EVALUATING TRAFFIC RESHAPING ATTACKS AGAINST MACHINE-LEARNING-BASED NETWORK INTRUSION DETECTION SYSTEMS

by

Luke Bushur

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science

Department of Computer Science

Fort Wayne, Indiana

August 2026

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Zesheng Chen, Chair

Department of Computer Science

Dr. Mohammadreza Hajiarbabi

Department of Computer Science

Dr. Jonathan Rusert

Department of Computer Science

Approved by:

Dr. Adolfo Coronado

ACKNOWLEDGMENTS

First, thank you to my advisor, Dr. Zesheng Chen. I am profoundly thankful for his unwavering support and guidance. His consistent and timely feedback, extensive expertise, and understanding were crucial to the success of this project. Dr. Chen's confidence in my abilities pushed me through difficult times so that I could see this project through to its completion. I will miss our weekly meetings.

Thank you also to Dr. Mohammadreza Hajiarbabi and Dr. Jonathan Rusert for serving on my thesis committee and for taking the time to read my thesis, listen to my defense, and provide valuable feedback.

Finally, thank you to my family for their unyielding support and steadfast encouragement. Their unending belief in me kept me going through my darkest moments. I could not have completed this project without them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

AD	anomaly detector
ARP	address resolution protocol
AUC	area under the curve
BB	black-box
BIM	basic iterative method
BNB	naive Bayes classifier for multivariate Bernoulli
CNN	convolutional neural network
CSV	comma-separated values
CW	Carlini and Wagner
DER	detection evasion rate
DoS	denial of service
DT	decision tree
FGSM	fast gradient sign method
FN	false negative
FP	false positive
FPR	false positive rate
GAN	generative adversarial network
GB	gray-box
GNN	graph neural network
HAA	hierarchical adversarial attack
IAT	inter-arrival time
IF	isolation forest
IoT	internet of things
IP	internet protocol
JSMA	Jacobian-based saliency map attack
KNN	k-nearest neighbors
LightGBM	light gradient boosting machine
LR	logistic regression

LSTM	long short-term memory
MAC	media access control
MAML	model-agnostic meta-learning
MANDA	manifold and decision boundary-based adversarial example detection scheme
MER	original malicious traffic evasion rate
ML	machine learning
MLP	multilayer perceptron
MMR	malicious features mimicry rate
NIDS	network intrusion detection system
OS	operating system
PCAP	packet capture
PCAPNG	packet capture next generation
PDR	malicious probability decline rate
PSO	particle swarm optimization
ReLU	rectified linear unit
RF	random forest
RMSE	root mean squared error
RNN	recurrent neural network
ROC	receiver operating characteristic
RWR	random walk with restart
SSDP	simple service discovery protocol
SSL	secure sockets layer
SVM	support vector machine
TANTRA	timing-based adversarial network traffic reshaping attack
TN	true negative
TP	true positive
TPR	true positive rate
TSTM	time-stacked state transition model
WB	white-box

ABSTRACT

Network intrusion attacks have become increasingly common in modern computing environments, motivating the use of Network Intrusion Detection Systems (NIDSs) to monitor traffic and detect intrusions. NIDSs based on Machine Learning (ML) have shown strong detection performance, but these systems are vulnerable to adversarial attacks. Prior research on adversarial attacks against NIDSs has often focused on feature-space attacks, where feature representations are perturbed directly. However, such attacks can be impractical because perturbations in the feature space may not translate to valid network traffic. This motivates the study of packet-space adversarial attacks, where malicious packets are modified directly. One practical packet-space method is traffic reshaping, in which observable properties of malicious traffic, such as packet timing, are modified to mimic benign traffic. Recent work such as TANTRA has shown that timing-based reshaping can evade ML-based NIDSs, but these approaches often rely on complex learned models. This raises the question of whether simpler reshaping strategies can achieve comparable results.

This thesis has three primary goals: (1) evaluate whether a simple fixed inter-arrival time (IAT) reshaping attack, referred to as the *Simple* reshaping attack, can approximate the effects of a more complex TANTRA-inspired Long Short-Term Memory (LSTM) reshaping attack, (2) determine whether supervised ML models can detect reshaped malicious traffic, and (3) identify which AfterImage features are most useful for distinguishing benign traffic from malicious traffic, as well as for distinguishing among benign, original malicious, and modified malicious traffic. To address these goals, network traffic packets are converted into AfterImage feature vectors, which are then evaluated by both the Kitsune anomaly detector and supervised ML models. Original malicious traffic, Simple-reshaped traffic, and LSTM-reshaped traffic are compared across several intrusion attacks. The supervised ML experiments evaluate Random Forest, LightGBM, and TabNet models in both binary and multiclass classification settings, with the binary classification experiments including reshaping-unaware and reshaping-aware training configurations.

The results show that the Simple reshaping attack often produces effects comparable to the LSTM-based reshaping attack, substantially reducing Kitsune's ability to detect mali-

cious traffic. In the binary supervised ML setting, models are generally effective when trained and tested on the same traffic condition, but some models fail to generalize across original and reshaped traffic. Models trained on both original and reshaped malicious traffic are the most robust overall, suggesting that reshaping-aware training improves detection performance. In the multiclass setting, supervised ML models can often distinguish among benign, original malicious, and reshaped malicious traffic, but performance varies by intrusion attack and model type. Feature-importance analysis reveals that binary classification relies heavily on host-to-host and directional source features, while multiclass classification uses a broader mixture of feature families, including timing-based jitter and socket-level features. These findings demonstrate that simple timing-based reshaping can pose a meaningful threat to ML-based NIDSs while also showing that reshaping-aware supervised training can improve robustness against such attacks.

1. INTRODUCTION

1.1 Background and Motivation

Network intrusion attacks have become increasingly prevalent in modern computing environments. Attackers conduct targeted attacks on business, government, or private networks to leak confidential information, obtain financial gain, or cause disruption. In these scenarios, attackers must connect to the targeted network to perform the intrusion. For example, in an active wiretap attack, an attacker may intercept traffic on a network and gather sensitive information from the traffic contents.

To defend against such attacks, Network Intrusion Detection Systems (NIDSs) are deployed on networks to monitor network traffic for intrusion attacks. Today, Machine Learning (ML) models serve as an increasingly popular choice for building NIDSs [1]. One widely studied ML-based NIDS is Kitsune [2–5]. Kitsune incorporates a widely-used feature extractor and an Anomaly Detector (AD) composed of an ensemble of autoencoders. The feature extractor, called AfterImage, extracts standardized feature representations from network traffic packets using damped incremental statistics. These feature representations serve as input to the AD, also known as KitNET, which learns to reconstruct benign traffic features and, in turn, distinguish between benign and malicious traffic.

Although ML-based NIDSs exhibit strong performance, they are also vulnerable to adversarial attacks. In these attacks, adversaries make small changes to model inputs to induce changes in model outputs. In the context of NIDSs, attackers modify malicious traffic inputs so that NIDSs misclassify them as benign while preserving the intrusion attack's functionality. Figure 1.1 demonstrates the effect of adversarial attacks on NIDS classification. Prior works on adversarial attacks against NIDSs have often focused on feature-space attacks, where attackers modify extracted feature vectors directly [6–8]. However, such attacks are often impractical because it may be difficult or impossible to make changes to network packets that produce the same perturbed feature vectors [4]. This motivates the study of packet-space or problem-space attacks, where malicious traffic is modified directly before feature extraction.

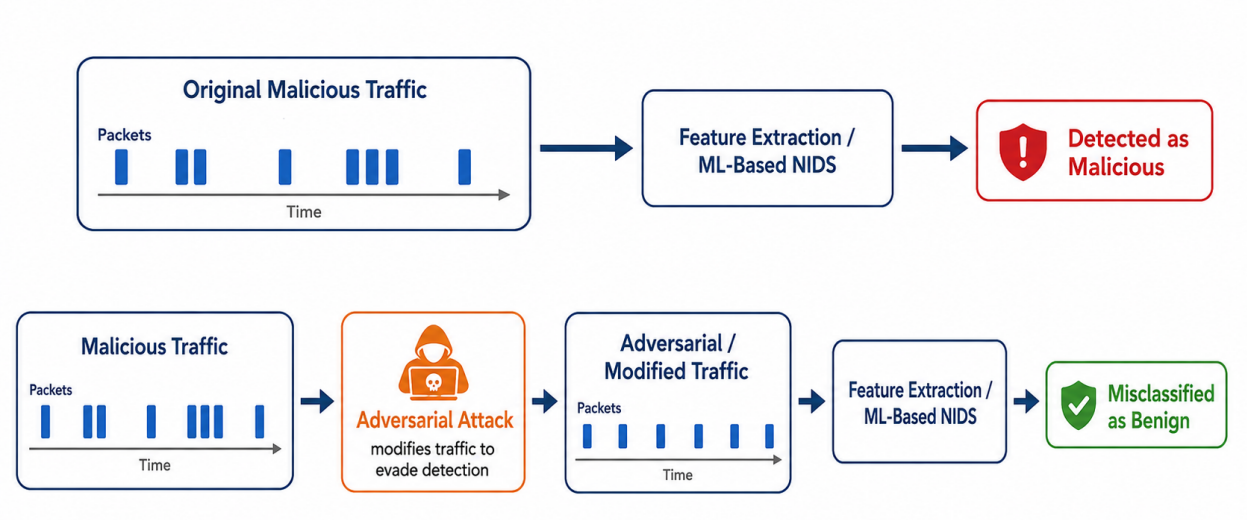

Figure 1.1. Impact of Adversarial Traffic on NIDS Classification. Image Generated by OpenAI's ChatGPT (GPT-5.6 Sol) [9]

One practical form of packet-space evasion that has achieved success against ML-based NIDSs is network traffic reshaping [4, 5]. In these reshaping attacks, observable properties of malicious traffic, such as packet timings or packet sizes, are modified to more closely resemble benign traffic. Timing-based reshaping is especially relevant because many ML-based NIDSs use time-dependent traffic statistics when making detection decisions [2, 10].

Sharon *et al.* [5] present a novel network traffic reshaping attack called the Timing-based Adversarial Network Traffic Reshaping Attack (TANTRA). In this attack, a Long Short-Term Memory (LSTM) model is trained on benign network traffic packets to learn the time differences, or Inter-Arrival Times (IATs), between benign packets. Then, this model is applied to malicious packets, changing the packets' IATs to mimic benign traffic with the goal of evading NIDSs.

TANTRA was shown to be extremely effective in evading ML-based NIDSs. The success of TANTRA and other learning-based reshaping attacks demonstrates the effectiveness of reshaping attacks in evading ML-based NIDSs [4]. However, the complexity of these methods, along with the lack of a simple baseline comparison in TANTRA, raises the question of whether simpler reshaping attacks can achieve similar levels of success. In addition, while

TANTRA evaluates its reshaping attack against ML-based NIDSs, it does not evaluate which features, timing-based or otherwise, the NIDSs rely on when making classification decisions.

This thesis addresses these gaps by introducing a simple network traffic reshaping attack and comparing its effectiveness against ML-based NIDSs to the effectiveness of a more complex, TANTRA-inspired LSTM reshaping attack. Because Kitsune and several related ML-based NIDS studies rely on AfterImage feature representations, this thesis also uses the AfterImage feature extractor to convert network traffic packets into feature vectors for detection and classification. This enables feature-importance analysis to determine which AfterImage features are most useful for distinguishing benign traffic from malicious traffic in a binary classification setting, and which features are most useful for distinguishing among benign traffic, unmodified malicious traffic, and modified malicious traffic in a multiclass classification setting.

1.2 Research Questions and Contributions

The central goals of this thesis are to (1) evaluate whether a simple fixed-IAT traffic reshaping attack can achieve effects comparable to a more complex TANTRA-inspired LSTM reshaping attack, (2) determine whether supervised ML-based NIDSs can detect reshaped malicious traffic, and (3) identify which AfterImage features are most useful for distinguishing among benign traffic, original malicious traffic, and reshaped malicious traffic. These goals break down into the following five research questions:

RQ1: Can a simple fixed-IAT traffic reshaping attack reduce Kitsune's detection performance in a way that is comparable to a more complex TANTRA-inspired LSTM reshaping attack?

RQ2: Can traffic reshaping attacks evade supervised ML models trained only on original malicious traffic?

RQ3: Can supervised ML models trained on both original and reshaped malicious traffic improve detection robustness in the binary classification setting?

RQ4: Can supervised ML models distinguish among benign traffic, original malicious traffic, and reshaped malicious traffic in a multiclass classification setting?

RQ5: Which AfterImage features are most useful for distinguishing among benign traffic, original malicious traffic, and reshaped malicious traffic?

To address these research questions, this thesis develops an experimental framework for evaluating traffic reshaping attacks against ML-based NIDSs. Network traffic packets are converted into AfterImage feature vectors, allowing both Kitsune and supervised ML models to evaluate traffic using the same feature representation. Two reshaping strategies are applied to malicious traffic: a simple fixed-IAT reshaping strategy, hereafter referred to as the *Simple* reshaping attack, and a TANTRA-inspired LSTM reshaping attack. The original, Simple-reshaped, and LSTM-reshaped traffic are then evaluated using both anomaly-based and supervised ML detection models.

The main contributions of this thesis are as follows:

1. **A low-complexity baseline for timing-based traffic reshaping.** This thesis introduces and evaluates the Simple reshaping attack, which modifies malicious packet timings using a fixed inter-arrival time derived from benign traffic. Unlike learned reshaping methods such as TANTRA, this approach does not require training an LSTM model or performing sequence prediction during reshaping. This provides a simple baseline for evaluating whether learned timing models are necessary for effective NIDS evasion.

2. **A direct comparison between Simple and LSTM-based reshaping against Kitsune.** This thesis compares the Simple reshaping attack with a TANTRA-inspired LSTM reshaping attack using the Kitsune anomaly detector. For each intrusion attack, Kitsune is evaluated on original, Simple-reshaped, and LSTM-reshaped malicious traffic using RMSE anomaly scores, threshold-based binary predictions, and standard performance metrics. The results show that both reshaping strategies substantially reduce Kitsune's detection performance, with the Simple strategy often performing comparably to the more complex LSTM-based strategy.

3. **A supervised binary classification evaluation under reshaping-aware and reshaping-unaware training conditions.** This thesis evaluates RF, LightGBM,

and TabNet models trained on original malicious traffic, reshaped malicious traffic, and combined original-plus-reshaped malicious traffic. This evaluation measures whether supervised ML models trained on one traffic condition generalize to another and whether including reshaped malicious examples during training improves robustness. The results show that models trained on both original and reshaped malicious traffic are generally the most robust.

4. **A multiclass evaluation of benign, original malicious, and reshaped malicious traffic.** This thesis extends the supervised ML evaluation beyond binary benign-versus-malicious classification by treating benign traffic, original malicious traffic, and reshaped malicious traffic as separate classes. This setting evaluates whether supervised models can distinguish reshaped malicious traffic as a distinct traffic condition rather than only detecting whether traffic is malicious. The results show that multiclass models can often separate all three classes, although performance depends on the intrusion attack and model type.

5. **A feature-importance analysis of AfterImage features under traffic reshaping.** This thesis analyzes feature importance values from RF, LightGBM, and TabNet models to identify which AfterImage feature families, feature templates, and effective time windows are most useful for classification. The binary analysis shows that host-to-host and directional source features are especially important, while the multiclass analysis relies on a broader mixture of feature families, including timing-based jitter and socket-level features. This analysis provides insight into how supervised models detect original and reshaped malicious traffic.

1.3 Thesis Organization

The rest of this thesis is organized as follows: Chapter 2 reviews background information on NIDS datasets, ML-based NIDSs, adversarial attacks against NIDSs, and defenses against adversarial attacks. Chapter 3 presents the system design of the proposed framework. Chapter 4 discusses the background of and details behind each of the evaluated traffic reshaping

attacks. Chapter 5 presents information about the supervised ML learning experiments, including problem formulations, dataset splitting details, the ML model types evaluated, training configurations, and the feature importance analysis performed. Chapter 6 reports results and analyses from each studied experiment; performances of reshaping attacks are compared, supervised ML model performance under different training configurations and classification settings is evaluated, and the most important AfterImage features are identified. Finally, Chapter 7 concludes the thesis, summarizing the main findings and offering directions for future work.

2. LITERATURE REVIEW

ML has emerged as a powerful approach for developing NIDSs, enabling the identification of both known and novel threats by analyzing patterns in network traffic. Unlike traditional signature-based systems, ML-based NIDSs can generalize from labeled examples and adapt to evolving attack strategies, making them well-suited for the dynamic nature of today's environments. As the field matures, research has increasingly focused on building systems that are not only accurate and scalable but also resilient against a growing spectrum of adversarial threats that seek to evade detection. This literature review explores the key components that underpin ML-based NIDS research: the datasets that support experimentation, the ML models employed for detection, the adversarial attacks designed to exploit model weaknesses, and the defenses developed to counter such threats.

A foundational aspect of any ML-based NIDS is the dataset used to train and evaluate its performance. The quality, diversity, and realism of these datasets directly influence a model's ability to generalize to real-world scenarios. Datasets vary in structure, such as whether they are packet-level or feature-level, and in scope, including traditional networks and Internet of Things (IoT)-specific environments. Benchmark datasets like Kitsune [2], CIC-IDS2017 [11], and NSL-KDD [12] have become widely adopted in the research community for evaluating intrusion detection capabilities. As adversarial ML becomes more prevalent, the need for datasets that reflect complex and evolving attack patterns has become even more pronounced.

Beyond datasets, a wide array of ML techniques has been applied to NIDSs, ranging from classical algorithms like Decision Trees (DTs) and Support Vector Machines (SVMs) to Deep Learning models such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). These techniques differ in their learning paradigms—supervised, unsupervised, semi-supervised, or ensemble-based—and each offers trade-offs in terms of accuracy, computational efficiency, and robustness. However, as these models grow more sophisticated, so too do the adversarial attacks aimed at undermining them. Recent research has demonstrated the feasibility of both feature-space and packet-space adversarial attacks, prompting the development of specialized defenses, including meta-learning approaches and

adversarial example detectors. The sections that follow provide a detailed examination of the datasets, ML techniques, attack strategies, and defense mechanisms that form the foundation of current research in ML-based NIDSs.

2.1 Datasets

Datasets form the foundation of research in ML-based NIDSs. The performance, robustness, and generalizability of a NIDS heavily depend on the quality and diversity of the datasets used for training and evaluation. As these systems are increasingly challenged by adversarial attacks, the choice of dataset becomes even more critical. A comprehensive dataset enables researchers to simulate realistic attack scenarios, benchmark detection capabilities, and evaluate adversarial resilience.

Datasets can be categorized by the network category. There are primarily two network categories: traditional (general) network and IoT network. A traditional network dataset attempts to simulate a network consisting of a wide range of normal devices, including multi-purpose devices such as desktops, laptops and others like printers [13]. On the other hand, an IoT network dataset attempts to simulate an IoT network, where physical objects called Things are connected via sensors or software [13]. Multi-purpose devices like laptops or desktops are not usually included in these networks; rather, many single-purpose devices, like cameras, sensors, and thermostats, are present in the topology. For this reason, IoT network traffic and traditional network traffic exhibit different patterns [13].

In addition to network category, datasets can be categorized by their types. There are primarily two types: packet-level and feature-level. Packet-level datasets contain the raw packets of the simulated network. These datasets usually store the data as one or many packet capture (PCAP) or packet capture next generation (PCAPNG) files. On the other hand, feature-level datasets contain higher-level information about connections or sessions within a packet capture. Features are most often calculated from packets' headers, containing statistics about the overall connection (such as duration), aggregate information derived from packets' discrete characteristics (such as number of bytes sent), and statistical information derived from packets' discrete characteristics (such as mean packet size) [13].

Over the years, numerous datasets have been developed to reflect a wide range of network environments and traffic patterns. Table 2.1 presents a collection of benchmark datasets used in recent NIDS research. Among these, Kitsune, CIC-IDS2017, and NSL-KDD stand out as some of the most utilized datasets in recent literature, particularly in studies exploring adversarial ML defenses [4, 5, 14].

- **Kitsune** [2]: Kitsune is a collection of packet-level datasets specifically designed for IoT security. Released in 2018, it contains nine network attack datasets that capture traffic from two networks: a real video surveillance network and a Wi-Fi network with nine IoT devices and three computers. Each dataset contains traffic with a particular attack, such as fuzzing, Simple Service Discovery Protocol (SSDP) Flood, and Mirai botnet. Over 41 million packets are captured across the entire dataset.

- **CIC-IDS2017** [11]: CIC-IDS2017 is a traditional network dataset introduced by the Canadian Institute for Cybersecurity. It includes both raw packet data in PCAP file format and feature-level data in Comma-Separated Values (CSV) file format resulting from feature extraction using CICFlowMeter. It aims to resemble real-world data and encompasses multiple attack vectors such as distributed denial of service, brute force, infiltration, and web attacks.

- **NSL-KDD** [12]: NSL-KDD is a refined version of the older KDDCUP'99 dataset [15] designed to address its major shortcomings such as redundant records and class imbalance. It is a feature-level dataset consisting of 148 517 connection records, each represented by 41 handcrafted features and labeled as either normal or one of several attack types, including Denial of Service (DoS), Probing, Remote to Local, and User to Root. Despite being somewhat dated, NSL-KDD remains a staple in academic research due to its simplicity, standardized format, and wide adoption in benchmark comparisons.

Table 2.1. Benchmark Datasets Used in Recent NIDS Research

Name	Network Category	Type	Year	Size
KDDCUP'99 [15]	Traditional	Feature-Level	1999	5 209 460 Records
NSL-KDD [12]	Traditional	Feature-Level	2009	148 517 Records
ISCXIDS2012 [16]	Traditional	Packet-Level	2012	84.42 GB
UNSW-NB15 [17]	Traditional	Both	2015	2 540 044 Records
USTC-TFC2016 [18]	Traditional	Packet-Level	2016	3.17 GB Benign, 816 MB Malware
CIC-IDS2017 [11]	Traditional	Both	2017	51.1 GB
Kitsune [2]	IoT	Both	2018	41 670 754 Packets
UNSW-SOSR2019 [19]	IoT	Packet-Level	2019	609 976 Samples
FSIDS-IoT [20]	IoT	Feature-Level	2023	78 Categories, Maximum of 5000 Samples/Category

2.2 ML-Based NIDSs

ML has become a cornerstone of modern NIDSs, offering flexible, scalable, and adaptive mechanisms for identifying malicious activities in network traffic. Unlike traditional signature-based systems that rely on predefined patterns, ML-based NIDSs can generalize from known attack types and potentially detect novel or evolving threats. These systems are particularly critical in the context of adversarial ML, where the ability to adapt and learn from changing attack patterns is essential.

ML techniques used in NIDSs can be categorized into supervised, unsupervised, semi-supervised, and ensemble-based approaches, each offering distinct advantages and trade-offs in terms of accuracy, generalization, and computational overhead. Supervised ML utilizes labeled data to detect intrusions, unsupervised ML uses unlabeled data, and semi-supervised ML utilizes partially labeled data [13]. In addition, ensemble techniques involve combining multiple ML algorithms to achieve better performance. Table 2.2 outlines the primary ML techniques applied in the field, their descriptions, and examples.

Table 2.2. ML Techniques Applied in NIDS Research

Technique	Type	Description	Examples
Artificial Neural Network [1]	Supervised	Network of connected artificial neurons whose weights are adjusted during training using backpropagation to minimize prediction error.	RNN, CNN
Autoencoder [13]	Unsupervised	Feed-forward neural network trained to reconstruct its input by minimizing the difference between the input and the reconstructed output.	Kitsune [2]
CNN [13]	Supervised	Used in tasks involving locally related data. Convolution layers capture local features with filters. Pooling layers abstract these features. Fully connected layers classify abstract features.	Few-Shot-based MAML + CNN [20]
DT [1]	Supervised	Recursively partition the feature space using attribute-based tests to create a tree that classifies instances through a sequence of decision rules.	ID3, C4.5, CART
Deep Neural Network [13]	Supervised	Multi-layer neural network trained to learn a mapping from input features x to output labels y by minimizing a loss function between predicted and ground-truth labels.	
Ensemble Methods [1]	Ensemble	Combine multiple ML algorithms to create a model with better predictive performance.	Boosting, Bagging, Stacking, RF

continued on next page

Table 2.2. *continued*

Technique	Type	Description	Examples
Fuzzy Logic [1]	Supervised	Based on degrees of uncertainty. Permits an instance to belong to multiple classes simultaneously.	
Generative Adversarial Network [4]	Unsupervised	Consists of two parts; the generator learns to generate realistic fake data while the discriminator learns to distinguish the generator's fake data from real data.	GAN + PSO [4]
Genetic Algorithm [1]	Supervised	Optimizes based on principles of evolution. Each possible solution is represented as a series of bits. The quality of the solution improves over time by selection and reproduction operators.	
Hidden Markov Model [1]	Supervised	Models a system as a Markov process with hidden states that produce observable features. After training on known attack behavior, incoming traffic can be scored and classified using a threshold.	
Hierarchical Clustering [1]	Unsupervised	Groups data points by building a hierarchy of clusters, either by progressively merging smaller clusters or by recursively splitting larger clusters.	Agglomerative, Divisive
K-means [1]	Unsupervised	Separate n data objects into k clusters. Each data object is selected in the cluster with the nearest mean.	

continued on next page

Table 2.2. *continued*

Technique	Type	Description	Examples
K-Nearest Neighbors [1]	Supervised	Classifies an unlabeled data sample based on the classes of its k nearest labeled neighbors, using a majority vote to determine the predicted class.	
Naive Bayes [1]	Supervised	Applies Bayes' theorem with a naive conditional independence assumption, classifying traffic based on the likelihood that its features belong to attack or benign classes.	Hidden Naive Bayes
RNN [13]	Supervised	Processes sequential data by maintaining memory of previous inputs, allowing earlier information to influence later predictions.	LSTM, Gated Recurrent Unit, Transformer
Semi-Supervised Learning [1]	Semi-Supervised	Falls between supervised learning and unsupervised learning where the training data is partially labeled.	Expectation Maximization, Self-Training, Co-Training
SVM [1]	Supervised	Use a kernel function to map training data into a higher-dimensional space to classify the intrusion linearly.	Kernels: Linear, Polynomial, Gaussian Radial Basis Function, Hyperbolic Tangent

In addition to standalone NIDSs, these techniques can also be seen in adversarial attacks against NIDSs and defenses against these attacks. TANTRA [5] uses an LSTM model, trained to learn time differences between benign packets, to modify traffic and avoid detection. Han *et al.* [4] utilize a Generative Adversarial Network (GAN) for generating adversarial features. In particular, the generator transforms a malicious feature into adversarial features that look benign while the discriminator distinguishes between adversarial features and benign features [4]. Lu *et al.* [20] present a few-shot-based Model-Agnostic Meta-Learning (MAML) + CNN approach to defending against adversarial attacks where they employ a CNN to classify network flow data. As a final example, Wang *et al.* [14] present their defense called the MANifold and Decision boundary-based Adversarial example detection scheme (MANDA) and evaluate it on several ML models, including MultiLayer Perceptron (MLP), Logistic Regression (LR), K-Nearest Neighbors (KNN), Naive Bayes classifier for multivariate Bernoulli (BNB), DT, SVM and CNN.

2.3 Adversarial Attacks Against NIDSs

As ML-based NIDSs have become increasingly sophisticated, so too have the methods for subverting them. Adversarial attacks are a particularly insidious form of evasion, in which an attacker deliberately modifies malicious inputs to appear benign to a detection system. In the context of NIDSs, such attacks may alter packet timing, traffic features, or payload structures, all while preserving the functional behavior of the original malicious activity. These attacks can operate under different knowledge assumptions—black-box (BB), gray-box (GB), or white-box (WB)—depending on what the attacker knows about the model or feature extractor.

