DATA CURATION AND QUALITY ASSURANCE FOR MACHINE LEARNING-BASED CYBER INTRUSION DETECTION

Haihua Chen
Department of Information Science
University of North Texas
Denton, TX 76203
haihua.chen@unt.edu

Ngan Tran
Department of Information Science
University of North Texas
Denton, TX 76203
NganTran4@my.unt.edu

Anand Sagar Thumati
Department of Information Science
University of North Texas
Denton, TX 76203
AnandSagarThumati@my.unt.edu

Jay Bhuyan
Computer Science Department
Tuskegee University
Tuskegee, AL 36088
jbhuyan@tuskegee.edu

Junhua Ding *
Department of Information Science
University of North Texas
Denton, TX 76203
junhua.ding@unt.edu

May 10, 2021

ABSTRACT

xxxxxxx.

1 Introduction

The increasing usage of digital devices in cyber-physical system (CPS) has enhanced the efficiency of operation systems, but also led to the vulnerability of cyber-attacks. Cyber assaults on process control and monitoring those intelligent systems could lead to a significant control failure [1] and even huge economic losses. This makes cyber security a major concern due to the high level of attacks on networks and systems for CPS [2]. Therefore, building an intrusion detection systems (IDS) to predict and respond to assaults against CPS, has become an essential task among the soft engineering community. However, it is very challenge because of the range of novelty involved in cyber-attacks [1]. Recently, machine learning and deep learning have been applied for intrusion detection at both operation system level (host-based intrusion detection systems, called HIDS) and network level (network-based intrusion detection systems, called NIDS). As pointed by Andrew Ng[2], both the models/algorithms and the quality of the data greatly impact the performance of machine learning systems. The computing rule of “garbage in, garbage out" is still applicable to machine learning [3] and the lacking of high-quality training data becomes a barrier for building high-performance machine learning systems [4]. Therefore, we should not only optimize the models but also systematically evaluate and ensure the data quality to improve performance for intrusion detection.

However, current studies on machine learning-based intrusion detection only focus on model construction and optimization. For example, Sahu et al. compared the performance of various linear and non-linear classifiers for NIDS using the KDD dataset[5], while Al-Maksousy compared deep neural networks (DNN) and verious traditional machine learning (ML) models for NIDS on the same dataset, finding that DNN outperformed ML models in terms of accuracy, running time, and false positive rate [6]. Hu et al proposed an incremental HMM training framework that incorporates a simple data pre-processing method for identifying and removing similar sub-sequences of system calls for HIDS [7]. This training strategy has been widely applied since it can save the training cost (especially on large data) without noticeable degradation of intrusion detection performance [7]. Convolutional neural network (CNN) and recurrent neural network (RNN) have also been used in HIDS [8, 2]. Recently, Liu and Lang conducted a comprehensive survey on machine

*Corresponding author.

[2] youtube.com/watch?v=06-AZXmwHjo

learning and deep learning methods for IDS [9]. Nevertheless, very few research has paid attention to data requirements, data quality issues and data quality assurance for IDS.

The input of machine learning-based intrusion detection is a collection of data instances, each of which is represented by only one feature or a group of features. The features can be binary, categorical, or continuous. Data instances can be related to each other, such as sequence data, spatial data, and graph data. Each instance can be labeled as normal class or anomaly class or is not labeled. The data requirements for different machine learning techniques include supervised learning, semi-supervised learning, and unsupervised learning vary. Training data set should be labeled as normal and anomaly classes for supervised learning. The model is trained on the labeled data, then any unseen data is input to the trained model to determine which class it belongs to [10]. Semi-supervised learning assume labeled instances only include either normal class or anomaly class, then the trained model is used to identify anomalies in the test data [10]. Label noisy, insufficient labeled data, class imbalance are the most common data quality issues for supervised learning and semi-supervised learning based-IDS. Differently, unsupervised learning does not require labeled data [10]. However, unsupervised learning techniques usually assume that normal instances are far more frequent than anomalies in the test data. Under this assumption, semi-supervised learning can be incorporated to unsupervised learning by taking a sample of the unlabeled data set as training data to improve the robust of the model. Other data quality issues such as inconsistent, duplication, incompleteness, incomprehension, no variety, imprecise timestamps might also exist of the data input of all the machine learning models.

Instead of optimizing the machine learning-based IDS from the model/algorithm perspective, this paper targets on the data-enteric IDS and discuss how to systematically evaluate and ensure the data quality to improve performance for intrusion detection. We will answer a few important questions regarding the data quality to the performance of machine learning-based IDS:

- How to prepare an appropriate dataset for a intrusion detection in a specific scenario? What are the data requirements? We investigate and compare the existing datasets used for different intrusion detection tasks to answer this questions.
- How to evaluate the data quality? How to judge whether the data quality or the machine learning model has a major effect on the intrusion detection performance? We conduct a case study on a host-based intrusion detection system, which aims to detect user anomalous behaviors in an operating system based on system call sequences. The machine learning techniques we implement include K-means, logistic regression, support vector machine, deep neural network, BERT, and GPT. A comprehensive analysis on the experiments is performed to answer the second question.
- What are the strategies to assure the data quality if data quality issues exist in the datasets? Based on discussions in the experiment results, we propose four quality attributes that are most critical to the "fit for purposes" of intrusion detection. We also discuss the approach to evaluate each quality attributes.

To the best of our acknowledge, this is the first study which explores the intrusion detection from data-centric rather than model-centric perspective. It will benefit the IDS reseacher and practitioner with new insights on improving the intrusion detection performance by enhancing the data quality. The rest of the paper is structured as follows: Section 2 presents the literature review related to machine learning-based intrusion detection systems and data quality. Section 3 discusses the data preparation and quality requirement for intrusion detection. Section 4 introduce the experiments, experiment results, and analysis. Section 5 proposes the data quality assessment and assurance strategies. Section 6 concludes the paper and discuss the future work. The code and datasets used in this research are available at:

2 Related work

2.1 Machine learning-based intrusion detection systems

Intrusion detection aims to detect the malicious activities or intrusions (break-ins, penetrations, and other forms of computer abuse) in a computer related system (operation system or network system) [10]. An intrusion is different from the normal behavior of the system, and hence the techniques used for anomaly detection can also be used for intrusion detection.

The machine learning techniques used for intrusion detection can be divided into supervised learning and unsupervised learning. Whether the labeling data sufficient or not became the key criteria of selecting the a machine learning technique. However, the detection performance of unsupervised learning methods is usually inferior to those of supervised learning methods [9]. Meanwhile, due to the issues that the data for intrusion detection typically comes in a streaming fashion and the data imbalance caused by the low false alarm rate, the usage of machine learning techniques in intrusion detection is more challenge than other anomaly detection applications. Table 4 summarizes the machine learning techniques and algorithms used for HIDS and NIDS in the last five years.

Table 1: Machine learning techniques for HIDS and NIDS. The full-names and the abbreviations of the models are introduced in Appendix A.

Technique used	Model	Datasets	HIDS/NIDS	Year	Reference
Supervised	KNN, SR	KDD-Cup99	NIDS	2017	[11]
Supervised	LR, RF	UNSW	NIDS	2017	[12]
Supervised	DNN, imbalanced network traffic, RF, VAE	CIDDS-001	NIDS	2018	[13]
Supervised	DNN	NSL-KDD	NIDS	2018	[14]
Supervised	DNN	NSL-KDD	NIDS	2018	[15]
Supervised	LR, NB, KNN, DT, AdaBoost, RF, CNN, CNN-LSTM, LSTM, GRU, SimpleRNN, DNN	CICIDS-2017, UNSW-NB15, ICS cyberattack	NIDS	2018	[16]
Unsupervised	Metric learning + clustering + SVM	Kyoto 2006, NSL-KDD	NIDS	2019	[17]
Supervised	NB, AODE, RBFN, MLP, J48 DT	UNSW-NB15	NIDS	2019	[18]
Supervised	Pruned exact linear time, quantile regression forests	NetFlow data	NIDS	2020	[19]
Unsupervised	Autoencoder, IF, KNN, K-Means, SCH, SVM	NSL-KDD, ISCX	NIDS	2020	[20]
Unsupervised	IF, HBOS, CBLOF, K-Means	BRO DNS, BRO CONN	NIDS	2020	[21]
Supervised	DNN	KDD-Cup99, NSL-KDD, UNSW-NB15	NIDS	2020	[22]
Supervised	KNN, RF, SVM-rbf, DNN, ResNet-50, one-vs-all classifier, multiclass classifier	NSL-KDD	NIDS	2020	[23]
Supervised	NB, DT, RF, ANN	KDD-Cup99	NIDS	2020	[24]
Supervised	SVM, MLP, NB, DT	ADFA-LD	HIDS	2017	[25]
Supervised	CNN	NGIDS-DS, ADFA-LD	HIDS	2017	[8]
Semi-Supervised	SC4ID	ADFA-LD, UNM dataset	HIDS	2018	[26]
Supervised	GRU, LSTM, CNN+GRU	ADFA-LD	HIDS	2019	[2]
Supervised	LR, SVM, DT, RF, ANN	DS2OS traffic traces	HIDS	2019	[19]
Supervised	NN, DT, linear discriminate analysis with the bagging algorithm	NSL-KDD	HIDS	2019	[27]
Supervised	NB, LR, KNN, SVM, IntruDTree	Kaggle cybersecurity datasets	HIDS	2020	[28]

As can be seen from table 4, most of existing studies are focusing on NIDS. More datasets have been created for NIDS and different machine learning algorithms have been explored. However, compared to NIDS, HIDS is more challenge due to [29]: (1) More labeled data is required to reduce the false positive alarm rate. (2) It difficult to design an efficient HIDS which can prevent the outgoing denial-of-service attacks. (3) In shared system environment the HIDS need to be work as an independent module since the shared parameters may cause the attack. Nowadays HIDS are becoming more important and plays a major role in most of the intrusion detection systems [29]. Even though some studies have been conducted on HIDS [25, 8, 26, 2, 19, 28], more attentions should be paid on HIDS. The applications of the state-of-the-art techniques such as combining powerful language models with deep learning might be a promising direction.

2.2 Datasets for intrusion detection

As shown in table 4, many datasets have been created for intrusion detection [30, 31]. Datasets for NIDS mainly include information from the packet itself and aims at detecting the malicious activity in network traffic using the content of individual packets, while datasets for HIDS usually include information about events or system calls/logs on a particular system with the purpose of detecting vulnerability exploits against a target application or computer system [2]. We conduct an investigation on the popular datasets used for NIDS and HIDS, as shown in table 2.

Table 2: Overview of public datasets for IDS. The detail description of the datasets are presented in Appendix B.

Dataset	Volume	Information	Format	Labeled	Balanced	Year
		NIDS				
KDD-Cup99 [32]	4.9 millions for training, 2 million for testing	41 features, 20 types of attacks	packet, logs	yes	no	1999
Kyoto 2006 [33]	3,054,682 for training, 1,563,923for testing	14 statistical features derived from the KDD-Cup99 and 10 additional features	packet, logs	yes	yes	2006
DARPA-2009 [34]	673931 records for training and 74880 records for testing	16 network features and 26 packet features; The dataset consists of 7000 pcap files	packets	yes	no	2009
NSL-KDD [35]	125,973 for training, 22,544 for testing	41 features, 22 types of attacks	packet, logs	yes	no	2009
ISCX [36]	30,814 normal traces and 15,375 attack traces for training, 13,154 normal traces and 6,580 attack traces for testing	1.5 million network traffic packets, with 20 features and covered seven days of network activity	packets	yes	no	2012
UNSW-NB15 [37]	175,341 records for training, 82,332 records for testing	49 features in pcap file format and 9 categories of attacks	packets	yes	no	2015
NGIDS-DS [38]	631,85,992 records for training and 34,987,493 records for testing	The dataset contains 90 million records, including 88,791,734 for benign and 1,262,426 records for malicious activities. It contiens 7 features for ground-truth cs; 9 features for the 99 csv files of host logs; and 18 features for NGIDS.pcap	packet, logs	yes	no	2016
CICIDS2017 [39]	75,561 records for training and 25,187 records for testing	The dataset contains 3119345 instances and 83 features containing 15 class labels	packets	yes	no	2017
		HIDS				

DARPA 98/99 [40]	4,898,431 records for training and 2,984,154 records for testing	consist of 409,021 records, 97,277 normal traces and 311,744 intrusion traces with 41 features and classes labeled as either normal or any of the 22 types of attacks	packet, logs	yes	no	1998
UNM dataset [41]	627 system-call sequences for training set and 3,136 system-call sequences for testing set	consists of 4,298 normal traces and 1,001 intrusion traces and 467 features	logs	yes	no	1998
KDD99 [32]	contains 494,021 and 311,029 records in the training and testing sets	consists of 4,898,430 records 1,033,372 normal traces, 4,176,086 attack traces and 41 features	packets	yes	no	1999
NSL-KDD [35]	consists of 1,152,281 distinct records from KDD99 dataset 860,725 normal traces and 291,556 attack traces	contains 41 features per record	packets	yes	no	2000
ADFA-LD [42]	833 traces and 308,077 system calls for training, 4373 traces and 2,122,085 system calls for testing	746 trace attack sequences and 317,388 system call attack sequences		yes	no	2014
ADFA-WD [42]	355 traces and 13,504,419 system calls for training, 1827 traces and 117,918,735 system calls for testing	5542 trace attack sequences and 74,202,804 system call attack sequences		yes	no	2014
DARPA-2009 [34]	673931 records for training and 74880 records for testing	16 network features and 26 packet features; The dataset consists of 7000 pcap files	packets	yes	no	2009
ADFA-IDS [43]	308,077 system calls and 833 traces for training and 212,2085 system calls and 4372 traces for testing	continues 15 attack types and 3, 6636 malicious system call traces	logs	yes	no	2013
NGIDS-DS [38]	313,926 records with 7 attributes in ground-truth csv file; 90,054,160 records (1,262,426 attack and 88,791,734 normal) with 9 attributes in 99 csv files of host logs; 1,094,231 capture packets with 18 unique IPs in NGIDS.pcap file	cyber normal and abnormal traffic scenarios for different enterprises	packet, host logs	yes	no	2017

DS2OS traffic traces [44]	61.52 MB	contains traces captured in the IoT environment DS2OS for different services: light controller, thermometer, movement sensors, washing machines, batteries, thermostats, smart doors and smart phones	–	yes	yes	2018
Kaggle cybersecurity datasets [28]	25,000 instances	3 qualitative features and 38 quantitative features	–	yes	yes	2020

Generally, the following properties are required when creating a IDS dataset: (1) Normal user behavior. The quality of an IDS is primarily determined by its attack detection rate and false alarm rate. Therefore, the presence of Normal user behavior is indispensable for evaluation an IDS [30]. (2) Attack traffic. The attack types in different scenarios varies, it is necessary to clarify the attacks in the IDS dataset. (3) Format. IDS dataset can be in different formats such as packet-based, flow-based, host-based log files, etc. (4) Anonymity. Some of the information is anonymized due to privacy concerns, this indicates which attributes will be affected. (5) Duration. It indicates the recording time (e.g., daytime vs. night or weekday vs. weekend) of the dataset since a behavior might be regarded as an attack only in a specific duration. (6) Labeled. Labeled datasets are necessary for training supervised learning and semi-supervised learning models and for evaluating supervised learning, semi-supervised learning, and unsupervised learning models. (7) Other information such as attack scenarios, network structure, IP addresses, recording environment, download url are also useful. Quality issues can easily appear in the above information. Those data quality issues, if not checked and eliminated appropriately, will greatly affect the intrusion detection performance. However, few studies have discussed the qualities of IDS datasets [36, 38] for machine learning, although data quality issues such as duplication, imbalance have been reported in KDD dataset [35].

