A Brief Study on Network Intrusion Detection Systems and Machine Learning Applications

Liam Keefe
ECE 6612
Georgia Institute of Technology

April 30, 2025

Abstract

Network Intrusion Detection Systems (NIDS) are critical cybersecurity components, increasingly leveraging Machine Learning (ML) to improve the detection of sophisticated attacks, particularly through anomaly detection. This paper presents a simulation study analyzing the performance of Kitsune, a lightweight, unsupervised ML-NIDS based on an ensemble of autoencoders (KitNet), designed for online packet processing. The study utilized the Python implementation of Kitsune with default parameters (‘maxAE=10‘, ‘FMgrace=5000‘, ‘ADgrace=50000‘) on the Mirai BotNet and the large-scale CIC-IDS 2017 datasets, executed on a local machine. Anomaly classification was performed by applying a threshold based on the 99th percentile of Root Mean Squared Error (RMSE) scores from normal traffic. Performance metrics (Accuracy, Precision, Recall, F1-Score) were calculated against ground truth, involving significant post-processing challenges to map packet-level results to flow-based labels for the CIC-IDS 2017 dataset. Results showed Kitsune achieved high performance on the Mirai dataset (Accuracy: 90.1%, Precision: 99.9%, Recall: 99.0%, F1: 93.7%). However, performance on the complex CIC-IDS 2017 dataset was significantly lower (Accuracy: 58.4%, Precision: 12.0%, Recall: 98.2%, F1: 21.4%), primarily due to very low precision despite high recall, indicating a high false positive rate. This study demonstrates Kitsune’s application and highlights practical challenges in evaluation.

1 Introduction

Confronted by the escalating complexity and sophistication of cyberattacks, traditional security measures are often limited. In conjunction with traditional security measures like antiviruses and firewalls, Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) have emerged as a critical component of modern cybersecurity strategies. Three solution IDS strategies are Host-based Intrusion Detection Systems (HIDS), Network-based Intrusion Detection Systems (NIDS), and a Hybrid Intrusion Detection Systems (Hybrid IDS) . HIDS (and Host Intrusion Preventions Systems (HIPS)) monitor and analyze the internals of a computing system, collecting data on system calls, process activity, file system modifications, system logs, etc. NIDS (and Network Prevention Detection Systems (NIPS)) on the other hand, monitor network traffic passing through a network segment for suspicious activity conducted via packets. NIDS can be deployed in several locations, either within the network or on the network perimeters, interacting closely with firewalls and other security devices. The main goal of NIDS is to detect and respond to potential threats in real-time, providing an additional layer of security to the network infrastructure [11]. Hybrid IDS combine the strengths of both HIDS and NIDS, providing a comprehensive security solution that can monitor both network traffic and host activity. By leveraging the capabilities of both systems, hybrid IDS can provide a more robust defense against a wide range of cyber threats.

IDS can be classified into two main categories: signature-based and anomaly-based detection. However, traditional IDS also have limitations. Signature-based detection, analogous to antivirus software which relies on known malware signatures. Signature-based approaches will maintain a database for known 'signatures', as it continuousy monitors network traffic, it compares incoming packets to the signatures in its database. If a match is found, an alert is generated. This method is effective for known attacks but may struggle with novel or zero-day attacks or new variants of existing threats [3]. Anomaly-based detection, on the other hand, establishes a baseline via several methods like statistical analysis, policy or protocol violations, and heuristic measurements of normal network behavior and flags any deviations from this baseline as potential threats. While this approach can identify previously unknown attacks, it may also generate false positives due to legitimate deviations from the established baseline. By flagging any suspicious activity, these analysis methods can be used to detect a wide range of attacks, including malware, denial-of-service (DoS) attacks, and unauthorized access attempts.

Several methods are currently implemented in a pursuit to lower the false positive rate and increase the detection rate. These methods include statistical analysis, machine learning, and deep learning techniques [16], among many others. Statistical analysis involves collecting and analyzing data to identify patterns and anomalies in network traffic. This can include monitoring network traffic for unusual spikes in activity, analyzing packet headers for unusual patterns, and examining system logs for signs of unauthorized access. Statistical analysis can be effective in identifying known attack patterns, but it may struggle with novel or zero-day attacks. Machine learning involves training algorithms on large datasets to identify patterns and anomalies in network traffic.

Among other methods, along with this paper, NIDS can be implemented and will be analyzed using machine learning. In a controlled simulation, a performance of IDS is typically evaluated using metrics such as accuracy, specificity, precision, recall, and F1-score. These metrics help assess the system's ability to correctly identify both normal and malicious traffic. Additionally, the trade-off between detection rate and false positive rate is crucial in determining the effectiveness of an IDS. A high detection rate with a low false positive rate is ideal, as it minimizes the risk of overlooking real threats while reducing the burden of false alarms on security analysts.