Adversarial attacks can also be classified according to the level at which the attack is applied. In particular, these attacks can be categorized as feature-space attacks or packet-space attacks. Feature-space attacks directly perturb the features resulting from the feature extractor rather than the network traffic. While this type of attack can be effective, it lacks practicality; because network traffic data is structured, adding a small perturbation to one feature may translate to invalid packets that violate traffic constraints [4]. In contrast,

packet-space attacks directly perturb network packets rather than network features. This type of attack is more practical, as generated traffic can be guaranteed to be valid and replayable.

Table 2.3 presents a selection of recent, state-of-the-art adversarial attacks targeting ML-based NIDSs. TANTRA evades NIDSs by using an LSTM to generate time differences between benign network packets and then reshaping malicious packets' IATs to mimic benign traffic [5]. Han *et al.* [4] propose a GB and BB attack method where they train a GAN to generate adversarial features and employ Particle Swarm Optimization (PSO) to mutate malicious traffic while maintaining the attack's maliciousness. Li *et al.* [21] propose a defense against network traffic analyzers and website fingerprinting attacks called Prism, which fingerprints each class using a Time-stacked State Transition Model (TSTM) and perturbs traffic in real time to cause analyzers to misclassify traffic. Zhou *et al.* [8] propose a novel Hierarchical Adversarial Attack (HAA) against Graph Neural Network (GNN)-based intrusion detection. This attack targets a set of vulnerable nodes in the network found using Random Walk with Restart (RWR), identifies critical feature elements, and modifies these elements with minimal perturbations.

Comparing attacks against each other can be challenging due to varying metrics, baseline attacks, baseline defenses, and other factors. Despite these differences, a comparative analysis can still offer valuable insight into the strengths, weaknesses, and real-world applicability of specific techniques. For example, in the work by Sharon *et al.* [5], the authors directly compare their method against Han *et al.*'s method [4]. Specifically, they compare the attack detection evasion rate when using TANTRA against the evasion rate when using Han *et al.*'s GAN + PSO method. They find that, against the fuzzing, brute force, Mirai (botnet), and SSDP flood attacks, TANTRA achieves greater than 0.99 evasion rate whereas the GAN + PSO method achieves less than 0.97 evasion rate on two attacks and less than 0.78 evasion rate on the rest. Compared to TANTRA, GAN + PSO, and Prism [21], the HAA presented by Zhou *et al.* [8] is the only one that is not packet-level, reducing its practicality compared to the rest. Also, all the attacks seem to consider similar mutation methods when perturbing packets and features, with packet IAT being a prominent perturbation target.

Table 2.3. State-of-the-Art Adversarial Attacks Against ML-Based NIDSs

Name	Datasets	Defenses Featured	Attack Type	Model/ Methods	Mutation Methods	Knowledge	Metrics
TANTRA [5]	Kitsune, CIC-IDS2017	Autoencoder, KitNET, IF	Packet-Space	LSTM	Timing-based reshaping attack	BB	Detection Rate, Anomaly Score
GAN + PSO [4]	Kitsune, CIC-IDS2017	KitNET, MLP, LR, DT, SVM, IF	Packet-Space	GAN, PSO	IAT shift; crafted-packet IAT/layers/size	GB, BB	MER, DER, PDR, MMR, Precision, Recall, F1 score
Prism [21]	ISCX2016, USTC-TFC2016	FS-Net, DeepCorr, WF-LSTM, MBTree, RBRN, FC-Net, RF, FlowPrint	Packet-Space	Power-law division, TSTM, packet-block level reconstruction	Packet size, packet intervals	BB	Accuracy, Precision, Recall, F1 score
HAA [8]	UNSW-SOSR2019	GCN, JK-Net	Feature-Space	Shadow GNN, RWR for node selection, generate adversarial examples based on saliency maps	Add perturbations to critical features	BB	Saliency Maps, Loss, Precision

2.4 Defenses Against Adversarial Attacks

As adversarial attacks against ML-based NIDSs have grown in sophistication, so too has the need for robust defense mechanisms capable of detecting or mitigating such attacks. These defenses must contend with a wide range of attack types and operate under varying knowledge assumptions about the attacker (e.g., WB vs. BB). A well-rounded defense strategy must therefore strike a balance between accuracy, adaptability, and generalizability to novel attack scenarios.

Table 2.4 summarizes several recent approaches to adversarial defense across different datasets, attack types, and evaluation settings. The methods presented vary significantly in terms of design philosophy—from manifold-based anomaly detection to meta-learning.

Table 2.4. Recent Approaches to Adversarial Defenses

Name	Datasets	Attacks Featured	Knowledge	Attack Types	Models	Methods	Metrics
MANDA [14]	NSL-KDD, CIC-IDS2017	CW, FGSM, BIM, JSMA	WB	Feature-Space	MLP, LR, KNN, BNB, DT, SVM, CNN	Manifold Learning + Decision Boundary	TPR, FPR, AUC-ROC
Few-Shot-based MAML + CNN [20]	FSIDS-IoT	None	N/A	N/A	MAML + CNN	Few-shot learning with MAML + CNN	Accuracy

MANDA exploits the properties of adversarial attacks and combines manifold and decision boundary components to achieve effective adversarial example detection [14]. Lu *et al.* [20] propose a few-shot-based intrusion detection method that employs a CNN to classify network flow data and MAML to obtain an optimal classifier.

While these defense mechanisms demonstrate promising results against a variety of adversarial attacks, they primarily focus on feature-space perturbations or general adversarial examples. To the best of our knowledge, there has been little to no follow-up work specif-

ically investigating defenses against TANTRA-style timing-based traffic reshaping attacks. Although the original TANTRA study briefly introduced a mitigation strategy, the broader research community has yet to establish effective defenses tailored to packet-space timing reshaping attacks. This gap highlights the need for methods that can maintain detection performance in the presence of adversarial traffic reshaping and motivates the focus of the present work.

3. SYSTEM DESIGN

The proposed framework evaluates the impact of network traffic reshaping attacks on both anomaly-based and supervised ML-based NIDSs. The system consists of six primary components: (1) network datasets, (2) data loading and preprocessing, (3) anomaly detection using Kitsune, (4) traffic reshaping attack generation, (5) supervised learning-based classification, and (6) performance evaluation. Figure 3.1 illustrates the overall architecture.

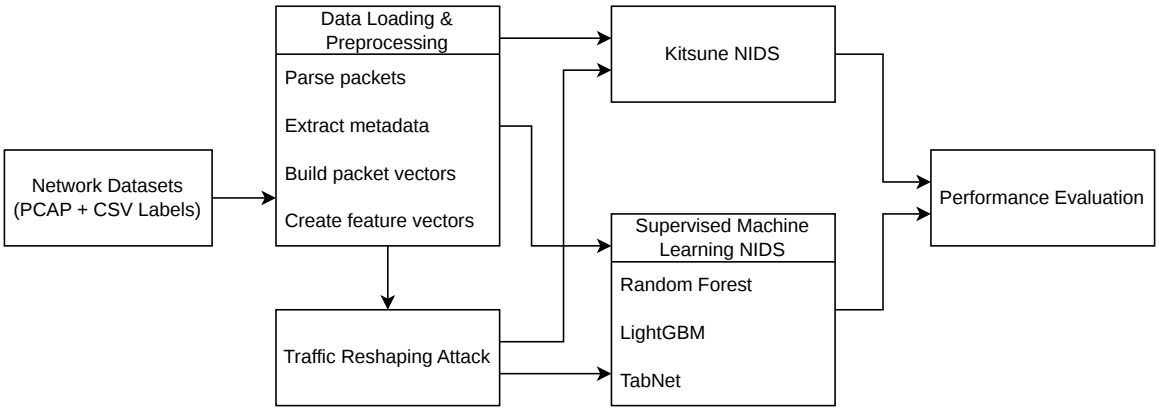

Figure 3.1. System Overview

The workflow begins by loading packet capture files and corresponding packet-level ground-truth labels. Packets are converted into a standardized feature representation and supplied to Kitsune for anomaly detection and supervised ML models for classification. Next, malicious traffic is modified using traffic reshaping techniques that alter packet timing characteristics while preserving packet contents. Then, the modified malicious traffic is converted into the same standardized feature representation used before and supplied to Kitsune for anomaly detection and supervised ML models for classification. Finally, the Kitsune anomaly scores and supervised ML class predictions are passed to the performance evaluation module to assess the effectiveness of the attack and defensive strategies.

3.1 Network Datasets Module

The first component and input to the system is an NIDS dataset consisting of two parts: a PCAP or PCAPNG file of the network traffic containing a malicious attack and a CSV

file containing packet-level labels for whether a packet is benign (not part of an attack) or malicious (part of an attack).

Experimentally, this work utilizes the Kitsune dataset [2], which consists of nine network traffic recordings collected from a video surveillance network under both normal operation and active cyber attacks. The dataset contains the following attack scenarios: active wiretap, Address Resolution Protocol (ARP) man-in-the-middle, fuzzing, Mirai, Operating System (OS) scan, SSDP flood, Secure Sockets Layer (SSL) renegotiation, SYN Denial of Service (DoS), and video injection.

Each recording is provided as a PCAP or PCAPNG packet capture file accompanied by a CSV file containing packet-level ground-truth labels indicating whether each packet is benign or malicious. This packet-level labeling enables both anomaly-based and supervised learning-based intrusion detection experiments to be performed on the same traffic traces.

The duration of the traffic recordings ranges from 28.15 to 118.9 minutes, with an average duration of 63.7 minutes. The number of packets per recording ranges from 764 137 to 6 084 492, with an average of 3 018 973 packets.

The Kitsune dataset is particularly well suited for evaluating traffic reshaping attacks because it was originally developed alongside the Kitsune NIDS and contains realistic packet-level traffic traces that can be processed using the complete Kitsune pipeline. Furthermore, Kitsune's AfterImage feature extractor derives numerous temporal and flow-based statistics from packet timestamps and communication patterns. Since traffic reshaping attacks primarily manipulate packet timing behavior while preserving packet contents, the dataset provides an appropriate environment for studying the effectiveness of such attacks and identifying which features are most useful for distinguishing benign traffic from malicious traffic.

In this work, experiments focus primarily on the active wiretap, fuzzing, Mirai, and SSDP flood attack scenarios. These attacks exhibit diverse traffic characteristics and provide representative examples for evaluating both anomaly-based and supervised ML detection approaches. In addition, prior traffic reshaping attacks have demonstrated effective evasion against these Kitsune attack scenarios, motivating their use in this study [4, 5].

3.2 Data Loading and Preprocessing Module

3.2.1 Packet Parsing

The second component of the framework loads network traffic recordings and their corresponding ground-truth labels from the Kitsune dataset. Each dataset consists of a packet capture file containing the recorded network traffic and a CSV file containing packet-level labels that identify whether each packet is benign or malicious.

During packet parsing, packets are extracted from the traffic recording while preserving their original ordering and timing information. Simultaneously, the corresponding labels are loaded and associated with the appropriate packets. The output of this stage is a synchronized packet stream in which each packet is paired with its ground-truth classification. This representation provides a common foundation for both anomaly-based and supervised learning experiments.

3.2.2 Metadata Extraction

Following packet parsing, relevant metadata is extracted from each packet. The extracted information includes network-layer and transport-layer addressing information, packet size, and packet timing information. These attributes describe the communication endpoints involved in each transmission as well as the temporal characteristics of the observed traffic.

The extracted metadata serves two primary purposes. First, it provides the information required by Kitsune's feature extraction process to characterize network behavior. Second, it enables traffic to be organized into communication flows that can be individually analyzed and modified during traffic reshaping attacks. In particular, packet timestamps are essential for computing IATs and other temporal statistics that are frequently used by NIDSs.

3.2.3 Packet Vector Construction

Following metadata extraction, each packet is converted into a standardized packet-vector representation that serves as the common data format throughout the framework.

The packet vector preserves all information required for Kitsune's feature extraction process while providing an efficient representation for flow analysis and traffic reshaping operations.

Each packet vector contains the following fields:

1. Internet Protocol (IP) type (IPv4 or IPv6)
2. Source Media Access Control (MAC) address
3. Destination MAC address
4. Source IP address
5. Source transport-layer port or protocol
6. Destination IP address
7. Destination transport-layer port or protocol
8. Packet size (bytes)
9. Packet timestamp

The addressing and protocol fields identify the communication endpoints associated with the packet and are used for flow construction and traffic analysis. The packet size and timestamp fields capture the spatial and temporal characteristics of the traffic and are subsequently used by Kitsune's AfterImage feature extractor to generate statistical traffic features.

Representing packets in this form provides two primary advantages. First, it enables efficient access to packet attributes without repeatedly processing the original packet capture files. Second, it allows traffic reshaping attacks to selectively modify packet timestamps while preserving all other packet characteristics. Consequently, malicious traffic can be altered to mimic benign timing behavior without modifying packet contents or communication patterns.

The resulting packet vectors are supplied to the feature generation stage, where Kitsune's AfterImage feature extractor transforms packet-level observations into statistical feature vectors suitable for anomaly detection and supervised ML.

3.2.4 Feature Vector Generation

After packet vectors have been constructed, they are supplied to Kitsune's AfterImage [2] feature extraction framework, which converts individual packet observations into statistical feature vectors describing recent network behavior. Rather than relying solely on the attributes of the current packet, AfterImage maintains a collection of incremental statistics that summarize communication patterns observed over multiple time scales.

The feature extraction process is based on damped incremental statistics. Unlike a traditional sliding window, which stores a fixed number of previous observations, AfterImage applies exponential decay to older observations so that recent traffic contributes more heavily to the computed statistics. The decay applied to each observation is controlled by the parameter λ according to

$$d_\lambda(t) = 2^{-\lambda t} \tag{3.1}$$

where t represents the elapsed time since the previous observation and λ controls the rate at which older observations lose influence. Larger values of λ correspond to shorter effective time windows that emphasize recent activity, while smaller values of λ correspond to longer time windows that capture long-term communication patterns.

To characterize network behavior across multiple temporal scales, AfterImage computes statistics using five decay factors: $\lambda \in \{5, 3, 1, 0.1, 0.01\}$ which correspond approximately to historical windows of 100ms, 500ms, 1.5s, 10s, and 60s, respectively. For each value of λ, AfterImage computes the same set of twenty statistical features. Table 3.1 describes the twenty unique feature templates produced for each time window.

Table 3.1. AfterImage Feature Templates Computed for Each Time Window

Feature Template	Description
`MI_dir_λ_weight`	Exponentially decayed count or weight of recent packets associated with the same source MAC address and source IP address.
`MI_dir_λ_mean`	Exponentially decayed mean packet size for traffic associated with the same source MAC address and source IP address.
`MI_dir_λ_std`	Exponentially decayed standard deviation of packet size for traffic associated with the same source MAC address and source IP address.
`HH_λ_weight`	Exponentially decayed count or weight of recent packets observed on a host-to-host IP channel.
`HH_λ_mean`	Exponentially decayed mean packet size observed on a host-to-host IP channel.
`HH_λ_std`	Exponentially decayed standard deviation of packet size observed on a host-to-host IP channel.
`HH_λ_radius_0_1`	Joint variability of paired host-to-host traffic streams, computed from the decayed variances of the two stream directions.
`HH_λ_magnitude_0_1`	Joint traffic-size magnitude of paired host-to-host traffic streams, computed from the decayed means of the two stream directions.
`HH_λ_covariance_0_1`	Exponentially decayed covariance between the packet-size behavior of paired host-to-host traffic streams.

continued on next page

Table 3.1. *continued*

Feature Template	Description
HH_λ_pcc_0_1	Exponentially decayed Pearson correlation coefficient between the packet-size behavior of paired host-to-host traffic streams.
HH_jit_λ_weight	Exponentially decayed count or weight of recent packets on a host-to-host channel when modeling IAT behavior.
HH_jit_λ_mean	Exponentially decayed mean IAT for packets on a host-to-host channel.
HH_jit_λ_std	Exponentially decayed standard deviation of IAT for packets on a host-to-host channel.
HpHp_λ_weight	Exponentially decayed count or weight of recent packets observed on a host/port-to-host/port channel.
HpHp_λ_mean	Exponentially decayed mean packet size observed on a host/port-to-host/port channel.
HpHp_λ_std	Exponentially decayed standard deviation of packet size observed on a host/port-to-host/port channel.
HpHp_λ_radius_0_1	Joint variability of paired host/port-to-host/port traffic streams, computed from the decayed variances of the paired streams.
HpHp_λ_magnitude_0_1	Joint traffic-size magnitude of paired host/port-to-host/port traffic streams, computed from the decayed means of the paired streams.
HpHp_λ_covariance_0_1	Exponentially decayed covariance between the packet-size behavior of paired host/port-to-host/port traffic streams.

continued on next page

Table 3.1. *continued*

Feature Template	Description
`HpHp_λ_pcc_0_1`	Exponentially decayed Pearson correlation coefficient between the packet-size behavior of paired host/port-to-host/port traffic streams.

In Table 3.1, λ represents one of the five decay factors $5, 3, 1, 0.1, 0.01$. Therefore, each of the twenty feature templates is instantiated once for each time window, producing a total feature vector containing one hundred features.

The prefix of each AfterImage feature template identifies the feature family, which describes the type of network relationship being summarized. Each AfterImage feature vector is composed of four feature families: `MI_dir`, `HH`, `HH_jit`, and `HpHp`. The `MI_dir` family describes directional traffic associated with a source MAC address and source IP address. The `HH` family describes host-to-host communication between source and destination IP addresses. The `HH_jit` family describes host-to-host timing behavior using IAT statistics rather than packet-size statistics. Finally, the `HpHp` family describes communication between source and destination host/port pairs, capturing more specific socket-level behavior.

The suffix `_0_1` indicates that the statistic is computed over the two paired directions or dimensions of a communication relationship. For example, radius, magnitude, covariance, and Pearson correlation coefficient features describe relationships between paired traffic streams rather than a single unidirectional statistic. This multi-scale representation allows Kitsune to simultaneously capture short-term traffic bursts, medium-term communication behavior, and long-term network trends. Consequently, both rapid attack activity and slow behavioral deviations can be reflected in the generated feature vectors.

The resulting 100-dimensional feature vectors form the input to both the Kitsune AD and the supervised ML models evaluated in this work.

3.3 Kitsune-Based Anomaly Detection Module

The third component performs anomaly detection using the Kitsune AD, also known as KitNET [2]. Whereas the previous stage generates 100-dimensional feature vectors that describe recent network behavior, the Kitsune AD is responsible for determining whether those observations conform to previously learned benign traffic patterns. The Kitsune AD operates in an online and unsupervised manner, requiring only benign traffic during training and no attack labels during model construction.

Unlike conventional ADs that rely on a single neural network, the Kitsune AD employs an ensemble of autoencoders organized into two layers. The first layer consists of multiple small autoencoders, each responsible for reconstructing a subset of the input feature vector. The second layer contains a single output autoencoder that receives the reconstruction errors produced by the ensemble layer and generates a final anomaly score. This architecture reduces computational complexity while preserving the ability to model complex relationships among network traffic features.

The anomaly detection process consists of three stages: feature mapping, AD training, and execution.

3.3.1 Feature Mapping Phase

During the feature mapping grace period, the Kitsune AD analyzes correlations among the generated features and automatically partitions the feature space into groups of related features. Each group is assigned to an individual autoencoder in the ensemble layer. Rather than training a single large autoencoder on all 100 features simultaneously, the feature mapping process decomposes the feature space into smaller subsets that can be modeled more efficiently.

The size of each feature subset is bounded by the maximum autoencoder size parameter, denoted as $maxAE$. In this work, $maxAE$ is set to 10, meaning that no individual autoencoder receives more than ten input features. This constraint reduces training and inference costs while enabling deployment on resource-constrained systems.

3.3.2 Anomaly Detector Training Phase

Following feature mapping, the Kitsune AD enters the AD grace period. During this phase, the ensemble of autoencoders is trained online using benign network traffic. Each autoencoder learns to reconstruct the normal patterns associated with its assigned feature subset. The reconstruction errors generated by the ensemble layer are subsequently used to train the output autoencoder.

As packets arrive, each feature vector is processed once and immediately discarded after updating the model. Consequently, the Kitsune AD maintains its online learning properties and does not require storage of historical traffic. At the conclusion of the AD grace period, the model represents the baseline behavior of the monitored network.

3.3.3 Execution Phase

After training is complete, the Kitsune AD transitions to execution mode. Incoming feature vectors are processed by the trained ensemble without further updating the model parameters. For each packet, the ensemble layer first attempts to reconstruct the corresponding feature vector. The reconstruction errors from the ensemble are then supplied to the output autoencoder, which produces a final RMSE reconstruction score.

The RMSE score quantifies the degree to which the observed traffic differs from the benign behavior learned during training. Small RMSE values indicate that the traffic closely resembles previously observed benign patterns, while larger RMSE values indicate increasing deviation from expected network behavior. Consequently, packets associated with high RMSE values are more likely to correspond to anomalous or malicious activity.

The final output of the anomaly detection module is a sequence of RMSE anomaly scores generated during the execution phase, where each score corresponds to an individual packet and represents its deviation from the learned model of benign network behavior.

3.4 Traffic Reshaping Attack Module

The fourth component modifies the timing characteristics of malicious network traffic in order to evaluate the robustness of intrusion detection systems against traffic reshaping attacks. Rather than altering packet contents, protocol information, or communication endpoints, this module operates exclusively on packet timestamps. Consequently, the underlying attack behavior remains unchanged while the temporal characteristics of the traffic are modified.

Prior to reshaping, the framework partitions network traffic into communication flows and identifies a target malicious flow for modification. The selected flow is isolated from the remaining traffic, while the benign traffic used for evaluation remains unchanged. This design allows the effect of timestamp manipulation to be evaluated independently of other traffic characteristics.

After isolation, the packet timestamps associated with the target malicious flow are modified according to a selected reshaping strategy. The modified timestamps define a new sequence of IATs intended to alter the temporal behavior of the flow. Throughout this process, packet ordering, packet sizes, protocol information, source and destination addresses, and packet payloads remain unchanged.

The reshaping process consists of the following stages:

1. Identify and isolate a target malicious flow.
2. Obtain a benign timing profile from reference traffic.
3. Generate modified packet timestamps according to the selected reshaping strategy.
4. Replace the original timestamps of packets in the target flow.
5. Produce a reshaped malicious flow for subsequent evaluation.

The output of this module is a modified version of the target malicious flow whose communication behavior remains functionally identical to the original attack, but whose timing characteristics differ from the original traffic. The reshaped flow is subsequently supplied

to the anomaly detection and supervised learning modules to evaluate the effectiveness of timing-based evasion techniques. For more information on traffic reshaping and the reshaping strategies evaluated in this thesis, see Chapter 4.

3.5 Supervised Learning Classification Module

The fifth component evaluates the impact of original malicious or reshaped malicious traffic on supervised ML classifiers. Whereas Kitsune performs anomaly detection in an unsupervised manner, this module learns explicit decision boundaries between benign and malicious traffic using labeled feature vectors generated by the AfterImage feature extraction process.

The input to the module consists of the 100-dimensional feature vectors described previously together with their corresponding packet labels. To support model training, validation, and testing, the dataset is partitioned into separate training, validation, and testing subsets. These partitions are constructed independently for benign and malicious traffic before being combined to form the final datasets used during experimentation.

The framework supports three training configurations:

- **Original Training**: Models are trained using feature vectors generated from the original malicious traffic.
- **Reshaped Training**: Models are trained using feature vectors generated from reshaped malicious traffic.
- **Combined Training**: Models are trained using feature vectors generated from both original and reshaped malicious traffic.

Three supervised learning algorithms are currently supported:

- Random Forest
- LightGBM
- TabNet

After training, each classifier is evaluated against both original and reshaped traffic. Testing on original traffic measures baseline detection performance, while testing on reshaped traffic evaluates the robustness of the classifier to timing-based evasion attacks. Comparing results across the three training configurations further enables the effectiveness of retraining-based defenses to be assessed.

In addition to packet-level predictions, the module generates feature importance information that can be used to identify which AfterImage features contribute most strongly to classification decisions. This capability supports the investigation of which network traffic characteristics are most useful for distinguishing between benign and malicious behavior.

The outputs of the supervised learning module consist of packet-level class predictions and feature importance measurements. These outputs are subsequently supplied to the performance evaluation component for quantitative analysis and comparison. For more details on supervised learning and on the ML models used in this thesis, see Chapter 5.

3.6 Performance Evaluation Module

The sixth and final component of the framework evaluates the effectiveness of both the traffic reshaping attacks and the corresponding defensive strategies. This module receives anomaly detection outputs from Kitsune and class predictions from the supervised learning models and compares them against the ground-truth packet labels provided by the dataset.

The evaluation process transforms raw model outputs into a collection of quantitative performance measures that characterize detection effectiveness. These measures include TPR, FPR, accuracy, and F1 score. For the multiclass classification experiments, the module also uses 3×3 confusion matrices and per-class one-versus-rest metrics for original malicious and modified malicious traffic. These metrics, discussed in greater detail in Chapter 6, are used to assess how traffic reshaping attacks influence intrusion detection performance and whether reshaping-aware training configurations can improve robustness against reshaped traffic.

To support comprehensive analysis, the module evaluates multiple experimental configurations, including anomaly-based detection using Kitsune, supervised learning models

trained on original traffic, supervised learning models trained on reshaped traffic, and supervised learning models trained on combined original and reshaped traffic. Results are generated for both original and reshaped traffic, enabling direct comparison of detector performance before and after traffic modification.

In addition to prediction-based evaluation, the module processes feature importance information generated by the supervised learning classifiers. These measurements provide insight into which AfterImage features contribute most strongly to distinguishing benign traffic from malicious traffic and support the analysis of how traffic reshaping affects the features relied upon by different models.

The outputs of the performance evaluation module include summary tables, comparative performance reports, confusion matrices, and feature importance rankings. These outputs provide the basis for the experimental analysis presented in Chapter 6.

4. TRAFFIC RESHAPING ATTACKS

4.1 Background and Motivation

ML-based NIDSs rely on statistical patterns extracted from observed network traffic to distinguish benign behavior from malicious activity. While many attacks against ML systems focus on modifying the features used by a classifier, network traffic introduces a unique challenge because many important features are derived from packet timing behavior rather than packet contents.

Traffic reshaping attacks exploit this characteristic by modifying packet timestamps while preserving packet contents, communication endpoints, and attack functionality. As a result, the underlying malicious behavior remains unchanged, but the resulting traffic may appear significantly different to an ML-based detector. Because these attacks operate at the timing level, they can influence the statistical features generated by an NIDS without requiring modification of packet payloads or protocol semantics. Figure 4.1 demonstrates the effect of traffic reshaping attacks on packet timing.

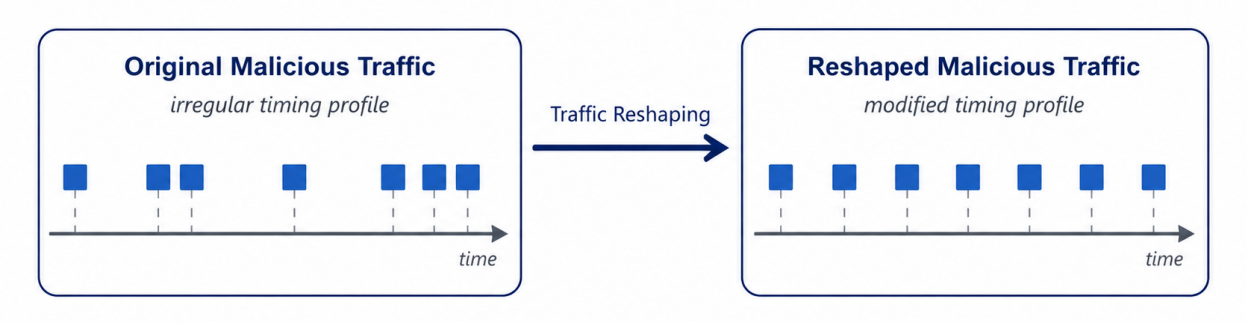

Figure 4.1. Effect of Traffic Reshaping on Packet Timing. Image Generated by OpenAI's ChatGPT (GPT-5.6 Sol) [9]

Kitsune is particularly susceptible to timing-based manipulation because its AfterImage feature extraction framework derives numerous statistics from packet timing behavior. These include communication-rate statistics, jitter measurements, traffic magnitudes, variances, and covariance measures computed across multiple temporal windows. Consequently, modifying packet IATs can substantially alter the resulting feature vectors supplied to the anomaly detection model, even though the attack traffic itself remains unchanged.