2.3 Data quality evaluation and assurance for machine learning

Poor data quality has a direct impact on the performance of the machine learning system that is built on the data. For example, a face recognition-based gender classification system that was implemented with machine learning algorithms produced 0.8% error rate for recognizing the faces of lighter-skinned males, but as high as 34.7% error rate for recognizing the faces of darker-skinned females [45]. The problem was due to the significant imbalance of the training datasets in skin colors [45]. Recently, a study showed that 10 of the most commonly-used computer vision, natural language, and audio datasets had serious data quality issues [46]. The computing rule of “garbage in, garbage out" is also applicable to machine learning-based anomaly detection. However, there is not much research on the analysis of low-quality training data and its impact on machine learning-based anomaly detection.

One of the fundamental issue might be the label noisy. To investigate the impact of the quality of the labelling process on the performance of the machine learning-based network intrusion detection, Lauría and Tayi compared two classification algorithms (DT and NB) which were trained on the data with poor quality [47]. The experiments showed that data with totally clean labels may not be required to train a classifier that performs at an acceptable level as a detector of network intrusions [47]. Class imbalance is another common issue in IDS. As pointed by Sahu at al., both the datasets CIDDS and KDD are imbalanced in classes, and the distribution of the KDD is even less uniform: the two most dominant classes both have more than 40,000 instances while the least dominant 16 classes have less than 1,000 instances [5]. Their experiments demonstrated that data balancing cannot improve its performance if the training dataset is less uniform and data balancing will benefit the neural network if improving the predictive accuracy of less dominant classes is desired [5]. Oversampling and undersampling have been used to deal with the class imbalance in IDS [48]. Missing information, duplicates data, attack diversity, dataset difficulty, and feature sparse can be other factors that reduce the performance of machine learning-based IDS [49, 5]. Different dimensions have been proposed to measure the data quality for machine learning systems. Fan argued that data consistency, data deduplication, information completeness, data currency and data accuracy are the central to data quality [50]. These dimensions are related to data itself. The dimensions related to users include accessibility, integrability, interpretability, rectifiability, relevance, and timeliness [51]. Recently, Chen, Chen, and Ding defined “data quality" as a measurement of data for fitting the purpose of building a machine learning system [52]. Dimensions such as comprehensiveness, correctness, and variety are critical to evaluate the data quality for machine learning systems [52]. Gradient boosted decision tree, data filtering, SVM, transfer learning have been investigated for data quality assurance and improvement [53, 52, 54].

3 Data preparation and quality requirement for intrusion detection

As discussed above, data quality is crucial for machine learning-base intrusion system. However, each stage of the data preparation can be plagued with problems of data quality, such as scattered presence of the data source, insufficient data during the data collecting, label noise during the data annotation, and overlapping issue during the training-testing data splitting. Therefore, it is necessary to identify the dataset attributes and potential, then develop a guideline for quality assurance during the data preparation.

3.1 Data preparation workflow

Data preparation for machine learning-based IDS mainly include four steps: (1) Selecting a data source or multiple data sources. (2) Collecting the data from the selected data source. (3) Labelling the data for training and testing. (4) Preprocessing the data as the model input. The workflow is shown in figure 1. The data sources for HIDS and NIDS are different: Data for HIDS can be collected from audits records, log files, Application Program Interface (API), rule patterns, system calls, while data for NIDS is usually collected from the simple network management protocol (SNMP), network packets (TCP/UDP/ICMP), management information base (MIB), router netFlow records. Data can be collected from one data source or by integrating multiple data sources. Data source is the foundation of accessing high-quality data. Once the data source is confirmed, we should collect information such as metadata, format, duration, etc. for further analysis. The data collecting procedure should be well designed to ensure the data quality. For example, when collecting the sequential events, they should be correctly organized by their order. Data labelling is an essential step for supervised machine learning-based IDS. Most existing machine learning algorithms consider the assumption that the training data feeding the algorithms is accurate (has no errors). However, errors in label data entry, lack of precision in expert judgement, imbalance data distribution in different categories in the process of labelling the training examples can impact the predictive accuracy of the classification algorithms [47]. Data preprocessing aims at removing the outliers, cleaning the data, extracting the useful features, and splitting the data for training and test. This process, if handled inappropriately, will cause the data quality issues such as data sparse and bias, overlapping between training and test, which can also reduce the performance of the machine learning algorithms.

3.2 Dataset properties and attributes

As pointed by [30], the certain properties of a dataset should be collected and evaluated based on a specific scenario. We believe that to build a machine learning-based IDS, the following information should be collected during the data curation.

3.2.1 General Information

General information of a dataset might include year of creation, public availability, metadata, format, data volume. Both network traffic and system calls face the issue of concept drift overtime, and new attacks might appear in any scenario. A machine learning model build in 1990 might not be used to predict the data in 2021. Therefore, the year of creation (age) of an intrusion detection datasets is critical for deciding the scope of an IDS. Intrusion detection datasets should be publicly available to serve as a basis for comparing different intrusion detection methods and for quality evaluation by third parties [30]. Metadata such as network structure, hosts, IP addresses, configuration of the clients, attack scenarios can provide users content-related explanations of the results. IDS datasets can be roughly divided into three formats: (1) packet, (2) flow, (3) logs. The processing of different data format is different. The format is directly associated with the volume of data. Data volume can be described with the number of packets, flows, points, instances, which is crucial for selecting machine learning models.

3.2.2 Duration

Duration refers to the timestamps that the data was collected. For eample, the MIT dataset includes live normal data for lpr for 2 weeks using 77 hosts, while the ISCX dataset consists of the 7 days of network activity (normal and malicious) from Information Security Center of Excellence (ISCX) at the University of New Brunswick. As mentioned in [55], in order to enable the evaluation of detection algorithms that consider the cyclostationary evolution of traffic, i.e., differences in traffic between daytime/nighttime or weekdays/weekends, a long duration trace is needed.

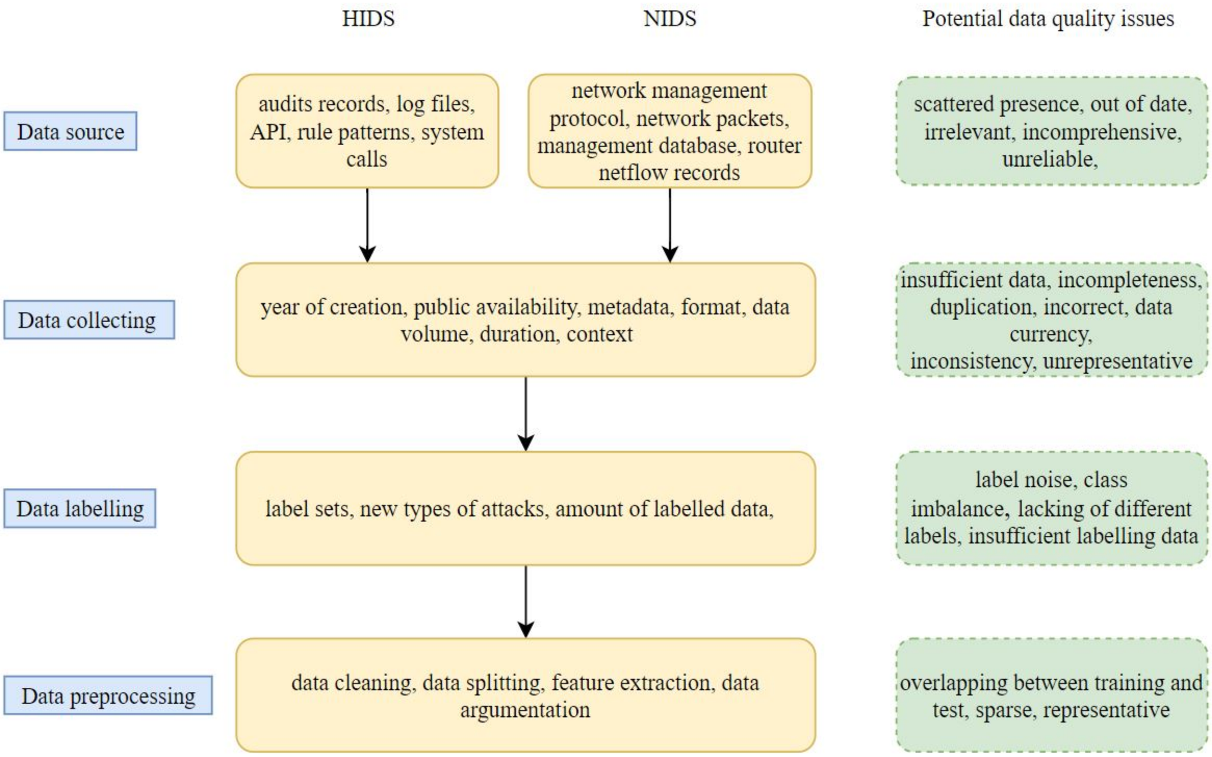

HIDS
NIDS
Potential data quality issues
Data source
audits records, log files, API, rule patterns, system calls
network management protocol, network packets, management database, router netflow records
scattered presence, out of date, irrelevant, incomprehensive, unreliable,
Data collecting
year of creation, public availability, metadata, format, data volume, duration, context
insufficient data, incompleteness, duplication, incorrect, data currency, inconsistency, unrepresentative
Data labelling
label sets, new types of attacks, amount of labelled data,
label noise, class imbalance, lacking of different labels, insufficient labelling data
Data preprocessing
data cleaning, data splitting, feature extraction, data argumentation
overlapping between training and test, sparse, representative

Figure 1: Data preparation workflow for machine learning-based IDS

3.2.3 Context

Context is related to the recording environment. It delineates the network environment and conditions in which the datasets are captured. As for NIDS, the kind of traffic, type of network, and complete network indicate the context information. HIDS is generally a software component located on the system being monitored is typically monitor a single system. The context information allows deeper understanding of processes and activities.

3.2.4 Normal traces and types of attacks for intrusion traces

This information is related to data labelling, which is the foundation of machine learning-based IDS. The training/validate/testing data should ideally be correctly labeled as normal or not. In case of attack records, the type of attack is also needed [55]. Maciá-Fernández et al applied the following strategy to label a instance: a) an attack label for the flows that they positively know that correspond to an attack, b) a normal label for those that are generated synthetically with normal patterns, and c) a background label for those which no one knows exactly if they are attacks or not [55]. Ring et al summarized the specific types of attacks used in different datasets.

3.2.5 Features

The input of machine learning-based IDS is a collection of data instances, each of which is represented by only one feature or a group of features. As shown in table 2, most of the datasets include the feature information, which can be qualitative features or quantitative features, network features or packet features, n-gram features, and other features. Features are the key components for developing an effective IDS.

3.3 Dataset quality principles and dimensions

Different principles, metrics, dimensions have been proposed to measure data quality [56, 57, 58, 59]. However, few of them are discussed in the context of building machine learning systems [52]. As for machine learning-based IDS, central to data quality are reputation, relevance, comprehensiveness, timeliness, variety, accuracy, consistency, deduplication. We will discuss these dimensions in the following:

Reputation. "Reputation" is related to reliability, believability, and trustworthiness, which lays the foundation of the data quality. Reputation is evaluated using an information-theoretic concept, the Kullback-Leibler distance. Data for IDS can be collected from different sources, including host logs, network traffic, and application data. For a single data source, we can use a collective measure of trustworthiness (in the sense of reliability) based on the referrals or ratings from members in a community to evaluate the reputation [60]. Pagerank is another method for reputation measurement [61]. If we generate the dataset by data fusion of multiple data sources, an "opinion" (a metric of the degree of belief) can be generated to represent the uncertainty in the aggregated result.

Relevance. "Relevance" indicates why the data is collected. Data should be collected and evaluated by "fit for purpose" [4]. For example, HIDS data should be collected from host system and audit sources, such as operating system, window server logs, firewalls logs, application system audits, or database logs. While NIDS data should be extracted from a network through packet capture, NetFlow, and other network data sources. Moreover, if the IDS targets on a specific type of attack such as DDoS attacks, then the data also needs to be relevant to this attack.

Comprehensiveness. Existing machine learning-based IDS frequently suffers from the issues of bias and lacking of robustness, which are mainly caused by the incomprehensiveness of the dataset. For example, the dataset doesn't contain balanced data for different normal or attack behavior, and the data cannot represent various features. Comprehensiveness requires a dataset contains all representative samples from the population [4]. For example, the NSL-KDD dataset includes a total of 39 attacks where each one of them is classified into one of the following four categories (DoS, R2L, U2R, and probe). Suppose all the attacks should have instance in the training set. However, 17 of these attacks is introduced only in the testing set. This dataset cannot be considered as comprehensive. The importance of the comprehensiveness of data to machine learning, especially deep learning, is well understand since a deep learning model normally includes millions of parameters that needs a large amount of data to train it. If a comprehensive datasets with as many different types of attacks included, similar to the ImageNet for computer vision, can be developed for IDS, it would enhance the performance of machine learning-based IDS.

Timeliness. "Timeliness" (also called "currency") refers to the extent to which the age of the data [56] is appropriate for the IDS task. Timeliness is an important factor to affect the performance of machine learning models since new types of attacks are emerging, and some existing datasets such as DARPA and KDD99 are too old to reflect these new attacks. Although ADFA dataset contains many new attacks, it cannot be considered as comprehensiveness. For that reason, testing of machine learning models for IDS using DARPA, KDD99, and ADFA datasets does not offer a real evaluation and could result in inaccurate claims for their effectiveness[31]. Ideally, datasets should include most of the common attacks and correspond to current network environments [9].