ML-NIDS goes above and beyond traditional NIDS, especially in anomaly-based detection, where analysis and alerts come as a result of statistical thresholds and rule violations [7]. NIDS are implemented in a multiple stage process, where the first stage is data collection, followed by data preprocessing, feature extraction, and finally classification. The data collection stage involves gathering network traffic data from various sources, such as packet captures and network flow data. This data is then preprocessed to remove noise and irrelevant information, often called 'features', ensuring that the model is trained on high-quality data. Feature extraction involves selecting and transforming relevant features from the raw data, which can include packet headers, flow statistics, and protocol information. This step is crucial for improving the model's performance, as it helps reduce the dimensionality of the data and focuses on the most informative features. Finally, classification involves applying machine learning algorithms to the preprocessed data to identify patterns and classify network traffic as normal or malicious. This stage can involve various machine learning techniques, such as supervised, unsupervised, or semi-supervised learning, depending on the availability of labeled data. Supervised learning requires labeled data for training, while unsupervised learning can identify patterns in unlabeled data. Semi-supervised learning combines both approaches, leveraging a small amount of labeled data alongside a larger set of unlabeled data.

The choice of machine learning algorithm depends on the specific requirements of the NIDS and the characteristics of the dataset being used. Commonly used algorithms include decision trees, support vector machines (SVM), random forests, and neural networks [1]. These algorithms can be trained to recognize patterns associated with various types of attacks. By continuously learning from new data and adapting to evolving threats, ML-NIDS can improve their detection capabilities over time. ML algorithms are used to learn from historical data, enabling them to adapt to evolving threats and improve detection accuracy over time. Generally, ML-NIDS are trained using classified datasets, often with labeled data, to identify patterns and anomalies in network traffic. Benchmarks are created using known datasets like KDD Cup 1999, UNSW-NB15, CI-CIDS 2017, and NSL-KDD. These datasets contain a variety of attack types, including DoS, probing, and remote-to-local attacks.

2 Related Work

IDS have been extensively investigated and developed over recent years, with numerous studies focusing on improving their detection capabilities and reducing false positive rates. NIDS in particular have gained significant attention due to their scalability and ability to monitor large networks. Several studies have explored the use of machine learning techniques to enhance the performance of NIDS. There are many approaches to improving NIDS, this is just a brief overview of some of the research that has been done in this area.

2.1 Prior Comparisons

Several studies have compared the performance of different machine learning algorithms for NIDS. For example, an extensive comparison was done by Hesford et al. [4] on the performance of various models and algorithms. Hesford et al. compared popular algorithms that were accessible through Github repositories, including Kitsune, HELAD [17], DNN [15], and Slip [2].

The study also looked at a variety of different datasets, including the UNSW-NB15 [10], CIC-IDS-2017, BoT-IoT [5], Stratosphere, and Mirai. The study found that the performance of the models varied significantly depending on the dataset used, with some models performing well on certain datasets but poorly on others. This does overlap with the study of this paper, as the performance of the Kitsune model was evaluated on two datasets, the Mirai BotNet dataset and the CIC-IDS 2017 dataset. However, the study done by Hesford et al. did not analyze the attack structure of their datasets, the pre-processing/post-processing of the datasets. Therefore, this study looks to improve on many of the limitations of the study done by Hesford et al. and provide a more comprehensive evaluation of the performance of the Kitsune model.

2.2 Autoencoders

Autoencoders are a type of neural network used for unsupervised learning, particularly in the context of anomaly detection [8]. Multiple variations are presented in an aim to lower false-positive rates and increase detection rates. They consist of an encoder that compresses input data into a lower-dimensional representation and a decoder that reconstructs the original data from this representation. By training the autoencoder on normal data, it learns to capture the underlying patterns and structures of the data. When presented with new data, the autoencoder can reconstruct the input. If the reconstruction error is significantly high, it indicates that the input data deviates from the learned patterns, suggesting an anomaly. This approach is particularly useful for detecting previously unknown attacks, as the autoencoder can identify deviations from normal behavior without relying on predefined signatures.

Considerations for using autoencoders include the data purity of the training data, the threshold for reconstruction error, and the architecture of the autoencoder itself. The data purity of the training data is crucial, as the autoencoder learns to reconstruct the patterns present in the training data. If the training data contains anomalies or noise, the autoencoder may learn to reconstruct these anomalies, leading to false positives during testing. Therefore, it is essential to ensure that the training data is representative of normal behavior and free from anomalies. The threshold for reconstruction error is another important consideration. Setting the threshold too low may result in false positives, while setting it too high may lead to missed detections. The choice of architecture can significantly impact the performance of the autoencoder, as different architectures may be better suited for different types of data. For example, convolutional autoencoders are often used for image data, while recurrent autoencoders are more suitable for sequential data.