Sharon *et al.* [5] introduced TANTRA, a timing-based adversarial attack against ML-based NIDSs that leverages an LSTM network to generate benign-like timing behavior for malicious traffic. Rather than modifying packet contents, TANTRA learns temporal patterns from benign traffic and applies these patterns to malicious flows. Their results demonstrate that timing manipulation alone can significantly reduce the effectiveness of ML-based intrusion detection systems.

While TANTRA demonstrates the feasibility of timing-based evasion, it relies on a relatively sophisticated sequence-learning model. The extent to which successful evasion depends on such complexity remains unclear. Since Kitsune derives many features from aggregate timing behavior, it is possible that simpler timing transformations may achieve similar effects. This observation motivates the primary objective of this chapter: to compare an LSTM-based traffic reshaping attack inspired by TANTRA with a considerably simpler fixed IAT reshaping attack.

4.2 LSTM-Based Traffic Reshaping Attack

The first reshaping strategy evaluated in this work is inspired by TANTRA, a timing-based adversarial attack that uses an LSTM neural network to learn benign packet timing behavior and apply that behavior to malicious traffic. The goal of the attack is to modify the IATs of malicious packets so that the resulting traffic appears more similar to benign communication while preserving the original packet contents and attack functionality.

In this work, the LSTM-based attack is implemented as an offline reshaping procedure. Rather than interacting with a live target network in real time, the model is trained on benign traffic from the dataset and then used to generate modified IATs for an isolated malicious flow. This allows the reshaped malicious flow to be evaluated directly against Kitsune and the supervised learning classifiers.

4.2.1 Training Data Preparation

Before training the LSTM model, packets are grouped into communication channels. A benign reference channel is selected, and the packets belonging to that channel are used to

construct the training data. For each packet in the selected benign channel, two values are retained: the packet size and the packet timestamp.

The timestamp sequence is converted into IATs by subtracting the timestamp of the previous packet from the timestamp of the current packet. This transformation allows the model to learn packet spacing behavior rather than absolute packet arrival times. Packet sizes are normalized by dividing each packet size by the maximum packet size observed across the training channels. The IAT values are also scaled during training to improve numerical stability.

A sliding-window approach is used to generate training examples. In this implementation, the window size is set to 50. Each training example consists of the previous 50 normalized packet sizes and the previous 50 IATs from the benign reference channel. The target output is the next IAT in the sequence. Conceptually, the model learns the mapping

$$(\Delta t_{\mathrm{i}-50}, \ldots, \Delta t_{\mathrm{i}-1}, s_{\mathrm{i}-50}, \ldots, s_{\mathrm{i}-1}) \rightarrow \Delta t_{\mathrm{i}}, \tag{4.1}$$

where Δt_{i} is the IAT of packet i and s_{i} is the normalized packet size of packet i.

4.2.2 Model Architecture

The implemented LSTM model receives two sequence inputs: one sequence containing IATs and one sequence containing normalized packet sizes. Each input has length 50. These two inputs are concatenated along the feature dimension so that each time step contains both timing and size information.

The model architecture is as follows:

- Input 1: sequence of 50 IATs
- Input 2: sequence of 50 normalized packet sizes
- Concatenation layer combining timing and size inputs
- LSTM layer with 32 hidden units and sequence output enabled
- LSTM layer with 16 hidden units

- Dense layer with 8 neurons and Rectified Linear Unit (ReLU) activation
- Dense output layer with 1 neuron and ReLU activation

The output of the network is a predicted IAT for the next packet. The use of recurrent layers allows the model to incorporate recent timing history when generating each prediction. The final ReLU output constrains the model toward nonnegative timing predictions, which is appropriate because IATs cannot be negative.

The model is trained using the Adam optimizer with mean squared error as the loss function. During training, a validation split is used to monitor generalization, and the model with the lowest validation loss is saved. Early stopping is also applied to terminate training when validation performance stops improving. After training, the best saved model is loaded and used to generate reshaped malicious traffic.

4.2.3 Generating Malicious Packet Timings

After training, the LSTM model is applied to the isolated malicious flow. The packet sizes of the malicious packets are normalized using the same maximum packet size value computed during training. These normalized packet sizes are then used as one of the model inputs during prediction.

Because the model requires a fixed-length timing history before it can generate predictions, the prediction process is initialized with a sequence of 50 IATs. In this implementation, the initial timing history is filled using the fixed benign-derived IAT used by the simple reshaping attack. After this initialization, the LSTM model predicts subsequent IATs iteratively.

At each prediction step, the model receives the most recent 50 generated IATs and the corresponding 50 normalized malicious packet sizes. The predicted IAT is appended to the generated timing sequence and used as part of the input window for future predictions. This process continues until IATs have been generated for the malicious flow.

The generated IATs are then converted back into packet timestamps. Beginning with the timestamp of the first malicious packet, each subsequent timestamp is produced by adding the next generated IAT to the current timestamp:

$$t'_{\mathrm{i}} = t'_{\mathrm{i}-1} + \widehat{\Delta t_{\mathrm{i}}}, \tag{4.2}$$

where t'_{i} is the reshaped timestamp of packet i and $\widehat{\Delta t_{\mathrm{i}}}$ is the LSTM-predicted IAT.

4.2.4 Resulting Reshaped Flow

The output of the LSTM-based attack is a reshaped version of the target malicious flow. The packet contents, packet sizes, protocol fields, and communication endpoints remain unchanged. Only the timestamps are modified. As a result, the attack behavior is preserved, but the temporal characteristics of the malicious flow are altered to more closely resemble the benign timing behavior learned by the LSTM model.

This LSTM-based attack serves as the more complex reshaping method evaluated in this thesis. It provides a comparison point for determining whether learned temporal modeling is necessary for evading Kitsune and supervised learning NIDSs, or whether simpler timing transformations can achieve similar effects.

4.3 Fixed IAT Traffic Reshaping Attack

Although TANTRA demonstrates that a learned sequence model can successfully generate benign-like timing behavior, it remains unclear whether such a sophisticated model is required to evade a detector such as Kitsune. Because AfterImage derives numerous statistical features from packet timing information, even a simple timing transformation may significantly alter the resulting feature vectors.

To investigate this possibility, this work introduces a fixed IAT attack. Unlike the LSTM-based approach, this attack does not require model training, hyperparameter tuning, or sequence prediction. Instead, the attack generates modified timestamps using a constant IAT derived from benign traffic observations.

As with the LSTM-based attack, a target malicious flow is first isolated from the network trace. A benign reference flow is then selected, and the average IAT of the benign packets is computed. This average value serves as the target timing interval for the attack.

Let $\overline{\Delta t}_{benign}$ denote the mean IAT observed within the benign reference flow. Beginning with the timestamp of the first malicious packet, subsequent packet timestamps are generated according to

$$t'_{\mathrm{i}} = t'_{\mathrm{i}-1} + \overline{\Delta t}_{benign}, \tag{4.3}$$

where t'_{i} represents the reshaped timestamp of packet i.

This process produces a malicious flow whose packet spacing is uniform and whose average communication rate resembles that of the benign reference traffic. Packet contents, protocol information, communication endpoints, and packet ordering remain unchanged.

The fixed IAT attack represents a significant simplification compared to TANTRA. No neural network is trained, no prediction process is performed, and no attempt is made to model complex temporal relationships. Instead, the attack tests whether simple manipulation of packet timing behavior is sufficient to alter the features used by Kitsune and supervised learning models.

Because the attack requires minimal computational resources and can be implemented without ML expertise, successful evasion using this approach would suggest that the vulnerability extends beyond sophisticated adversarial learning techniques and may arise from a more fundamental reliance on timing-based features.

The fixed IAT reshaping attack and the previously discussed LSTM reshaping attack are evaluated using both the Kitsune AD and multiple supervised ML classifiers. By comparing detector performance before and after reshaping, the experimental results provide insight into the extent to which timing manipulation can influence intrusion detection systems and whether complex reshaping strategies offer meaningful advantages over simpler alternatives.

5. SUPERVISED ML MODELS AGAINST TRAFFIC RESHAPING ATTACKS

To complement the anomaly-based evaluation performed using Kitsune, this work additionally investigates the impact of traffic reshaping attacks on supervised ML models. Unlike ADs, supervised classifiers learn explicit decision boundaries between benign and malicious traffic using labeled examples. This chapter examines whether timing-based reshaping attacks can evade supervised classifiers and whether retraining strategies can improve robustness against reshaped traffic.

The supervised learning experiments are designed around two primary questions. First, can traffic reshaping attacks evade supervised ML models trained on unmodified malicious traffic? Second, can supervised ML models be trained to detect both original and reshaped malicious traffic? These questions evaluate both the offensive effectiveness of traffic reshaping and the defensive potential of training models with reshaped examples.

Three supervised models are selected for this study: Random Forest (RF), Light Gradient Boosting Machine (LightGBM), and TabNet. These models represent different approaches to learning from tabular network traffic features. RF provides a bagging-based ensemble baseline. LightGBM represents a boosting-based tree ensemble designed for efficient learning on structured data. TabNet provides a neural architecture designed specifically for tabular data through attention-based feature selection. Together, these models enable comparisons across bagging, boosting, and deep learning approaches.

5.1 Supervised Learning Formulation

The supervised learning models operate on the same feature representation used by the Kitsune-based anomaly detection pipeline. Each packet is represented using the 100-dimensional feature vector produced by Kitsune's AfterImage feature extractor. These features describe packet behavior and recent network activity across multiple temporal windows. Each feature vector is paired with a packet-level label derived from the dataset.

For the primary binary classification task, the label space is defined as

$$y = \begin{cases} 0, & \text{benign packet,} \\ 1, & \text{malicious packet.} \end{cases} \tag{5.1}$$

Thus, each model learns a function

$$f : \mathbb{R}^{100} \to \{0, 1\}, \tag{5.2}$$

where the input is a 100-dimensional AfterImage feature vector and the output is a binary prediction indicating whether the packet is benign or malicious.

The binary classification task is used to evaluate whether reshaped malicious traffic remains detectable when the classifier is trained under different data conditions. This is important because traffic reshaping does not alter packet labels: a reshaped malicious packet remains malicious even though its timestamp-derived features may change.

5.2 Dataset Construction and Splitting

For each experiment, the available traffic is separated into benign and malicious portions. The benign portion is held constant between the original and reshaped datasets, while the malicious portion differs depending on whether the original or reshaped flow is used. This setup ensures that the primary difference between the two datasets is the timing behavior of the malicious flow.

The data is split independently within the benign and malicious portions before being recombined into train, validation, and test sets. The first 70% of benign and malicious samples are used for training, the next 10% are used for validation, and the final 20% are used for testing. This produces class-consistent partitions while preserving the chronological structure of the packet stream within each class.

The binary experiments construct three training configurations:

1. **Original Traffic Training**: The model is trained on benign traffic and original malicious traffic.

2. **Reshaped Traffic Training**: The model is trained on benign traffic and reshaped malicious traffic.

3. **Combined Traffic Training**: The model is trained using benign traffic, original malicious traffic, and reshaped malicious traffic.

The combined training configuration is intended to evaluate whether exposure to both original and reshaped malicious examples improves robustness. In this setting, the benign traffic is not duplicated unnecessarily; instead, the combined dataset augments the original training data with reshaped malicious examples.

While some datasets exhibit class imbalance, no class-balancing parameters are applied in the supervised learning experiments. Although RF, LightGBM, and TabNet support class weighting or imbalance-handling options, these options are left disabled so that each model is evaluated under the same unweighted training conditions. This choice avoids introducing model-specific class-balance adjustments and keeps the comparison focused on the effects of traffic reshaping and training-data composition. In preliminary experimentation, class-balancing parameters reduced performance in some cases, further motivating their exclusion from the final experiments.

Each trained model is evaluated on two test sets. The original test set contains benign traffic and original malicious traffic, while the reshaped test set contains the same benign traffic and reshaped malicious traffic. This paired evaluation design allows performance degradation under reshaping to be measured directly.

5.3 ML Models

5.3.1 Random Forest

Random Forest (RF) is an ensemble learning algorithm that constructs a collection of decision trees and combines their predictions through majority voting [22]. Each tree is trained on a bootstrap sample of the data, and candidate split features are randomly subsampled during tree construction. These sources of randomness reduce correlation among trees and improve generalization relative to a single decision tree.

RF is included as a robust tree-based baseline for supervised intrusion detection. It is well suited to tabular data, can capture nonlinear feature interactions, and requires relatively little preprocessing. It also provides feature importance values that can be used to interpret which AfterImage features contribute most strongly to classification decisions.

In this work, the RF classifier is trained separately under the original, reshaped, and combined training configurations. For the binary classification experiments, the model is trained using the combined training and validation sets. After training, the saved classifier is evaluated on both original and reshaped test sets. This produces a direct comparison between performance on unmodified malicious traffic and performance on reshaped malicious traffic.

In addition to predictions, the RF model outputs feature importance values based on the trained ensemble. These values are later used to identify the AfterImage features most influential to the model's binary classification decisions.

The primary strengths of RF in this study include:

- Strong baseline performance on structured tabular data;
- Ability to model nonlinear relationships among AfterImage features;
- Robustness to noise and irrelevant features;
- Reduced overfitting through bootstrap aggregation;
- Built-in feature importance estimates.

5.3.2 LightGBM

LightGBM is a gradient boosting framework that constructs an ensemble of decision trees sequentially [23]. Unlike RF, where trees are trained independently, each LightGBM tree is trained to reduce the errors of the existing ensemble. The final prediction is obtained by aggregating the outputs of the boosted trees.

LightGBM is selected because gradient-boosted decision trees frequently perform well on structured datasets, including network intrusion detection data. The model is computation-

ally efficient and can learn complex nonlinear relationships among features. This makes it suitable for evaluating whether the 100-dimensional AfterImage representation contains sufficient information to distinguish benign, original malicious, and reshaped malicious traffic.

For the binary classification experiments, LightGBM is trained under the same original, reshaped, and combined configurations used for RF. Training uses a validation set and binary log-loss as the evaluation metric. The maximum number of boosted trees (`n_estimators`) is set to 1000; however, early stopping is applied so that training terminates when validation performance no longer improves. The best validation iteration is saved and used for testing.

During testing, LightGBM outputs a probability estimate for the malicious class. A threshold of 0.5 is used to convert this probability into a binary class prediction:

$$\hat{y} = \begin{cases} 1, & P(y = 1 \mid x) \geq 0.5, \\ 0, & P(y = 1 \mid x) < 0.5. \end{cases} \tag{5.3}$$

Feature importance is computed using both split-based and gain-based importance. Split importance measures how frequently a feature is used in tree splits, whereas gain importance measures the improvement in the objective function contributed by splits using that feature. Because gain importance more directly reflects the contribution of a feature to reducing model loss, it is particularly useful when interpreting predictive importance.

The primary strengths of LightGBM in this study include:

- High predictive performance on tabular datasets;
- Efficient training and inference;
- Ability to model nonlinear feature interactions;
- Validation-based early stopping;
- Feature importance analysis using split and gain metrics.

5.3.3 TabNet

TabNet is a deep neural network architecture designed for supervised learning on tabular data [24]. Traditional neural networks often process all input features simultaneously, which can make them less effective on structured datasets than tree-based methods. TabNet addresses this limitation through sequential attention mechanisms that learn to select informative subsets of features during prediction.

At each decision step, TabNet produces a sparse feature mask that determines which input features should receive attention. These masks allow the model to focus on different feature subsets at different stages of inference. This design provides both predictive flexibility and a degree of interpretability, since feature importance can be estimated from the learned attention behavior.

TabNet is included in this work as a neural alternative to RF and LightGBM. Its inclusion allows the experiments to evaluate whether a deep learning model designed for tabular data responds differently to traffic reshaping attacks than tree-based ensemble methods.

For the binary classification experiments, TabNet is trained using the same original, reshaped, and combined training configurations. Training uses a validation set with log-loss as the evaluation metric. Early stopping is applied with a patience parameter to reduce overfitting. The implementation uses a maximum of 200 epochs, validation-based early stopping, and minibatch training. The batch size and virtual batch size are selected based on the size of the training data.

After training, the saved TabNet model is evaluated on both the original and reshaped test sets. The model outputs packet-level class predictions and feature importance values. These feature importance values are included in the later feature analysis to compare the features emphasized by TabNet against those emphasized by the tree-based models.

The primary strengths of TabNet in this study include:

- Neural architecture designed specifically for tabular data;
- Sequential attention-based feature selection;
- Ability to model complex nonlinear relationships;

- Feature importance estimation through learned attention;
- Comparison point against tree-based supervised models.

5.4 Experimental Usage

For each supervised model, experiments are conducted under three training scenarios. The purpose of these scenarios is to evaluate both vulnerability to reshaping and potential robustness gained through retraining.

5.4.1 Original Traffic Training

In the original training configuration, models are trained using benign traffic and unmodified malicious traffic. These models are then evaluated on both original and reshaped test traffic.

This setting answers the question of whether traffic reshaping attacks can evade supervised ML models that have only learned the original attack distribution. If performance is high on original traffic but degrades on reshaped traffic, then the reshaping attack has successfully shifted the malicious feature distribution away from what the classifier learned during training.

5.4.2 Reshaped Traffic Training

In the reshaped training configuration, models are trained using benign traffic and reshaped malicious traffic. These models are again evaluated on both original and reshaped test traffic.

This setting evaluates whether supervised models can learn to recognize reshaped malicious traffic when such examples are available during training. It also tests whether a model trained on reshaped attacks generalizes back to original unmodified attacks.

5.4.3 Combined Traffic Training

In the combined training configuration, models are trained using benign traffic, original malicious traffic, and reshaped malicious traffic. This configuration is intended to evaluate a retraining-based defense strategy in which the supervised model is exposed to both clean and reshaped attack distributions.

If combined training performs well on both original and reshaped test traffic, then supervised learning may provide a viable mechanism for detecting traffic reshaping attacks, provided that representative reshaped examples are included in training.

5.5 Feature Importance Analysis

In addition to evaluating predictive performance, this work investigates which AfterImage features are most useful for distinguishing benign and malicious traffic. This question is important because traffic reshaping attacks operate by modifying packet timing behavior, and AfterImage includes multiple timing-sensitive feature categories across several temporal windows.

Feature importance analysis is performed for each supervised model and training configuration. For RF, feature importance is obtained from the trained ensemble. For LightGBM, both split-based and gain-based importance are computed, with gain importance providing a measure of how much each feature contributes to reducing the model's loss. For TabNet, feature importance is obtained from the model's learned attention-based feature selection.

The feature importance values are ranked to identify the most influential AfterImage features for each model. These rankings are then compared across models and training configurations. This analysis addresses the question of which features from Kitsune's AfterImage feature extractor are most useful for distinguishing benign traffic from malicious traffic.

The analysis also helps determine whether models rely primarily on timing-sensitive features, size-based features, host-to-host communication statistics, or socket-level statistics. If timing-related features dominate the rankings, then this may explain why traffic reshaping attacks are effective. Conversely, if non-timing features remain highly informative, supervised models may be able to detect malicious traffic even after timestamp manipulation.

The feature importance analysis therefore serves two purposes. First, it provides interpretability by identifying which features drive model decisions. Second, it provides insight into the relationship between feature reliance and robustness against traffic reshaping.

5.6 Multiclass Classification Extension

The primary supervised learning experiments formulate intrusion detection as a binary classification task: benign versus malicious. However, this formulation does not distinguish between original malicious traffic and reshaped malicious traffic. To explore this distinction, this work also includes a multiclass classification extension.

In the multiclass setting, the label space is defined as

$$y = \begin{cases} 0, & \text{benign packet,} \\ 1, & \text{original malicious packet,} \\ 2, & \text{reshaped malicious packet.} \end{cases} \tag{5.4}$$

The objective is no longer simply to detect whether traffic is malicious. Instead, the model must determine whether a packet is benign, part of the original malicious flow, or part of the reshaped malicious flow. This formulation evaluates whether supervised models can learn differences between unmodified and reshaped attack traffic.

The multiclass dataset is constructed using the unchanged benign traffic, the original malicious flow, and the reshaped malicious flow. As in the binary experiments, the data is divided into training, validation, and testing partitions using 70%, 10%, and 20% splits. The combined multiclass dataset contains benign, original malicious, and modified malicious examples.

RF, LightGBM, and TabNet are extended to the three-class setting. For LightGBM, the objective is changed to multiclass classification with three output classes. During prediction, LightGBM produces a probability distribution over the three classes, and the predicted class is selected using the largest probability:

$$\hat{y} = \arg \max_{c \in \{0,1,2\}} P(y = c \mid x). \tag{5.5}$$

The multiclass extension addresses a more detailed version of the supervised learning question: Can supervised ML models be trained to distinguish among benign traffic, original malicious traffic, and reshaped malicious traffic? If the models can reliably separate all three classes, then reshaped traffic retains identifiable artifacts that supervised learning can exploit. If reshaped malicious traffic is frequently classified as benign, then the reshaping attack successfully conceals the attack from the model. If reshaped malicious traffic is classified as original malicious, then the model detects malicious behavior but does not distinguish the reshaped distribution from the original one.

Together, the binary and multiclass experiments provide a comprehensive view of supervised learning under traffic reshaping attacks. The binary experiments evaluate detection robustness, while the multiclass extension evaluates whether models can distinguish reshaped attacks as a separate traffic condition.

6. PERFORMANCE EVALUATION

This chapter evaluates the effectiveness of traffic reshaping attacks and the robustness of both anomaly-based and supervised ML intrusion detection models. First, the evaluation metrics for the binary and multiclass classification settings are defined. The thresholding procedure used to convert Kitsune RMSE anomaly scores into binary predictions is also described. Next, the chapter evaluates the effectiveness of the Simple and LSTM-based traffic reshaping strategies against the Kitsune AD. The chapter then evaluates the performance of supervised ML models in detecting intrusion attacks under different traffic reshaping strategies in both binary and multiclass classification settings. Finally, feature-importance data from each trained model is analyzed to identify which AfterImage features are most useful for distinguishing benign traffic from malicious traffic in the binary setting, and which features are most useful for distinguishing among benign traffic, original malicious traffic, and modified malicious traffic in the multiclass setting.

6.1 Evaluation Metrics

Evaluation uses packet-level ground-truth labels to compare model predictions against the true class of each packet. Because the experiments feature both binary and multiclass classification settings, the metrics are defined separately for each case. The section also describes how Kitsune RMSE anomaly scores are converted into binary predictions using attack-specific detection thresholds.

6.1.1 Binary Classification Metrics

For the binary classification experiments, packets are labeled as either benign or malicious. A malicious packet is treated as the positive class, while a benign packet is treated as the negative class.

For each experiment, the predictions can be summarized using four confusion-matrix outcomes:

- **True Positive (TP)**: a malicious packet correctly classified as malicious.

- **True Negative (TN)**: a benign packet correctly classified as benign.
- **False Positive (FP)**: a benign packet incorrectly classified as malicious.
- **False Negative (FN)**: a malicious packet incorrectly classified as benign.

From these values, several performance metrics are computed. The True Positive Rate (TPR), also referred to as recall, measures the proportion of malicious packets correctly detected:

$$TPR = \frac{TP}{TP + FN}. \tag{6.1}$$

The False Positive Rate (FPR) measures the proportion of benign packets incorrectly flagged as malicious:

$$FPR = \frac{FP}{FP + TN}. \tag{6.2}$$

Accuracy measures the overall proportion of correct predictions:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \tag{6.3}$$

Precision measures the proportion of packets predicted as malicious that were actually malicious:

$$Precision = \frac{TP}{TP + FP}. \tag{6.4}$$

Finally, the F1 score is the harmonic mean of precision and recall:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}. \tag{6.5}$$

These metrics provide complementary views of detector behavior. TPR is especially important when evaluating attack detection because it measures the proportion of malicious packets that are successfully identified. FPR characterizes the cost imposed on benign traffic, indicating whether normal packets are incorrectly flagged as attacks. Precision measures the

reliability of malicious predictions, while F1 score provides a single summary metric that balances precision and recall.

6.1.2 Multiclass Classification Metrics

For the multiclass classification experiments, packets are labeled as benign, original malicious, or modified (reshaped) malicious. In this setting, benign packets are assigned class 0, original malicious packets are assigned class 1, and modified malicious packets are assigned class 2. Model predictions are summarized using a 3×3 confusion matrix, where each row represents the ground-truth class and each column represents the predicted class.

In addition to overall accuracy, per-class one-versus-rest metrics are computed for the original malicious and modified malicious classes. For a class c, the one-versus-rest formulation treats packets belonging to class c as the positive class and all other packets as the negative class. For example, for the original malicious class, original malicious packets are treated as positive examples, while benign and modified malicious packets are treated as negative examples.

For each malicious class c, the one-versus-rest confusion-matrix outcomes are defined as follows:

- **True Positive ($\boldsymbol{TP_c}$)**: a packet from class c correctly classified as class c.
- **True Negative ($\boldsymbol{TN_c}$)**: a packet from another class correctly classified as not belonging to class c.
- **False Positive ($\boldsymbol{FP_c}$)**: a packet from another class incorrectly classified as class c.
- **False Negative ($\boldsymbol{FN_c}$)**: a packet from class c incorrectly classified as another class.

Using these one-versus-rest values, TPR_c, FPR_c, and $F1_c$ are computed using the same equations as in the binary classification setting. In the multiclass performance tables, the subscript O denotes the original malicious class, while the subscript M denotes the modified malicious class. Thus, TPR_O, FPR_O, and $F1_O$ describe one-versus-rest performance

for original malicious traffic, while TPR_M, FPR_M, and $F1_M$ describe one-versus-rest performance for modified malicious traffic. To provide a single summary value for multiclass malicious-traffic classification performance, the multiclass performance tables also report $F1_A$, the average F1 score across the original malicious and modified malicious classes:

$$F1_A = \frac{F1_O + F1_M}{2}. \tag{6.6}$$

This metric summarizes how well a multiclass model performs across the two malicious classes while excluding the benign class from the average. The benign class is often much larger than either malicious class, so this metric provides a better indication of malicious-class performance than overall accuracy alone.

6.1.3 Kitsune Threshold Selection

For the Kitsune AD, predictions are produced by applying a threshold to the RMSE anomaly scores generated during the execution phase. Packets with RMSE values above the selected threshold are classified as malicious, while packets with RMSE values below the threshold are classified as benign. The threshold is selected separately for each intrusion attack. For the Mirai and SSDP flood experiments, the threshold is computed as the mean plus one standard deviation of the benign RMSE values observed after the feature mapping and AD grace periods and before the first malicious packet:

$$\phi = \mu_{benign} + \sigma_{benign}. \tag{6.7}$$

For the fuzzing and active wiretap experiments, using the mean plus one standard deviation method results in substantially higher FPR values. As a result, a different method is used; the threshold is selected according to a target FPR on benign execution-phase traffic. Specifically, the threshold is chosen using the benign RMSE values observed after Kitsune enters execution mode and before the first malicious packet is encountered. For fuzzing experiments, the threshold is selected as the lowest RMSE value that produces an FPR of 0 on

this benign interval. For active wiretap experiments, the threshold is selected as the lowest RMSE value that produces an FPR of 0.001 on the same type of benign interval.

Although these threshold values are not necessarily optimal for maximizing Kitsune's overall detection performance, they provide a consistent basis for comparing reshaping attacks. The goal is not to tune Kitsune separately for each attack condition, but to establish a fixed baseline threshold for each intrusion attack and then evaluate how the original, fixed IAT reshaped, and LSTM-reshaped malicious flows behave under that same threshold.