Variety. "Variety" concerns about the coverage of the instances on the selected features. For example, the KDD-Cup99 dataset has 41 features, it suppose to be normal distribution in the selected features with known mean and standard deviation in the real world. Otherwise, it will induce the data sparse issue. Moreover, to improve the robustness in machine learning models, the instances in the validate data and test data should be variety enough to test the training model. Variety is considered as a subset of comprehensiveness in the scenario of constructing a machine learning system for intrusion detection.

Accuracy. According to definition from Wang and Strong [56], "accuracy" means "the extent to which data are correct, reliable and certified." Generally, accuracy can be distinguished from two aspects: syntactic and semantic [9]. However, for machine learning systems, the labelling accuracy should also be taken into consideration. Syntactic accuracy aims to check whether a value is any one of the values of or how close it is to the elements of the corresponding defined feature, while semantic accuracy requires an instance to be semantically represented appropriately. Labelling accuracy means that an instance should be correctly labelled as normal or any type of attacks.

Consistency. "Data consistency" refers to the validity and integrity of data representing real-world entities [58]. It aims to detect errors such as inconsistencies and conflicts in the data, typically identified as violations of data dependencies [58]. For example, the system call for HIDS should be represented to ensure the sequential order, and the value of an attribute's domain (feature) should be constrained to the range of admissible values.

Deduplication. Data deduplication is the problem of identifying the same instances. It is a longstanding issue that has been studied for decades. Duplication was usually caused by "data from a large number of (heterogeneous) data sources was not fused appropriately." Recently, Panigrahi and Borah reported that the CICIDS2017 dataset contains many redundant records which seems to be irreverent for training any IDS [62]. However, duplication does not necessarily a data quality issue for machine learning-based IDS. For example, if one of the record (a data instance and its label) appears multiple times in training data, but it reflects the probability of this attack behavior. This will not be considered as a duplication issue. However, having a large overlap between the training and the test data can potentially introduce bias in the model and contribute to high accuracy, as pointed by Lee et al. [63]. Transfer learning [4], distance-based approach [58], rule-based approach [58], and probabilistic [58] can be used for data deduplication.

In Section 5, we will conduct a case study of the datasets used in the experiment study using data quality dimensions discussed above. Based on the case study, we will figure out how data quality affect the performance of machine learning-based IDS, thereby understand the strategies to assure the data quality.

4 Experiment on machine learning-based intrusion detection

4.1 Data collection

4.1.1 Data cleaning and prepossessing

Since the goal of this case study is to develop a host-based intrusion detection (HIDS), we conduct the machine learning experiments on UNM, MIT and ADFA-LD datasets. A detail introduction of these datasets can be found in Appendix B. Regardless of a slight difference in ADFA and UNM data format, we use similar data cleaning and augmentation techniques to create a dataset for each class. Processing data for pre-trained language models vectors such as BERT [64] and GPT-2 [65] is similar for normal machine learning algorithms. We use tokenizer to parse data into system call sequences with a length of 6. Since UNM datasets contain system calls from concurrently running processes, we group them by PID to ensure their sequential order. On the other hand, as ADFA-LD dataset is already organized by different processes, and there is no PID provided, we do not need to group system calls together. Once the data is in order, we tokenize them into a sequence of six. Let's say a trace is:

open, read, mmap, fork, getpid, open, write, close, close.

After tokenizing into 6-grams, four different sequences will be produced:

Sequence 1: open, read, mmap, fork, getpid, open.

Sequence 2: read, mmap, fork, getpid, open, write.

Sequence 3: mmap, fork, getpid, open, write, close.

Sequence 4: fork, getpid, open, write, close, close.

This process is applied to normal sequences and intrusion sequences from UNM, MIT and ADFA-LD datasets. By tokenizing into a sequence of 6-grams, we increase the amount of data for training as well as testing purposes. In addition, the number of features will decrease when a trace is tokenized into smaller chunks, this will increase the efficiency of training as well as testing performances. [41].

We proceed to clean data by removing any rows or sequences that appear in both normal data and intrusion data. This step draws distinctive characteristics between the 2 classes and effectively boost machine learning performance. A row with normal sequence is labeled 0, whereas the one with intrusion sequence is labeled 1. We use normal data and intrusion data from each dataset to create a sample pool. If it is imbalanced, we use bootstrapping method create a balanced sample of normal sequences and intrusion sequences. Then, we split the sample into training and testing sets in a 70-30 ratio. By training with only signature sequences from both classes, we increase the model accuracy and recall (true positive rate) as well as decrease its false positive rate.

4.1.2 Descriptive analysis of the datasets

In order to compare the difference between the two classes, we graph an overlaid histogram of normal data and intrusion data from each dataset - Figure 2. We use all of UNM datasets (except Sendmail), MIT Live Lpr and ADFA-LD datasets. Each dataset has two histogram versions. The first one displays the dispersion of original traces that have not been cleaned nor processed yet, so it can show the actual difference between normal and intrusion sequences. After cleaning duplicated sequences that exist in both classes, the overlaid histograms of most datasets have changed significantly. Therefore, the second one exhibits a more distinguished difference between the two classes than the original data. The goal is to differentiate normal sequences from intrusion sequences as much as possible so that the algorithms can learn to distinguish them from one another. Therefore, the more distinctive the two classes are, the better the candidate algorithms can perform.

Figure 2a shows that there is a slight difference between normal data and intrusion data from Synthetic Sendmail dataset. However, most of them are overlapped each other. After cleaning duplicating sequences, the dispersion of normal system calls and intrusion system calls have changed in Figure 3a. Normal system calls have expanded from two wide ranges (68 to 83 and 101 to 118) to multiple specific ranges. Particularly, sequences that contain system call numbers ranging between 13 and 15, 64 and 77, 100 and 102, 128 and 134, 150 and 177 are most likely normal. This increases the homogeneity of normal data and reduces false negatives. Additionally, after being processed, intrusion system calls have expanded to more specific ranges. For example, sequences that contain system call numbers from 17 to 27, 77 to 84, 102 to 128 are most likely intrusion. This narrows down possible intrusion likelihoods and reduces false positives in overlapped system calls.

In Synthetic Ftp dataset – Figure 2b, intrusion data are more scattered, and most of them are overlapped with normal data. After being processed – Figure 3b, the difference between the two classed have become more noticeable. In some ranges, number of intrusion system calls are increased while that of normal system calls are decreases, which increases the homogeneity in intrusion data in that area, and vice versa. Originally, most intrusion system calls range between 18 and 148. Comparing to after being processed, that range has been reduced to between 33 and 82, and between 95 and 141. This narrows down possible intrusion likelihoods and reduces false positives in overlapped system calls.

The overlaid histogram of UNM Synthetic Lpr and Live Lpr datasets – Figures 2c and 2d, have very similar layout, where normal sequences and intrusion sequences are mostly overlapped each other on the same system call ranges. Normal sequences tend to contain more system calls ranging from 2 to 19, while intrusion sequences contain more system calls between 19 and 168. However, their histograms of processed data – Figures 3c and 3d, are completely different from each other. Normal data continues dominating system calls between 2 and 19 in Synthetic Lpr, whereas it only dominates system calls between 7 and 18 in Live Lpr. In Synthetic Lpr, intrusion data mostly contains system calls from 19 to 30, 58 to 73, and 87 to 115, while in Live Lpr, it mostly contains system calls from 2 to 6, and 27 to 32. This difference indicates that data with similar system calls distribution do not necessarily turn out to be the same after being processed.

Similarly, MIT Live Lpr dataset – Figure 2e, have very similar layout with UNM Synthetic Lpr and Live Lpr histograms – Figures 2c and 2d, yet its processed data histogram – Figures 3e, is the same from the original one. This is because there is no duplicating rows that exist in both classes. In other words, normal sequences and intrusion sequences are completely different from each other.

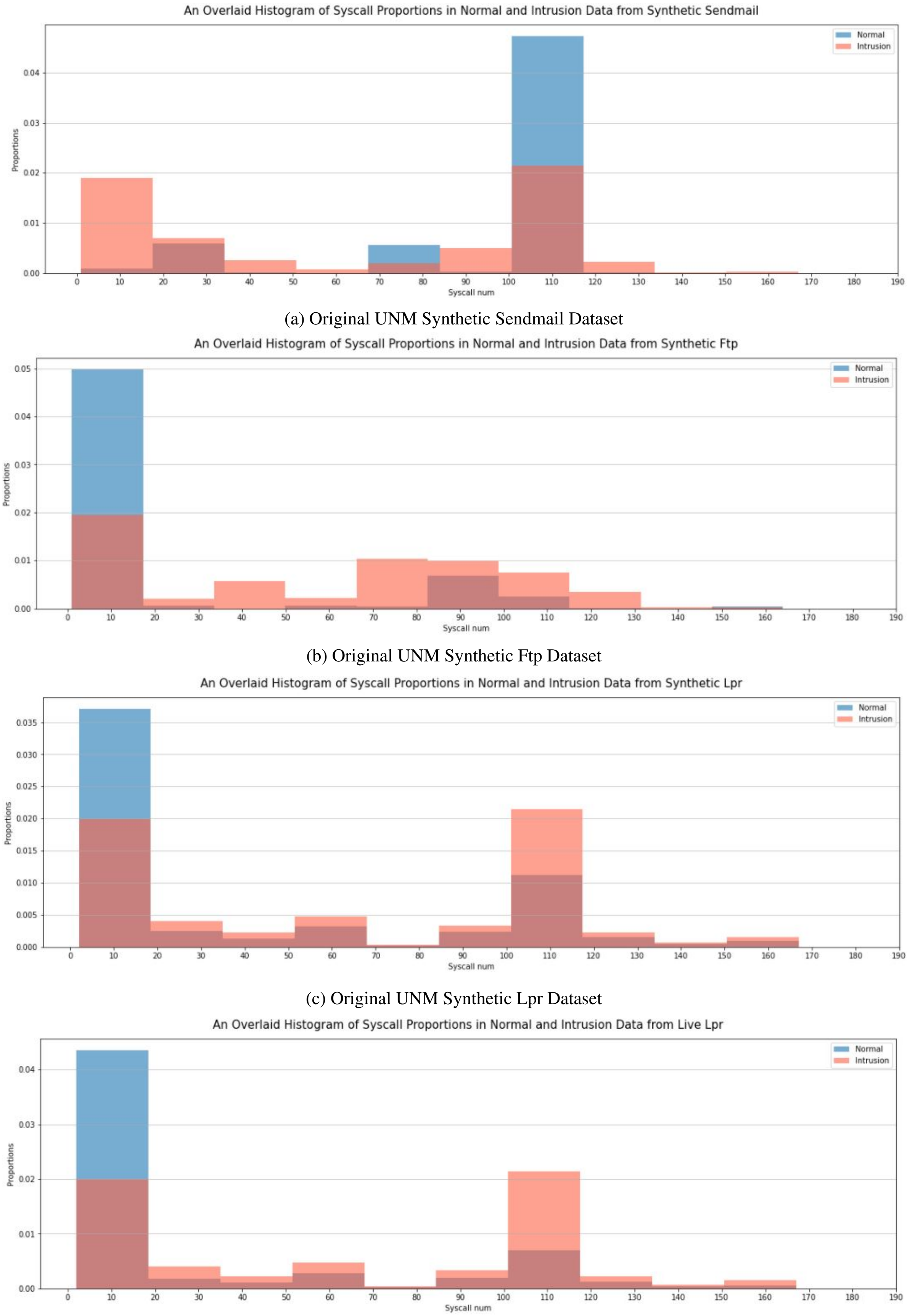
The difference between normal and intrusion data is more visible in the original Xlock dataset – Figure 2f . The two classes mostly overlap between 1 and 14, 69, and between 122 and 131; other than that, they have very distinctive difference. As a result, the processed data histogram – Figure 3f, is not very different from the original histogram. The only change is that their difference is magnified a little bit more in Figure 3f. For example, normal data containing system calls from 54 to 99 have been removed, and normal system calls from 115 to 132 have been increased in Figure 3f. This increases the homogeneity of normal data and reduces false negatives.

Likewise, the difference between normal and intrusion system calls also exists in Live Named dataset – Figure 2g. Normal sequences only dominate from system call numbers 69 to 109 and 122 to 136, whereas intrusion sequences dominate mostly the rest of the system calls. The dispersion of normal system calls has changed in Figure 3g, where most normal data contain system calls from 29 to 43, 87 to 89, and 102 to 114. This transformation also increases the homogeneity of data from each class and reduces false negatives.

The difference between the two classes in Login and Ps dataset is very minimal in Figure 2h, where most of them overlap each other. However, their difference becomes more visible after data is cleaned - Figure 3h. Normal data is more dominant in system call ranges from 15 to 30, 45 to 59, and 100 to 115.

The dispersion of intrusion class in Inetd, and Stide datasets – Figures 2i and 2j, is barely perceptible due to lack of data. As the data is processed, the distribution of intrusion data becomes more clear in both datasets – Figures 3i and 3j. In Inetd, intrusion data dominate system call ranging from 52 to 64, 68 to 83, and 109 to 122, whereas in Stide, it dominates system calls from 0 to 11, 39 to 40, and 51 to 62. Notice that normal data in Inetd – Figure 3i, still have the same shape and proportion. while it changes drastically in Stide – Figure 3j. This is because we did not remove any duplicating normal sequences since all of them exist in intrusion class. If we removed them all, there would be no normal data for training and testing. Therefore, we only removed duplicating intrusion sequences.

On top of that, normal and intrusion data from ADFA-LD dataset – Figure 2k, are very different from each other. Besides some minor overlaps, intrusion sequences mostly have system call numbers from 138 to 170, and 237 to 274. Therefore, its processed histogram – Figure 3k, barely changes from the original one – Figure 2k.