2.3 Dataset Selection

The choice of dataset is critical for evaluating the performance of NIDS. Several datasets are commonly used in the research community, including KDD Cup 1999 [14], UNSW-NB15, CIC-IDS-2017, and NSL-KDD [13]. These datasets contain a variety of attack types, including DoS, probing, and remote-to-local attacks. The choice of dataset can significantly impact the performance of the model, as different datasets may contain different types of attacks and network conditions. Therefore, it is essential to select a dataset that is representative of the target environment and contains a diverse range of attack types. The CIC-IDS-2017 dataset is a more recent dataset for evaluating the performance of NIDS. The dataset contains a variety of attack types, including DoS, probing, and remote-to-local attacks. Given the complexity of the dataset, it is especially useful for determining the performance of the Kitsune model, in a more contemporary attack environment. The dataset is available from the Canadian Institute for Cybersecurity (CIC) and contains a large amount of network traffic data, making it suitable for training and evaluating machine learning models. The Mirai BotNet dataset is a popular dataset for evaluating the performance of Kitsune NIDS, and will be used as a verification case in this study.

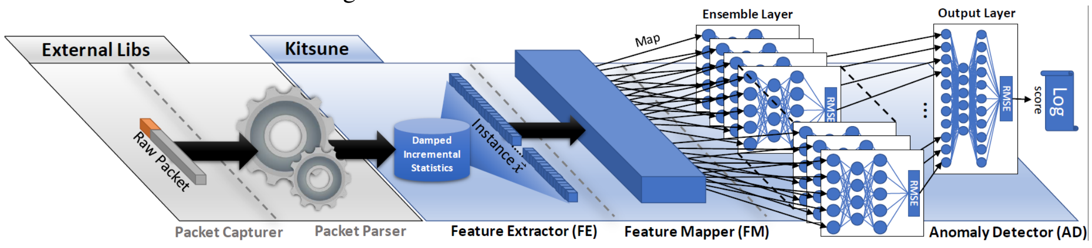
3 Approach

NIDS deployed out in the field will typically run during the duration of uptime of the network, collecting data and analyzing it in real-time. NIDS are also typically deployed in chokepoint environments where they can monitor all incoming and outgoing traffic. This is not the case for this paper, as the simulation will be run on a local machine, using a dataset of network traffic data. However, to decrease the complexity of the simulation and the necessary overhead to completely understand Machine Learning principles and paradigms, the simulation will be run using a relatively popular and well known NIDS. Kitsune [9] is a NIDS that uses machine learning techniques to detect anomalies in network traffic. It is designed to be lightweight and efficient, making it suitable for deployment in various environments. The implementors of Kitsune have also published the python code for the model, making it easy to use and modify. Kitsune uses unsupervised learning techniques to identify patterns in network traffic and detect anomalies.

Kitsune's core KitNet algorithm learns a baseline from normal data in an unsupervised manner, not requiring attack labels for this initial training phase. The model uses a combination of clustering and classification techniques to identify patterns in network traffic and detect anomalies. Kitsune also doesn't feature a signature-based detection system, which is a common feature of traditional NIDS. Kitsune uses an incremental based approach, in where the reconstruction of the previous packet does not affect the next packet. This allows Kitsune to run with relatively low memory and lowers the complexity of the reconstruction. The model is designed to be lightweight and efficient, making it suitable for deployment in various environments.

For the purposes of this paper, a simulation will be used to analyze the performance of NIDS. The simulation will be run on a local machine, using a dataset of network traffic data. The dataset used will be the Mirai BotNet dataset [9] that Kitsune has been popularized for. The dataset contains a variety of attack types, including DoS, probing, and remote-to-local attacks. In addition to the CIC-IDS 2017 dataset [12]. The dataset contains a variety of attack types, including DoS, probing, and remote-to-local attacks. The dataset will establish a baseline by using a period of normal traffic data, which will be used to create the threshold needed to ascertain RMSE driven anomalies. Then using attack filled periods of the dataset, the model will determine whether or not the packets are normal or malicious. The model will be evaluated using the RMSE score, which is a measure of how different the input data is from the learned patterns. The model will be evaluated using various metrics, including accuracy, precision, recall, and F1-score. These metrics will help assess the system's ability to correctly identify both normal and malicious traffic. Additionally, the trade-off between detection rate and false positive rate will be analyzed to determine the effectiveness of the NIDS. A high detection rate with a low false positive rate is ideal, as it minimizes the risk of overlooking real threats while reducing the burden of false alarms on security analysts.

