing) and *Irq_softirq_raise* (software interrupts), reflect resource management that could be disrupted by exploitation efforts targeting memory or interrupt handling.

By focusing on these host-based events, we aim to capture a comprehensive view of the EVSE-B system's internal behavior under various conditions. Analyzing HPC and Kernel Events in tandem allows for a nuanced approach to behavioral profiling and anomaly detection, as they together reveal both hardware-level and OS-level irregularities. This in-depth focus on host data is expected to yield effective detection techniques by uncovering subtle, system-level indications of compromise that might go unnoticed at the network level alone. This approach will contribute significantly to developing more proactive and robust security measures for EVSE systems.

4 Methodology

In this project, we aim to follow the transformer architecture [9] to implement a DDoS classifier, leveraging the model's strengths in processing sequential data. The rise of the transformer model has significantly advanced the field of machine learning, particularly in natural language processing and other sequential data tasks. Unlike traditional RNN-based approaches, which process sequences token by token, transformers utilize self-attention mechanisms to allow each token to attend to all others in the sequence simultaneously.

This architecture eliminates the vanishing gradient issues commonly faced by RNNs and enables transformers to capture long-range dependencies more effectively. As a result, transformers have shown superior performance across various tasks, from text

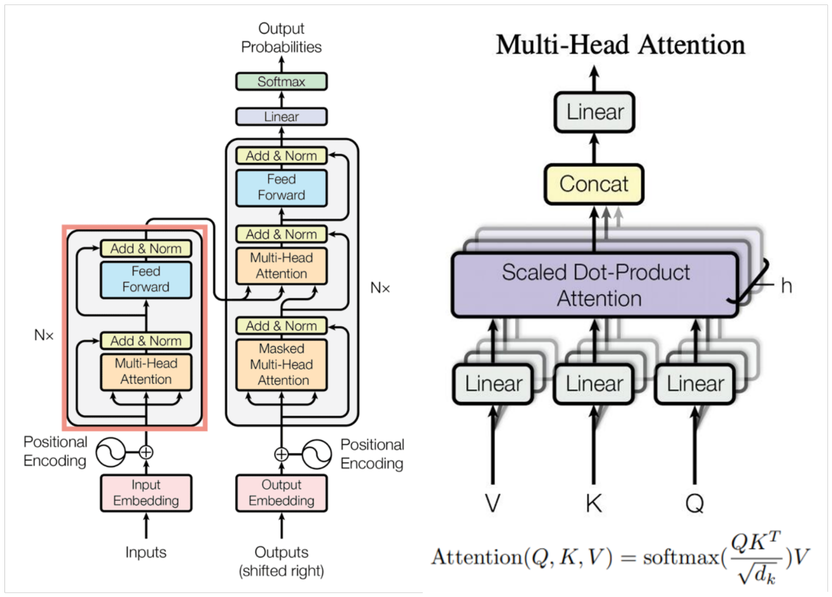


Figure 2: The encoder block of the Transformer architecture.[9]

translation and sentiment analysis to time-series prediction and anomaly detection. In the network anomaly detection domain, several implementation of DDoS detection based on transformer models also indicates such model can provide competitive performance compared to RNNs.[10][11]

The transformer encoder processes input sequences by using a combination of self-attention and feedforward layers to capture complex dependencies between tokens. Initially, each token in the sequence is converted into an embedding, which is then combined with positional encoding to retain the order of tokens. The encoder consists of multiple layers, where each layer includes a multi-head self-attention mechanism and a feedforward neural network. In the self-attention step, each token attends to all other tokens in the sequence, allowing the model to weigh the importance of each token relative to others. This enables the encoder to capture both local and long-range relationships within the sequence.

After self-attention, a feedforward layer processes each token representation individually, enhancing non-linear transformations. Residual connections and layer normalization are applied at each layer to stabilize training and retain information. The encoder outputs a rich, context-aware representation of the input sequence, suitable for downstream tasks such as classification or, in a full transformer model, for passing to a decoder.

Data Preprocessing

In the data preprocessing stage, we begin by normalizing all numerical data fields using min-max normalization, scaling values to a range between 0 and 1. This step ensures that features with varying scales do not disproportionately impact the model and that all numerical inputs are standardized for consistency across sequences.