An Overlaid Histogram of Syscall Proportions in Normal and Intrusion Data from Synthetic Sendmail

(a) Original UNM Synthetic Sendmail Dataset

An Overlaid Histogram of Syscall Proportions in Normal and Intrusion Data from Synthetic Ftp

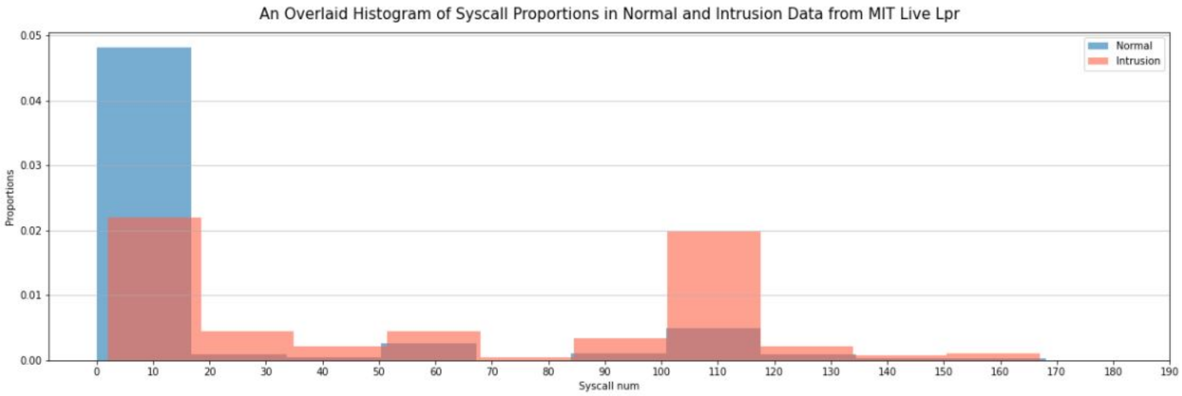
(b) Original UNM Synthetic Ftp Dataset

An Overlaid Histogram of Syscall Proportions in Normal and Intrusion Data from Synthetic Lpr

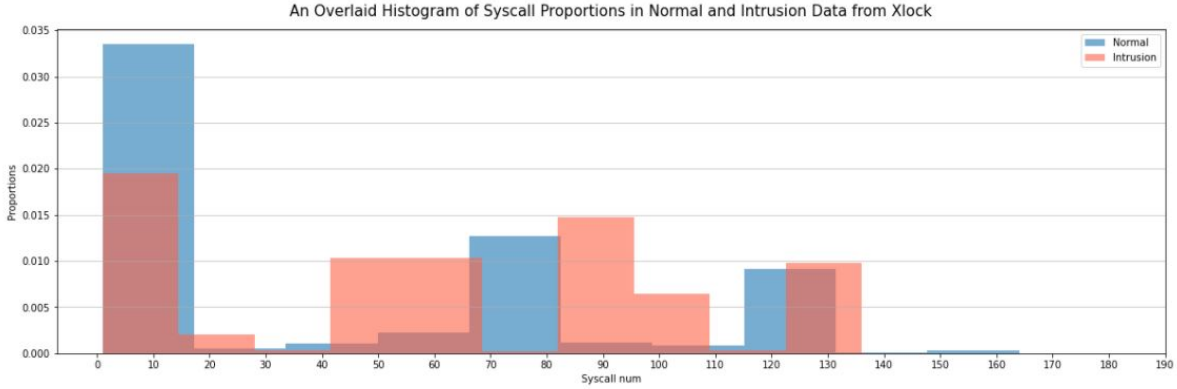
(c) Original UNM Synthetic Lpr Dataset

An Overlaid Histogram of Syscall Proportions in Normal and Intrusion Data from Live Lpr

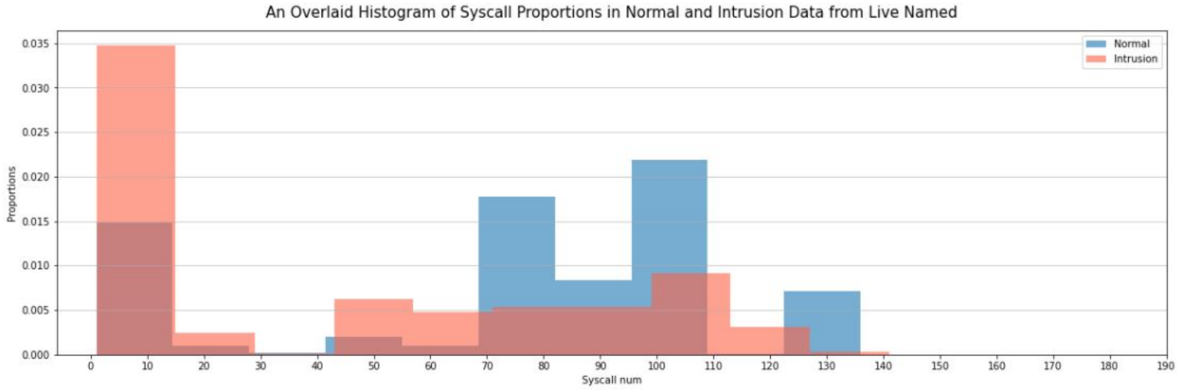
(d) Original UNM Live Lpr Dataset



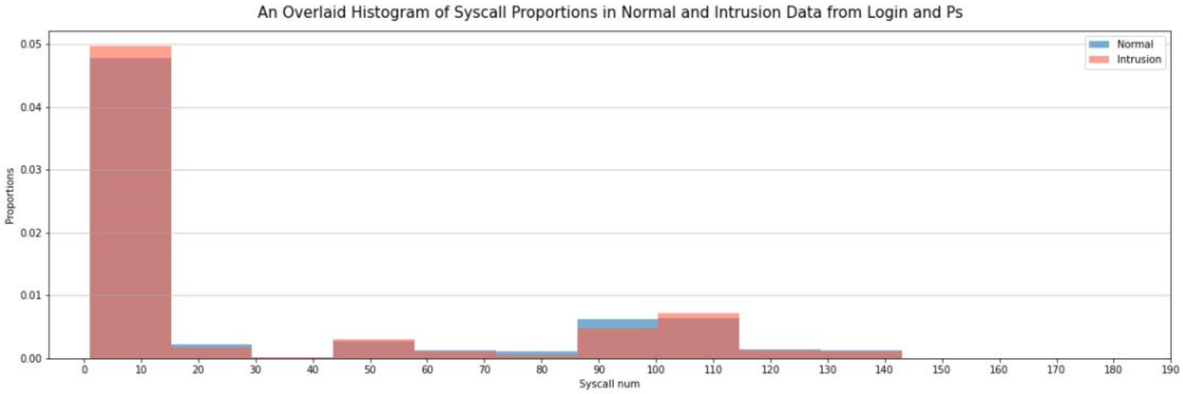
(e) Original MIT Live Lpr Dataset



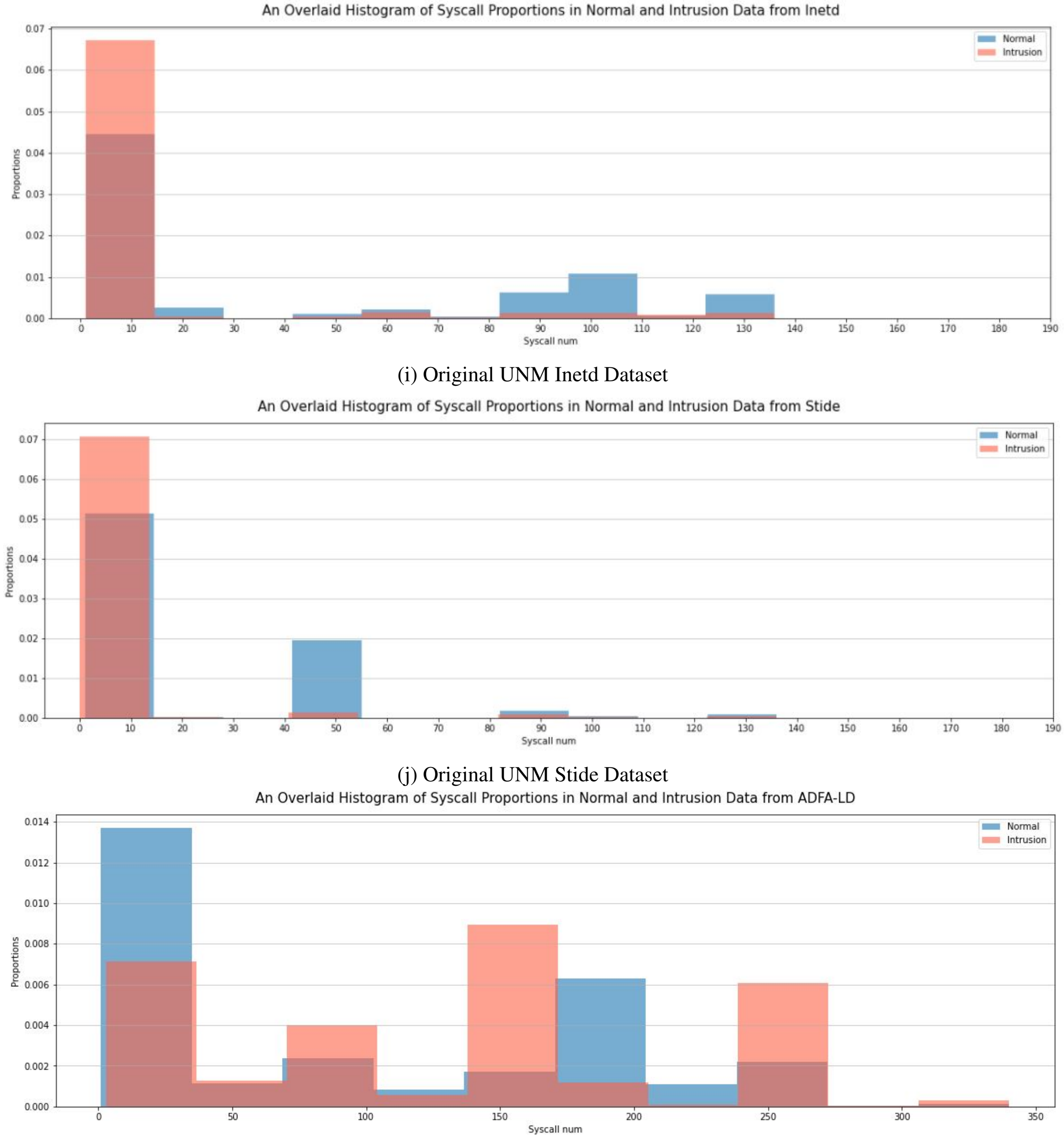
(f) Original UNM Xlock Dataset



(g) Original UNM Live Named Dataset



(h) Original UNM Login and Ps Dataset

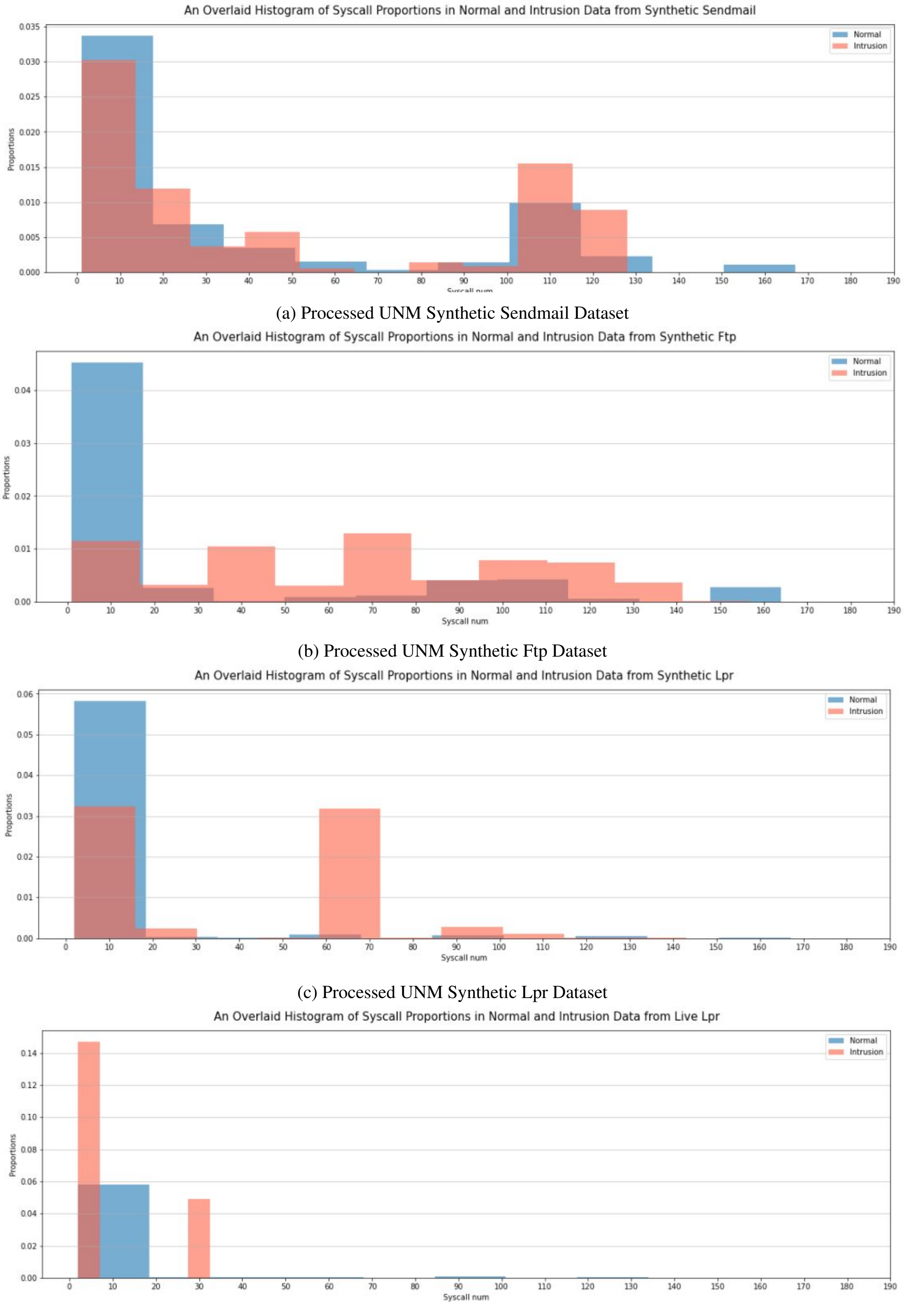


(i) Original UNM Inetd Dataset

(j) Original UNM Stide Dataset

(k) Original ADFA-LD Dataset

Figure 2: Overlaid Histograms of Original System Calls in Normal Data and Intrusion Data From Different Datasets.