For the supervised ML models, predictions are produced directly by the trained classifiers rather than by applying an RMSE threshold. These predictions are evaluated using the binary and multiclass metrics described above, depending on the experiment.

6.2 Traffic Reshaping Attack Results and Analysis

The first set of experiments uses the Kitsune AD to compare the effectiveness of two traffic reshaping strategies: a simple fixed IAT attack and an LSTM-based attack inspired by TANTRA. Rather than focusing solely on whether reshaped traffic can evade Kitsune, the primary goal of these experiments is to determine whether a simple reshaping strategy can achieve results comparable to a more complex learned reshaping model. By comparing Kitsune's performance across unmodified, simple-reshaped, and LSTM-reshaped malicious traffic, these experiments address whether simpler reshaping strategies can achieve the same goal as TANTRA.

For each intrusion attack, three traffic conditions are evaluated. The first condition, *Unmodified*, contains the original benign traffic and a single unmodified malicious flow. This condition establishes Kitsune's baseline detection performance against the selected malicious flow. The second condition, *Simple*, contains the same benign traffic and the same malicious flow, but the malicious flow is reshaped using the fixed IAT attack. The third condition, *LSTM*, contains the same benign traffic and malicious flow, but the malicious flow is reshaped using the LSTM-based traffic reshaping attack. Because the benign traffic is held constant across all three conditions, changes in Kitsune performance can be attributed to modifications in the malicious flow's timing behavior.

During execution, Kitsune produces a sequence of RMSE anomaly scores that can be graphed using a logarithmic scale to visualize how strongly the detector reacts to different traffic conditions. Figures 6.1–6.3 show Kitsune anomaly scores for the SSDP flood attack under the unmodified, Simple-reshaped, and LSTM-reshaped conditions. Successful detection is indicated by malicious (red) traffic points appearing above the detection threshold, shown as a horizontal black dashed line. A successful reshaping attack aims to fool the AD and reduce RMSE scores of malicious traffic such that these packets' anomaly scores fall below the threshold and are incorrectly classified as benign. For graphs of Kitsune anomaly scores for other intrusion attacks, see appendix A.

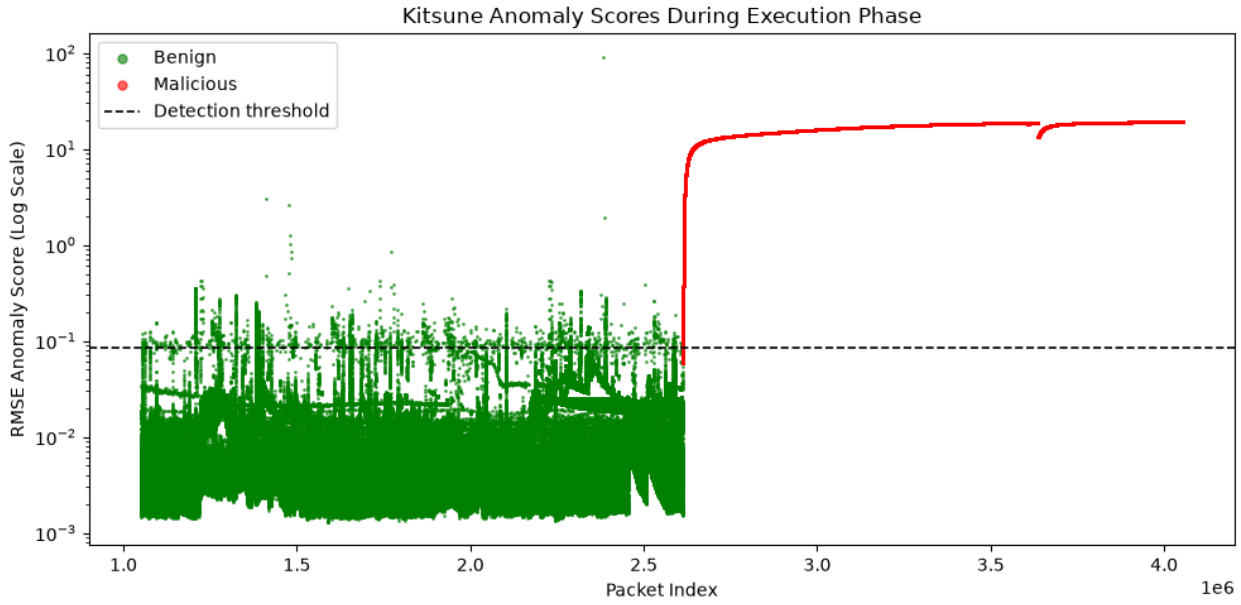

Figure 6.1. Kitsune Anomaly Scores for the Unmodified SSDP Flood Attack

Table 6.1 displays the number of benign and malicious packets used in each evaluated intrusion attack. The table reports both the total number of packets in each class, after isolating the malicious flow, and the number of packets included specifically in the test set, which corresponds to the final 20% of each class. These counts provide context for interpreting the performance metrics reported in later tables, particularly accuracy and FPR, since the relative number of benign and malicious packets differs substantially across attacks.

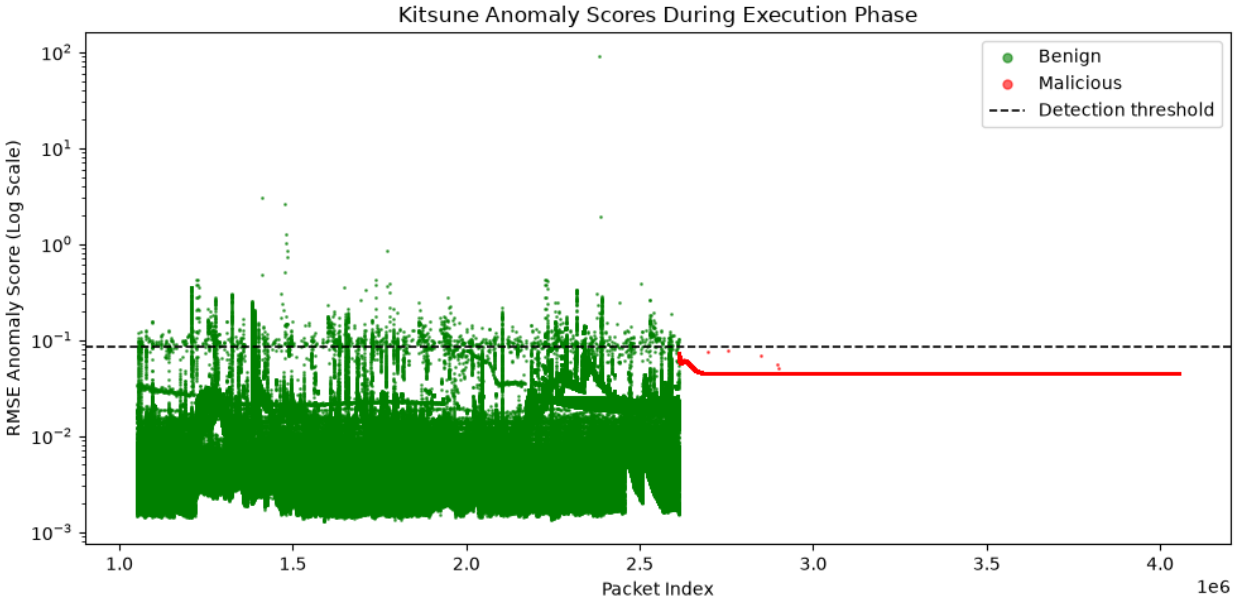

Figure 6.2. Kitsune Anomaly Scores for the SSDP Flood Attack Under Simple Reshaping

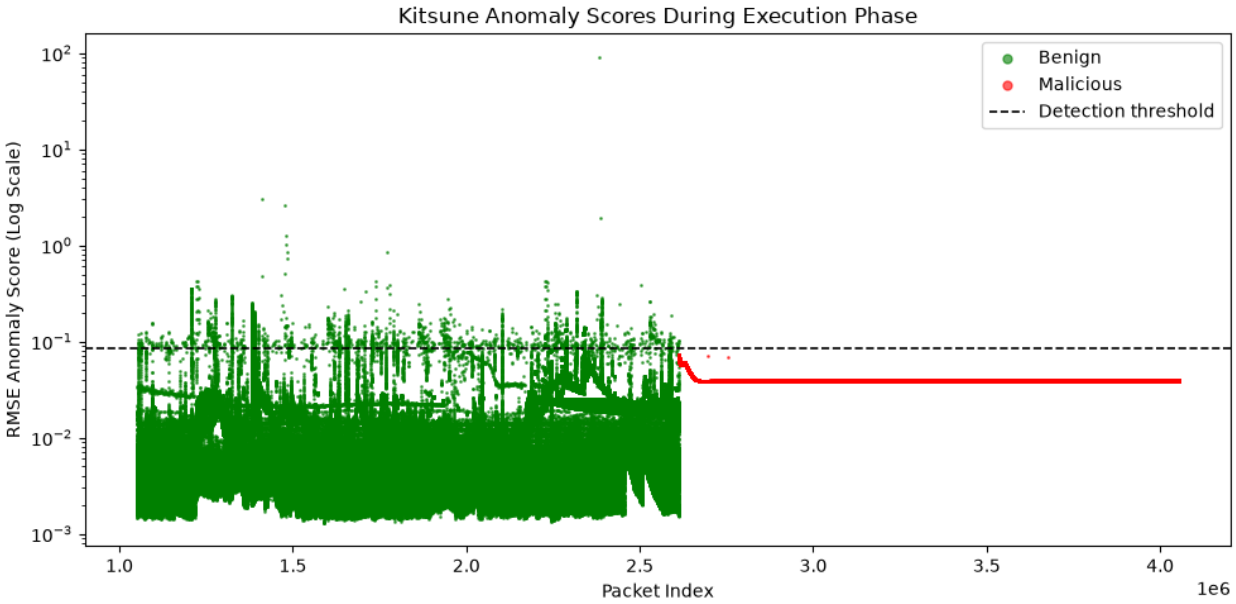

Figure 6.3. Kitsune Anomaly Scores for the SSDP Flood Attack Under LSTM Reshaping

Table 6.1. Evaluation Packet Counts by Intrusion Attack

Attack	Total Packets		Test Set Packets	
	Benign	Malicious	Benign	Malicious
SSDP Flood	2 613 442	1 439 599	522 689	287 920
Mirai	121 621	550 967	24 325	110 194
Fuzzing	1 300 005	8324	260 001	1665
Active Wiretap	1 300 000	48 014	260 000	9603

By applying a selected detection threshold to the RMSE anomaly scores generated by Kitsune, binary predictions can be produced; packets with RMSE scores above the threshold are classified as malicious while packets with RMSE scores below the threshold are classified as benign. Table 6.2 summarizes Kitsune's performance across all evaluated attacks and reshaping strategies after applying each attack's chosen detection threshold and comparing predicted classes against ground-truth labels. For consistency with the supervised ML results in Section 6.3, Kitsune performance metrics are calculated using the test-set packets shown in Table 6.1. These packets correspond to the last 20% of the benign traffic and the last 20% of the malicious traffic for each reshaping strategy.

Table 6.2. Kitsune Performance on Traffic Reshaping Attacks

Attack	Reshaping	TPR	FPR	Accuracy	F1 Score
SSDP Flood	Unmodified	1.0000	0.0026	0.9983	0.9976
SSDP Flood	Simple	0.0000	0.0026	0.6431	0.0000
SSDP Flood	LSTM	0.0000	0.0026	0.6431	0.0000
Mirai	Unmodified	1.0000	0.0082	0.9985	0.9991
Mirai	Simple	0.0000	0.0082	0.1794	0.0000
Mirai	LSTM	0.0000	0.0082	0.1794	0.0000
Fuzzing	Unmodified	1.0000	0.0000	1.0000	1.0000
Fuzzing	Simple	0.5165	0.0000	0.9969	0.6812
Fuzzing	LSTM	0.5255	0.0000	0.9970	0.6890
Active Wiretap	Unmodified	0.7221	0.0010	0.9892	0.8261
Active Wiretap	Simple	0.5106	0.0010	0.9816	0.6645
Active Wiretap	LSTM	0.4660	0.0010	0.9801	0.6247

Table 6.2 demonstrates that both evaluated traffic reshaping strategies substantially reduce Kitsune's ability to detect malicious traffic. For SSDP flood and Mirai, Kitsune detects the unmodified malicious flows with a TPR of 1.0000. However, after either Simple or LSTM-based reshaping, the TPR falls to 0.0000 and the F1 score falls to 0.0000. This indicates that both reshaping attacks completely evade Kitsune under the selected thresholds for these attacks.

For the fuzzing and active wiretap attacks, reshaping does not completely eliminate detection, but it still reduces performance. In the fuzzing experiment, the unmodified malicious traffic is detected perfectly, while the Simple- and LSTM-reshaped versions reduce TPR to 0.5165 and 0.5255, respectively. Similarly, for active wiretap, TPR decreases from 0.7221 on unmodified traffic to 0.5106 under Simple reshaping and 0.4660 under LSTM reshaping. These results demonstrate that reshaping attacks can reduce Kitsune's detection capability even when they do not fully evade detection.

One possible explanation for the lower detection performance for the fuzzing and active wiretap attacks is that SSDP flood and Mirai exhibit more pronounced timing- and volume-based behavior than fuzzing and active wiretap. Flooding and botnet-style traffic often involve many packets generated over a short period, producing distinctive rate, count, and host-to-host communication patterns in the AfterImage feature space. In contrast, fuzzing and active wiretap may produce traffic patterns that are less dominated by packet timing or volume alone and therefore overlap more strongly with benign traffic. This overlap can make these attacks more difficult for ML models to distinguish from normal behavior.

Comparison between the Simple and LSTM reshaping attacks shows that the simple fixed IAT attack performs similarly to the LSTM-based method. For the Mirai and SSDP flood attacks, both reshaping strategies reduce the TPR from 1.0000 to 0.0000. For the fuzzing attack, the Simple reshaping attack is slightly more effective than the LSTM reshaping attack, reducing TPR to 0.5165 compared with 0.5255. Conversely, for the active wiretap attack, the LSTM reshaping attack is slightly more effective, reducing TPR and F1 score to 0.4660 and 0.6247, respectively, compared with 0.5106 and 0.6645 for the Simple reshaping attack.

Overall, these results show that the simple fixed IAT attack performs comparably to the more complex LSTM-based reshaping attack. Although the LSTM reshaping attack performs slightly better for active wiretap, the Simple reshaping attack performs slightly better for fuzzing, and both attacks perform identically for SSDP flood and Mirai. In each case, the difference between the two reshaping strategies is small compared to the overall drop in Kitsune detection performance from the unmodified traffic to the reshaped traffic. These results suggest that a learned LSTM timing model may not be necessary for successful traffic reshaping. Instead, a much simpler fixed IAT strategy can still substantially reduce Kitsune's ability to detect malicious traffic.

6.3 Supervised ML Binary Classification Models Against Traffic Reshaping Attacks

As demonstrated in the previous set of experiments, the Kitsune AD can struggle to detect reshaped malicious traffic at certain decision thresholds. The second set of experiments explored in this section aims to address the effectiveness of the same traffic reshaping strategies against supervised ML models in a binary classification setting. In particular, two core questions are addressed: (1) can traffic reshaping attacks evade supervised ML models trained on unmodified malicious traffic and (2) can supervised ML models be trained to detect both original and reshaped malicious traffic? In addition, each supervised ML model's feature importance attributes are leveraged to investigate which AfterImage features are most useful for distinguishing between benign and malicious traffic.

Unlike the Kitsune AD, which is trained only on benign traffic, the supervised models are trained using labeled benign and malicious examples. For each intrusion attack and reshaping strategy, a set of nine model configurations is trained and evaluated. As mentioned in Chapter 5, the three ML models evaluated are RF, LightGBM, and TabNet. In addition, for each model, there are three training configurations. In the *Original* configuration, the model is trained on benign traffic and original malicious traffic; in the *Modified* configuration, the model is trained on benign traffic and reshaped malicious traffic; and in the *Combined* configuration, the model is trained on benign traffic, original malicious traffic, and

reshaped malicious traffic. For more information about dataset construction and splitting, see Section 5.2.

Figures 6.4 and 6.5 show representative training and validation loss curves for LightGBM and TabNet. The LightGBM model is trained on Combined original and LSTM-reshaped active wiretap traffic while the TabNet model is trained on Simple-reshaped (Modified) fuzzing traffic. For LightGBM, validation loss is used with early stopping to select the best boosting iteration. For TabNet, validation log loss is monitored across epochs, and early stopping is used to help reduce overfitting. Both curves show a sharp initial decline in training and validation loss, followed by a plateau near zero for training loss. In Figure 6.4, the validation loss begins to increase after reaching its minimum, which suggests that the model begins to overfit after this point. However, because early stopping is used, the model iteration with the best validation loss is selected for test-set evaluation. In Figure 6.5, the validation loss remains close to the training loss plateau, suggesting less separation between training and validation performance for this representative run. No loss curve is produced for RF because RF does not learn through iterative optimization in the same way as LightGBM and TabNet. Since RF does not explicitly use validation data during training, both the training and validation splits are incorporated into the RF training data.

6.3.1 Supervised ML Binary Model Performance Results and Analysis

For each intrusion attack and reshaping strategy, every model is evaluated on two test sets; the *Original* test set contains benign traffic and original malicious traffic, and the *Modified* test set contains benign traffic and reshaped malicious traffic. Tables 6.3 and 6.4 display confusion matrices demonstrating a Modified-trained LightGBM model's performance on the Original and Modified test sets, respectively. On the Original test set, the Modified-trained LightGBM model correctly identifies 75 996 malicious packets but misses 211 924 malicious packets, producing many false negatives. On the Modified test set, the same model correctly identifies all 287 920 malicious packets and all 522 689 benign packets. This example illustrates an important trend in the supervised learning experiments: a model may perform

well when the test traffic matches the type of malicious traffic it was trained on, but may perform poorly when evaluated on a different malicious traffic condition.

Table 6.3. Original Test Set Confusion Matrix for the Binary LightGBM Model Trained on LSTM-Reshaped (Modified) SSDP Flood Attack

	Predicted Positive	**Predicted Negative**
Ground Truth Positive	75 996	211 924
Ground Truth Negative	0	522 689

Table 6.4. Modified Test Set Confusion Matrix for the Binary LightGBM Model Trained on LSTM-Reshaped (Modified) SSDP Flood Attack

	Predicted Positive	**Predicted Negative**
Ground Truth Positive	287 920	0
Ground Truth Negative	0	522 689

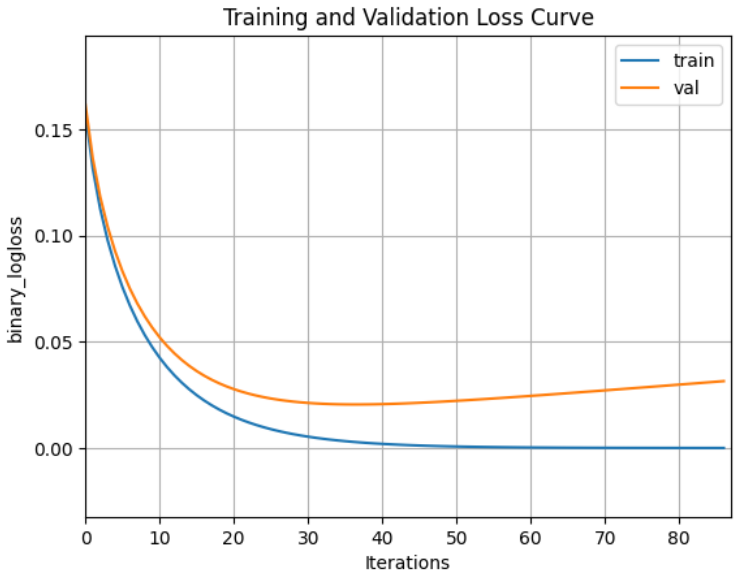

Figure 6.4. Training and Validation Loss for the LightGBM Model Trained on Combined Original and LSTM-Reshaped Active Wiretap Attack

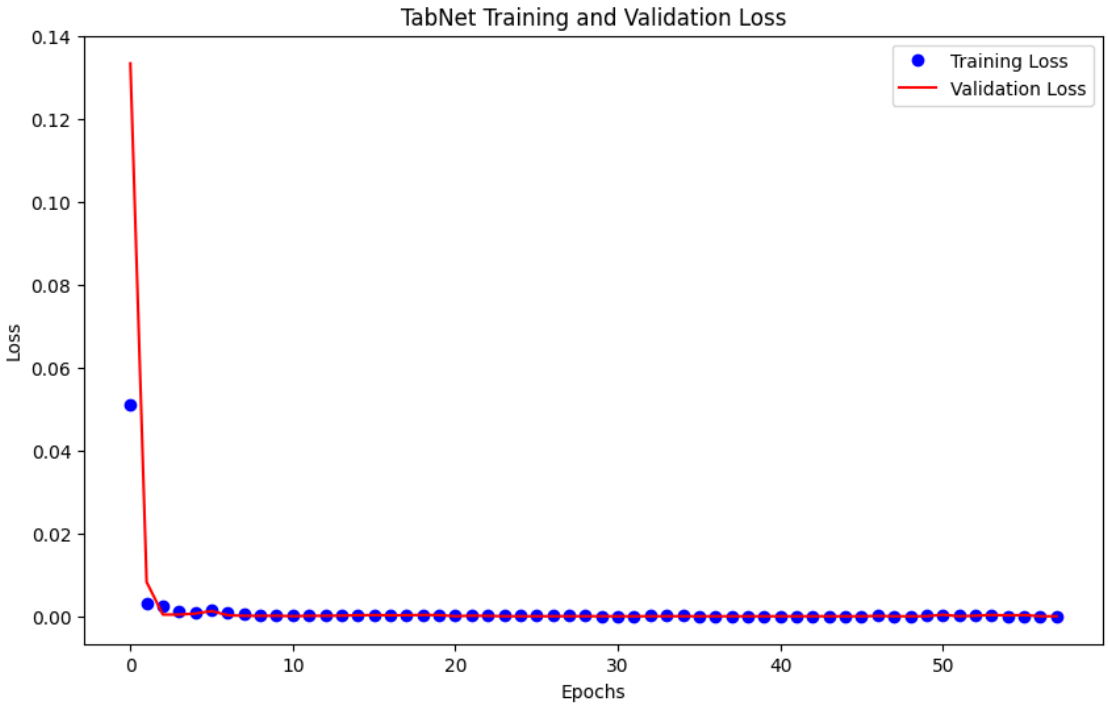

Figure 6.5. Training and Validation Loss for the TabNet Model Trained on Simple-reshaped (Modified) Fuzzing Attack

For the full binary classification evaluation, Tables 6.5–6.12 report the TPR, FPR, accuracy, and F1 score for RF, LightGBM, and TabNet models. These tables include models trained under the *Original*, *Modified*, and *Combined* configurations for both Simple and LSTM reshaping strategies with performance evaluated on both the *Original* and *Modified* test sets. These tables are used to evaluate whether reshaped malicious traffic can evade supervised models trained only on original malicious traffic, and whether combined training can improve detection across both original and reshaped traffic.

Table 6.5. Supervised ML Binary Models Performance on SSDP Flood Attack Under Simple Reshaping

Model	Train Type	Test Set	TPR	FPR	Accuracy	F1 Score
Random Forest	Original	Original	1.0000	0.0000	1.0000	1.0000
Random Forest	Original	Modified	1.0000	0.0000	1.0000	1.0000
Random Forest	Modified	Original	1.0000	0.0000	1.0000	1.0000
Random Forest	Modified	Modified	1.0000	0.0000	1.0000	1.0000
Random Forest	Combined	Original	1.0000	0.0000	1.0000	1.0000
Random Forest	Combined	Modified	1.0000	0.0000	1.0000	1.0000
LightGBM	Original	Original	1.0000	0.0000	1.0000	1.0000
LightGBM	Original	Modified	1.0000	0.0000	1.0000	1.0000
LightGBM	Modified	Original	1.0000	0.0000	1.0000	1.0000
LightGBM	Modified	Modified	1.0000	0.0000	1.0000	1.0000
LightGBM	Combined	Original	1.0000	0.0000	1.0000	1.0000
LightGBM	Combined	Modified	1.0000	0.0000	1.0000	1.0000
TabNet	Original	Original	1.0000	0.0000	1.0000	1.0000
TabNet	Original	Modified	1.0000	0.0000	1.0000	1.0000
TabNet	Modified	Original	0.0000	0.0000	0.6448	0.0000
TabNet	Modified	Modified	1.0000	0.0000	1.0000	1.0000
TabNet	Combined	Original	1.0000	0.0000	1.0000	1.0000
TabNet	Combined	Modified	1.0000	0.0000	1.0000	1.0000

Table 6.6. Supervised ML Binary Models Performance on SSDP Flood Attack Under LSTM Reshaping

Model	Train Type	Test Set	TPR	FPR	Accuracy	F1 Score
Random Forest	Original	Original	1.0000	0.0000	1.0000	1.0000
Random Forest	Original	Modified	1.0000	0.0000	1.0000	1.0000
Random Forest	Modified	Original	1.0000	0.0000	1.0000	1.0000
Random Forest	Modified	Modified	1.0000	0.0000	1.0000	1.0000
Random Forest	Combined	Original	1.0000	0.0000	1.0000	1.0000
Random Forest	Combined	Modified	1.0000	0.0000	1.0000	1.0000
LightGBM	Original	Original	1.0000	0.0000	1.0000	1.0000
LightGBM	Original	Modified	1.0000	0.0000	1.0000	1.0000
LightGBM	Modified	Original	0.2639	0.0000	0.7386	0.4177
LightGBM	Modified	Modified	1.0000	0.0000	1.0000	1.0000
LightGBM	Combined	Original	1.0000	0.0000	1.0000	1.0000
LightGBM	Combined	Modified	1.0000	0.0000	1.0000	1.0000
TabNet	Original	Original	1.0000	0.0000	1.0000	1.0000
TabNet	Original	Modified	1.0000	0.0000	1.0000	1.0000
TabNet	Modified	Original	0.0000	0.0000	0.6448	0.0000
TabNet	Modified	Modified	1.0000	0.0000	1.0000	1.0000
TabNet	Combined	Original	1.0000	0.0000	1.0000	1.0000
TabNet	Combined	Modified	1.0000	0.0000	1.0000	1.0000

Table 6.7. Supervised ML Binary Models Performance on Mirai Attack Under Simple Reshaping

Model	Train Type	Test Set	TPR	FPR	Accuracy	F1 Score
Random Forest	Original	Original	1.0000	0.0000	1.0000	1.0000
Random Forest	Original	Modified	0.0000	0.0000	0.1808	0.0000
Random Forest	Modified	Original	1.0000	0.0000	1.0000	1.0000
Random Forest	Modified	Modified	1.0000	0.0000	1.0000	1.0000
Random Forest	Combined	Original	1.0000	0.0000	1.0000	1.0000
Random Forest	Combined	Modified	1.0000	0.0000	1.0000	1.0000
LightGBM	Original	Original	1.0000	0.0000	1.0000	1.0000
LightGBM	Original	Modified	0.0000	0.0000	0.1808	0.0000
LightGBM	Modified	Original	1.0000	0.0000	1.0000	1.0000
LightGBM	Modified	Modified	1.0000	0.0000	1.0000	1.0000
LightGBM	Combined	Original	1.0000	0.0000	1.0000	1.0000
LightGBM	Combined	Modified	1.0000	0.0000	1.0000	1.0000
TabNet	Original	Original	1.0000	0.0000	1.0000	1.0000
TabNet	Original	Modified	0.0000	0.0000	0.1808	0.0000
TabNet	Modified	Original	0.0000	0.0000	0.1808	0.0000
TabNet	Modified	Modified	1.0000	0.0000	1.0000	1.0000
TabNet	Combined	Original	1.0000	0.0000	1.0000	1.0000
TabNet	Combined	Modified	1.0000	0.0000	1.0000	1.0000

Table 6.8. Supervised ML Binary Models Performance on Mirai Attack Under LSTM Reshaping