To enhance data quality, we implement a cleaning step where data examples with corrupted labels are filtered out, reducing noise and ensuring the model learns from accurate, high-integrity data. Finally, we prepare the dataset for training by applying five-fold cross-validation, where the data is split into five subsets. Each subset is used as a test set once while the remaining subsets serve as the training data, ensuring robust evaluation of model performance and minimizing the risk of overfitting. This preprocessing pipeline prepares the data for effective input to the transformer model, maximizing both the quality and reliability of the learning process.

1. **Data scale**
   · Total data examples: 8468x914
   · Within 8468 examples, there are 865 rows are 'DoS'

2. **Data Correlation** - Dimensionality Reduction
   · Step1: Subtract the Mean: For each value in column X and Y, subtract their respective means $\bar{X}$ and $\bar{Y}$

$$X_i - \bar{X}, \quad Y_i - \bar{Y} \tag{1}$$

   · Step2: Compute the Covariance: Multiply these mean-centered values for XX and YY and sum them up

$$Cov(X, Y) = \sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y}) \tag{2}$$

   · Step3: Calculate the Standard Deviations

$$\sigma_X = \sqrt{\sum_{i=1}^{n} (X_i - \bar{X})^2} \qquad \sqrt{\sum_{i=1}^{n} (Y_i - \bar{Y})^2} \tag{3}$$

· Step4: Normalize the Covariance: Divide the covariance by the product of the standard deviations of X and Y

$$\frac{Cov(X,Y}{\sigma_X \cdot \sigma_Y} \tag{4}$$

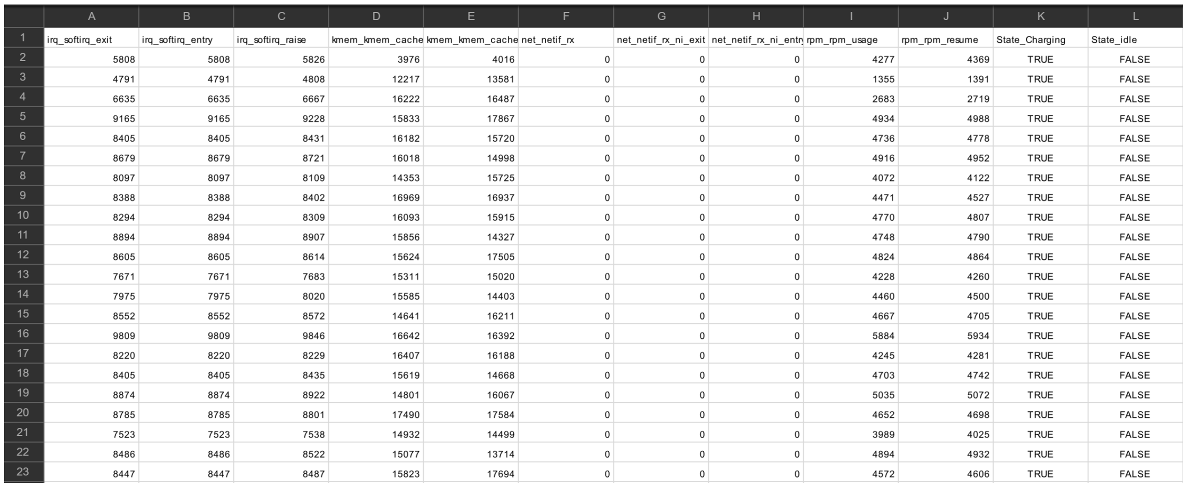
3. **Identifying the Top 10 Correlated Columns**
Once the correlation coefficients have been calculated for each numerical column with respect to the label column, the next step is to identify the top 10 columns with the highest correlation. The correlation values are computed, and their absolute values are taken to focus on the strength of the relationships, regardless of the direction. These values are then sorted in descending order, and the top 10 columns with the highest absolute correlation are selected for further analysis.