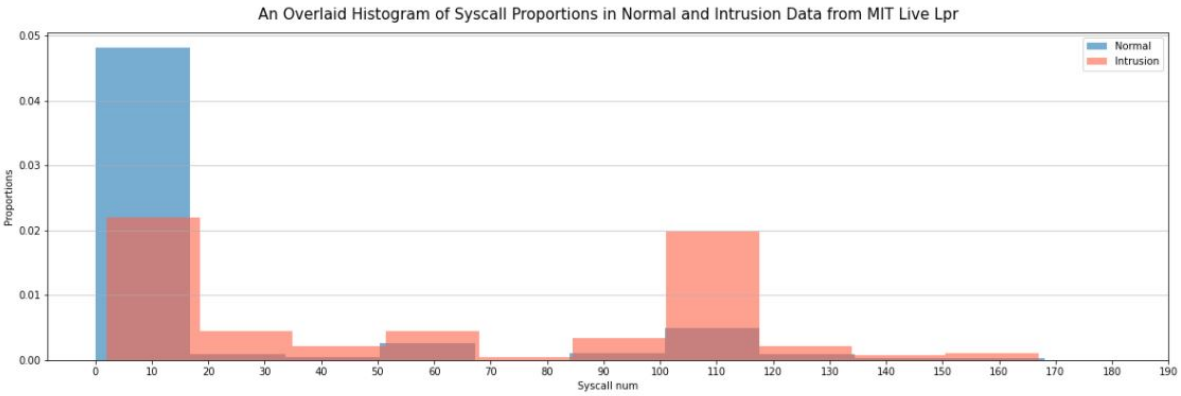


(a) Processed UNM Synthetic Sendmail Dataset

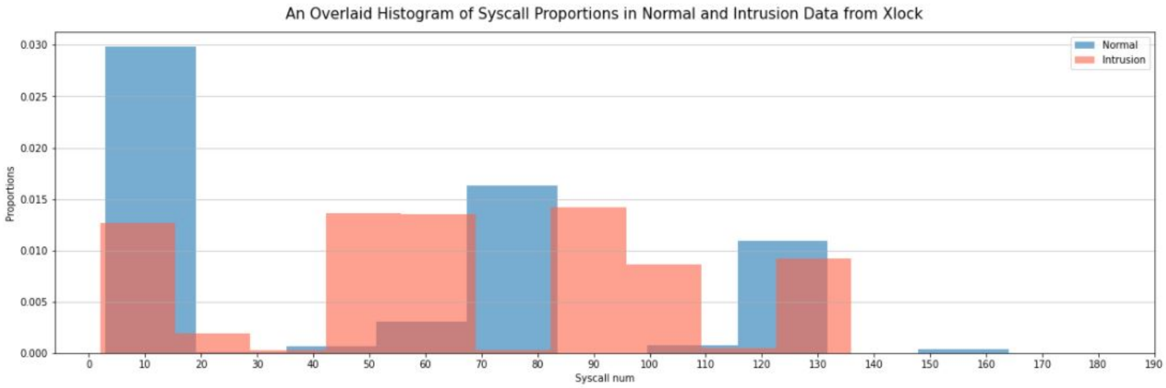
(b) Processed UNM Synthetic Ftp Dataset

(c) Processed UNM Synthetic Lpr Dataset

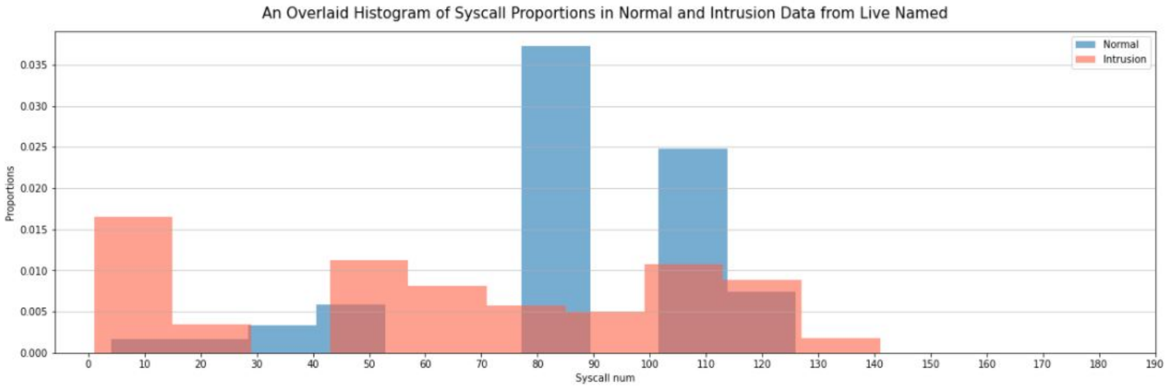
(d) Processed UNM Live Lpr Dataset



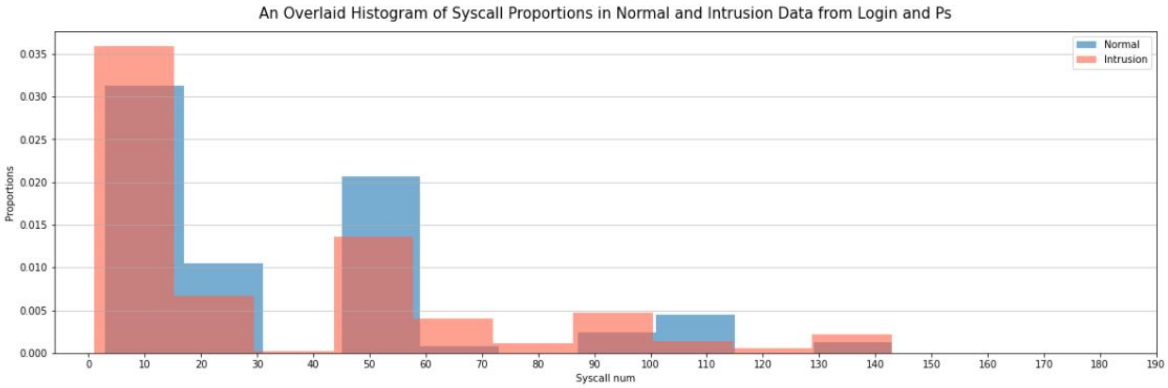
(e) Processed MIT Live Lpr Dataset



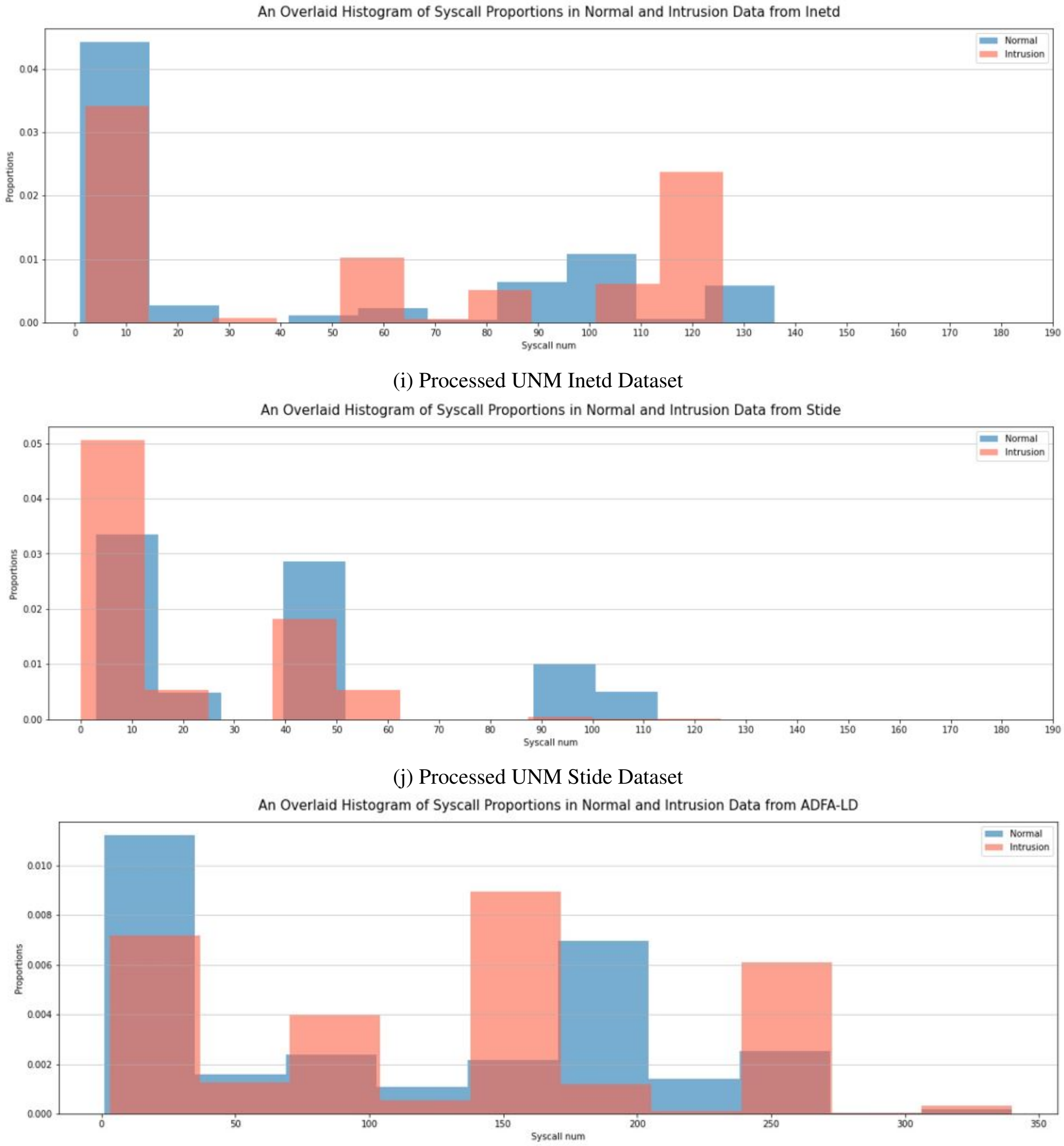
(f) Processed UNM Xlock Dataset



(g) Processed UNM Live Named Dataset



(h) Processed UNM Login and Ps Dataset



(i) Processed UNM Inetd Dataset

(j) Processed UNM Stide Dataset

(k) Processed ADFA-LD Dataset

Figure 3: Overlaid Histograms of System Calls in Normal Data and Intrusion Data From Different Datasets After Cleaning.

4.2 Machine learning algorithms

K-means. K-means is one of the common approaches for anomaly detection, which groups similar characteristics into a cluster. We choose the number of clusters to be 2 because there are two categories in this dataset: normal (0) and intrusion (1). Normal sequences are expected to be clustered together, apart from intrusion cluster. The rest of K-means parameters are set as default from scikit-learn library. For each dataset, we choose roughly 7,000 normal sequences and 7,000 intrusion sequences to split into training and testing sets with a 70-30 ratio. Since K-means is an unsupervised algorithm, it is trained without labels, and tested with the testing set. The predicted labels of the testing set are then compared against the true labels to get the testing accuracy.

Logistic Regression. Logistic Regression (LR) is another potential candidate for detecting anomaly using maximum likelihood. Since this is a supervised model, we use sequences from the training set with labels to train the model. The model is evaluated on the testing set, where the true labels are compared against the predicted ones to measure the model performance. All parameters of this model are set as default in scikit-learn library. Logistic Regression is trained with 9,800 sequences and tested with 4,200 sequences from both classes in each dataset.

Support Vector Machine. Support Vector Machine (SVM) is a supervised machine learning algorithm which is good at detecting anomaly by separating normal behavior from the other using different kernel types (Linear, Polynomial and RBF). We choose Polynomial kernel with 3 degrees and 1.0 regularization to represent a SVM model. Similarly, the model is trained with 9,800 sequences from the training set, and tested with 4,200 sequences from the testing set of each dataset. The testing result is used to evaluate the model performance on each dataset.

Neural Network. Neural Network (NN) is efficient at learning underlying complex relationships because of its composite architecture. The model is comprised of 3 layers: an input layer of 6 features, where each of them represents a system call from a sequence of 6; a ReLU layer with 6 hidden nodes containing extracted information from each system call; and, a Sigmoid output layer with 2 output nodes, where each one represents a probability of a sequence belonging to either class. Whichever class has higher chance in the output node will be the predicted label of the given input sequence. Likewise, we train and test the model with 9,800 sequences and 4,200 sequences respectively from each dataset.

Decision Tree. Decision Tree (DT) relies on a set of rules, derived from the training process, to partition data into groups that are as homogeneous as possible. The goal is to generate a Decision Tree to classify normal sequences from intrusion sequences at the lowest possible error rate. That way, it can be generalized on the testing set as well as future unseen data. For ever split, the model considers at most a square root number of features and decides where to split using GINI criterion with a minimum of ten samples per split and five samples per leaf. Each leaf node contains sequences, where most of them belonging to the same class. Likewise, we train and test the model with 9,800 sequences and 4,200 sequences respectively from each dataset.

Random Forest. Random Forest (RF) distinguishes normal sequences from intrusion sequences using ensemble learning method. RF creates multiple Decision Trees to classify the same observation. The final decision is based on the wisdom of the crowd that is the majority predicted class of that particular observation. Similarly, this model has the same parameters as the Decision Tree model above. The only difference is that this model allows bootstrap samples to build multiple trees. We train and test the model with 9,800 sequences and 4,200 sequences respectively from each dataset.

K Nearest Neighbor. K Nearest Neighbor (KNN) classifies an observation by counting the majority classes of the nearest neighbors or sequences. This model enhances its performance by minimizing intra-variability within a group while maximizing inter-variability between different groups. For simplicity, we choose the number of neighbors to be 3. Still, any odd number of k is recommended to avoid a tie situation. Additionally, the weights parameter is set to be uniform, where all observations in each neighborhood are weighted equally regardless of their distances. The model is trained and tested with 9,800 sequences and 4,200 sequences respectively from each dataset.

Naïve Bayes. Naïve Bayes (NB) is known for classifying data based on conditional probabilities without making any assumption. Given lots of labeled sequences, we hope that this model can efficiently distinguish intrusion sequences from normal sequences. A predicted label is determined based on the highest probability of a class. The parameters of this model are set as default from scikit-learn library. NB is trained and tested using 9,800 sequences and 4,200 sequences respectively from each dataset.

BERT. Bidirectional Encoder Representations from Transformers (BERT), developed by the google AI language, using transformer architecture with attention mechanisms for learning context. BERT is a bidirectional unsupervised language model, which takes the texts before and after the token into contextual account. Therefore, BERT is good at sentimental analysis, language modeling, name entity recognition, summarizing, question-answering, and translation. There are two versions of BERT models based on structure size: BERT Base with 110 million parameters and BERT Large with 345 million parameters [64]. In this research, the processed string in section 4.1.1 is passed to tokenizer and each string of system call sequence is tokenized by the BERT's tokenize (bert-base-uncased). Each sequence is padded with [CLS] and [SEP] at beginning and end or the sequence. The output of the BERT model is then passed to a linear network with 768 neurons and output of the linear layer is passed to a single neuron with SoftMax activation. The model is trained and tuned, using normal and intrusion sequences from each dataset, with 16 epochs and 16 batch size.