Figure 1: The Kitsune Model Architecture



The architecture of the Kitsune model is taken from Miresky et al [9]. The model consists of a Feature Extraction stage, Feature Mapper, an Ensemble Layer (Anomaly Detection).

3.1 Model Architecture

Kitsune uses a simple model architecture as shown in Figure 1, consisting of an input layer, several hidden layers, and an output layer. The input layer will take in the relatively unprocessed network traffic data. Kitsune is built with deployment to routers/network traffic monitors in mind and is intended to be plug-and-play, and therefore, the input layer takes in raw network traffic data in the form of .pcap filetype. The PCAP filetype stands for Packet Capture, and is the standard file format for network traffic data. PCAP files are used to capture and store network packets, allowing for analysis and monitoring of network traffic. PCAP files can be generated using various tools, such as Wireshark, tcpdump, and tshark. The PCAP file format is widely used in network analysis and security research, as it provides a standardized way to capture and store network packets. In comparison to popularly used labelled datasets, raw network traffic data tends to be several times larger in packet number. This is due to the fact that labelled datasets are often preprocessed and filtered to remove noise and irrelevant information. The use of network monitoring tools like Wireshark/TShark/CICFlowMeter/Scapy are used to parse packets and separate the raw network features into the network flows. Network flows are groupings of packets that share common characteristics, and use characteristics like source and destination IP addresses, source and destination ports, and protocol information to identify the flow of data between devices on a network. This preprocessed data is then used to train the traditional ML-NIDS models and the number of network flow characteristics is often 80 or more. This can amount to a significant reduction of the size of the dataset, as the number of packets in the dataset is reduced to a fraction of the original size. Therefore, when using a NIDS like Kitsune, a considerable amount of post processing is required to understand the performance metrics for a given dataset.

The initial stages involve feature extraction using the AfterImage technique, which calculates statistics incrementally, and is considered the pre-ensemble stage. Feature extraction involves selecting and transforming relevant features from the raw data, which can include packet headers, flow statistics, and protocol information as it pertains to the network flow. This step is crucial for improving the model's performance, as it helps reduce the dimensionality of the data and focuses on the most informative features. Kitsune identifies features such as the timestamp, frame length, source IP address, destination IP address, among many others. The feature extraction uses damped incremental statistics to identify the features in the network traffic data. Thereby, the feature mapper assigns vectors to feature clusters.

The core ML-NIDS stage is within the Kitsune NIDS tool is the anomaly detection stage, which uses KitNet, an autoencoder ensemble model to perform anomaly detection via reconstruction error. The unsupervised learning techniques used in Kitsune are based on the concept of clustering, where the model groups similar data points together based on their features. The presets according to the Kitsune model are set to:

- maxAE = 10
- FMgrace = 5000
- ADgrace = 50000

The numbers used are presets, but are adjustable depending on the situation. These parameters will stay consistent throughout the simulation, and are not being tuned for the optimization of the dataset. The maxAE parameter determines the maximum size of any autoencoder in the ensemble layer, while the FMgrace and ADgrace parameters determine the number of instances used to learn the feature mapping and train the anomaly detector, respectively.

At the core of this anomaly detection stage, is the machine learning algorithm 'KitNet'. This algorithm uses small ensembles of autoencoders to learn the normal behavior of the network and identify anomalies. Using the clustered data from the feature extraction stage, KitNet creates a set of autoencoders that are trained on the normal data. The autoencoders are designed to learn the underlying patterns and structures of the data, allowing them to reconstruct the input data. When presented with new data, the autoencoders can reconstruct the input. If the reconstruction error is significantly high, it indicates that the input data deviates from the learned patterns, suggesting an anomaly. The autoencoders learn to reconstruct the input data, and the reconstruction error, measured in Root Mean Squared Error (RMSE) is used to identify anomalies. This RMSE is directly used to calculate the anomaly score, which is a measure of how different the input data is from the learned patterns. A score above a certain threshold indicates an anomaly, while a score below the threshold indicates normal behavior.

The output layer will produce the RMSE score, the reconstruction error, of the network traffic. A creation of a post processing output layer is needed to indicate whether it is normal or malicious. The output layer of Kitsune is not included with the Kitsune model. Therefore, in order to evaluate the performance of the model, a custom output layer will be created. A RMSE threshold score will be required to be set to determine the boundary of what is considered an anomaly. The RMSE threshold score will be set to 99th percentile of the observed normal traffic data. Indicating that 99 percent of the normal traffic data passed the criterion for being classified as benign from an RMSE score standpoint. The threshold score will be used to classify the network traffic as either normal or malicious. The output layer will also include the calculation of performance metrics of the model, including accuracy, precision, recall, and F1-score. To determine these metrics, each anomaly needs to be correlated to the ground truth of the dataset. This is done by comparing the determined anomalies to the known attacks packets or flow in the labelled dataset. The performance metrics will be calculated using the following formulas:

- True Positives (TP): The number of correctly identified anomalies (malicious traffic).
- True Negatives (TN): The number of correctly identified normal traffic.
- False Positives (FP): The number of normal traffic incorrectly identified as anomalies.
- False Negatives (FN): The number of anomalies incorrectly identified as normal traffic.