Model	Train Type	Test Set	TPR	FPR	Accuracy	F1 Score
Random Forest	Original	Original	1.0000	0.0000	1.0000	1.0000
Random Forest	Original	Modified	0.0000	0.0000	0.1808	0.0000
Random Forest	Modified	Original	1.0000	0.0000	1.0000	1.0000
Random Forest	Modified	Modified	1.0000	0.0000	1.0000	1.0000
Random Forest	Combined	Original	1.0000	0.0000	1.0000	1.0000
Random Forest	Combined	Modified	1.0000	0.0000	1.0000	1.0000
LightGBM	Original	Original	1.0000	0.0000	1.0000	1.0000
LightGBM	Original	Modified	0.0000	0.0000	0.1808	0.0000
LightGBM	Modified	Original	1.0000	0.0000	1.0000	1.0000
LightGBM	Modified	Modified	1.0000	0.0000	1.0000	1.0000
LightGBM	Combined	Original	1.0000	0.0000	1.0000	1.0000
LightGBM	Combined	Modified	1.0000	0.0000	1.0000	1.0000
TabNet	Original	Original	1.0000	0.0000	1.0000	1.0000
TabNet	Original	Modified	0.0000	0.0000	0.1808	0.0000
TabNet	Modified	Original	0.0000	0.0000	0.1808	0.0000
TabNet	Modified	Modified	1.0000	0.0000	1.0000	1.0000
TabNet	Combined	Original	1.0000	0.0000	1.0000	1.0000
TabNet	Combined	Modified	1.0000	0.0000	1.0000	1.0000

Table 6.9. Supervised ML Binary Models Performance on Fuzzing Attack Under Simple Reshaping

Model	Train Type	Test Set	TPR	FPR	Accuracy	F1 Score
Random Forest	Original	Original	1.0000	0.0000	1.0000	1.0000
Random Forest	Original	Modified	1.0000	0.0000	1.0000	1.0000
Random Forest	Modified	Original	0.0000	0.0000	0.9936	0.0000
Random Forest	Modified	Modified	1.0000	0.0000	1.0000	1.0000
Random Forest	Combined	Original	1.0000	0.0000	1.0000	1.0000
Random Forest	Combined	Modified	1.0000	0.0000	1.0000	1.0000
LightGBM	Original	Original	1.0000	0.0000	1.0000	1.0000
LightGBM	Original	Modified	1.0000	0.0000	1.0000	1.0000
LightGBM	Modified	Original	0.1532	0.0002	0.9945	0.2602
LightGBM	Modified	Modified	0.9994	0.0002	0.9998	0.9878
LightGBM	Combined	Original	1.0000	0.0000	1.0000	1.0000
LightGBM	Combined	Modified	1.0000	0.0000	1.0000	1.0000
TabNet	Original	Original	1.0000	0.0000	1.0000	0.9988
TabNet	Original	Modified	0.9976	0.0000	1.0000	0.9976
TabNet	Modified	Original	0.6787	0.0000	0.9980	0.8086
TabNet	Modified	Modified	1.0000	0.0000	1.0000	1.0000
TabNet	Combined	Original	1.0000	0.0000	1.0000	1.0000
TabNet	Combined	Modified	1.0000	0.0000	1.0000	1.0000

Table 6.10. Supervised ML Binary Models Performance on Fuzzing Attack Under LSTM Reshaping

Model	Train Type	Test Set	TPR	FPR	Accuracy	F1 Score
Random Forest	Original	Original	1.0000	0.0000	1.0000	1.0000
Random Forest	Original	Modified	1.0000	0.0000	1.0000	1.0000
Random Forest	Modified	Original	0.0000	0.0000	0.9936	0.0000
Random Forest	Modified	Modified	1.0000	0.0000	1.0000	1.0000
Random Forest	Combined	Original	1.0000	0.0000	1.0000	1.0000
Random Forest	Combined	Modified	1.0000	0.0000	1.0000	1.0000
LightGBM	Original	Original	1.0000	0.0000	1.0000	1.0000
LightGBM	Original	Modified	1.0000	0.0000	1.0000	1.0000
LightGBM	Modified	Original	0.5471	0.0000	0.9971	0.7067
LightGBM	Modified	Modified	1.0000	0.0000	1.0000	0.9994
LightGBM	Combined	Original	1.0000	0.0000	1.0000	1.0000
LightGBM	Combined	Modified	1.0000	0.0000	1.0000	1.0000
TabNet	Original	Original	1.0000	0.0000	1.0000	0.9988
TabNet	Original	Modified	0.9976	0.0000	1.0000	0.9976
TabNet	Modified	Original	0.9904	0.0000	0.9999	0.9922
TabNet	Modified	Modified	0.9988	0.0000	1.0000	0.9964
TabNet	Combined	Original	1.0000	0.0000	1.0000	1.0000
TabNet	Combined	Modified	1.0000	0.0000	1.0000	1.0000

Table 6.11. Supervised ML Binary Models Performance on Active Wiretap Attack Under Simple Reshaping

Model	Train Type	Test Set	TPR	FPR	Accuracy	F1 Score
Random Forest	Original	Original	1.0000	0.0120	0.9884	0.8604
Random Forest	Original	Modified	1.0000	0.0120	0.9884	0.8604
Random Forest	Modified	Original	0.0868	0.0127	0.9553	0.1215
Random Forest	Modified	Modified	1.0000	0.0127	0.9878	0.8536
Random Forest	Combined	Original	1.0000	0.0085	0.9918	0.8973
Random Forest	Combined	Modified	1.0000	0.0085	0.9918	0.8973
LightGBM	Original	Original	0.6817	0.0221	0.9673	0.5977
LightGBM	Original	Modified	0.0000	0.0221	0.9430	0.0000
LightGBM	Modified	Original	0.0860	0.0179	0.9502	0.1095
LightGBM	Modified	Modified	1.0000	0.0179	0.9827	0.8049
LightGBM	Combined	Original	0.8189	0.0176	0.9766	0.7133
LightGBM	Combined	Modified	1.0000	0.0176	0.9830	0.8074
TabNet	Original	Original	1.0000	0.0001	0.9999	0.9986
TabNet	Original	Modified	1.0000	0.0001	0.9999	0.9986
TabNet	Modified	Original	0.9742	0.0027	0.9964	0.9512
TabNet	Modified	Modified	1.0000	0.0027	0.9974	0.9643
TabNet	Combined	Original	1.0000	0.0000	1.0000	1.0000
TabNet	Combined	Modified	1.0000	0.0000	1.0000	1.0000

Table 6.12. Supervised ML Binary Models Performance on Active Wiretap Attack Under LSTM Reshaping

Model	Train Type	Test Set	TPR	FPR	Accuracy	F1 Score
Random Forest	Original	Original	1.0000	0.0120	0.9884	0.8604
Random Forest	Original	Modified	1.0000	0.0120	0.9884	0.8604
Random Forest	Modified	Original	0.5589	0.0132	0.9715	0.5830
Random Forest	Modified	Modified	1.0000	0.0132	0.9872	0.8480
Random Forest	Combined	Original	1.0000	0.0132	0.9872	0.8479
Random Forest	Combined	Modified	1.0000	0.0132	0.9872	0.8479
LightGBM	Original	Original	0.6817	0.0221	0.9673	0.5977
LightGBM	Original	Modified	0.0000	0.0221	0.9430	0.0000
LightGBM	Modified	Original	0.0000	0.3858	0.5923	0.0000
LightGBM	Modified	Modified	1.0000	0.3858	0.6279	0.1607
LightGBM	Combined	Original	0.8189	0.0142	0.9799	0.7438
LightGBM	Combined	Modified	1.0000	0.0142	0.9864	0.8392
TabNet	Original	Original	1.0000	0.0001	0.9999	0.9986
TabNet	Original	Modified	0.9860	0.0001	0.9994	0.9916
TabNet	Modified	Original	1.0000	0.0002	0.9998	0.9971
TabNet	Modified	Modified	1.0000	0.0002	0.9998	0.9971
TabNet	Combined	Original	1.0000	0.0008	0.9993	0.9897
TabNet	Combined	Modified	1.0000	0.0008	0.9993	0.9897

A major trend across the binary classification results, hinted at earlier when discussing the confusion matrices, is that supervised models generally detect the type of traffic they are trained on. When a model is trained on original traffic and tested on original traffic, performance is usually high. Similarly, when a model is trained on modified traffic and tested on modified traffic, performance is also usually high. For example, Tables 6.5 and 6.6 show that in the SSDP flood attack, almost every combination of model, train type, and test type achieves perfect performance. The main exceptions are Modified-trained TabNet evaluated on the Original test set under both reshaping strategies and Modified-trained LightGBM evaluated on the Original test set under LSTM reshaping. Similarly, Tables 6.7 and 6.8 demonstrate that, in the Mirai attack, all models trained on Modified traffic achieve perfect performance on the Modified test set. In the fuzzing attack, models trained on Modified traffic also perform well on the Modified test set, as Tables 6.9 and 6.10 show that RF, LightGBM, and TabNet reach perfect or near-perfect performance in these training and testing configurations. Tables 6.11 and 6.12 tell a similar story for the active wiretap attack. Although models seem to struggle the most against this intrusion attack, models trained on Modified traffic are generally able to detect Modified traffic well. The only major exception is the Modified LightGBM model under LSTM reshaping. On the Modified test set, the model detects all malicious packets, achieving a TPR of 1.0000; however, it also incorrectly labels many benign packets as malicious, producing a high FPR of 0.3858 and reducing the F1 score to 0.1607.

Another discovery is that whether reshaped traffic can evade models trained only on original traffic depends on the attack and the model. For example, in the Mirai attack under both Simple and LSTM reshaping, as shown in Tables 6.7 and 6.8, all three models trained on Original traffic fail when evaluated on the Modified test set, exhibiting a TPR and F1 score of 0.0000. However, in the SSDP flood and fuzzing attacks, models trained on Original traffic usually detect Modified traffic very well. Under both SSDP flood reshaping strategies, as shown in Tables 6.5 and 6.6, all three models trained on Original traffic achieve a TPR of 1.0000 on the Modified test set. The fuzzing attack results, displayed in Tables 6.9 and 6.10, show a similar trend, with Original-trained RF and LightGBM achieving a TPR of 1.0000 and Original-trained TabNet achieving a TPR of 0.9976 on the Modified test set. For the

active wiretap attack, Tables 6.11 and 6.12 reveal that the results are model-dependent. RF and TabNet trained on Original traffic detect Modified traffic well, but LightGBM fails under both Simple and LSTM reshaping, achieving a TPR and F1 score of 0.0000.

One of the most consistent trends discovered is that Combined training performs best across both Original and Modified test sets. For the SSDP flood, Mirai, and fuzzing attacks, models trained on Combined traffic almost always achieve perfect results on both Original and Modified test sets. In particular, all Combined-trained models in the SSDP flood, Mirai, and fuzzing attack tables achieve a TPR and F1 score of 1.0000 on both Original and Modified test sets. Although active wiretap proves to be a more difficult attack to detect across all models, Combined training still improves robustness. For example, the Combined-trained TabNet model achieves perfect performance under the Simple reshaping strategy and near-perfect performance under the LSTM reshaping strategy. In addition, while Combined-trained LightGBM does not reach perfect performance, it still performs better across both test sets than LightGBM trained only on Original or only on Modified traffic.

In comparing performance across the three supervised ML model types, RF and TabNet models perform the best overall while LightGBM is less stable and consistent. RF is very stable across many attacks and often achieves perfect detection, even when evaluated on a test set containing traffic it was not trained on. For example, in the SSDP flood attack under both reshaping strategies, all three RF models achieve perfect performance on both Original and Modified test sets, achieving a TPR and F1 score of 1.0000. In addition, Combined-trained RF achieves perfect performance on both Original and Modified test sets across the SSDP flood, Mirai, and fuzzing attacks. Although RF struggles against the active wiretap attack, the Combined-trained models still achieve an F1 score of more than 0.8400 under both reshaping strategies, performing better than LightGBM in both cases. On the other hand, TabNet appears to be the strongest under Combined training and particularly performs well on the active wiretap attack. Across the SSDP flood, Mirai, and fuzzing attacks, Combined-trained TabNet achieves a TPR and F1 score of 1.0000. Combined-trained TabNet also achieves perfect performance on the active wiretap attack under Simple reshaping with an F1 score of 1.0000 and near-perfect performance on the active wiretap attack under LSTM reshaping with an F1 score of 0.9897. However, TabNet is not uniformly robust in cross-

distribution settings. For example, according to the Mirai attack results, while Modified-trained RF achieves perfect performance on the Original test set, Modified-trained TabNet completely fails on the Original test set with a TPR of 0.0000.

LightGBM also performs well in many cases, especially when the training and testing distributions match. For example, LightGBM achieves perfect or near-perfect performance when trained and tested on the same traffic condition for the SSDP flood, Mirai, and fuzzing attacks. In addition, Combined-trained LightGBM achieves perfect performance on both Original and Modified test sets for the SSDP flood, Mirai, and fuzzing attacks under both reshaping strategies. However, LightGBM is less consistent than RF and TabNet because it experiences larger performance drops in several cross-distribution settings. For example, in the active wiretap experiments, Original-trained LightGBM fails to detect Modified traffic under both Simple and LSTM reshaping, producing a TPR and F1 score of 0.0000. LightGBM also performs poorly when trained on Modified active wiretap traffic under LSTM reshaping, where it achieves a TPR of 1.0000 on the Modified test set but also produces a high FPR of 0.3858, resulting in an F1 score of only 0.1607. Therefore, while LightGBM can perform very well when the training data is representative of the test condition, it appears to be more sensitive to distribution shifts between original and reshaped traffic than RF and TabNet.

Another notable trend discovered through these experiments is that the supervised ML results under the Simple reshaping strategy are often similar to the results under the LSTM reshaping strategy. For example, for the Mirai attack, the Simple and LSTM results are identical. The results are also mostly identical for SSDP flood with the exception of Modified-trained LightGBM evaluated against the Original test set. For the fuzzing attack, LSTM reshaping produces better Modified-to-Original generalization for LightGBM and TabNet than Simple reshaping. For the active wiretap attack, LSTM reshaping is more difficult for LightGBM, especially because of the high FPR in the Modified-trained LightGBM case. These results suggest that the specific reshaping strategy has less impact on supervised ML performance than the relationship between the training distribution and the test distribution. In other words, the simple fixed IAT reshaping attack often produces detection trends similar to the more complex LSTM-based reshaping attack.

As another notable observation, supervised ML models seem to struggle more against the active wiretap attack than the other three evaluated intrusion attacks. One possible explanation for the lower overall performance on active wiretap is that this attack produces a less separable feature distribution than the other evaluated attacks. Attacks such as SSDP flood, Mirai, and fuzzing appear to create more distinct changes in the AfterImage feature space, allowing the supervised models to separate benign and malicious traffic more easily. In contrast, active wiretap may produce traffic patterns that are closer to benign behavior, especially because the attack is not primarily characterized by an obvious flooding or high-volume communication pattern. As a result, the models are more likely to confuse benign and malicious traffic, leading to higher false positive rates and lower F1 scores. This is especially visible for LightGBM, which is more sensitive to the active wiretap distribution than RF and TabNet.

6.3.2 Most Important Features Results and Analysis

In addition to evaluating binary classification performance, supervised ML models are used to investigate which AfterImage features are most useful for distinguishing benign traffic from malicious traffic, either unmodified or reshaped. Leveraging feature importance attributes from each of the evaluated supervised ML model types, the top five feature-importance rankings of each trained model are identified across every intrusion attack and reshaping strategy. Table 6.13 shows the top five most important features of each trained model in the SSDP flood attack under the Simple reshaping strategy, where RF stands for Random Forest, LGBM stands for LightGBM, and TN stands for TabNet. For feature-importance rankings for the rest of the attacks and reshaping strategies, see Appendix B. Across these eight feature-importance tables, there are 360 total top-five feature entries, corresponding to four intrusion attacks, two reshaping strategies, three model types, three training configurations, and five ranked features per model configuration. To improve readability, this section summarizes rankings by feature template, feature family, time window, model type, and training configuration. For descriptions of the AfterImage feature names and feature templates, see Section 3.2.4.

Table 6.13. Supervised ML Binary Models Top 5 Most Important Features in SSDP Flood Attack Under Simple Reshaping

Model	Rank	Feature Name	Importance (%)
RF Original	1	HH_3_radius_0_1	11.9961
RF Original	2	HH_jit_5_weight	10.0553
RF Original	3	HH_5_radius_0_1	7.9997
RF Original	4	MI_dir_3_weight	6.9960
RF Original	5	MI_dir_5_weight	5.9995
RF Modified	1	HH_5_magnitude_0_1	11.9978
RF Modified	2	MI_dir_0.01_std	10.9954
RF Modified	3	MI_dir_0.1_std	9.9945
RF Modified	4	HH_5_radius_0_1	8.9923
RF Modified	5	HH_3_magnitude_0_1	6.1487
RF Combined	1	HH_5_magnitude_0_1	11.9978
RF Combined	2	MI_dir_0.01_std	11.0041
RF Combined	3	MI_dir_0.1_std	10.0785
RF Combined	4	HH_5_radius_0_1	8.9987
RF Combined	5	HH_1_magnitude_0_1	6.1369
LGBM Original	1	HH_5_magnitude_0_1	99.9315
LGBM Original	2	MI_dir_5_weight	0.0631
LGBM Original	3	HH_jit_0.01_weight	0.0030
LGBM Original	4	HpHp_0.01_weight	0.0022
LGBM Original	5	MI_dir_0.01_std	0.0001
LGBM Modified	1	MI_dir_0.01_std	99.9115
LGBM Modified	2	HH_0.1_weight	0.0885
LGBM Modified	3	MI_dir_5_std	0.0000
LGBM Modified	4	HH_3_magnitude_0_1	0.0000
LGBM Modified	5	HH_3_radius_0_1	0.0000
LGBM Combined	1	HH_1_magnitude_0_1	99.9257
LGBM Combined	2	HH_0.1_weight	0.0554
LGBM Combined	3	HpHp_0.01_weight	0.0145
LGBM Combined	4	HH_jit_0.01_weight	0.0042
LGBM Combined	5	MI_dir_0.01_std	0.0001
TN Original	1	HpHp_0.01_weight	15.4483
TN Original	2	MI_dir_1_weight	10.7436
TN Original	3	MI_dir_3_std	9.3740
TN Original	4	HpHp_0.1_pcc_0_1	8.6292
TN Original	5	HH_jit_0.01_weight	7.5202
TN Modified	1	HpHp_0.01_weight	19.2372
TN Modified	2	HpHp_0.01_covariance_0_1	14.3335
TN Modified	3	HH_0.1_radius_0_1	13.5649
TN Modified	4	HH_5_radius_0_1	10.8643
TN Modified	5	HpHp_0.01_radius_0_1	9.6101
TN Combined	1	HH_5_radius_0_1	11.6349
TN Combined	2	HH_5_weight	10.9265
TN Combined	3	MI_dir_3_mean	8.4907
TN Combined	4	HpHp_1_weight	7.9583
TN Combined	5	HpHp_0.1_mean	6.9433

Across the three supervised ML models, feature importance is computed differently. RF and TabNet both produce normalized feature-importance values while LightGBM produces non-normalized split-based and gain-based importance values. Gain-based importance values more directly reflect the contribution of a feature to reducing model loss, so these values are normalized and used as LightGBM's feature importance values. Gain-based importance values are often highly concentrated in a small number of features, which is why relative importance values are often substantially unbalanced for LightGBM's most important features. For example, in Table 6.13, the relative importance of the `HH_5_magnitude_0_1` feature for the Original-trained LightGBM model is 99.93%.

Table 6.14 shows the feature template counts across all 360 top-five feature entries in the binary classification feature-importance results. In each feature template, λ represents any of the five AfterImage decay factors: $5, 3, 1, 0.1, 0.01$. The template counts indicate that the most used features overall are those that convey packet-count behavior, host-to-host traffic magnitude, packet-size variability, and directional source behavior. The `MI_dir` family is particularly frequently used, with all three statistics ranking within the top six. The two most frequent templates, `MI_dir_`λ`_weight` and `HH_`λ`_magnitude_0_1`, account for nearly one fourth of all top-five feature entries, suggesting that source MAC/IP packet-count behavior and host-to-host traffic-size magnitude are highly informative. In addition, the high rankings of `MI_dir_`λ`_std`, `HH_`λ`_radius_0_1`, and `HH_jit_`λ`_std` demonstrate that variability-based features are frequently important.

The `HH_jit` feature family templates consistently rank in the middle of the table, suggesting that explicit timing features may be useful in some situations but are not the only important indicators of malicious traffic. This is significant because while the evaluated reshaping attacks modify packet timing, the most common feature templates also include packet-count, packet-size, magnitude, and source-directional statistics. In addition, the `HpHp` family shows a more uneven pattern. With the exception of `HpHp_`λ`_weight`, the `HpHp` templates appear near the bottom of the table, occupying ranks 14–19. These rankings suggest that socket-level features are less consistently selected among the top five features.

Table 6.14. Feature Template Counts Across Binary Classification Feature-Importance Results

Rank	Feature Template	Count	Percent (%)
1	MI_dir_λ_weight	45	12.5
2	HH_λ_magnitude_0_1	44	12.2
3	MI_dir_λ_std	38	10.6
4	HpHp_λ_weight	33	9.2
5	HH_λ_radius_0_1	27	7.5
6	MI_dir_λ_mean	27	7.5
7	HH_λ_weight	21	5.8
8	HH_jit_λ_std	21	5.8
9	HH_jit_λ_mean	18	5.0
10	HH_λ_pcc_0_1	17	4.7
11	HH_jit_λ_weight	16	4.4
12	HH_λ_std	16	4.4
13	HH_λ_mean	9	2.5
14	HpHp_λ_std	8	2.2
15	HpHp_λ_radius_0_1	6	1.7
16	HpHp_λ_magnitude_0_1	5	1.4
17	HpHp_λ_covariance_0_1	4	1.1
18	HpHp_λ_pcc_0_1	3	0.8
19	HpHp_λ_mean	2	0.6
20	HH_λ_covariance_0_1	0	0.0

Table 6.15 summarizes the top five feature-importance rankings by feature family for each intrusion attack. Across all attacks, the most common feature families are `MI_dir` and `HH`, appearing a total of 110 and 134 times, respectively. This suggests that directional MAC-IP statistics and host-to-host communication statistics are the most consistently useful feature families in this binary classification setting.

Table 6.15. Top-Five Binary Classification Feature Counts by Feature Family and Attack

Attack	MI_dir	HH	HH_jit	HpHp
SSDP Flood	30	34	8	18
Mirai	44	23	9	14
Fuzzing	17	54	12	7
Active Wiretap	19	23	26	22
Total	110	134	55	61

Looking at feature family counts at the attack level reveals that models trained on different intrusion attack traffic rely on different feature families. SSDP flood and Mirai models rely the most on `MI_dir` and `HH` features, with Mirai relying more heavily on `MI_dir` features than `HH` features. This suggests both host-to-host communication features and source-directional behavior features are especially important for detecting these attacks. Fuzzing models rely most heavily on `HH` features, suggesting that host-to-host communication magnitude and variability may be the strongest indicators for this attack.

Active wiretap differs the most from the other attacks because its feature-family counts are more evenly distributed across `MI_dir`, `HH`, `HH_jit`, and `HpHp`. The most frequent feature family is `HH_jit`, suggesting that active wiretap is more timing-sensitive than the other attacks. In addition, because the counts are spread across all four feature families, active wiretap appears to rely on a broader set of detection cues rather than one dominant feature family. This may help explain why the supervised ML models generally performed worse against active wiretap than against the other three attacks.

Table 6.16 summarizes the top five feature-importance rankings by decay factor for each intrusion attack. With each decay factor corresponding to a different effective time window,

this table assists in evaluating whether important features depend on the effective time window.

Table 6.16. Top-Five Binary Classification Feature Counts by Time Window and Attack

Attack	$\boldsymbol{\lambda = 0.01}$	$\boldsymbol{\lambda = 0.1}$	$\boldsymbol{\lambda = 1}$	$\boldsymbol{\lambda = 3}$	$\boldsymbol{\lambda = 5}$
SSDP Flood	25	14	9	13	29
Mirai	6	39	24	11	10
Fuzzing	24	18	21	14	13
Active Wiretap	27	20	2	21	20
Total	82	91	56	59	72

The decay factor counts show that important features do depend on the effective time window, but no single window dominates every attack. Features using $\lambda = 0.1$ and $\lambda = 0.01$ appear most often, demonstrating that features capturing traffic statistics over longer periods of time are frequently useful. At the attack level, models in the SSDP flood attack setting rely heavily on $\lambda = 5$ features, conveying the importance of short-term burst behavior information. Mirai models rely strongly on $\lambda = 0.1$ and $\lambda = 1$ features, indicating a reliance on medium-term source behavior. For the fuzzing attack, models use a more balanced set of windows, but $\lambda = 0.01$ features are still the most frequently used. For the active wiretap attack, models use features from several effective time windows, especially $\lambda \in \{0.01, 0.1, 3, 5\}$. These results suggest that the most informative effective time window depends on the attack.

Another important question is whether models trained on original malicious traffic and models trained on modified malicious traffic rely on the same features. Table 6.17 summarizes the average overlap between top-five feature sets for the Original, Modified, and Combined training configurations for each matching intrusion attack, reshaping strategy, and model. Feature overlap in the table is based on exact feature names, including the λ value, and ranking order is ignored.

The low overlap between Original and Modified feature rankings indicates that models trained on original malicious traffic and models trained on reshaped malicious traffic often rely on different AfterImage features. On average, the top-five feature lists for Original-trained and Modified-trained models share fewer than one feature, suggesting that traffic

reshaping attacks change the feature cues used by supervised ML models. Combined-trained models share more features with the corresponding Modified-trained models than with the corresponding Original-trained models, suggesting that reshaped traffic introduces distinctive patterns that remain important even when the model is trained on both original and modified malicious traffic.

Table 6.17. Average Top-Five Binary Classification Feature Overlap Between Training Configurations

Comparison	Average Shared Features
Original vs. Modified	0.83 out of 5
Original vs. Combined	0.96 out of 5
Modified vs. Combined	1.83 out of 5

Table 6.18 summarizes the top five feature-importance rankings by feature family for each supervised ML model type. The results suggest that the three model types rely on related but not identical feature families. RF relies most heavily on `HH` features followed by `MI_dir` features, suggesting a reliance on host-to-host communication behavior and directional source behavior information. LightGBM selects the largest number of `MI_dir` features and also relies more on `HH_jit` features than either RF or TabNet, indicating that it is more sensitive to source-directional and timing-related cues. TabNet relies substantially on `HH` features but also selects more `HpHp` features than the other two models, conveying a greater use of socket-level communication patterns.

Table 6.18. Top-Five Binary Classification Feature Counts by Feature Family and Model Type

Model	`MI_dir`	`HH`	`HH_jit`	`HpHp`
Random Forest	39	54	17	10
LightGBM	43	31	26	20
TabNet	28	49	12	31

Table 6.19 summarizes the top five feature-importance rankings by feature family for each evaluated reshaping strategy. Under Simple reshaping, the top-five rankings contain 55

MI_dir, 66 HH, 29 HH_jit, and 30 HpHp features. Under LSTM reshaping, the corresponding counts are 55, 68, 26, and 31. The highly similar corresponding counts suggest that the Simple and LSTM reshaping strategies produce similar broad feature-importance patterns. Therefore, the feature-importance results further suggest that the Simple fixed IAT reshaping attack produces feature-importance trends that are similar to those produced by the more complex LSTM-based reshaping attack.