GPT-2. Generative Pretrained Transformers (GPT) is developed by Microsoft's Open AI, using the decoder part of the transformer architecture [65]. There are four types of GPT-2 depending on the model structure size, which can be Small – 117M parameters, Medium – 345M parameters, Large – 762M parameters and Extra Large – 1542M parameters. GPT-2 is autoregressive which reuses its own output as an input sequence. Hence, this model does a great job in text generation at any given length and even more advanced tasks such as answering questions, summarizing paragraphs and translating text from one language to another. In this experiment, we use Small GPT-2 to train on processed data from both classes of each dataset using 16 epochs and 16 batch size. Unlike BERT, GPT-2 does not require additional output layer; instead, it outputs the sequence's likelihood in each class.

4.3 Experiment results

4.3.1 Overall results

Table 3 shows the performance of different machine learning algorithms on 11 datasets from UNM, MIT and ADFA-LD regrading performance indicator accuracy, recall, precision, macro-f1 score, false positive rate (meaning the percentage of malicious behaviours that are labelled as benign behaviours), and AUC score [31]. Our goal is to find the best candidate model with high accuracy, high recall but low FPR. High performance measures are bolded so that we can easily identify the best candidate model. We make the following observations:

- Almost all of the algorithms achieved a better performance on Synthetic Lpr, Live Lpr, Xlock, Live Named, Inetd and Stide datasets than the others, indicating that the data quality of the former group of datasets might be higher than the latter group.
- Decision Tree, Random Forest, KNN, BERT and GPT-2 are the best candidate algorithms since they achieved higher accuracy, recall and precision, yet at a lower false positive rate on all datasets.
- BERT and GPT-2 are the best algorithm for HIDS on all of the datasets since it outperformed other algorithms in terms of recall and false positive rate, which are one of the most important metrics in anomaly detection.
- BERT and GPT-2 performances are excellent on all of the datasets. Their recalls are above 0.90, and FPR are below 0.06.
- Class imbalance is not a major issue for the HIDS datasets used in this paper since we use bootstrapping technique to generate a balanced population of both classes. This population is then split into training and testing sets with a 70-30 ratio to train and test the models.

Table 3: Model performance regarding accuracy, recall, precision, macro-F1, false positive rate (FPR), and AUC score on different HIDS datasets.

Dataset	Model	Accuracy	Recall	Precision	Macro-F1	FPR	AUC score
Synthetic Sendmail	K-means	0.63	0.61	0.64	0.63	0.36	0.63
	Logistic Regression	0.63	0.57	0.62	0.61	0.35	0.63
	SVM	0.73	0.57	0.85	0.73	0.10	0.73
	Neural Network	0.66	0.63	0.68	0.67	0.29	0.66
	Decision Tree	**0.98**	**1.00**	**0.97**	**0.98**	**0.03**	**0.98**
	Random Forest	**0.99**	**1.00**	**0.98**	**0.99**	**0.02**	**0.99**
	KNN	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
	Naïve Bayes	0.63	0.57	0.61	0.61	0.36	0.63
	BERT	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
	GPT-2 Network	**1.00**	**1.00**	**0.99**	**1.00**	**0.01**	**1.00**

Synthetic Ftp	K-means	0.20	0.07	0.10	0.20	0.66	0.20
	Logistic Regression	0.74	0.77	0.74	0.74	0.28	0.74
	SVM	0.79	0.67	**0.90**	0.79	**0.08**	0.79
	Neural Network	0.80	**0.89**	0.76	0.80	0.30	0.80
	Decision Tree	**0.99**	**1.00**	**0.99**	**0.99**	**0.02**	**0.99**
	Random Forest	**0.99**	**1.00**	**0.99**	**0.99**	**0.01**	**0.99**
	KNN	**0.99**	**1.00**	**0.99**	**0.99**	**0.01**	**0.99**
	Naïve Bayes	0.77	0.82	0.75	0.77	0.28	0.77
	BERT	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
	GPT-2 Network	**0.99**	**0.99**	**1.00**	**0.99**	**0.00**	**0.99**
Synthetic Lpr	K-means	0.55	0.10	**0.93**	0.54	**0.01**	0.54
	Logistic Regression	**0.97**	**0.99**	**0.95**	**0.97**	**0.05**	**0.97**
	SVM	**0.99**	**0.99**	**0.99**	**0.99**	**0.01**	**0.99**
	Neural Network	**0.97**	**0.99**	**0.95**	**0.97**	**0.05**	**0.97**
	Decision Tree	**0.99**	**1.00**	**0.98**	**0.99**	**0.02**	**0.99**
	Random Forest	**1.00**	**0.99**	**1.00**	**1.00**	**0.00**	**1.00**
	KNN	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
	Naïve Bayes	**0.96**	**1.00**	**0.93**	**0.96**	**0.07**	**0.96**
	BERT	**1.00**	**1.00**	**0.99**	**1.00**	**0.01**	**1.00**
	GPT-2 Network	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
Live Lpr	K-means	0.86	0.76	**0.96**	0.86	**0.03**	0.86
	Logistic Regression	**0.98**	**1.00**	**0.97**	**0.98**	**0.03**	**0.98**
	SVM	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
	Neural Network	**0.98**	**1.00**	**0.95**	**0.98**	**0.05**	**0.98**
	Decision Tree	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
	Random Forest	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
	KNN	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
	Naïve Bayes	**0.96**	**1.00**	**0.93**	**0.96**	**0.07**	**0.96**
	BERT	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
	GPT-2 Network	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
MIT Live Lpr	K-means	0.70	0.54	0.79	0.70	0.15	0.70
	Logistic Regression	0.74	0.67	0.78	0.74	0.18	0.74
	SVM	0.76	0.69	0.79	0.75	0.17	0.75
	Neural Network	0.75	0.70	0.77	0.75	0.20	0.75
	Decision Tree	**1.00**	**1.00**	**0.99**	**1.00**	**0.01**	**1.00**
	Random Forest	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
	KNN	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
	Naïve Bayes	0.73	0.66	0.78	0.73	0.19	0.73
	BERT	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
	GPT-2 Network	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
Xlock	K-means	0.37	0.31	0.36	0.37	0.56	0.37
	Logistic Regression	0.77	0.67	0.84	0.77	0.13	0.77
	SVM	0.84	**0.97**	0.77	0.84	0.29	0.84
	Neural Network	0.79	0.74	0.82	0.79	0.17	0.79
	Decision Tree	**0.99**	**1.00**	**0.97**	**0.99**	**0.03**	**0.99**
	Random Forest	**0.99**	**1.00**	**0.98**	**0.99**	**0.02**	**0.99**
	KNN	**0.99**	**1.00**	**0.99**	**0.99**	**0.01**	**0.99**
	Naïve Bayes	0.68	0.69	0.68	0.68	0.32	0.68
	BERT	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
	GPT-2 Network	**0.99**	**1.00**	**0.99**	**0.99**	**0.01**	**0.99**
Live Named	K-means	0.21	0.38	0.29	0.21	0.96	0.21
	Logistic Regression	0.83	0.75	**0.90**	0.83	**0.08**	0.83
	SVM	**0.92**	**0.87**	**0.99**	**0.92**	**0.01**	**0.92**
	Neural Network	0.84	0.72	**0.95**	0.84	**0.04**	0.84
	Decision Tree	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
	Random Forest	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
	KNN	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
	Naïve Bayes	0.87	0.81	**0.92**	**0.87**	**0.07**	**0.87**

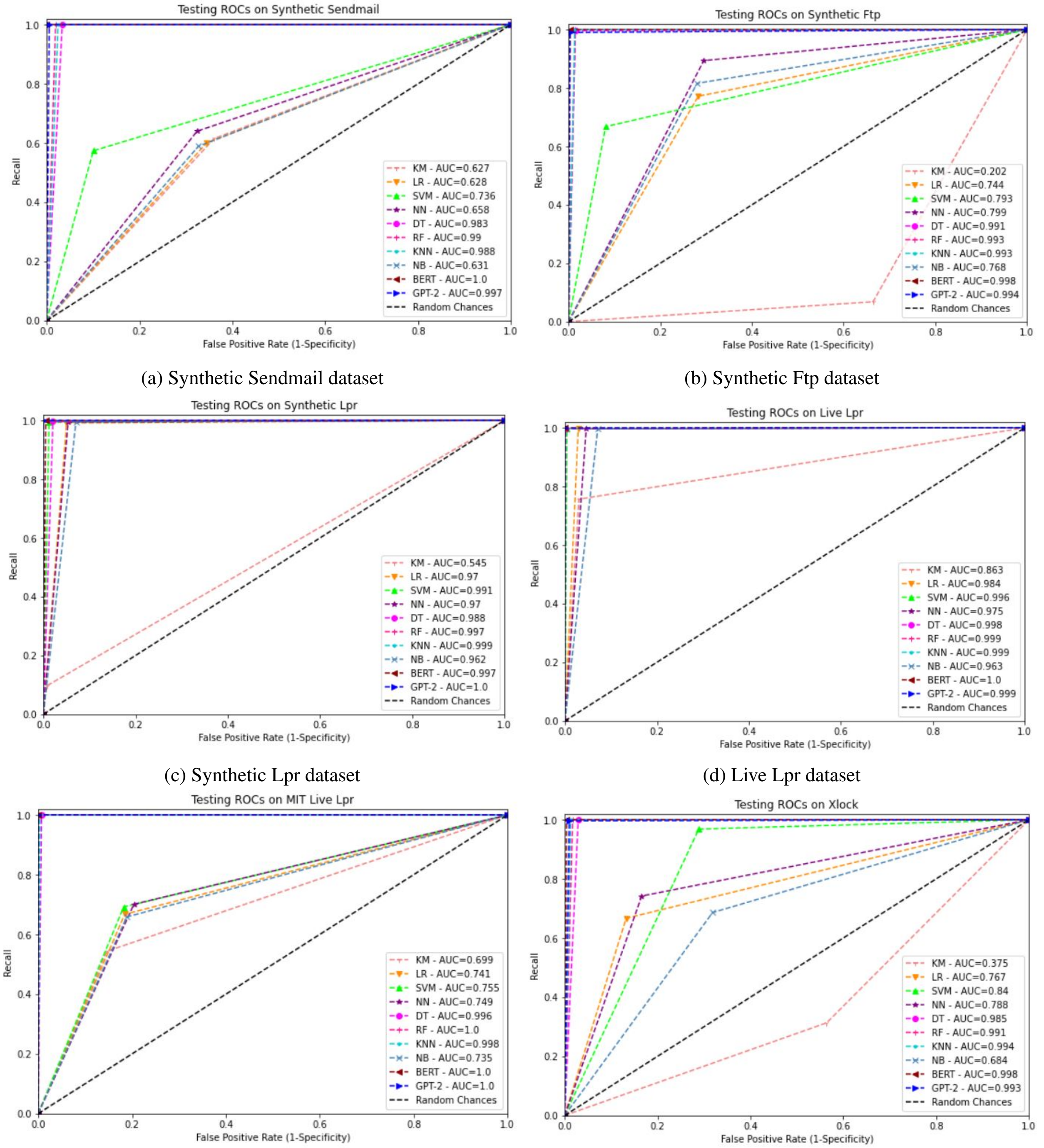
	BERT	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
	GPT-2 Network	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
Login and Ps	K-means	0.52	0.64	0.53	0.51	0.61	0.51
	Logistic Regression	0.68	0.55	0.77	0.68	0.18	0.68
	SVM	**0.92**	**0.94**	**0.92**	**0.92**	**0.09**	**0.92**
	Neural Network	0.71	0.60	0.78	0.71	0.18	0.71
	Decision Tree	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
	Random Forest	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
	KNN	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
	Naïve Bayes	0.63	0.55	0.67	0.63	0.29	0.63
	BERT	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
	GPT-2 Network	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
Inetd	K-means	0.68	0.82	0.64	0.68	0.46	0.68
	Logistic Regression	0.70	0.75	0.69	0.70	0.33	0.70
	SVM	**0.96**	**0.95**	**0.97**	**0.96**	**0.03**	**0.96**
	Neural Network	0.86	**0.95**	0.80	0.86	0.24	0.86
	Decision Tree	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
	Random Forest	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
	KNN	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
	Naïve Bayes	0.71	0.81	0.68	0.71	0.39	0.71
	BERT	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
	GPT-2 Network	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
Stide	K-means	0.54	0.66	0.53	0.54	0.59	0.54
	Logistic Regression	0.80	0.76	0.82	0.80	0.17	0.80
	SVM	**0.91**	**0.99**	**0.85**	**0.91**	0.18	**0.91**
	Neural Network	0.80	0.77	0.82	0.80	0.17	0.80
	Decision Tree	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
	Random Forest	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
	KNN	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
	Naïve Bayes	0.76	0.80	0.74	0.76	0.29	0.76
	BERT	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
	GPT-2 Network	**1.00**	**1.00**	**1.00**	**1.00**	**0.00**	**1.00**
ADFA-LD	K-means	0.61	0.62	0.60	0.61	0.39	0.61
	Logistic Regression	0.61	0.62	0.59	0.61	0.41	0.61
	SVM	0.65	0.53	0.69	0.65	0.23	0.65
	Neural Network	0.60	0.70	0.58	0.60	0.49	0.60
	Decision Tree	**0.85**	**0.84**	**0.85**	**0.85**	**0.14**	**0.85**
	Random Forest	**0.88**	**0.87**	**0.89**	**0.88**	**0.11**	**0.88**
	KNN	**0.80**	**0.70**	**0.88**	**0.80**	**0.09**	**0.80**
	Naïve Bayes	0.61	0.61	0.60	0.61	0.38	0.61
	BERT	**0.94**	**0.92**	**0.95**	**0.94**	**0.05**	**0.94**
	GPT-2 Network	**0.93**	**0.92**	**0.94**	**0.93**	**0.06**	**0.93**

4.3.2 ROC Curve

Figures 3 shows the testing ROC curves of all candidate algorithms on 11 datasets separately. Overall, Random Forest, Decision Tree, KNN, BERT and GPT-2 generate the highest results and outperform the others on all datasets. Their average AUC scores are 0.986 (RF), 0.982 (DT) and 0.980 (KNN), 0.995 (BERT) and 0.993 (GPT-2) demonstrating near perfect performances. These five algorithms can truly learn and effectively distinguish between normal sequences and intrusion sequences. Contrarily, most of the other candidates only perform well on certain datasets such as Synthetic Ftp, Synthetic Lpr, Live Lpr, Xlock, Live Named, Inetd and Stide datasets. Besides the five best candidates, there is no other algorithm perform well when trained and tested on Synthetic Sendmail, MIT Live Lpr and ADFA-LD datasets.