   (a) irq_softirq_exit
   (b) irq_softirq_entry
   (c) irq_softirq_raise
   (d) kmem_kmem_cache_free
   (e) kmem_kmem_cache_alloc
   (f) net_netif_rx
   (g) net_netif_rx_ni_exit
   (h) net_netif_rx_ni_entry
   (i) rpm_rpm_usage
   (j) rpm_rpm_resume

4. **One-hot Encoding** - Convert categorical data into a numerical format
For a categorical variable C with k unique categories, one-hot encoding converts a given category $c \in C$ into a k-dimensional binary vector $\mathbf{v}$ where:

$$v_i = \begin{cases} 1, & \text{if } i = \text{c's index} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

for $i = 1, 2, ..., k$.



	A	B	C	D	E	F	G	H	I	J	K	L
1	irq_softirq_exit	irq_softirq_entry	irq_softirq_raise	kmem_kmem_cache	kmem_kmem_cache	net_netif_rx	net_netif_rx_ni_exit	net_netif_rx_ni_entr	rpm_rpm_usage	rpm_rpm_resume	State_Charging	State_idle
2	5808	5808	5826	3976	4016	0	0	0	4277	4369	TRUE	FALSE
3	4791	4791	4808	12217	13581	0	0	0	1355	1391	TRUE	FALSE
4	6635	6635	6667	16222	16487	0	0	0	2683	2719	TRUE	FALSE
5	9165	9165	9228	15833	17867	0	0	0	4934	4988	TRUE	FALSE
6	8405	8405	8431	16182	15720	0	0	0	4736	4778	TRUE	FALSE
7	8679	8679	8721	16018	14998	0	0	0	4916	4952	TRUE	FALSE
8	8097	8097	8109	14353	15725	0	0	0	4072	4122	TRUE	FALSE
9	8388	8388	8402	16969	16937	0	0	0	4471	4527	TRUE	FALSE
10	8294	8294	8309	16093	15915	0	0	0	4770	4807	TRUE	FALSE
11	8894	8894	8907	15856	14327	0	0	0	4748	4790	TRUE	FALSE
12	8605	8605	8614	15624	17505	0	0	0	4824	4864	TRUE	FALSE
13	7671	7671	7683	15311	15020	0	0	0	4228	4260	TRUE	FALSE
14	7975	7975	8020	15585	14403	0	0	0	4460	4500	TRUE	FALSE
15	8552	8552	8572	14641	16211	0	0	0	4667	4705	TRUE	FALSE
16	9809	9809	9846	16642	16392	0	0	0	5884	5934	TRUE	FALSE
17	8220	8220	8229	16407	16188	0	0	0	4245	4281	TRUE	FALSE
18	8405	8405	8435	15619	14668	0	0	0	4703	4742	TRUE	FALSE
19	8874	8874	8922	14801	16067	0	0	0	5035	5072	TRUE	FALSE
20	8785	8785	8801	17490	17584	0	0	0	4652	4698	TRUE	FALSE
21	7523	7523	7538	14932	14499	0	0	0	3989	4025	TRUE	FALSE
22	8486	8486	8522	15077	13714	0	0	0	4894	4932	TRUE	FALSE
23	8447	8447	8487	15823	17694	0	0	0	4572	4606	TRUE	FALSE

Figure 3: Merged data

5. **Integration**
   Merge selected numerical columns and encoded categorical columns

DDoS Detection

To detect host attacks using kernel events and high-performance computing (HPC) statistics, we first structure the data into sequences using a 5-second time interval, capturing snapshots of host activity at high granularity. Each sequence aggregates kernel event data and HPC metrics (e.g., CPU and memory usage, disk I/O), creating a continuous representation of system behavior. For each unique type of kernel event and HPC metric, we assign a specific token or identifier, effectively building a vocabulary of possible events. These tokens are then mapped to numerical embeddings that represent each event and metric within a fixed-dimensional space, encoding event type and associated features such as CPU or memory usage at the time of occurrence.