Table 6.19. Top-Five Binary Classification Feature Counts by Feature Family and Reshaping Strategy

Reshaping	MI_dir	HH	HH_jit	HpHp
Simple	55	66	29	30
LSTM	55	68	26	31

6.4 Supervised ML Multiclass Classification Models Against Traffic Reshaping Attacks

Moving on from the binary classification setting, the third and final set of experiments explored in this section aims to address the effectiveness of traffic reshaping strategies against supervised ML models in a multiclass classification setting. These experiments address the question of whether supervised ML models can be trained to distinguish among benign traffic, original malicious traffic, and reshaped malicious traffic. In addition, each supervised ML model's feature importance attributes are leveraged to investigate which AfterImage features are most useful for distinguishing among benign traffic, original malicious traffic, and reshaped malicious traffic.

The supervised ML models in this set of experiments are trained using examples labeled either benign (class 0), original malicious (class 1), or modified malicious (class 2). For each intrusion attack and reshaping strategy, an RF model, a LightGBM model, and a TabNet model are trained and evaluated. For more information about the multiclass classification setting and dataset, see Section 5.6.

6.4.1 Supervised ML Multiclass Model Performance Results and Analysis

All supervised ML multiclass classification models, hereafter referred to as multiclass models, are evaluated on a test set containing benign traffic, original malicious traffic, and reshaped malicious traffic. Table 6.20 displays the 3×3 confusion matrix demonstrating the performance of a multiclass TabNet model on the fuzzing attack trained under the LSTM reshaping strategy. The model correctly identifies 259 382 examples as benign, 1663 examples as original malicious, and 1665 examples as modified malicious. However, it incorrectly identifies 619 benign examples as original malicious, 1 original malicious example as benign, and 1 original malicious example as modified malicious. Therefore, the model is effective in separating original malicious traffic from modified malicious traffic, but the false positives from benign traffic into the original malicious class reduce the original malicious F1 score.

Table 6.20. Confusion Matrix for the Multiclass TabNet Model on the Fuzzing Attack Under LSTM Reshaping

	Predicted Benign	**Predicted Original Malicious**	**Predicted Modified Malicious**
Ground Truth Benign	259 382	619	0
Ground Truth Original Malicious	1	1663	1
Ground Truth Modified Malicious	0	0	1665

Tables 6.21–6.24 report the multiclass performance results for RF, LightGBM, and TabNet across the SSDP flood, Mirai, fuzzing, and active wiretap attacks and the Simple and LSTM reshaping strategies (the Re. column indicates the reshaping strategy while the Acc. column reports the accuracy). The metrics TPR_O, FPR_O, and $F1_O$ refer to the original malicious class, while TPR_M, FPR_M, and $F1_M$ refer to the modified malicious class. These class-specific one-versus-rest metrics are used to determine whether the multiclass models can distinguish among benign traffic, original malicious traffic, and reshaped malicious traffic rather than only distinguishing between benign traffic and malicious traffic. In addition, the metric $F1_A$ reports the average of $F1_O$ and $F1_M$, providing a single summary measure of performance across the two malicious classes.

Table 6.21. Supervised ML Multiclass Models Performance on SSDP Flood Attack (Re. = Reshaping, Acc. = Accuracy)

Model	Re.	Acc.	TPR_O	TPR_M	FPR_O	FPR_M	$F1_O$	$F1_M$	$F1_A$
RF	Simple	1.0000	1.0000	1.0000	0.0000	0.0000	1.0000	1.0000	1.0000
RF	LSTM	1.0000	1.0000	1.0000	0.0000	0.0000	1.0000	1.0000	1.0000
LGBM	Simple	1.0000	1.0000	1.0000	0.0000	0.0000	1.0000	1.0000	1.0000
LGBM	LSTM	1.0000	1.0000	1.0000	0.0000	0.0000	1.0000	1.0000	1.0000
TN	Simple	1.0000	1.0000	1.0000	0.0000	0.0000	1.0000	1.0000	1.0000
TN	LSTM	1.0000	1.0000	1.0000	0.0000	0.0000	1.0000	1.0000	1.0000

Table 6.22. Supervised ML Multiclass Models Performance on Mirai Attack

Model	Re.	Acc.	TPR_O	TPR_M	FPR_O	FPR_M	$F1_O$	$F1_M$	$F1_A$
RF	Simple	1.0000	1.0000	1.0000	0.0000	0.0000	1.0000	1.0000	1.0000
RF	LSTM	1.0000	1.0000	1.0000	0.0000	0.0000	1.0000	1.0000	1.0000
LGBM	Simple	1.0000	1.0000	1.0000	0.0000	0.0000	1.0000	1.0000	1.0000
LGBM	LSTM	1.0000	1.0000	1.0000	0.0000	0.0000	1.0000	1.0000	1.0000
TN	Simple	1.0000	1.0000	1.0000	0.0000	0.0000	1.0000	1.0000	1.0000
TN	LSTM	1.0000	1.0000	1.0000	0.0000	0.0000	1.0000	1.0000	1.0000

Table 6.23. Supervised ML Multiclass Models Performance on Fuzzing Attack

Model	Re.	Acc.	TPR_O	TPR_M	FPR_O	FPR_M	$F1_O$	$F1_M$	$F1_A$
RF	Simple	1.0000	1.0000	1.0000	0.0000	0.0000	1.0000	1.0000	1.0000
RF	LSTM	1.0000	1.0000	1.0000	0.0000	0.0000	1.0000	1.0000	1.0000
LGBM	Simple	0.9994	1.0000	0.9964	0.0001	0.0005	0.9914	0.9631	0.9773
LGBM	LSTM	1.0000	1.0000	1.0000	0.0000	0.0000	1.0000	1.0000	1.0000
TN	Simple	0.9990	0.8511	0.9886	0.0001	0.0009	0.9142	0.9250	0.9196
TN	LSTM	0.9976	0.9988	1.0000	0.0024	0.0000	0.8427	0.9997	0.9212

Table 6.24. Supervised ML Multiclass Models Performance on Active Wiretap Attack

Model	Re.	Acc.	TPR_O	TPR_M	FPR_O	FPR_M	$F1_O$	$F1_M$	$F1_A$
RF	Simple	0.9933	1.0000	1.0000	0.0069	0.0000	0.9113	1.0000	0.9556
RF	LSTM	0.9978	1.0000	1.0000	0.0023	0.0000	0.9689	1.0000	0.9844
LGBM	Simple	0.9792	0.8189	1.0000	0.0149	0.0002	0.7318	0.9976	0.8647
LGBM	LSTM	0.9806	0.8189	1.0000	0.0124	0.0012	0.7553	0.9835	0.8694
TN	Simple	1.0000	1.0000	1.0000	0.0000	0.0000	0.9993	1.0000	0.9997
TN	LSTM	1.0000	1.0000	1.0000	0.0000	0.0000	0.9997	1.0000	0.9998

One major trend across the multiclass results is that multiclass models can often distinguish among benign, original malicious, and reshaped malicious traffic with very high accuracy. For example, in the SSDP flood and Mirai attacks, Tables 6.21 and 6.22 show that all three model types achieve perfect performance under both Simple and LSTM reshaping strategies. For both attacks, RF, LightGBM, and TabNet achieve an accuracy, TPR_O, TPR_M, and $F1_A$ of 1.0000, with both FPR_O and FPR_M equal to 0.0000. These results suggest that, for SSDP flood and Mirai, the AfterImage feature space contains enough information for supervised models to distinguish original malicious traffic from reshaped malicious traffic while also separating both classes from benign traffic.

The fuzzing attack is also largely separable in the multiclass setting, but its results show more variation across model types. As displayed in Table 6.23, RF achieves perfect performance under both Simple and LSTM reshaping. LightGBM also performs very well, achieving perfect performance under LSTM reshaping and near-perfect performance under Simple reshaping. For example, under Simple reshaping, LightGBM achieves an accuracy of 0.9994, $TPR_O = 1.0000$, and $TPR_M = 0.9964$, indicating that it detects original malicious traffic perfectly but misses a small portion of modified malicious traffic. TabNet performs well overall, but its class-specific results reveal that it is less consistent than RF. Under Simple reshaping, TabNet achieves $TPR_O = 0.8511$ and $F1_O = 0.9142$, indicating that some original malicious examples are not correctly assigned to the original malicious class. Under LSTM reshaping, TabNet achieves $TPR_O = 0.9988$ and $TPR_M = 1.0000$, but $F1_O$ is only 0.8427 due to low precision in the original malicious class, as shown in the confusion matrix in Table 6.20. Thus, despite some performance drops in the TabNet models, multiclass models can still largely distinguish among benign, original malicious, and modified malicious traffic in the fuzzing attack.

Just like in the binary classification results, the active wiretap attack remains the most difficult attack for the supervised ML models. Table 6.24 demonstrates that TabNet performs the best out of all the model types on this attack, achieving perfect or near-perfect results under both reshaping strategies. RF also performs strongly, achieving $TPR_O = 1.0000$ and $TPR_M = 1.0000$ under both Simple and LSTM reshaping. However, its $F1_O$ is lower than its $F1_M$, especially under Simple reshaping, where $F1_O = 0.9113$. This indicates

that some benign traffic is incorrectly assigned to the original malicious class. LightGBM struggles the most on active wiretap. Under both reshaping strategies, LightGBM achieves $TPR_M = 1.0000$, demonstrating that it detects modified malicious traffic well, but TPR_O is only 0.8189. Its original malicious F1 scores are also lower, with $F1_O = 0.7318$ under Simple reshaping and $F1_O = 0.7553$ under LSTM reshaping. These results suggest that LightGBM has more difficulty distinguishing original active wiretap traffic from the other classes than RF or TabNet, supporting the results found in the binary classification experiments.

Another trend made apparent by these multiclass experiments is that the modified malicious class is generally detected very well. Across most attacks, models achieve $TPR_M = 1.0000$ or near 1.0000, as the results for the SSDP flood, Mirai, and active wiretap attacks demonstrate. In contrast, the original malicious class is more likely to show reduced F1 scores, particularly for fuzzing and active wiretap. These results suggest that, in the multiclass setting, reshaped malicious traffic often forms a distinct enough pattern to be identified as its own class.

The multiclass results also show that the Simple and LSTM reshaping strategies produce similar performance trends. For the SSDP flood and Mirai attacks, the results are identical across both reshaping strategies. For the fuzzing attack, the trends are dependent on the model, as RF remains perfect under both strategies, LightGBM performs better under LSTM reshaping than under Simple reshaping, and TabNet shows different class-specific tradeoffs across the two strategies. For the active wiretap attack, the Simple and LSTM results are also similar, with TabNet performing best, RF performing strongly, and LightGBM showing the weakest original malicious class performance. These results suggest that the specific reshaping strategy has less impact on multiclass performance than the attack type and model type.

In comparing model types, RF is the most consistently strong model across the multiclass experiments. It achieves perfect performance on the SSDP flood, Mirai, and fuzzing attacks under both reshaping strategies and remains strong on the active wiretap attack. TabNet also performs very well, especially on the active wiretap attack, where it outperforms both RF and LightGBM. However, TabNet shows some weakness on the fuzzing attack, where its $F1_O$ score decreases under both reshaping strategies. LightGBM performs perfectly on the

SSDP flood and Mirai attacks and performs well on the fuzzing attack, but it is the least consistent model overall because of its weaker performance on the active wiretap attack, especially for the original malicious class.

6.4.2 Most Important Features Results and Analysis

In addition to evaluating multiclass classification performance, the supervised ML models are used to investigate which AfterImage features are most useful for distinguishing among benign traffic, original malicious traffic, and modified malicious traffic. Leveraging feature importance attributes from each of the evaluated supervised ML model types, the top five feature-importance rankings of each trained multiclass model are identified across every intrusion attack and reshaping strategy. The complete top-five feature-importance rankings for each multiclass model are provided in Appendix C.

Across the four feature-importance tables, there are 120 total top-five feature entries, corresponding to four intrusion attacks, two reshaping strategies, three model types, and five ranked features per model configuration. As in the binary classification analysis, RF and TabNet both produce normalized feature-importance values. For LightGBM, gain-based importance values are normalized and used as the feature importance values. To summarize the main trends, this section aggregates the top-five rankings by feature template, feature family, model type, reshaping strategy, and effective time window. For descriptions of the AfterImage feature names and feature templates, see Section 3.2.4.

Table 6.25 shows the feature template counts across all 120 top-five feature entries in the multiclass classification feature-importance results. In each feature template, λ represents any of the five AfterImage decay factors: $5, 3, 1, 0.1, 0.01$. The results indicate that the most frequently selected features in the multiclass setting include socket-level packet counts, host-to-host traffic magnitude, timing variability, and directional source behavior, as the most common feature templates are `HpHp_λ_weight`, followed by `HH_λ_magnitude_0_1`, `HH_jit_λ_std`, `MI_dir_λ_weight`, `HH_jit_λ_weight`, and `MI_dir_λ_std`.

Compared with the binary classification feature-importance results, the multiclass feature-template distribution is more balanced across feature families. In the binary setting, the most

common templates were strongly dominated by MI_dir and HH features. In the multiclass setting, HH_jit features appear more prominently, and some HpHp features have also moved up in the rankings compared to the binary classification results. This difference suggests that distinguishing original malicious traffic from modified malicious traffic may require more information about timing behavior and socket-level communication patterns.

Table 6.25. Feature Template Counts Across Multiclass Classification Feature-Importance Results

Rank	Feature Template	Count	Percent (%)
1	HpHp_λ_weight	17	14.2
2	HH_λ_magnitude_0_1	15	12.5
3	HH_jit_λ_std	14	11.7
4	MI_dir_λ_weight	13	10.8
5	HH_jit_λ_weight	11	9.2
6	MI_dir_λ_std	10	8.3
7	HH_λ_radius_0_1	8	6.7
8	HH_λ_weight	6	5.0
9	HH_jit_λ_mean	6	5.0
10	MI_dir_λ_mean	5	4.2
11	HpHp_λ_std	5	4.2
12	HpHp_λ_magnitude_0_1	3	2.5
13	HH_λ_pcc_0_1	3	2.5
14	HpHp_λ_mean	1	0.8
15	HH_λ_std	1	0.8
16	HH_λ_covariance_0_1	1	0.8
17	HpHp_λ_radius_0_1	1	0.8
18	HH_λ_mean	0	0.0
19	HpHp_λ_covariance_0_1	0	0.0
20	HpHp_λ_pcc_0_1	0	0.0

Table 6.26 summarizes the top-five feature-importance rankings by feature family for each intrusion attack. Across all attacks, the counts are relatively balanced: MI_dir appears 28 times, HH appears 34 times, HH_jit appears 31 times, and HpHp appears 27 times. This differs from the binary classification setting, where HH and MI_dir were more clearly dominant. These results suggest that, in the multiclass setting, all four feature families contribute meaningfully to distinguishing among benign, original malicious, and modified malicious traffic.

Table 6.26. Top-Five Multiclass Classification Feature Counts by Feature Family and Attack

Attack	MI_dir	HH	HH_jit	HpHp
SSDP Flood	8	13	6	3
Mirai	8	4	4	14
Fuzzing	7	14	4	5
Active Wiretap	5	3	17	5
Total	28	34	31	27

At the attack level, the feature family counts reveal different feature-importance patterns for different attacks. Consistent with the binary classification analysis, SSDP flood models rely more heavily on `MI_dir` and `HH` features, and fuzzing models rely most on `HH` features. This suggests that host-to-host traffic patterns serve as an important indicator for SSDP flood and fuzzing multiclass models, and directional source behavior is also useful for SSDP flood multiclass models. Mirai models rely most heavily on `HpHp` features, differing from the binary classification analysis and indicating that socket-level traffic behavior is particularly informative for the multiclass Mirai setting. Active wiretap is the clearest timing-sensitive case, as `HH_jit` features appear 17 times—far more than any other feature family for that attack. This supports the earlier observation that active wiretap is more timing-sensitive than the other evaluated attacks.

Table 6.27 summarizes the top-five feature-importance rankings by decay factor for each intrusion attack. The results indicate that while the most informative effective time window depends on the attack, short-term and longer-term features appear most frequently. $\lambda = 5$ appears most frequently, followed by $\lambda = 0.1$ and $\lambda = 0.01$, suggesting the usefulness of short- and long-term traffic behavior in the multiclass setting. Looking at the results for each intrusion attack, SSDP flood models rely most on short-term burst behavior, as $\lambda = 5$ features appear most often. For Mirai models, medium- and short-term features dominate, with the most frequent decay factors being $\lambda = 0.1$, $\lambda = 3$, and $\lambda = 5$. For fuzzing and active wiretap models, $\lambda = 0.01$ features appear the most, indicating the importance of long-term

traffic behavior in distinguishing among benign, original malicious, and modified malicious traffic for these attacks.

Table 6.27. Top-Five Multiclass Classification Feature Counts by Time Window and Attack

Attack	$\lambda = 0.01$	$\lambda = 0.1$	$\lambda = 1$	$\lambda = 3$	$\lambda = 5$
SSDP Flood	3	4	7	3	13
Mirai	1	9	4	8	8
Fuzzing	11	6	4	3	6
Active Wiretap	9	6	4	5	6
Total	24	25	19	19	33

Table 6.28 summarizes the top-five feature-importance rankings by feature family for each evaluated supervised ML model type. The results support the binary classification observation that the three model types rely on related but not identical feature families. RF relies most heavily on `HH` features, followed by `MI_dir` and `HH_jit` features. These results suggest that RF primarily uses host-to-host communication behavior, while also incorporating source-directional and timing-based information. LightGBM selects the largest number of `HH_jit` features, suggesting that timing-related features are especially important for its multiclass decisions. TabNet relies heavily on both `HH` and `HpHp` features, suggesting that it uses a mixture of host-to-host communication behavior and socket-level communication patterns.

Table 6.28. Top-Five Multiclass Classification Feature Counts by Feature Family and Model Type

Model	`MI_dir`	`HH`	`HH_jit`	`HpHp`
Random Forest	11	14	10	5
LightGBM	10	6	14	10
TabNet	7	14	7	12

Table 6.29 summarizes the top five feature-importance rankings by feature family for each evaluated reshaping strategy. Under Simple reshaping, the top-five rankings contain 14 `MI_dir`, 16 `HH`, 16 `HH_jit`, and 14 `HpHp` features. Under LSTM reshaping, the corre-

sponding counts are 14, 18, 15, and 13. The highly similar corresponding counts reinforce the binary classification analysis; the Simple and LSTM reshaping strategies produce similar broad feature-importance patterns. These feature-importance results further suggest that the simple fixed IAT reshaping attack produces feature-importance trends that are similar to those produced by the more complex LSTM-based reshaping attack.

Table 6.29. Top-Five Multiclass Classification Feature Counts by Feature Family and Reshaping Strategy

Reshaping	MI_dir	HH	HH_jit	HpHp
Simple	14	16	16	14
LSTM	14	18	15	13

6.5 Summary of Main Findings

This chapter evaluated the effectiveness of traffic reshaping attacks against both anomaly-based and supervised ML-based intrusion detection models. The results show that both the Simple fixed IAT reshaping strategy and the LSTM-based reshaping strategy substantially reduce Kitsune's ability to detect malicious traffic. For the SSDP flood and Mirai attacks, both reshaping strategies completely evade Kitsune under the selected thresholds, while the fuzzing and active wiretap reshaping attacks show partial but still meaningful reductions in detection performance. Across these anomaly detection experiments, the Simple reshaping strategy performs comparably to the more complex LSTM-based strategy.

The supervised ML binary classification results demonstrate that traffic reshaping can also affect supervised ML models, but the impact depends on the attack, model type, and training configuration. Models trained only on original malicious traffic can fail to detect reshaped malicious traffic in some scenarios, like for the Mirai attack. However, models trained on combined original and reshaped malicious traffic are generally the most robust across both Original and Modified test sets. These results suggest that supervised ML models can be made more resilient to reshaping attacks when reshaped traffic is included during training.

The binary feature-importance results show that the most frequently selected AfterImage features are primarily host-to-host communication features and directional source MAC-IP features. However, the most informative feature families and effective time windows vary by attack. Active wiretap relies more heavily on timing-related `HH_jit` features than the other attacks, which helps explain why it is more difficult for the supervised models to classify consistently. The feature-overlap results also show that models trained on original malicious traffic and models trained on reshaped malicious traffic often rely on different feature cues.

The supervised ML multiclass classification results show that models can often distinguish among benign traffic, original malicious traffic, and modified malicious traffic. SSDP flood and Mirai attack traffic are perfectly separated by all evaluated multiclass models, while the fuzzing and active wiretap attacks show more model-dependent behavior. The modified malicious class is generally detected very well, but the original malicious class is more likely to show reduced F1 scores, especially for the fuzzing and active wiretap attacks. RF is the most consistently strong multiclass model overall, while TabNet performs especially well on the active wiretap attack. LightGBM is also less stable for the active wiretap attack.

The multiclass feature-importance results demonstrate that distinguishing among benign, original malicious, and modified malicious traffic uses a broader range of AfterImage feature families than binary benign-versus-malicious classification. In particular, `HH_jit` and `HpHp` features appear more prominently in the multiclass setting. This suggests that separating original malicious traffic from reshaped malicious traffic requires more information about timing behavior and socket-level communication patterns. Finally, both the performance results and the feature-importance results show that the Simple and LSTM reshaping strategies often produce similar trends, further supporting the conclusion that a simple fixed IAT attack can approximate the effects of a more complex learned reshaping strategy.

7. CONCLUSION

This thesis evaluated the effectiveness of a simple fixed-IAT traffic reshaping attack against ML-based NIDSs and compared its behavior to a more complex TANTRA-inspired LSTM reshaping attack. The central goal was to determine whether a low-complexity timing-based reshaping strategy could approximate the effects of a learned reshaping model. In addition, this thesis evaluated whether supervised ML models can detect reshaped malicious traffic in both binary and multiclass classification settings. Finally, feature-importance analysis was performed to investigate which AfterImage features are most useful for distinguishing benign traffic from malicious traffic, as well as for distinguishing among benign traffic, original malicious traffic, and modified malicious traffic.

The first major finding is that the simple fixed-IAT reshaping attack can substantially reduce Kitsune's ability to detect malicious traffic. Across the evaluated intrusion attacks, both the Simple and LSTM-based reshaping strategies reduce Kitsune detection performance, and in most cases, the Simple strategy performs comparably to the more complex LSTM-based strategy. This suggests that learned timing models may not always be necessary to achieve effective traffic reshaping. From a security perspective, this finding is important because it shows that even low-complexity packet-space attacks can pose a meaningful threat to ML-based NIDSs.

The binary supervised ML experiments show that supervised models can often detect malicious traffic effectively, but their robustness depends on the relationship between the training and testing traffic distributions. Models trained only on original malicious traffic do not always generalize to reshaped malicious traffic, and models trained only on reshaped malicious traffic do not always generalize back to original malicious traffic. However, models trained on both original and reshaped malicious traffic are generally the most robust. In addition, the multiclass classification experiments further demonstrate that supervised ML models can often distinguish among benign traffic, original malicious traffic, and reshaped malicious traffic. The multiclass results show strong performance for several attacks, although performance varies by intrusion attack and model type. These findings support

the conclusion that reshaping-aware supervised training can improve detection performance against traffic reshaping attacks.

The feature-importance analysis provides additional insight into why models behave differently across attacks and classification settings. In the binary setting, host-to-host communication features and directional source features are especially important, and commonly selected feature templates capture packet-count behavior, host-to-host traffic magnitude, and packet-size variability. The results also show that important features depend on the effective time window, with no single decay factor dominating every attack. In the multiclass setting, the selected features are more evenly distributed across feature families, with timing-based jitter features and socket-level features appearing more prominently. This suggests that distinguishing original malicious traffic from reshaped malicious traffic requires a broader set of feature cues than binary benign-versus-malicious classification.

Overall, the results show that simple timing-based reshaping can pose a meaningful threat to ML-based NIDSs, but they also show that supervised models can be made more robust when reshaped traffic is included during training. These findings support the need to evaluate ML-based NIDSs not only against original malicious traffic but also against modified malicious traffic that has been intentionally reshaped to evade detection. They also demonstrate the value of feature-level analysis for understanding how models detect reshaped traffic and why certain attacks are more difficult to classify than others.

Several limitations should be considered when interpreting these results. First, the experiments focus on a subset of the Kitsune intrusion attacks and therefore may not capture the full range of behaviors present across all attack scenarios. Second, the reshaping attacks modify packet timing but do not evaluate other possible packet-space modifications, such as packet-size changes or combined timing-and-size reshaping. Third, the experiments are performed in an offline evaluation setting, so they do not fully capture practical deployment constraints such as latency, throughput, packet ordering, or whether reshaped traffic preserves attack functionality in a live network environment. Finally, each reshaping experiment isolates a single malicious flow from the surrounding traffic. Because AfterImage features capture relationships among packets, hosts, channels, and recent traffic history, isolating one malicious flow may alter the resulting feature values compared to a setting in which the

full attack traffic and background traffic are evaluated together. As a result, the measured detection performance reflects the controlled single-flow evaluation used in this thesis rather than every possible deployment scenario.

These limitations suggest several directions for future work. One direction is to evaluate additional intrusion attacks from the Kitsune dataset, including ARP man-in-the-middle, OS scan, SSL renegotiation, SYN DoS, and video injection, or additional datasets, such as CIC-IDS2017 [11]. These additional evaluations would provide a more complete understanding of when simple timing-based reshaping is effective and help determine whether the trends observed in this thesis generalize beyond the Kitsune dataset. Future work could also evaluate additional reshaping strategies, including those that modify other observable properties of packets, such as packet sizes. These experiments would help determine whether the effectiveness of the Simple attack comes from fixed timing specifically or from a broader vulnerability to low-complexity packet-space modifications. Another possible direction is to evaluate reshaping attacks and defenses in a live deployment environment to determine whether reshaped traffic preserves attack functionality. Finally, future work could explore stronger defenses against traffic reshaping attacks. The results of this thesis suggest that reshaping-aware training improves robustness, but additional defensive strategies may further improve performance.

In summary, this thesis shows that simple timing-based reshaping attacks can meaningfully reduce the effectiveness of ML-based NIDSs, even without the complexity of a learned reshaping model. At the same time, the supervised ML results show that reshaping-aware training can improve robustness against these attacks. These findings highlight the need to evaluate ML-based NIDSs against realistic packet-space adversarial modifications and demonstrate that feature-level analysis can provide useful insight into how models detect original and reshaped malicious traffic.

REFERENCES

[1] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, p. 20, Jul. 2019, ISSN: 2523-3246. DOI: 10.1186/s42400-019-0038-7. [Online]. Available: https://doi.org/10.1186/s42400-019-0038-7.

[2] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in *Proceedings 2018 Network and Distributed System Security Symposium*, San Diego, CA: Internet Society, 2018. DOI: 10.14722/ndss.2018.23204.

[3] M. J. Hashemi, G. Cusack, and E. Keller, "Towards evaluation of NIDSs in adversarial setting," in *Proceedings of the 3rd ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks*, ser. BigDAMA '19, Orlando, FL, USA: Association for Computing Machinery, 2019, pp. 14–21, ISBN: 9781450369992. DOI: 10.1145/3359992.3366642. [Online]. Available: https://doi.org/10.1145/3359992.3366642.

[4] D. Han *et al.*, "Evaluating and improving adversarial robustness of machine learning-based network intrusion detectors," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 8, pp. 2632–2647, 2021. DOI: 10.1109/JSAC.2021.3087242.

[5] Y. Sharon, D. Berend, Y. Liu, A. Shabtai, and Y. Elovici, "TANTRA: Timing-based adversarial network traffic reshaping attack," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3225–3237, 2022. DOI: 10.1109/TIFS.2022.3201377.

[6] O. Ibitoye, O. Shafiq, and A. Matrawy, "Analyzing adversarial attacks against deep learning for intrusion detection in IoT networks," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6. DOI: 10.1109/GLOBECOM38437.2019.9014337.

[7] J. Clements, Y. Yang, A. A. Sharma, H. Hu, and Y. Lao, "Rallying adversarial techniques against deep learning for network security," in *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2021, pp. 01–08. DOI: 10.1109/SSCI50451.2021.9660011.