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

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

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

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

3.2 Dataset Selection

The dataset used for this simulation is the Mirai BotNet dataset, which is a popular dataset for evaluating the performance of Kitsune NIDS. The dataset contains a variety of attack types, including DoS, probing, and remote-to-local attacks. This dataset is mostly used in this context to verify the performance of the Kitsune model, as it is well-suited for confirming the setup of the Kitsune NIDS. The Kitsune Mirai BotNet dataset in PCAP format is available on the GitHub repository of the Kitsune model. The ease of use of this particular dataset cannot be overstated, as the available labels often correspond directly to the packet capture without requiring complex flow-based feature extraction or mapping or filtering to extract flow-based features from the network traffic data. Therefore, each RMSE score can be directly correlated to the labelled dataset.

The more experimental dataset used in this simulation is the CIC-IDS 2017 dataset, which is a more recent dataset for evaluating the performance of NIDS. The CIC-IDS 2017 dataset contains a variety of attack types, including DoS, probing, and remote-to-local attacks. This dataset is used to evaluate the performance of the Kitsune model in a more realistic environment, as it contains a wider range of attack types and network conditions. The CIC-IDS 2017 dataset PCAP format is available from the Canadian Institute for Cybersecurity (CIC). The raw form of this dataset is on the order 54 million packets, while the available labeled dataset contains approximately 2.8 million flows. The labelled dataset is available in CSV format, which is a more common format for NIDS datasets. The tool used to reduce the size of the dataset is called CICFlowMeter [6], which is a flow-based feature extraction tool that converts PCAP files into CSV format. The tool is designed to extract flow-based features from network traffic data, allowing for the analysis and monitoring of network traffic. The tool is available on the GitHub repository of the CIC. This dataset requires significant post-processing of the RMSE data, to map corresponding RMSE packet data to the relevant labeled dataset, which represents aggregated flow data and is thus much smaller in data size due to its flow-based features.

4 Results

The simulation was run on a local machine, using a dataset of network traffic data. The simulation was run using the Kitsune model, with the parameters set to the default values. The simulation was run for several days, with the Mirai BotNet dataset taking about 3 hours to run and the CIC-IDS 2017 dataset taking about 3 days to run. The performance metrics were calculated using the RMSE threshold score of 99th percentile of the normal (benign) traffic data. The performance metrics are shown in Table 1.

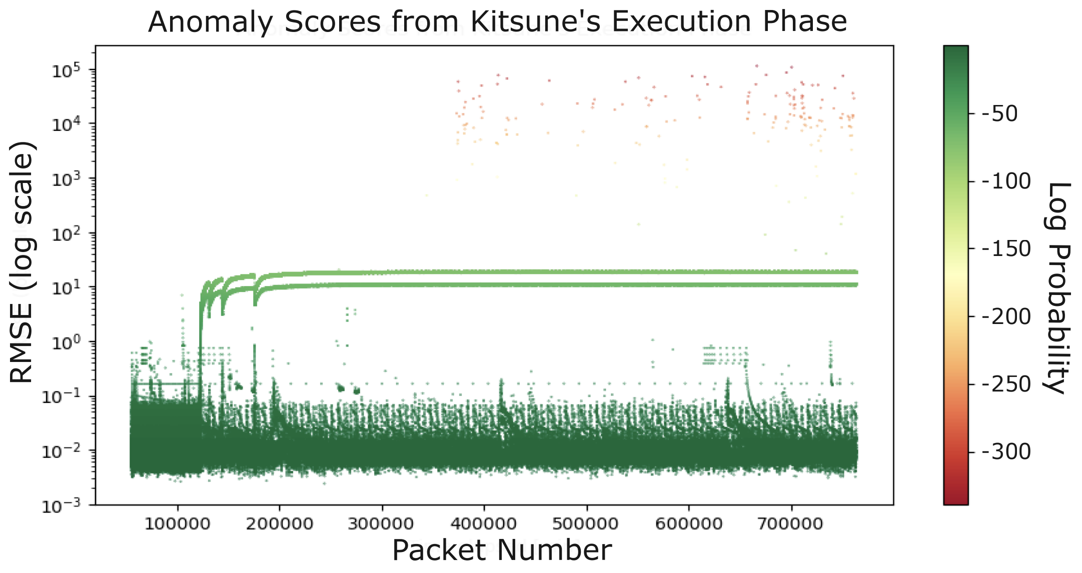
Table 1: Kitsune Model Simulation Performance Metrics

Dataset	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Mirai BotNet	90.1	99.9	99.0	93.7
CIC-IDS-2017	58.4	12.0	98.2	21.4

The performance metrics are calculated using the RMSE threshold score of 99th percentile of the normal (benign) traffic data.