To ensure temporal order within each 5-second sequence, we add positional encoding to each embedding, allowing the transformer model to recognize the sequence order of events and metrics. Since sequence lengths may vary, we apply padding where necessary, masking these positions to ensure they do not affect model learning. Additionally, we prepend a [CLS] token to each sequence, which will serve as an aggregated representation for the entire sequence, capturing key patterns indicative of normal or abnormal behavior. During training, the transformer encoder processes these sequences, with the output from the [CLS] token fed into a classification layer that predicts whether the system is under attack. This methodology enables the transformer to capture both event type dependencies and temporal patterns in host data, facilitating accurate attack detection.

1. **Data Process**
   · Min-max normalization
   · Shuffle the data examples
   · 7:1:2 data splitting

2. **Training Parameters**
   Loss Function: Binary Cross Entropy (BCE):
   · Binary classification problems
   · Measure the difference between the predicted probability and the actual binary class (0 or 1)

$$BCE\ Loss = -\frac{1}{N}\sum_{i=1}^{N}[y_i \cdot \log(\hat{y}_i + (1 - y_i) \cdot \log(1 - \hat{y}_i)] \tag{6}$$

N	- Total number of samples
y_i	- True label (0 or 1)
$\hat{y}_i$	- Predicted probability for class 1

   Optimizer: Adam with Learning Rate of 1e-4:
   · Adjust the learning rate for each parameter
   Early Stopping Mechanism with Patience = 5:
   · Prevent overfitting by halting training if the validation
   · Performance doesn't improve after a set number of epochs (patience)
   Epoch = 1000:
   · Ensure the model has sufficient opportunity to learn

5 Results and Analysis

Result
The model's performance is evaluated using three key metrics: Accuracy, Precision, and Recall.
· **Accuracy**: Measures the overall correctness of a model. Best suited when the class distribution is balanced [4].

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN} = 0.9903$$

· **Precision**: Evaluates the reliability of positive predictions. Useful in scenarios where false positives are costly [4].

$$\text{Precision} = \frac{TP}{TP + FP} = 0.9371$$

· **Recall**: Measures how well the model identifies positive samples. Important in cases where missing positive samples (false negatives) is critical [4].

$$\text{Recall} = \frac{TP}{TP + FN} = 0.9939$$

Analysis
The model demonstrates excellent performance in detecting Denial of Service (DoS) attack instances, with a high Recall value of 0.9939, indicating that it correctly identifies nearly all such instances. However, there is a trade-off in terms of Precision, which stands at 0.9371, suggesting that some benign traffic is incorrectly classified as a DoS attack (false positives). Despite this, the model's sensitivity to DoS attacks remains highly effective, though it could result in unnecessary alarms due to the increased rate of false positives.
The training curves, as shown in Figure 4, demonstrate that the model is learning effectively, with both the training and validation curves converging and approaching stability. This behavior indicates strong generalization to new data and suggests that the model is training effectively without significant overfitting, performing robustly on

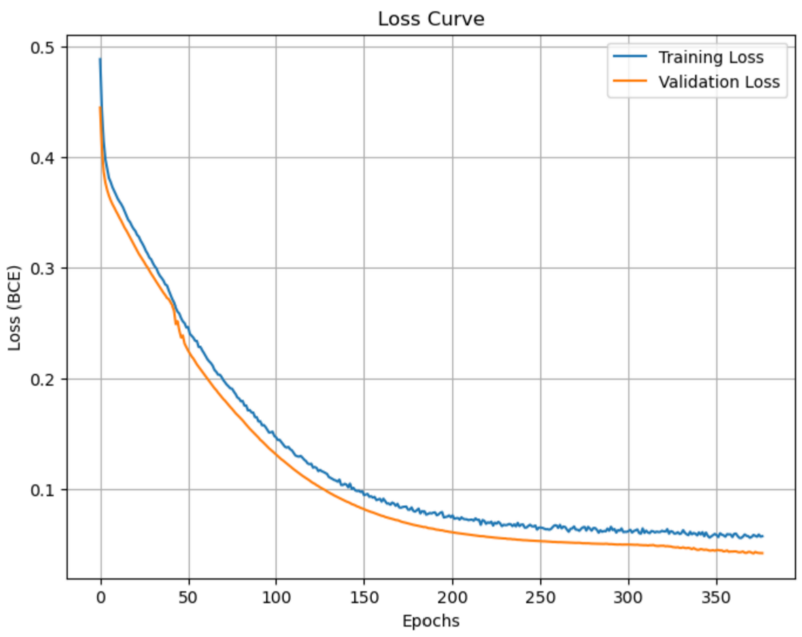


Figure 4: Loss curve

the data it has been trained on.