[8] X. Zhou, W. Liang, W. Li, K. Yan, S. Shimizu, and K. I.-K. Wang, "Hierarchical adversarial attacks against graph-neural-network-based IoT network intrusion detection system," *IEEE Internet of Things Journal*, vol. 9, no. 12, pp. 9310–9319, 2022. DOI: 10.1109/JIOT.2021.3130434.

[9] OpenAI, *ChatGPT (GPT-5.6 Sol)*, Jul. 2026. [Online]. Available: https://chatgpt.com/.

[10] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related features," in *Proceedings of the 2nd International Conference on Information Systems Security and Privacy - ICISSP*, INSTICC, SciTePress, 2016, pp. 407–414, ISBN: 978-989-758-167-0. DOI: 10.5220/0005740704070414.

[11] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy - ICISSP*, INSTICC, SciTePress, 2018, pp. 108–116, ISBN: 978-989-758-282-0. DOI: 10.5220/0006639801080116.

[12] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6. DOI: 10.1109/CISDA.2009.5356528.

[13] K. He, D. D. Kim, and M. R. Asghar, "Adversarial machine learning for network intrusion detection systems: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 538–566, 2023. DOI: 10.1109/COMST.2022.3233793.

[14] N. Wang, Y. Chen, Y. Xiao, Y. Hu, W. Lou, and Y. T. Hou, "MANDA: On adversarial example detection for network intrusion detection system," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 2, pp. 1139–1153, 2023. DOI: 10.1109/TDSC.2022.3148990.

[15] *KDD Cup 1999 data*, https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, Accessed: 2026-06-22, 1999.

[16] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers & Security*, vol. 31, no. 3, pp. 357–374, 2012, ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2011.12.012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404811001672.

[17] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, 2015, pp. 1–6. DOI: 10.1109/MilCIS.2015.7348942.

[18] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *2017 International Conference on Information Networking (ICOIN)*, 2017, pp. 712–717. DOI: 10.1109/ICOIN.2017.7899588.

[19] A. Hamza, H. H. Gharakheili, T. A. Benson, and V. Sivaraman, "Detecting volumetric attacks on IoT devices via SDN-based monitoring of MUD activity," in *Proceedings of the 2019 ACM Symposium on SDN Research*, ser. SOSR '19, San Jose, CA, USA: Association for Computing Machinery, 2019, pp. 36–48, ISBN: 9781450367103. DOI: 10.1145/3314148.3314352. [Online]. Available: https://doi.org/10.1145/3314148.3314352.

[20] C. Lu, X. Wang, A. Yang, Y. Liu, and Z. Dong, "A few-shot-based model-agnostic meta-learning for intrusion detection in security of internet of things," *IEEE Internet of Things Journal*, vol. 10, no. 24, pp. 21 309–21 321, 2023. DOI: 10.1109/JIOT.2023.3283408.

[21] W. Li *et al.*, "Prism: Real-time privacy protection against temporal network traffic analyzers," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 2524–2537, 2023. DOI: 10.1109/TIFS.2023.3267885.

[22] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, Oct. 2001, ISSN: 1573-0565. DOI: 10.1023/A:1010933404324. [Online]. Available: https://doi.org/10.1023/A:1010933404324.

[23] G. Ke *et al.*, "LightGBM: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, I. Guyon *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[24] S. O. Arik and T. Pfister, *TabNet: Attentive interpretable tabular learning*, 2020. arXiv: 1908.07442 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1908.07442.

A. KITSUNE ANOMALY SCORE GRAPHS

Figures A.1–A.9 display Kitsune anomaly scores for the Mirai, fuzzing, and active wiretap attacks under the unmodified, simple-reshaped, and LSTM-reshaped conditions.

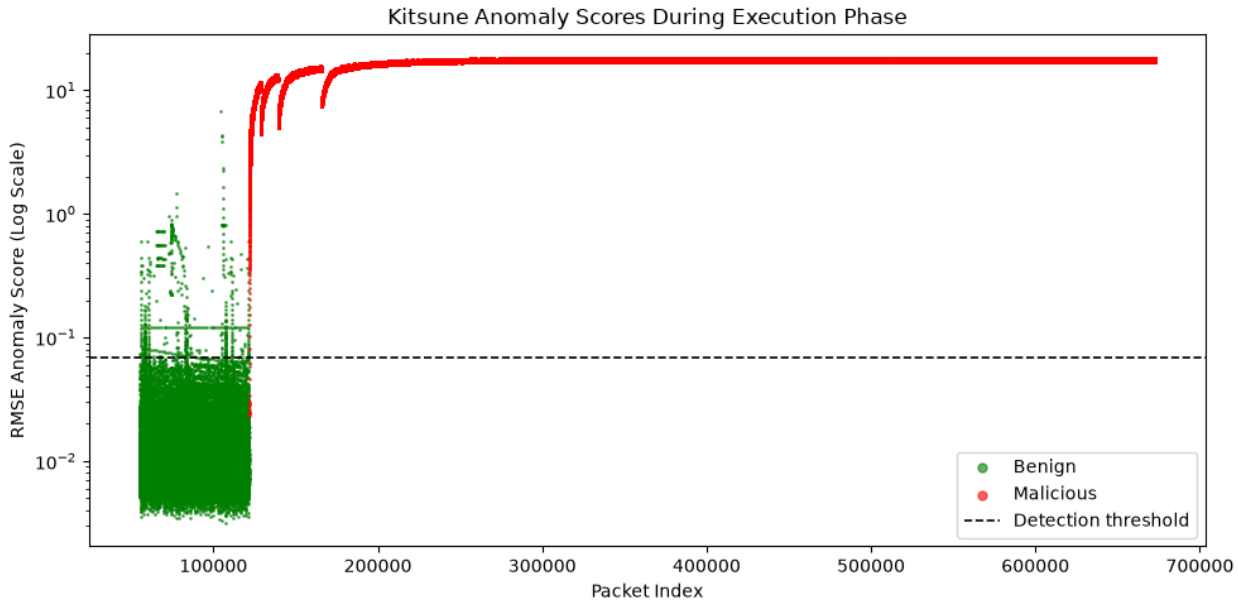

Figure A.1. Kitsune Anomaly Scores for the Unmodified Mirai Attack

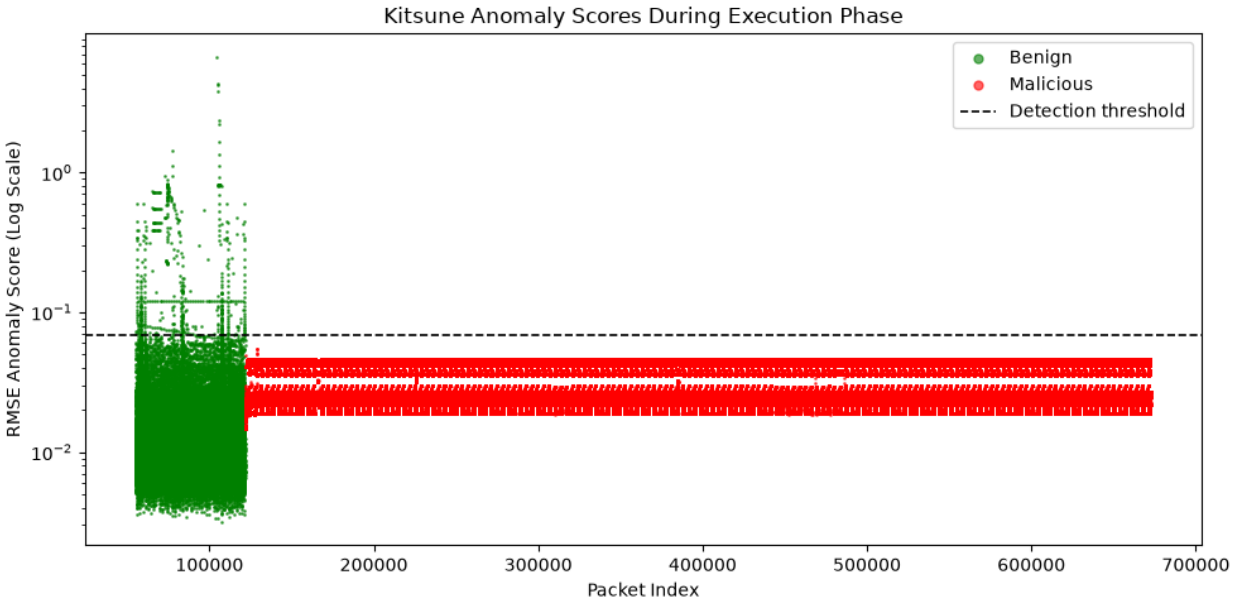

Figure A.2. Kitsune Anomaly Scores for the Mirai Attack Under Simple Reshaping

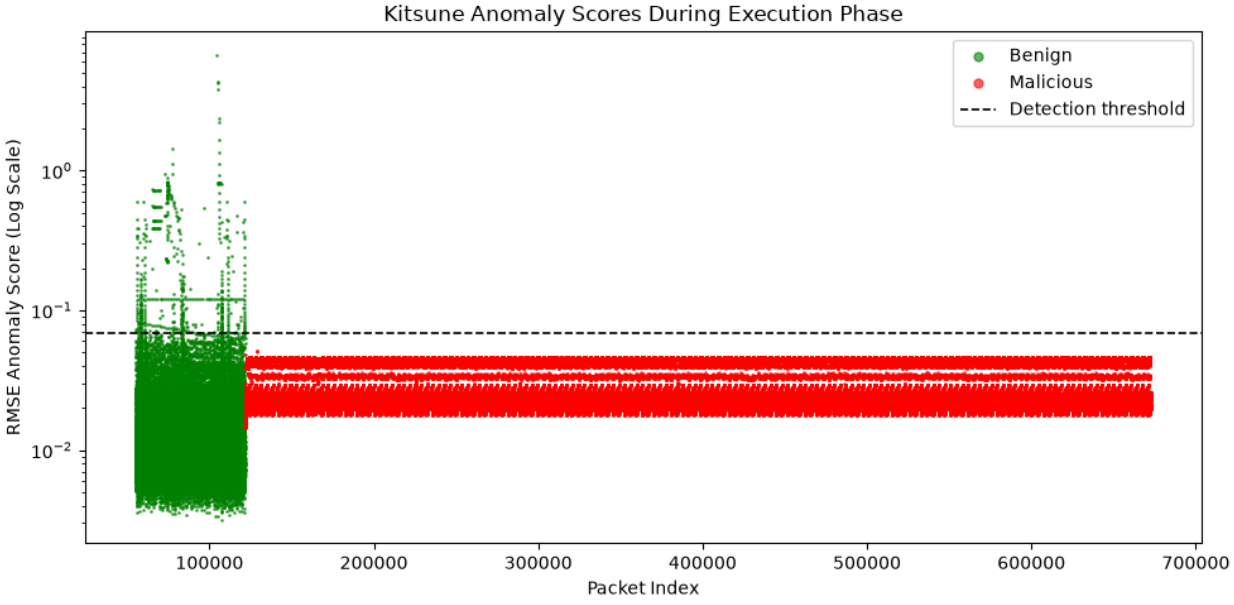

Figure A.3. Kitsune Anomaly Scores for the Mirai Attack Under LSTM Reshaping

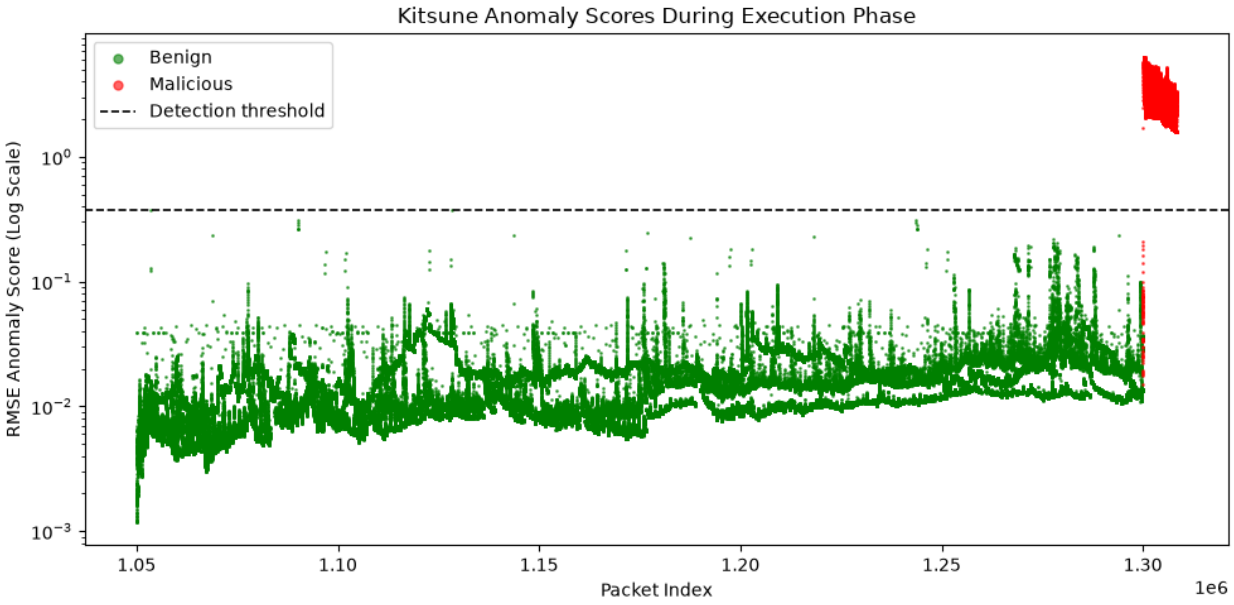

Figure A.4. Kitsune Anomaly Scores for the Unmodified Fuzzing Attack

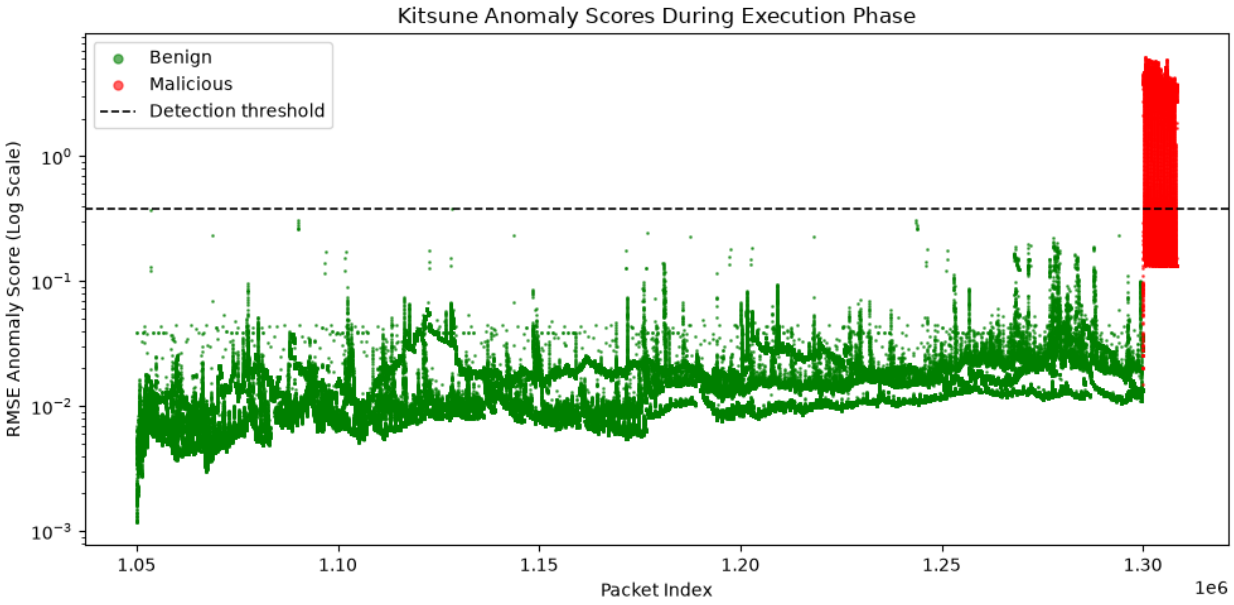

Figure A.5. Kitsune Anomaly Scores for the Fuzzing Attack Under Simple Reshaping

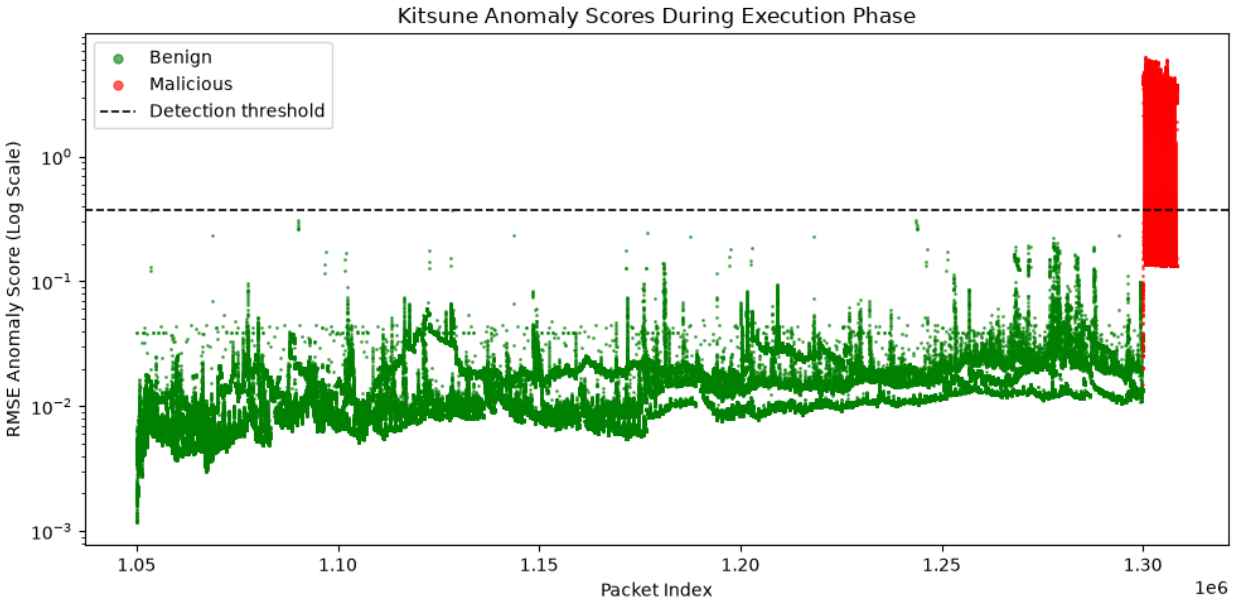

Figure A.6. Kitsune Anomaly Scores for the Fuzzing Attack Under LSTM Reshaping

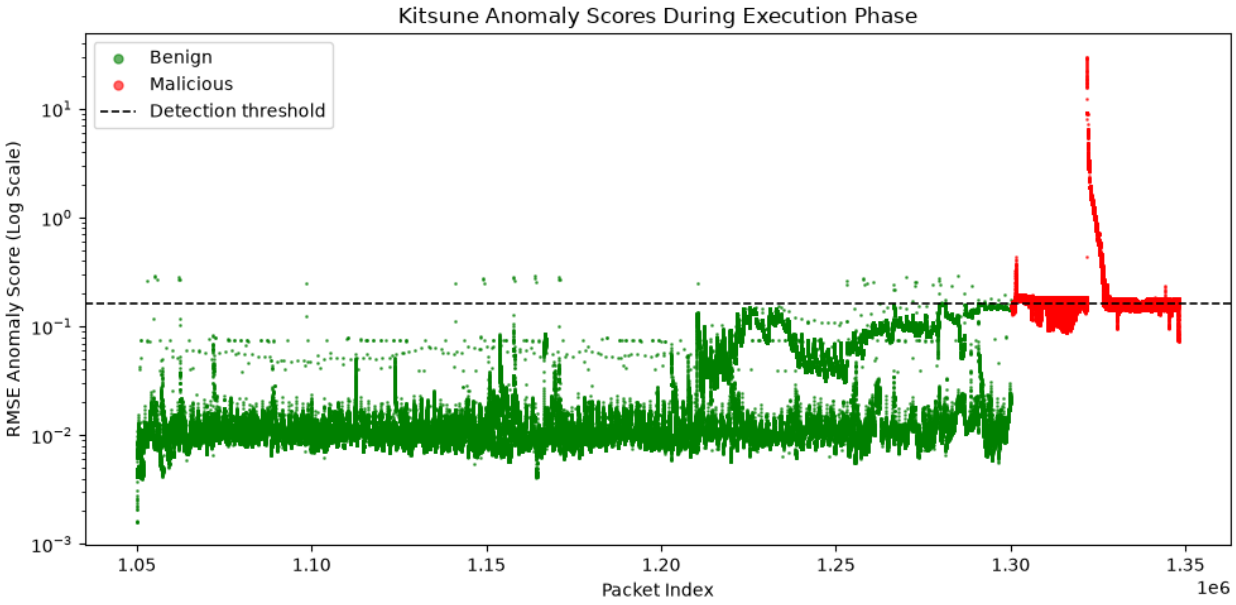

Figure A.7. Kitsune Anomaly Scores for the Unmodified Active Wiretap Attack

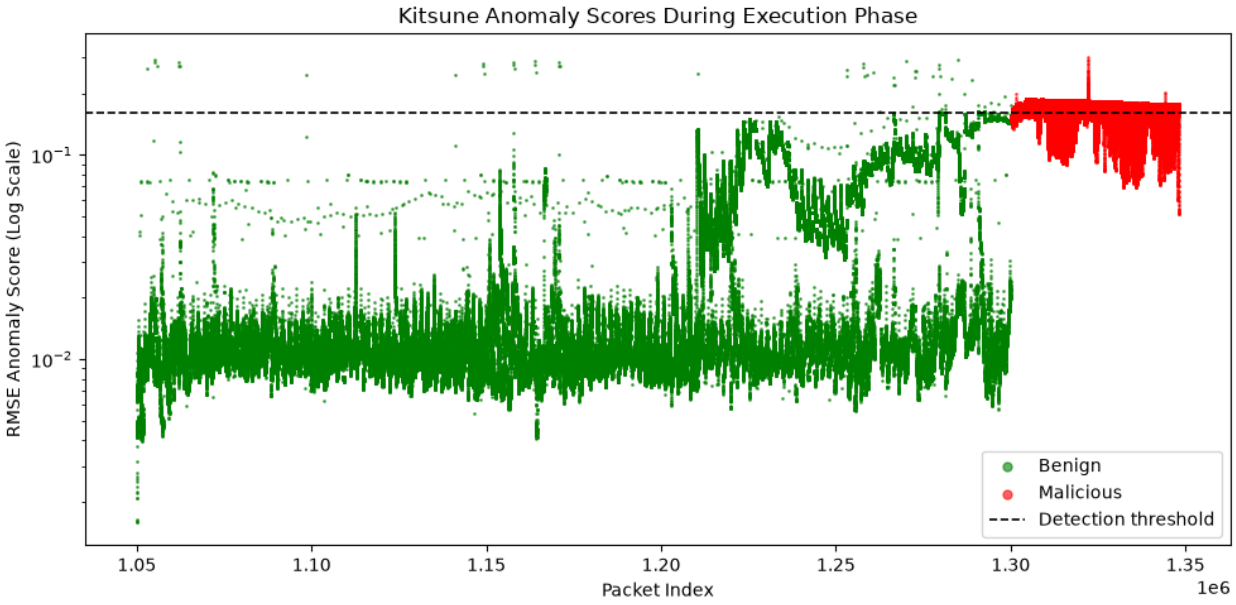

Figure A.8. Kitsune Anomaly Scores for the Active Wiretap Attack Under Simple Reshaping

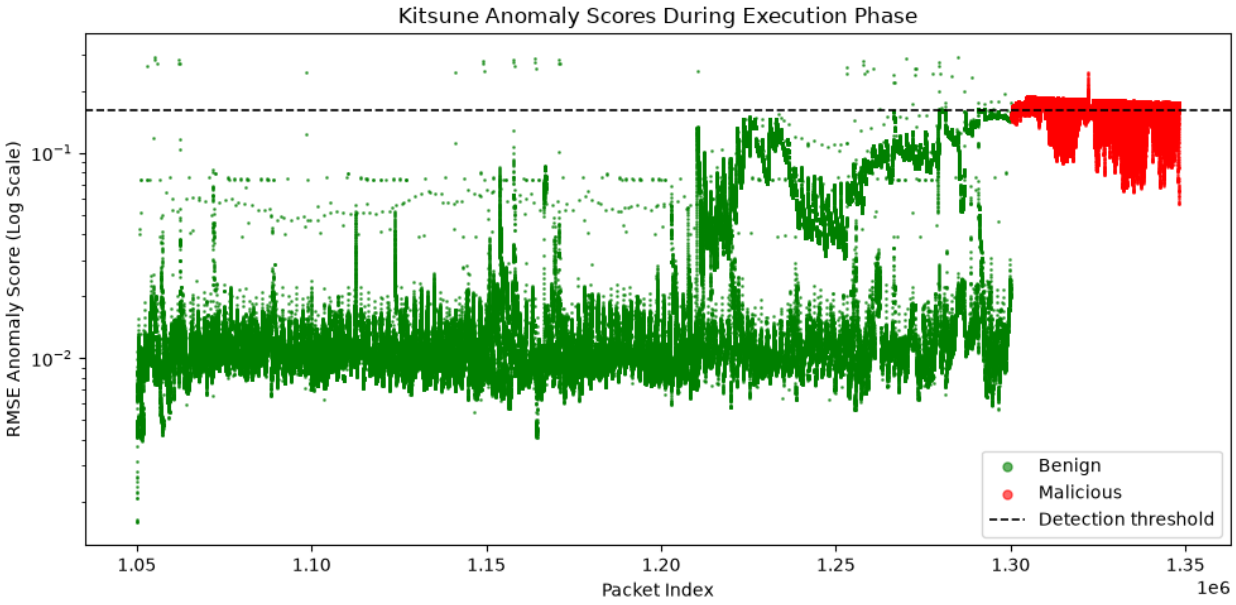

Figure A.9. Kitsune Anomaly Scores for the Active Wiretap Attack Under LSTM Reshaping

B. TOP FIVE MOST IMPORTANT FEATURES TABLES FOR SUPERVISED ML BINARY CLASSIFICATION

Tables B.1–B.7 display the top five most important features of each trained binary classification model in the SSDP flood attack under the LSTM reshaping strategy and in the Mirai, fuzzing, and active wiretap attacks under both the Simple and LSTM reshaping strategies.