When using Synthetic Ftp dataset, in addition to the best candidates, SVM, Neural Network and Naïve Bayes perform well with AUC scores close to 0.80. On Synthetic Lpr dataset, Logistic Regression, SVM, Neural Network and Naïve Bayes perform very well besides the best candidates, and their AUC scores are above 0.96. On Live Lpr dataset, all candidates perform extremely well with AUC scores above 0.96, except K-means, whose AUC score is 0.86. On Xlock dataset, only Logistic Regression, SVM and Neural Network have AUC scores above 0.77, other than the best candidates. Most candidates perform well on Live Named dataset, where their AUC scores are above 0.84, except

K-means, whose AUC score is only 0.21. In addition to the best candidates, only SVM performs very well with an AUC score of 0.925 on Login and Ps dataset. The rest of the candidates, whose AUC scores are between 0.56 and 0.73, perform slightly better than random chance on this dataset. On top of the best candidates, only SVM and Neural Network perform well on Inetd dataset with AUC scores above 0.86. Most candidates perform well on Stide dataset with AUC scores above 0.78, except K-means whose AUC score is only 0.54. The candidates' performance on ADFA-LD dataset is not as high as on the other datasets. Only the best candidates perform well with AUC scores above 0.80. Contrarily, the other candidates perform slightly better than random chance with AUC scores around 0.65. Next, we will use the log ratio of recall over false positive rate to determine the best and worst intrusion detection algorithms in section 4.3.3.



(a) Synthetic Sendmail dataset

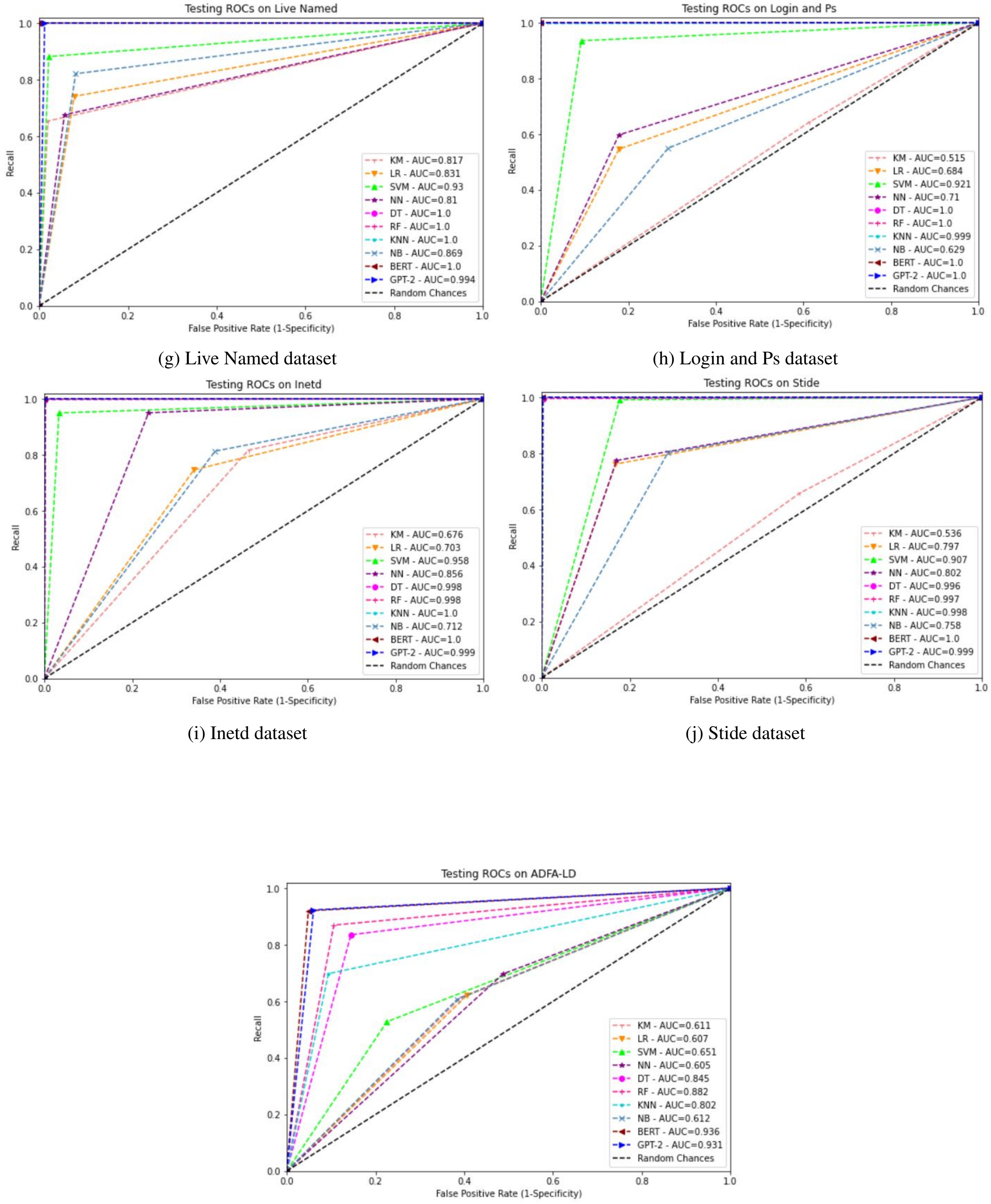
(b) Synthetic Ftp dataset

(c) Synthetic Lpr dataset

(d) Live Lpr dataset

(e) MIT Live Lpr dataset

(f) Xlock dataset



(g) Live Named dataset

(h) Login and Ps dataset

(i) Inetd dataset

(j) Stide dataset

(k) ADFA-LD dataset

Figure 4: ROC Curves on all the 11 datasets (test data) regarding different machine learning algorithms.

4.3.3 Ratio of Recall over False Positive Rate

To find out the best algorithm among Decision Tree, Random Forest, KNN, BERT and GPT-2, we take the average of log ratios between recall (true positive rate) and false positive rate from Table 3 and demonstrate it in Figure 5. The highest bar shows the best performing model. Therefore, BERT is the best model with the highest average of log ratio between recall and false positive, which is 2.75. This indicates that BERT yields the highest true positive (recall) at very low false positive rate, which is our primary goal in an intrusion detection system. BERT achieves 0.00 false positive rate on 9 out of 11 datasets (except Synthetic Lpr and ADFA-LD). The model's false positive on Xlock data is 0.01 and on ADFA-LD is 0.05. BERT also achieves the highest recalls on all datasets. Additionally, GPT-2 is the second-best model with the second highest average of log ratio, which is 2.74. GPT-2 achieves the highest recalls on 10 out of 11 datasets. Furthermore, KNN is the third-best model with 2.72 average of log ratio. This is because KNN has higher FPR than Decision Tree and Random Forest on Synthetic Ftp, Xlock and ADFA-LD datasets. On the other hand, K-means is the lowest performing model with the average log ratio of 0.19. Naïve Bayes is the second-lowest performing model, whose average log ratio is 0.56. This indicates that these models cannot distinguish between normal sequences and intrusion sequences. Since low false positive rate is more important than recall and accuracy, we conclude that BERT is the best candidate in detecting intrusions.

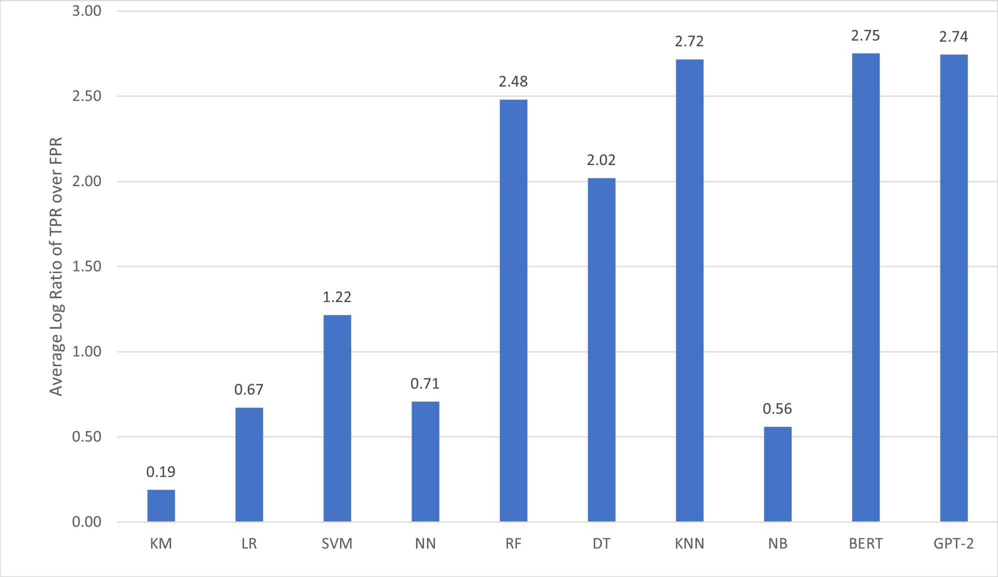


Figure 5: Log ratio of recall over false positive rate using different machine learning models

4.4 Discussion

To compare the overall performance on different datasets, we have created a clustered bar chart of average recalls (True Positive Rate - TPR) and average FPR - Figure 6. This figure is plotted using the performance metrics of processed data and of original data, and it is sorted decreasingly by the average FPR. The blue bars are the average recalls of original data, while the orange bars are the average recalls of processed data. The grey bars are the average FPR of original data, whereas the yellow bars are the average of the processed data. By visualizing the difference in performances among the datasets, we can identify the best dataset as well as the worst dataset. After being processed, Live Lpr dataset yields the highest TPR and the lowest FPR. Likewise, Synthetic Lpr yields the second highest TPR and the second lowest FPR. Therefore, Live Lpr is the best, and Synthetic Lpr is the second-best datasets for HIDS.

Besides that, Figure 6 is also helpful in delineating the difference in performances before and after a dataset is processed. Based on this figure, the best-performing processed datasets with the lowest average FPR are Live Lpr, Synthetic Lpr, MIT Lpr and Live Named in a decreasing order. This indicates that most candidate algorithms yield very low FPR on these datasets. This is also confirmed by the performance metrics from Table 3. The magnitude of the grey bars indicates that these datasets does not yield such low FPR before being processed. On average, the original Live Lpr dataset has lower quality, and therefore, yields a FPR 20 times higher than the processed one. Similarly, the original Synthetic Lpr yields a FPR 27.5 times higher than the processed one, the original Live Named yields a FR 5 times higher, and the original MIT Lpr yields a FPR 1.8 times higher. Furthermore, these datasets yield higher recall after being processed. On average, after being processed, Live Lpr yields recall 1.22 times higher, Synthetic Lpr yields recall almost 1.5 times higher. Processed MIT Lpr only increases its average recall by 0.006 because there is no duplication in this dataset. Therefore, the data cleaning process has no effect on this data; hence, its performance was not increased. Although processed Live Named has lower recall than the original data, its FPR is significantly decreased from 0.644 to 0.116. Therefore, it is safe to say that effective data cleaning has lowered FPR and increased TPR and therefore, improved the models' performance on these datasets. On the other hand, ADFA-LD dataset has the lowest average recall and the highest FPR. Therefore, it is the worst dataset for HIDS. Based on these observations, we can derive the best data characteristics for HIDS in the next section.

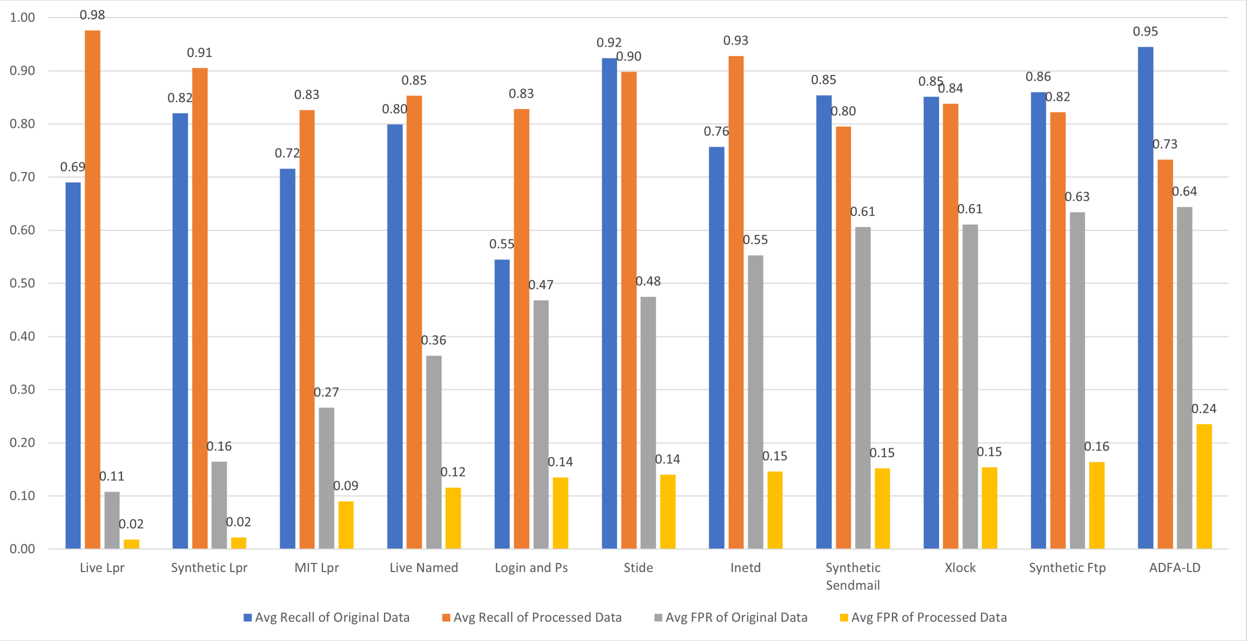


Figure 6: A Clustered Bar Chart of Average Recalls and Average FPR from Unprocessed and Processed Data

5 Data quality assessment and assurance

Our analysis in Sections 4.3 and 4.4 has revealed that KNN, Random Forest and Decision Tree are the best performing candidate models, and that most models perform best on Live Lpr and Synthetic Lpr datasets. On the other hand, they perform poorly on ADFA-LD dataset. In this section, we conduct data quality assessment and assurance to determine the best characteristics that a HIDS dataset should possess in order to yield the best possible result.