However, it is important to note that the model's performance is based on synthetic data, which may not fully capture the real-world patterns of DoS attacks and benign traffic. This discrepancy could lead to a mismatch between the model's behavior in controlled environments and its performance in actual operational settings. Similar challenges are often observed in recent studies that apply machine learning techniques to detect network anomalies and intrusions, where the limitations of synthetic datasets in representing real-world scenarios are frequently noted [12].

6 Conclusions

This study demonstrated the effectiveness of transformers in capturing long-range dependencies in network traffic data. The model showed strong potential in detecting anomalies, highlighting its suitability for real-world cybersecurity applications. By leveraging the transformer architecture, which excels in handling sequential information, the model was able to identify complex patterns in network behavior, proving its ability to address various anomaly detection tasks effectively.

In future work, there are several key areas to explore. First, the model could be extended to real-world environments, where it can be tested with live network traffic to evaluate its performance in dynamic, operational settings. Additionally, incorporating a broader range of cyberattack types beyond Denial of Service (DoS) attacks will improve the model's robustness and generalization capabilities. Furthermore, optimization techniques such as hyperparameter tuning and more efficient training strategies could be explored to enhance the model's performance and scalability. Finally, investigating hybrid models that combine transformers with traditional machine learning techniques, such as decision trees or support vector machines, could yield further performance improvements by leveraging the strengths of both approaches.

References

[1] IEA. By 2030 evs represent more than 60% of vehicles sold globally, and require an adequate surge in chargers installed in buildings. `https://www.iea.org/reports/by-2030-evs-represent-more-than-60-of-vehicles-sold-globally-and-require-an-adequ` 2022. Accessed: 5-Oct-2024.

[2] D. Ronanki and H. Karneddi. Electric vehicle charging infrastructure: Review, cyber security considerations, potential impacts, countermeasures, and future trends. *IEEE Journal of Emerging and Selected Topics in Power Electronics*, 12(1):242–256, 2024.

[3] S. Hamdare, O. Kaiwartya, M. Aljaidi, M. Jugran, Y. Cao, S. Kumar, M. Mahmud, D. Brown, and J. Lloret. Cybersecurity risk analysis of electric vehicles charging stations. *Sensors*, 23(15):6716, 2023.

[4] Emmanuel Dana Buedi et al. Enhancing ev charging station security using a multi-dimensional dataset: Cicevse2024. In *IFIP Annual Conference on Data and Applications Security and Privacy*, Cham, 2024. Springer Nature Switzerland.

[5] Canadian Institute for Cybersecurity. Evse dataset 2024, 2024. Accessed: 5-Oct-2024.

[6] N. Shone et al. A deep learning approach to network intrusion detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(1):41–50, 2018.

[7] R. Vinayakumar, K. P. Soman, and P. Poornachandran. Applying deep learning approaches for network traffic prediction. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1537–1543. IEEE, 2017.

[8] J. Byrnes et al. A modern implementation of system call sequence based host-based intrusion detection systems. In *2020 Second IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, pages 123–129. IEEE, 2020.

[9] A. Vaswani et al. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

[10] H. Wang and W. Li. Ddostc: A transformer-based network attack detection hybrid mechanism in sdn. *Sensors*, 21(15):5047, 2021.

[11] Z. Long et al. A transformer-based network intrusion detection approach for cloud security. *Journal of Cloud Computing*, 13(1):5, 2024.

[12] Z. Li, A. L. Gonzalez Rios, and Lj. Trajkovic. Machine learning for detecting anomalies and intrusions in communication networks. *IEEE Journal on Selected Areas in Communications*, 39(7):2254–2264, Jul. 2021.

7 Contribution

· Literature survey - All
· Data exploration and preprocessing - Daegil / Cheng-Lin Wu
· Model implementation and training - Cheng-Lin Wu / Shiyu
· Performance evaluation and analysis - All
· Written final report - All