In Figure 2 , the RMSE score and log probability is plotted against the time of the simulation for the Mirai BotNet dataset. The RMSE score is shown to be relatively stable over time, with a few spikes in the score. The CIC-IDS 2017 dataset is not shown, as the complexity and high density of events in the CIC-IDS-2017 dataset make packet-level RMSE visualization difficult at this scale to properly display the attack packets.

Figure 2: RMSE Score and Log Probability vs. Packet Number for Mirai BotNet Dataset



RMSE values plotted with log probability over packet numbers for the Mirai BotNet dataset. The RMSE values are shown to be relatively stable over time, with a few spikes in the score. The spikes in the score correspond to the time of the attacks in the dataset.

4.1 Performance Metrics

The performance metrics of the Kitsune model on the Mirai BotNet dataset are shown in Table 1. The model achieved a high accuracy rate of 90.1%, with a precision of 99.9%, recall of 99.0%, and F1-score of 93.7%. This indicates that the model was able to accurately identify the majority of the attacks in the dataset, with a low false positive rate. The model struggled with the CIC-IDS 2017 dataset, achieving an accuracy rate of 58.4%, with a precision of 12.0%, recall of 98.2%, and F1-score of 21.4%. This indicates that the model was able to identify a significant portion of the attacks in the dataset, but also had a higher false positive rate compared to the Mirai BotNet dataset. The model had difficulty when the attack complexity was high, as it struggled to identify the attacks in the dataset.

4.2 Attack RMSE Distinction

The spikes in the score correspond to the time of the attacks in the dataset, in this case the Mirai BotNet dataset features attacks that start with approximately packet number 120000, another attack at approximately packet number 130000, a subsequent attack at approximately packet number 140000, and a final attack at approximately packet number 180000. Attack packets feature higher RMSE values, indicating that the model was able to identify attack packets with a scaling RMSE score with packet number.

5 Discussion

A primary finding of this study was the performance of the Kitsune model on the Mirai BotNet dataset. The model achieved a high accuracy rate of 90.1%, with a precision of 99.8%, recall of 99.9%, and F1-score of 99.8%. This indicates that the model was able to accurately identify the majority of the attacks in the dataset, with a low false positive rate. However, the model did struggle with the CIC-IDS 2017 dataset, achieving an accuracy rate of 58.4%, with a precision of 12.0%, recall of 98.2%, and F1-score of 10.7%. This indicates that the model was able to identify a significant portion of the attacks in the dataset, but also had a higher false positive rate compared to the Mirai BotNet dataset. The Kitsune model had difficulty when the attack complexity was high. The Mirai BotNet dataset is binary in nature, in that there are two periods of the dataset, one with the attack and one without the attack. This makes it easy for the model to achieve high accuracy as there is only one transistion point between the two periods. The majority of the packet The CIC-IDS 2017 dataset is more complex, as it contains a variety of attack types and network conditions. There also were attacks that contained malformed packets, which caused the simulation to stop running. However, in a production environment, the model would require to filter these packets out, as they would not be able to be processed by the model. Therefore, detailing that the model and/or the feature extraction had trouble running into complex situations, that could be facilitated by either attack structure, hardware vulnerabilities, or collection errors.

5.1 Data Processing Limitations

A large part of the simulation is the data preprocessing and post-processing of the Kitsune model. The pre-processing of the data is necessary to remove any breaking changes in the data, such as when packets are corrupted or malformed. When the data is malformed, the simulation will stop running, often due to an segmentation fault. Therefore, with complex datasets like the CIC-IDS 2017 dataset, there becomes a necessary step in pre-screening the data and remove or fix any packets that are malformed. Post-data processing is also important, as the RMSE data needs to be mapped to the labelled dataset. The PCAP data file, in order to get a ground truth table for any given packet, is processed using the CICFlowMeter tool. The tool is designed to extract flow-based features from network traffic data, allowing for the analysis and monitoring of network traffic. However, this flow-based feature compresses the data significantly. The flow-based features are used to create a used to create a summary of network flows of the network traffic data by packet characteristics or features. This flow table is not directly correlated to an individual packet, but rather a collection of packets that share common characteristics. This individual packet representation is a project in itself. Therefore, an inaccurate mapping of the RMSE data to the labelled dataset was performed, matching an X-tuple of features (source IP, destination IP, source port, destination port, protocol) of the PCAP data to the RMSE output data.