Table B.1. Supervised ML Binary Models Top 5 Most Important Features in SSDP Flood Attack Under LSTM Reshaping

Model	Rank	Feature Name	Importance (%)
RF Original	1	HH_3_radius_0_1	11.9961
RF Original	2	HH_jit_5_weight	10.0553
RF Original	3	HH_5_radius_0_1	7.9997
RF Original	4	MI_dir_3_weight	6.9960
RF Original	5	MI_dir_5_weight	5.9995
RF Modified	1	HH_5_magnitude_0_1	11.9946
RF Modified	2	MI_dir_0.01_std	10.9973
RF Modified	3	MI_dir_0.1_std	8.9935
RF Modified	4	HpHp_0.01_weight	7.1280
RF Modified	5	HH_5_radius_0_1	6.9997
RF Combined	1	HH_5_magnitude_0_1	11.9980
RF Combined	2	MI_dir_0.01_std	11.0035
RF Combined	3	MI_dir_0.1_std	10.0786
RF Combined	4	HH_5_radius_0_1	8.9993
RF Combined	5	HH_1_magnitude_0_1	6.1371
LGBM Original	1	HH_5_magnitude_0_1	99.9315
LGBM Original	2	MI_dir_5_weight	0.0631
LGBM Original	3	HH_jit_0.01_weight	0.0030
LGBM Original	4	HpHp_0.01_weight	0.0022
LGBM Original	5	MI_dir_0.01_std	0.0001
LGBM Modified	1	MI_dir_3_std	99.8358
LGBM Modified	2	HH_0.1_weight	0.0992
LGBM Modified	3	HH_1_radius_0_1	0.0631
LGBM Modified	4	MI_dir_5_mean	0.0005
LGBM Modified	5	MI_dir_0.1_std	0.0004
LGBM Combined	1	HH_5_magnitude_0_1	99.8780
LGBM Combined	2	HpHp_0.01_weight	0.0622
LGBM Combined	3	MI_dir_5_weight	0.0528
LGBM Combined	4	HH_jit_0.01_weight	0.0054
LGBM Combined	5	HH_0.01_radius_0_1	0.0008
TN Original	1	HpHp_0.01_weight	15.4483
TN Original	2	MI_dir_1_weight	10.7436
TN Original	3	MI_dir_3_std	9.3740
TN Original	4	HpHp_0.1_pcc_0_1	8.6292
TN Original	5	HH_jit_0.01_weight	7.5202
TN Modified	1	MI_dir_3_std	22.8745
TN Modified	2	HH_0.1_radius_0_1	17.2540
TN Modified	3	HpHp_0.1_pcc_0_1	16.9800
TN Modified	4	HH_5_radius_0_1	8.9601
TN Modified	5	MI_dir_1_std	8.1277
TN Combined	1	HH_5_radius_0_1	15.9820
TN Combined	2	HpHp_5_mean	13.4388
TN Combined	3	HpHp_3_magnitude_0_1	11.2913
TN Combined	4	HH_5_weight	10.4460
TN Combined	5	HpHp_1_weight	9.6264

Table B.2. Supervised ML Binary Models Top 5 Most Important Features in Mirai Attack Under Simple Reshaping

Model	Rank	Feature Name	Importance (%)
RF Original	1	HpHp_0.1_weight	12.1004
RF Original	2	MI_dir_3_weight	12.0237
RF Original	3	HpHp_3_weight	11.1229
RF Original	4	HH_3_weight	10.0190
RF Original	5	MI_dir_1_weight	6.0134
RF Modified	1	HH_0.1_std	11.9695
RF Modified	2	MI_dir_0.1_std	8.9950
RF Modified	3	MI_dir_0.1_mean	7.1621
RF Modified	4	MI_dir_0.01_std	6.9044
RF Modified	5	HH_jit_0.1_weight	6.0034
RF Combined	1	HH_0.1_std	12.0719
RF Combined	2	MI_dir_0.1_std	8.9590
RF Combined	3	MI_dir_0.1_mean	7.3363
RF Combined	4	MI_dir_1_mean	6.1780
RF Combined	5	HH_1_std	5.9868
LGBM Original	1	MI_dir_3_weight	39.9375
LGBM Original	2	HpHp_5_weight	32.7330
LGBM Original	3	MI_dir_1_weight	27.1910
LGBM Original	4	MI_dir_0.1_weight	0.0822
LGBM Original	5	MI_dir_5_mean	0.0336
LGBM Modified	1	MI_dir_0.1_mean	99.7942
LGBM Modified	2	MI_dir_0.1_weight	0.0852
LGBM Modified	3	MI_dir_1_mean	0.0720
LGBM Modified	4	HH_jit_5_mean	0.0472
LGBM Modified	5	HH_jit_0.1_weight	0.0012
LGBM Combined	1	MI_dir_0.1_mean	99.7202
LGBM Combined	2	HpHp_5_weight	0.1407
LGBM Combined	3	MI_dir_0.1_weight	0.0786
LGBM Combined	4	MI_dir_1_mean	0.0300
LGBM Combined	5	MI_dir_5_mean	0.0145
TN Original	1	HH_1_weight	40.9320
TN Original	2	HH_1_radius_0_1	18.6906
TN Original	3	HpHp_0.1_magnitude_0_1	16.7241
TN Original	4	HpHp_1_weight	5.5562
TN Original	5	MI_dir_1_weight	4.9669
TN Modified	1	MI_dir_0.01_mean	45.0389
TN Modified	2	HH_0.01_weight	26.9048
TN Modified	3	HH_0.1_mean	10.8066
TN Modified	4	HH_0.1_std	10.0686
TN Modified	5	HH_0.1_pcc_0_1	1.6314
TN Combined	1	MI_dir_0.1_weight	34.3814
TN Combined	2	HH_0.1_mean	14.8686
TN Combined	3	HH_0.01_std	11.1054
TN Combined	4	MI_dir_0.1_std	9.3968
TN Combined	5	HH_jit_3_std	9.2104

Table B.3. Supervised ML Binary Models Top 5 Most Important Features in Mirai Attack Under LSTM Reshaping

Model	Rank	Feature Name	Importance (%)
RF Original	1	HpHp_0.1_weight	12.1004
RF Original	2	MI_dir_3_weight	12.0237
RF Original	3	HpHp_3_weight	11.1229
RF Original	4	HH_3_weight	10.0190
RF Original	5	MI_dir_1_weight	6.0134
RF Modified	1	HH_0.1_std	12.0371
RF Modified	2	MI_dir_0.1_std	8.9970
RF Modified	3	MI_dir_0.1_mean	7.0950
RF Modified	4	HH_jit_3_mean	7.0079
RF Modified	5	MI_dir_0.01_std	6.9020
RF Combined	1	HH_0.1_std	12.0720
RF Combined	2	MI_dir_0.1_std	8.9591
RF Combined	3	MI_dir_0.1_mean	7.3364
RF Combined	4	MI_dir_1_mean	6.2036
RF Combined	5	HH_1_std	5.9868
LGBM Original	1	MI_dir_3_weight	39.9375
LGBM Original	2	HpHp_5_weight	32.7330
LGBM Original	3	MI_dir_1_weight	27.1910
LGBM Original	4	MI_dir_0.1_weight	0.0822
LGBM Original	5	MI_dir_5_mean	0.0336
LGBM Modified	1	MI_dir_0.1_mean	99.7943
LGBM Modified	2	MI_dir_0.1_weight	0.0852
LGBM Modified	3	MI_dir_1_mean	0.0719
LGBM Modified	4	HH_jit_5_mean	0.0467
LGBM Modified	5	HH_jit_0.1_weight	0.0016
LGBM Combined	1	MI_dir_0.1_mean	99.7223
LGBM Combined	2	HpHp_5_weight	0.1385
LGBM Combined	3	MI_dir_0.1_weight	0.0784
LGBM Combined	4	MI_dir_1_mean	0.0287
LGBM Combined	5	HpHp_1_weight	0.0164
TN Original	1	HH_1_weight	40.9320
TN Original	2	HH_1_radius_0_1	18.6906
TN Original	3	HpHp_0.1_magnitude_0_1	16.7241
TN Original	4	HpHp_1_weight	5.5562
TN Original	5	MI_dir_1_weight	4.9669
TN Modified	1	HH_0.1_mean	28.2725
TN Modified	2	HH_1_weight	21.5258
TN Modified	3	HH_0.01_pcc_0_1	19.2694
TN Modified	4	HH_jit_1_weight	12.9800
TN Modified	5	HH_1_radius_0_1	5.6588
TN Combined	1	HH_0.1_mean	28.4481
TN Combined	2	HpHp_5_weight	24.8533
TN Combined	3	MI_dir_0.1_weight	17.9261
TN Combined	4	HH_jit_3_mean	16.2395
TN Combined	5	MI_dir_0.1_std	4.9688

Table B.4. Supervised ML Binary Models Top 5 Most Important Features in Fuzzing Attack Under Simple Reshaping

Model	Rank	Feature Name	Importance (%)
RF Original	1	HH_0.01_magnitude_0_1	11.1054
RF Original	2	HH_0.1_magnitude_0_1	9.6183
RF Original	3	MI_dir_0.1_std	8.0587
RF Original	4	HH_1_magnitude_0_1	7.4371
RF Original	5	MI_dir_0.01_std	7.0909
RF Modified	1	HH_0.01_magnitude_0_1	10.2808
RF Modified	2	HH_0.1_magnitude_0_1	9.2088
RF Modified	3	HH_0.1_weight	7.7224
RF Modified	4	HH_1_magnitude_0_1	6.8504
RF Modified	5	MI_dir_3_weight	6.7200
RF Combined	1	HH_0.01_magnitude_0_1	10.9508
RF Combined	2	HH_0.1_magnitude_0_1	9.9088
RF Combined	3	MI_dir_3_weight	7.4181
RF Combined	4	HH_1_magnitude_0_1	7.3178
RF Combined	5	HH_5_magnitude_0_1	5.7558
LGBM Original	1	HH_0.1_magnitude_0_1	80.2107
LGBM Original	2	HH_1_radius_0_1	19.7790
LGBM Original	3	HH_jit_5_mean	0.0031
LGBM Original	4	HH_5_radius_0_1	0.0017
LGBM Original	5	HH_0.01_std	0.0016
LGBM Modified	1	MI_dir_0.1_mean	27.2952
LGBM Modified	2	HH_jit_1_std	10.3029
LGBM Modified	3	HpHp_1_radius_0_1	9.8369
LGBM Modified	4	MI_dir_0.01_mean	8.5960
LGBM Modified	5	HH_jit_1_mean	8.2193
LGBM Combined	1	HH_0.01_magnitude_0_1	94.6111
LGBM Combined	2	MI_dir_0.01_std	2.5948
LGBM Combined	3	HH_1_magnitude_0_1	1.4074
LGBM Combined	4	HpHp_3_radius_0_1	0.6129
LGBM Combined	5	HH_jit_3_std	0.3998
TN Original	1	HH_1_magnitude_0_1	26.6899
TN Original	2	HH_0.01_std	14.2709
TN Original	3	HH_5_mean	12.5748
TN Original	4	HH_jit_0.1_std	8.9790
TN Original	5	HH_3_std	7.4560
TN Modified	1	HH_5_std	16.4922
TN Modified	2	HH_5_radius_0_1	15.8133
TN Modified	3	HH_1_std	14.7999
TN Modified	4	HH_jit_0.01_weight	14.0006
TN Modified	5	HpHp_3_std	7.2932
TN Combined	1	HH_0.01_magnitude_0_1	38.6054
TN Combined	2	MI_dir_5_std	15.9590
TN Combined	3	HpHp_1_radius_0_1	15.8153
TN Combined	4	MI_dir_3_weight	9.6067
TN Combined	5	HH_3_weight	8.9473

Table B.5. Supervised ML Binary Models Top 5 Most Important Features in Fuzzing Attack Under LSTM Reshaping

Model	Rank	Feature Name	Importance (%)
RF Original	1	HH_0.01_magnitude_0_1	11.1054
RF Original	2	HH_0.1_magnitude_0_1	9.6183
RF Original	3	MI_dir_0.1_std	8.0587
RF Original	4	HH_1_magnitude_0_1	7.4371
RF Original	5	MI_dir_0.01_std	7.0909
RF Modified	1	HH_0.01_magnitude_0_1	10.2720
RF Modified	2	HH_0.1_magnitude_0_1	9.1576
RF Modified	3	HH_0.1_weight	7.7218
RF Modified	4	HH_1_magnitude_0_1	6.8549
RF Modified	5	MI_dir_3_weight	6.7195
RF Combined	1	HH_0.01_magnitude_0_1	10.9505
RF Combined	2	HH_0.1_magnitude_0_1	9.9182
RF Combined	3	MI_dir_3_weight	7.4181
RF Combined	4	HH_1_magnitude_0_1	7.3163
RF Combined	5	HH_5_magnitude_0_1	5.7558
LGBM Original	1	HH_0.1_magnitude_0_1	80.2107
LGBM Original	2	HH_1_radius_0_1	19.7790
LGBM Original	3	HH_jit_5_mean	0.0031
LGBM Original	4	HH_5_radius_0_1	0.0017
LGBM Original	5	HH_0.01_std	0.0016
LGBM Modified	1	HH_jit_3_std	97.1615
LGBM Modified	2	HH_5_magnitude_0_1	1.5684
LGBM Modified	3	HH_jit_0.01_std	0.3933
LGBM Modified	4	HH_1_radius_0_1	0.2567
LGBM Modified	5	MI_dir_0.1_std	0.1898
LGBM Combined	1	HH_0.01_magnitude_0_1	94.6226
LGBM Combined	2	MI_dir_0.01_std	2.5784
LGBM Combined	3	HH_1_magnitude_0_1	1.4204
LGBM Combined	4	HpHp_3_radius_0_1	0.9549
LGBM Combined	5	HpHp_0.01_std	0.3479
TN Original	1	HH_1_magnitude_0_1	26.6899
TN Original	2	HH_0.01_std	14.2709
TN Original	3	HH_5_mean	12.5748
TN Original	4	HH_jit_0.1_std	8.9790
TN Original	5	HH_3_std	7.4560
TN Modified	1	HH_1_magnitude_0_1	30.7686
TN Modified	2	MI_dir_5_weight	18.2845
TN Modified	3	HH_jit_3_mean	11.4395
TN Modified	4	HH_0.01_pcc_0_1	8.7987
TN Modified	5	MI_dir_0.1_mean	5.5277
TN Combined	1	HH_0.1_pcc_0_1	23.8869
TN Combined	2	HH_jit_1_mean	16.6159
TN Combined	3	HH_1_magnitude_0_1	16.3288
TN Combined	4	HH_0.01_pcc_0_1	14.2595
TN Combined	5	HpHp_0.01_weight	7.0887

Table B.6. Supervised ML Binary Models Top 5 Most Important Features in Active Wiretap Attack Under Simple Reshaping

Model	Rank	Feature Name	Importance (%)
RF Original	1	HH_jit_0.1_std	9.3957
RF Original	2	HH_jit_0.01_std	7.5032
RF Original	3	HH_3_pcc_0_1	5.2821
RF Original	4	HH_5_pcc_0_1	3.7107
RF Original	5	HpHp_0.01_std	3.5233
RF Modified	1	HH_jit_5_std	13.6925
RF Modified	2	HH_jit_3_std	12.8745
RF Modified	3	HH_jit_1_std	9.1901
RF Modified	4	HpHp_0.01_weight	6.2538
RF Modified	5	HH_jit_5_mean	3.6133
RF Combined	1	HH_jit_5_mean	6.8157
RF Combined	2	HH_jit_3_mean	6.0883
RF Combined	3	MI_dir_3_weight	4.8452
RF Combined	4	MI_dir_5_weight	4.6449
RF Combined	5	HH_3_weight	4.5465
LGBM Original	1	HH_jit_0.1_std	36.3747
LGBM Original	2	HH_jit_0.01_std	33.9283
LGBM Original	3	HH_jit_0.01_weight	11.1431
LGBM Original	4	HH_5_pcc_0_1	6.3334
LGBM Original	5	MI_dir_0.01_mean	3.9304
LGBM Modified	1	HH_jit_5_std	98.2965
LGBM Modified	2	HpHp_0.01_weight	1.0196
LGBM Modified	3	HH_0.1_pcc_0_1	0.6839
LGBM Modified	4	HH_jit_3_std	0.0000
LGBM Modified	5	MI_dir_0.1_mean	0.0000
LGBM Combined	1	HH_jit_5_mean	73.8624
LGBM Combined	2	HpHp_0.01_std	25.4191
LGBM Combined	3	HH_0.1_pcc_0_1	0.4072
LGBM Combined	4	HpHp_0.01_weight	0.3113
LGBM Combined	5	MI_dir_5_weight	0.0000
TN Original	1	MI_dir_3_weight	22.3640
TN Original	2	HpHp_0.01_magnitude_0_1	16.9041
TN Original	3	MI_dir_5_weight	16.5567
TN Original	4	HH_0.1_weight	15.6959
TN Original	5	HpHp_3_covariance_0_1	12.3088
TN Modified	1	MI_dir_3_weight	20.7423
TN Modified	2	HH_0.01_pcc_0_1	18.4586
TN Modified	3	HpHp_3_radius_0_1	18.4409
TN Modified	4	HpHp_0.1_weight	9.3864
TN Modified	5	HH_jit_5_std	7.5662
TN Combined	1	MI_dir_3_weight	36.4952
TN Combined	2	HpHp_0.01_weight	9.8433
TN Combined	3	HH_3_radius_0_1	7.9311
TN Combined	4	HpHp_0.01_std	6.7839
TN Combined	5	HH_0.01_weight	6.2712

Table B.7. Supervised ML Binary Models Top 5 Most Important Features in Active Wiretap Attack Under LSTM Reshaping

Model	Rank	Feature Name	Importance (%)
RF Original	1	HH_jit_0.1_std	9.3957
RF Original	2	HH_jit_0.01_std	7.5032
RF Original	3	HH_3_pcc_0_1	5.2821
RF Original	4	HH_5_pcc_0_1	3.7107
RF Original	5	HpHp_0.01_std	3.5233
RF Modified	1	HpHp_0.1_weight	18.3285
RF Modified	2	HpHp_0.01_weight	7.7259
RF Modified	3	HH_0.1_weight	4.6664
RF Modified	4	HH_jit_3_mean	4.2530
RF Modified	5	HH_3_pcc_0_1	3.8511
RF Combined	1	HH_jit_3_mean	7.4004
RF Combined	2	MI_dir_3_weight	6.1103
RF Combined	3	HH_jit_5_mean	5.7301
RF Combined	4	HH_3_weight	5.6095
RF Combined	5	HH_0.01_mean	4.8755
LGBM Original	1	HH_jit_0.1_std	36.3747
LGBM Original	2	HH_jit_0.01_std	33.9283
LGBM Original	3	HH_jit_0.01_weight	11.1431
LGBM Original	4	HH_5_pcc_0_1	6.3334
LGBM Original	5	MI_dir_0.01_mean	3.9304
LGBM Modified	1	HpHp_0.1_weight	53.8580
LGBM Modified	2	HpHp_0.01_weight	35.5241
LGBM Modified	3	HH_0.1_pcc_0_1	6.9930
LGBM Modified	4	MI_dir_0.1_mean	3.5423
LGBM Modified	5	MI_dir_5_weight	0.0826
LGBM Combined	1	HH_jit_5_mean	73.8013
LGBM Combined	2	HpHp_0.01_std	25.4248
LGBM Combined	3	HH_0.1_pcc_0_1	0.4504
LGBM Combined	4	HpHp_0.01_weight	0.3121
LGBM Combined	5	HH_jit_0.1_mean	0.0082
TN Original	1	MI_dir_3_weight	22.3640
TN Original	2	HpHp_0.01_magnitude_0_1	16.9041
TN Original	3	MI_dir_5_weight	16.5567
TN Original	4	HH_0.1_weight	15.6959
TN Original	5	HpHp_3_covariance_0_1	12.3088
TN Modified	1	MI_dir_0.01_std	35.4781
TN Modified	2	HpHp_0.1_covariance_0_1	27.3848
TN Modified	3	HpHp_0.01_std	12.1032
TN Modified	4	HH_1_weight	11.0901
TN Modified	5	HH_5_mean	5.6162
TN Combined	1	MI_dir_3_weight	49.1543
TN Combined	2	MI_dir_0.1_std	16.6930
TN Combined	3	MI_dir_5_std	11.0750
TN Combined	4	HH_jit_0.1_weight	7.2250
TN Combined	5	HH_5_mean	6.7203

C. TOP FIVE MOST IMPORTANT FEATURES TABLES FOR SUPERVISED ML MULTICLASS CLASSIFICATION

Tables C.1–C.4 display the top five most important features of each trained multiclass model in the SSDP flood, Mirai, fuzzing, and active wiretap attacks under both Simple and LSTM reshaping strategies.

Table C.1. Supervised ML Multiclass Models Top 5 Most Important Features in SSDP Flood Attack

Model	Reshaping	Rank	Feature Name	Importance (%)
RF	Simple	1	HH_5_magnitude_0_1	7.1078
RF	Simple	2	MI_dir_0.01_std	6.5285
RF	Simple	3	MI_dir_0.1_std	5.8781
RF	Simple	4	HH_5_radius_0_1	5.3518
RF	Simple	5	HH_jit_5_weight	4.9417
RF	LSTM	1	HH_5_magnitude_0_1	7.0602
RF	LSTM	2	MI_dir_0.01_std	6.5868
RF	LSTM	3	MI_dir_0.1_std	5.9033
RF	LSTM	4	HH_5_radius_0_1	5.3774
RF	LSTM	5	HH_jit_5_weight	5.3347
LightGBM	Simple	1	HH_1_magnitude_0_1	36.1135
LightGBM	Simple	2	HH_jit_5_std	31.6778
LightGBM	Simple	3	HH_jit_3_weight	18.0368
LightGBM	Simple	4	MI_dir_1_weight	13.6857
LightGBM	Simple	5	MI_dir_0.1_std	0.1940
LightGBM	LSTM	1	HH_5_magnitude_0_1	36.0952
LightGBM	LSTM	2	HH_jit_5_weight	31.7455
LightGBM	LSTM	3	HH_0.1_magnitude_0_1	21.6858
LightGBM	LSTM	4	HH_1_radius_0_1	9.5335
LightGBM	LSTM	5	MI_dir_5_weight	0.2774
TabNet	Simple	1	HH_3_magnitude_0_1	23.4770
TabNet	Simple	2	HpHp_1_magnitude_0_1	15.5614
TabNet	Simple	3	HpHp_1_weight	10.7468
TabNet	Simple	4	HH_5_weight	7.6168
TabNet	Simple	5	HH_jit_3_weight	7.0613
TabNet	LSTM	1	HH_5_magnitude_0_1	17.4711
TabNet	LSTM	2	HpHp_0.01_weight	12.9951
TabNet	LSTM	3	MI_dir_1_weight	12.0614
TabNet	LSTM	4	HH_5_radius_0_1	10.8315
TabNet	LSTM	5	HH_1_radius_0_1	10.2269

Table C.2. Supervised ML Multiclass Models Top 5 Most Important Features in Mirai Attack

Model	Reshaping	Rank	Feature Name	Importance (%)
RF	Simple	1	HpHp_0.1_weight	9.9913
RF	Simple	2	HpHp_3_weight	9.0489
RF	Simple	3	MI_dir_3_weight	8.9380
RF	Simple	4	HH_3_weight	7.4927
RF	Simple	5	HH_1_weight	5.1350
RF	LSTM	1	HpHp_0.1_weight	10.2673
RF	LSTM	2	MI_dir_3_weight	8.9378
RF	LSTM	3	HpHp_3_weight	8.4938
RF	LSTM	4	HH_3_weight	7.4906
RF	LSTM	5	HH_jit_3_mean	5.0891
LightGBM	Simple	1	MI_dir_5_weight	38.6750
LightGBM	Simple	2	HH_jit_0.1_mean	38.5882
LightGBM	Simple	3	MI_dir_0.1_mean	22.5115
LightGBM	Simple	4	HpHp_0.1_weight	0.0488
LightGBM	Simple	5	HpHp_5_weight	0.0358
LightGBM	LSTM	1	HpHp_0.1_weight	38.6643
LightGBM	LSTM	2	HH_jit_1_weight	38.5756
LightGBM	LSTM	3	MI_dir_0.1_mean	22.5113
LightGBM	LSTM	4	MI_dir_1_mean	0.1347
LightGBM	LSTM	5	HpHp_5_weight	0.0815
TabNet	Simple	1	HpHp_5_mean	18.5420
TabNet	Simple	2	HpHp_5_magnitude_0_1	15.6338
TabNet	Simple	3	HpHp_5_weight	15.5064
TabNet	Simple	4	HH_0.1_std	15.1042
TabNet	Simple	5	MI_dir_0.1_std	11.1903
TabNet	LSTM	1	MI_dir_1_weight	34.6371
TabNet	LSTM	2	HpHp_5_weight	20.7993
TabNet	LSTM	3	HpHp_5_magnitude_0_1	13.2195
TabNet	LSTM	4	HpHp_3_std	8.1402
TabNet	LSTM	5	HH_jit_0.01_std	8.1161

Table C.3. Supervised ML Multiclass Models Top 5 Most Important Features in Fuzzing Attack

Model	Reshaping	Rank	Feature Name	Importance (%)
RF	Simple	1	HH_0.01_magnitude_0_1	7.9142
RF	Simple	2	HH_0.1_magnitude_0_1	7.6546
RF	Simple	3	MI_dir_5_weight	6.6364
RF	Simple	4	MI_dir_3_weight	6.0470
RF	Simple	5	HH_1_magnitude_0_1	5.5434
RF	LSTM	1	HH_0.01_magnitude_0_1	7.9140
RF	LSTM	2	HH_0.1_magnitude_0_1	7.6507
RF	LSTM	3	MI_dir_5_weight	6.5095
RF	LSTM	4	MI_dir_3_weight	6.0469
RF	LSTM	5	HH_1_magnitude_0_1	5.5372
LightGBM	Simple	1	MI_dir_0.01_std	22.5121
LightGBM	Simple	2	MI_dir_0.01_mean	17.8340
LightGBM	Simple	3	HpHp_0.01_std	9.7261
LightGBM	Simple	4	HpHp_5_std	9.4247
LightGBM	Simple	5	HpHp_0.01_weight	8.7271
LightGBM	LSTM	1	HH_0.01_magnitude_0_1	32.0186
LightGBM	LSTM	2	HH_jit_3_std	31.9725
LightGBM	LSTM	3	HH_jit_0.1_std	26.2518
LightGBM	LSTM	4	HH_0.1_radius_0_1	6.7681
LightGBM	LSTM	5	MI_dir_0.01_std	0.8732
TabNet	Simple	1	HH_0.01_magnitude_0_1	40.9282
TabNet	Simple	2	HH_0.01_pcc_0_1	17.3844
TabNet	Simple	3	HpHp_1_weight	8.7580
TabNet	Simple	4	HH_jit_0.1_weight	7.7940
TabNet	Simple	5	HH_5_radius_0_1	6.5108
TabNet	LSTM	1	HpHp_1_weight	21.7289
TabNet	LSTM	2	HH_5_radius_0_1	15.6979
TabNet	LSTM	3	HH_0.1_weight	13.2928
TabNet	LSTM	4	HH_0.01_pcc_0_1	10.3961
TabNet	LSTM	5	HH_jit_5_weight	9.7066

Table C.4. Supervised ML Multiclass Models Top 5 Most Important Features in Active Wiretap Attack

Model	Reshaping	Rank	Feature Name	Importance (%)
RF	Simple	1	HH_jit_3_std	8.4345
RF	Simple	2	HH_jit_5_std	6.8001
RF	Simple	3	HH_jit_1_std	6.0408
RF	Simple	4	HH_jit_0.1_std	5.4463
RF	Simple	5	HH_jit_5_mean	4.8177
RF	LSTM	1	HpHp_0.1_weight	9.0914
RF	LSTM	2	MI_dir_3_weight	4.7342
RF	LSTM	3	HH_jit_3_mean	4.7056
RF	LSTM	4	HH_3_weight	4.1273
RF	LSTM	5	HH_jit_1_std	3.6682
LightGBM	Simple	1	HH_jit_5_std	30.4505
LightGBM	Simple	2	HH_jit_5_mean	27.0131
LightGBM	Simple	3	HpHp_0.01_std	11.9904
LightGBM	Simple	4	HH_jit_0.1_std	11.9115
LightGBM	Simple	5	HH_jit_0.01_std	10.1477
LightGBM	LSTM	1	HH_jit_5_mean	26.9774
LightGBM	LSTM	2	HpHp_0.1_weight	19.3555
LightGBM	LSTM	3	HpHp_0.01_std	11.9934
LightGBM	LSTM	4	HH_jit_0.1_std	11.9121
LightGBM	LSTM	5	HH_jit_0.01_std	10.1430
TabNet	Simple	1	HH_jit_1_weight	14.1224
TabNet	Simple	2	MI_dir_5_std	13.4407
TabNet	Simple	3	HH_0.1_covariance_0_1	13.1488
TabNet	Simple	4	HH_0.01_pcc_0_1	7.9339
TabNet	Simple	5	MI_dir_3_weight	7.3585
TabNet	LSTM	1	HH_jit_1_weight	28.9146
TabNet	LSTM	2	MI_dir_0.01_std	14.6371
TabNet	LSTM	3	HpHp_0.01_radius_0_1	11.9570
TabNet	LSTM	4	HH_jit_0.01_weight	8.8111
TabNet	LSTM	5	MI_dir_0.01_mean	8.6916