5.2 Project Limitations

The limitations of this paper include the use of a only two datasets for evaluation, the lack of real-time testing, and the limited scope of the analysis. There are many other datasets available for evaluating the performance of NIDS, and the use of only two datasets may not provide a comprehensive evaluation of the performance of the Kitsune model. However, a significant time restraint comes with the simulation, as the simulation is run on a local machine and the datasets are large in size. The limit of two datasets comes in the interest of time and resources, the Kitsune model is ran on my local machine and uses a CPU. The CIC-IDS 2017 dataset is on the approximately 54 million packets, took about three full days to run the simulation, excluding time for preprocessing of the PCAP input data or the post-processing mapping of the RMSE data to the labelled dataset. With dedicated GPU resources and a C-implementation of Kitsune, the simulation could be run in a fraction of the time.

While post-processing of the RMSE data was a significant part of the simulation, especially for the CIC-IDS-2017 dataset, the mapping of the RMSE data to the labelled dataset was not perfect. The flow-based features used in the CICFlowMeter tool are not directly correlated to an individual packet, but rather a collection of packets that share common characteristics. Creating a tool for individual packet labeling would increase the accuracy of the mapping process, and therefore, the accuracy of the performance metrics. Additionally, the analysis of the performance metrics was limited to the overall accuracy, precision, recall, and F1-score of the model. Distinguishing between the different types of attacks in the dataset would yield a more comprehensive evaluation of the performance of the Kitsune model. Furthermore, determining in which part (transistions between attacks, idle periods, start of attack, end of attack) of the attack the model struggled with could provide valuable insights into the limitations of the model and areas for improvement.

The analysis of the CIC-IDS-2017 dataset was limited overall. Furthermore, visualizations of RMSE scores over time, similar to Figure 2, were not generated for the full CIC-IDS 2017 dataset due to its size and the associated processing time. Therefore, attack patterns similar to that of the Mirai BotNet dataset were not able to be visualized further analyzed. A solution to this would be to run the simulation on either a smaller subset for plot visualization, or correlate the simulation to a day-to-day basis. The CIC-IDS-2017 dataset is a attack array spanning four separate days. Therefore, running the simulation on a day-to-day basis would allow for a more comprehensive analysis of whether Kitsune could detect separate attacks.

The Kitsune model is simplistic in its design, and does not include a signature-based detection system, which is a common feature of traditional NIDS. This may limit the model's ability to detect certain types of attacks, as it relies solely on unsupervised learning techniques to identify anomalies. My own personal limitations also come into play, as I am not a machine learning expert and have limited experience with NIDS. The implementation of current state-of-the-art NIDS is a complex task, but future work could include more advanced machine learning techniques, such as deep learning or reinforcement learning, to improve the performance of the model. Additionally, the use of more advanced feature extraction techniques could also improve the performance of the model.

6 Conclusion

This study aimed to provide a brief analysis of the Kitsune Network Intrusion Detection System's performance through simulation on benchmark datasets. By applying Kitsune's autoencoder-based KitNet algorithm to the Mirai BotNet and CIC-IDS 2017 datasets on a local machine, we evaluated its anomaly detection capabilities using standard metrics derived from an RMSE threshold set at the 99th percentile of normal traffic.

The findings indicate a stark contrast in performance between the two datasets. Kitsune demonstrated strong detection capabilities on the Mirai dataset, achieving high Precision (99.9%) and Recall (99.0%), suggesting effectiveness in scenarios with relatively clear distinctions between normal and attack phases. However, its performance on the larger, more complex CIC-IDS 2017 dataset was considerably poorer, marked by extremely low Precision (12.0%) despite high Recall (98.2%), resulting in low overall Accuracy (58.4%) and F1-Score (21.4%). This suggests a high rate of false positives when dealing with the diverse traffic and attack types in CIC-IDS 2017 using the chosen static threshold and parameters.

The simulation also underscored significant practical challenges. Processing large PCAP datasets like CIC-IDS 2017 (54MM packets) with the Python implementation of Kitsune proved computationally intensive, requiring approximately three days on a standard CPU. Furthermore, the evaluation process was complicated by the difficulty of accurately mapping packet-level RMSE anomaly scores generated by Kitsune to the flow-based labels provided with the CIC-IDS 2017 dataset, a limitation impacting the precision of the calculated metrics.

Acknowledged limitations include the use of only two datasets, the offline simulation environment, the potentially inaccurate packet-to-flow mapping, and the fixed anomaly threshold. Future work should focus on evaluating Kitsune with optimized implementations (e.g., C++) or on GPU hardware, exploring adaptive thresholding techniques, developing robust packet-to-flow correlation tools, performing per-attack class analysis, and testing on a wider variety of contemporary datasets and potentially in real-time environments.

In conclusion, while Kitsune's unsupervised, packet-based approach shows promise, particularly on datasets with clear anomaly separation like Mirai, its practical application and evaluation on complex, large-scale datasets reveal significant challenges related to performance, resource requirements, and evaluation methodology that need further research and development.

References

[1] Mohammed Acheddad. *AI-Based Network IDS_ML-DL*. Accessed: 2025-04-30. 2025. URL: https://github.com/mohammedAcheddad/AI-Based-Network-IDS_ML-DL.

[2] Sebastian Garcia et al. *Slips, behavioral machine learning-based Python IPS*. Version X.Y.Z. 2025. URL: https://github.com/stratosphereips/StratosphereLinuxIPS.

[3] Yang Guo. "A review of Machine Learning-based zero-day attack detection: Challenges and future directions". In: *Computer Communications* 198 (2023), pp. 175–185. ISSN: 0140-3664. DOI: https://doi.org/10.1016/j.comcom.2022.11.001. URL: https://www.sciencedirect.com/science/article/pii/S0140366422004248.

[4] Jake Hesford et al. *Expectations Versus Reality: Evaluating Intrusion Detection Systems in Practice*. 2024. arXiv: 2403.17458 [cs.CR]. URL: https://arxiv.org/abs/2403.17458.

[5] Nicholas Koroniotis et al. "Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analysis: Bot-IoT". In: *Future Generation Computer Systems* 100 (2019), pp. 779–796. DOI: 10.1016/j.future.2019.05.041.

[6] Arash Habibi Lashkari et al. "Characterization of Tor Traffic Using Time Based Features". In: *Proceedings of the 3rd International Conference on Information System Security and Privacy (ICISSP 2017)*. Porto, Portugal: SCITEPRESS – Science and Technology Publications, 2017, pp. 253–262. DOI: 10.5220/0006105602530262.

[7] Richard Martin and Martin Roesch. "Snort - Lightweight Intrusion Detection for Networks". In: *Proceedings of the 13th USENIX Conference on System Administration*. USA: USENIX Association, 1999, pp. 229–238.

[8] Alberto Miguel-Diez et al. *A systematic literature review of unsupervised learning algorithms for anomalous traffic detection based on flows*. Mar. 2025. DOI: 10.48550/arXiv.2503.08293.

[9] Yisroel Mirsky et al. "Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection". In: Jan. 2018. DOI: 10.14722/ndss.2018.23211.

[10] Nour Moustafa and Jill Slay. "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)". In: *Military Communications and Information Systems Conference (MilCIS)*. 2015, pp. 1–6. DOI: 10.1109/MilCIS.2015.7348942.

[11] B. Mukherjee, L.T. Heberlein, and K.N. Levitt. "Network intrusion detection". In: *IEEE Network* 8.3 (1994), pp. 26–41. DOI: 10.1109/65.283931.

[12] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization". In: *4th International Conference on Information Systems Security and Privacy (ICISSP)*. SciTePress, 2018, pp. 108–116. DOI: 10.5220/0006639801080116.

[13] Mahbod Tavallaee et al. "A Comparative Study of Signature Generation Techniques for Intrusion Detection". In: *Proceedings of the 5th Annual Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference (CEAS)*. 2008.

[14] Mahbod Tavallaee et al. "A Detailed Analysis of the KDD CUP 99 Data Set". In: *Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*. 2009, pp. 1–6. DOI: 10.1109/CISDA.2009.5356528.

[15] Rahul K. Vigneswaran et al. "Evaluating Shallow and Deep Neural Networks for Network Intrusion Detection Systems in Cyber Security". In: *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. 2018, pp. 1–6. DOI: 10.1109/ICCCNT.2018.8494096.

[16] Li Yang and Abdallah Shami. "IDS-ML: An open source code for Intrusion Detection System development using Machine Learning". In: *Software Impacts* 14 (2022), p. 100446. ISSN: 2665-9638. DOI: https://doi.org/10.1016/j.simpa.2022.100446. URL: https://www.sciencedirect.com/science/article/pii/S2665963822001300.

[17] Ying Zhong et al. "HELAD: A novel network anomaly detection model based on heterogeneous ensemble learning". In: *Computer Networks* 169 (2020), p. 107049. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2019.107049. URL: https://www.sciencedirect.com/science/article/pii/S1389128619304086.