Table 4: Data quality evaluation of the datasets used in the experiment regarding the data quality dimensions discussed in Section 3.3. A: Reputation; B: Relevance; C: Comprehensiveness; D: Timeliness; E: Variety; F: Accuracy; G: Consistency; H: Duplication. Performance represents the Average of Log Ratioof TPR over FPR per dataset.

Dataset	Data quality evaluation	Performance
Synthetic Sendmail	**A:** Collected from sendmail program using strace on Sun SPARC stations running unpatched SunOS 4.1.1 and 4.1.4. **B:** To monitor normal usage and detect sunsendmailcp intrusion, decode intrusion and forwarding loops error. **C:** Both original data (1.8 million normal vs. 6755 intrusion system calls) and processed data (7,759 normal vs. 451 intrusion sequences) are imbalanced. **D:** Collected in 1996. **E:** The signature of both classes covers system call numbers from 1 to 168. **F:** Yes, data is correctly labeled. Normal data and intrusion data are stored in separate folders. **G:** Yes, the data is consistent. Features are extracted sequentially by PIDs. **H:** There is 99.5% of overlaps between normal sequences and intrusion sequences.	0.97

Synthetic Ftp	**A:** Collected from Washington University ftpd server using strace on a Linux machine. **B:** To monitor normal usage and detect misconfiguration vulnerability. **C:** Both original data (180,315 normal vs. 1,363 intrusion system calls) and processed data (28,415 normal vs. 376 intrusion sequences) are extremely imbalanced. **D:** Collected in 1998. **E:** The signature of both classes covers system call numbers from 1 to 164. **F:** Yes, data is correctly labeled. Normal data and intrusion data are stored in separate folders. **G:** Yes, the data is consistent. Features are extracted sequentially by PIDs. **H:** There is 84.15% of overlaps between normal sequences and intrusion sequences.	1.23
Synthetic Lpr	**A:** Collected from lpr program using strace on Sun SPARC stations running unpatched SunOS 4.1.4. **B:** To monitor normal usage data and detect lprcp intrusion signature. **C:** Original data is extremely imbalanced (2,400 normal vs. 164,232 intrusion system calls). Processed data is imbalanced (975 normal vs. 2,232 intrusion sequences). **D:** Collected in 1991. **E:** The signature of both classes covers system call numbers from 2 to 168. **F:** Yes, data is correctly labeled. Normal data and intrusion data are stored in separate folders. **G:** Yes, the data is consistent. Features are extracted sequentially by PIDs. **H:** There is 98.08% of overlaps between normal sequences and intrusion sequences.	1.82
Live Lpr	**A:** Live data were collected over 3 months from a SunOS 4.1.4 machine at UNM. **B:** To monitor normal usage data and lprcp intrusion signature from the same MIT Lpr scripted attack. **C:** Both original data (187,102 normal vs. 164,232 intrusion system calls) and processed data (108,700 normal vs. 4,000 intrusion sequences) are balanced. **D:** Collected in 1996. **E:** The signature of both classes covers system call numbers from 1 to 168. **F:** Yes, data is correctly labeled. Normal data and intrusion data are stored in separate folders. **G:** Yes, the data is consistent. Features are extracted sequentially by PIDs. **H:** There is 67.92% of overlaps between normal sequences and intrusion sequences.	2.08
MIT Live Lpr	**A:** Live data were collected over 2 weeks from 77 hosts on SunOS 4.1.4 machines at the MIT lab. **B:** To monitor normal usage data and lprcp intrusion signatures from scripted attacks. **C:** Original data is balanced (174,260 normal vs. 165,248 intrusion system calls). **D:** Collected in 1997. **E:** The signature of both classes covers system call numbers from 0 to 169. **F:** Yes, data is correctly labeled. Normal data and intrusion data are stored in separate folders. **G:** Yes, the data is consistent. Features are extracted sequentially by PIDs. **H:** No duplication.	1.22
Xlock	**A:** Both live and synthetic data from Xlock were collected on a Linux machine over 2 days. **B:** To monitor normal usage of xlock command and detect a buffer overflow exploit signature. **C:** Original data is extremely imbalanced (339,177 normal vs. 949 intrusion system calls). Therefore, we only use 25,000 normal system calls. Processed data is imbalanced (19,487 normal vs. 635 intrusion sequences). **D:** Collected in 1997. **E:** The signature of both classes covers system call numbers from 1 to 164. **F:** Yes, data is correctly labeled. Normal data and intrusion data are stored in separate folders. **G:** Yes, the data is consistent. Features are extracted sequentially by PIDs. **H:** There is 23.09% of overlaps between normal sequences and intrusion sequences.	0.94

Live Named	**A:** Live data was collected over a month from Named program on a UNM Linux 2.0.35 kernel. **B:** To monitor normal usage data and detect a buffer overflow exploit. **C:** Original data is extremely imbalanced (9.2 million normal vs. 1,800 intrusion system calls). Therefore, we only use 2,000 normal system calls to create a more balanced dataset. Processed data is imbalanced (99 normal vs. 273 intrusion sequences). **D:** Colected in 1998. **E:** The signature of both classes covers system call numbers from 1 to 141. **F:** Yes, data is correctly labeled. Normal data and intrusion data are stored in separate folders. **G:** Yes, the data is consistent. Features are extracted sequentially by PIDs. **H:** There is 90.18% of overlaps between normal sequences and intrusion sequences.	1.96
Login and ps	**A:** Both live and synthetic data were collected on a 2.0.35 Linux kernel over a month. The Login version is from Red Hat util-linux-2.5.38. The Ps verson is from Red Hat procps v.1.01. **B:** To monitor normal usage data and detect Trojan intrusion signature. **C:** Original data is balanced (15,050 normal vs. 11,825 intrusion system calls). Processed data is imbalanced (176 normal vs. 714 intrusion sequences). **D:** Collected in 1998. **E:** The signature of both classes covers system call numbers from 1 to 142. **F:** Yes, data is correctly labeled. Normal data and intrusion data are stored in separate folders. **G:** Yes, the data is consistent. Features are extracted sequentially by PIDs. **H:** There is 96.69% of overlaps between normal sequences and intrusion sequences.	1.44
Inetd	**A:** Live data was collected from Inetd program on a Linux 2.0.35 kernel at UNM. **B:** To monitor normal usage data and detect the signature of a DoS attack which ties up all network connection resources. **C:** Original data is imbalanced (541 normal vs. 8,371 intrusion system calls). Processed data is balanced (536 normal vs. 487 intrusion sequences). **D:** Collected in 1999. **E:** The signature of both classes covers system call numbers from 1 to 137. **F:** Yes, data is correctly labeled. Normal data and intrusion data are stored in separate folders. **G:** Yes, the data is consistent. Features are extracted sequentially by PIDs. **H:** There is 94.53% of overlaps between 2 classes, and all normal data appear in intrusion data. Therefore, we only remove the overlapped intrusion sequences.	1.26
Stide	**A:** Live Stide data was collected from a modified Linux 2.0.35 kernel at UNM. **B:** To monitor normal usage data and detect the signature of a DoS attack which affects requesting memory from other programs. **C:** Original data is extremely unbalanced (15.6 million normal vs. 206,000 intrusion system calls). Therefore, we only used 1.1 million normal system calls. Processed data is imbalanced (17,182 normal vs. 1,562 intrusion sequences). **D:** Collected in 1999. **E:** The signature of both classes covers system call numbers from 1 to 136. **F:** Yes, data is correctly labeled. Normal data and intrusion data are stored in separate folders. **G:** Yes, the data is consistent. Features are extracted sequentially by PIDs. **H:** There is 98.57% of overlaps between normal sequences and intrusion sequences.	1.46

ADFA-LD	**A:** Live data was collected from an UbuntuOS version 11.04. **B:** To monitor normal usage data and detect different types of attack signatures. There are 6 attack types included in the testing set: brute force attack over open FTP ports and SSH ports, unauthorized root user creation, target host compromise through Jave and Linux meterpreter payloads, privilege escalation over webshell. **C:** Original data is balanced (308,077 normal vs. 317,388 intrusion system calls). Processed data is balanced (161,400 normal vs. 194,000 intrusion sequences). **D:** Colected in 2013. **E:** The signature of both classes covers system call numbers from 1 to 325 in Linux kernel 2.6.38. **F:** Yes, data is correctly labeled. Training and Validating sets only contain normal data, and testing set only contains intrusion data. **G:** Yes, the data is consistent. Features are extracted sequentially by specific processes. **H:** There is 43.16% of overlaps between normal sequences and intrusion sequences.	0.46

6 Conclusion and future work

References

[1] Abiodun Ayodeji, Yong-kuo Liu, Nan Chao, and Li-qun Yang. A new perspective towards the development of robust data-driven intrusion detection for industrial control systems. *Nuclear Engineering and Technology*, 2020.

[2] Ashima Chawla, Brian Lee, Sheila Fallon, and Paul Jacob. Host based intrusion detection system with combined cnn/rnn model. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 149–158. Springer, 2018.

[3] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91, 2018.

[4] Haihua Chen, Jiangping Chen, and Junhua Ding. Data evaluation and enhancement for quality improvement of machine learning. In *2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS)*, pages 13–13, 2020.

[5] Abhijeet Sahu, Zeyu Mao, Katherine Davis, and Ana E Goulart. Data processing and model selection for machine learning-based network intrusion detection. In *2020 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR)*, pages 1–6. IEEE, 2020.

[6] Hassan Hadi Al-Maksousy, Michele C Weigle, and Cong Wang. Nids: Neural network based intrusion detection system. In *2018 IEEE International Symposium on Technologies for Homeland Security (HST)*, pages 1–6. IEEE, 2018.

[7] Jiankun Hu, Xinghuo Yu, Dong Qiu, and Hsiao-Hwa Chen. A simple and efficient hidden markov model scheme for host-based anomaly intrusion detection. *IEEE network*, 23(1):42–47, 2009.

[8] Nam Nhat Tran, Ruhul Sarker, and Jiankun Hu. An approach for host-based intrusion detection system design using convolutional neural network. In *International Conference on Mobile Networks and Management*, pages 116–126. Springer, 2017.

[9] Hongyu Liu and Bo Lang. Machine learning and deep learning methods for intrusion detection systems: A survey. *applied sciences*, 9(20):4396, 2019.

[10] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[11] Shengchu Zhao, Wei Li, Tanveer Zia, and Albert Y Zomaya. A dimension reduction model and classifier for anomaly-based intrusion detection in internet of things. In *2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, pages 836–843. IEEE, 2017.

[12] Tara Salman, Deval Bhamare, Aiman Erbad, Raj Jain, and Mohammed Samaka. Machine learning for anomaly detection and categorization in multi-cloud environments. In *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, pages 97–103. IEEE, 2017.

[13] Razan Abdulhammed, Miad Faezipour, Abdelshakour Abuzneid, and Arafat AbuMallouh. Deep and machine learning approaches for anomaly-based intrusion detection of imbalanced network traffic. *IEEE sensors letters*, 3(1):1–4, 2018.

[14] Sidharth Behera, Ayush Pradhan, and Ratnakar Dash. Deep neural network architecture for anomaly based intrusion detection system. In *2018 5th International Conference on Signal Processing and Integrated Networks (SPIN)*, pages 270–274. IEEE, 2018.

[15] Sheraz Naseer, Yasir Saleem, Shehzad Khalid, Muhammad Khawar Bashir, Jihun Han, Muhammad Munwar Iqbal, and Kijun Han. Enhanced network anomaly detection based on deep neural networks. *IEEE access*, 6:48231–48246, 2018.

[16] Nebrase Elmrabit, Feixiang Zhou, Fengyin Li, and Huiyu Zhou. Evaluation of machine learning algorithms for anomaly detection. In *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, pages 1–8. IEEE, 2020.

[17] Roya Aliakbarisani, Abdorasoul Ghasemi, and Shyhtsun Felix Wu. A data-driven metric learning-based scheme for unsupervised network anomaly detection. *Computers & Electrical Engineering*, 73:71–83, 2019.

[18] Mukrimah Nawir, Amiza Amir, Naimah Yaakob, and Ong Bi Lynn. Effective and efficient network anomaly detection system using machine learning algorithm. *Bulletin of Electrical Engineering and Informatics*, 8(1):46–51, 2019.

[19] Marina Evangelou and Niall M Adams. An anomaly detection framework for cyber-security data. *Computers & Security*, 97:101941, 2020.

[20] Jorge Meira, Rui Andrade, Isabel Praça, João Carneiro, Verónica Bolón-Canedo, Amparo Alonso-Betanzos, and Goreti Marreiros. Performance evaluation of unsupervised techniques in cyber-attack anomaly detection. *Journal of Ambient Intelligence and Humanized Computing*, 11(11):4477–4489, 2020.

[21] Ayush Hariharan, Ankit Gupta, and Trisha Pal. Camlpad: Cybersecurity autonomous machine learning platform for anomaly detection. In *Future of Information and Communication Conference*, pages 705–720. Springer, 2020.

[22] Sarika Choudhary and Nishtha Kesswani. Analysis of kdd-cup'99, nsl-kdd and unsw-nb15 datasets using deep learning in iot. *Procedia Computer Science*, 167:1561–1573, 2020.

[23] Maonan Wang, Kangfeng Zheng, Yanqing Yang, and Xiujuan Wang. An explainable machine learning framework for intrusion detection systems. *IEEE Access*, 8:73127–73141, 2020.

[24] Hamed Alqahtani, Iqbal H Sarker, Asra Kalim, Syed Md Minhaz Hossain, Sheikh Ikhlaq, and Sohrab Hossain. Cyber intrusion detection using machine learning classification techniques. In *International Conference on Computing Science, Communication and Security*, pages 121–131. Springer, 2020.

[25] Basant Subba, Santosh Biswas, and Sushata Karmakar. Host based intrusion detection system using frequency analysis of n-gram terms. In *TENCON 2017-2017 IEEE Region 10 Conference*, pages 2006–2011. IEEE, 2017.

[26] Pierre-François Marteau. Sequence covering for efficient host-based intrusion detection. *IEEE Transactions on Information Forensics and Security*, 14(4):994–1006, 2018.

[27] Elham Besharati, Marjan Naderan, and Ehsan Namjoo. Lr-hids: logistic regression host-based intrusion detection system for cloud environments. *Journal of Ambient Intelligence and Humanized Computing*, 10(9):3669–3692, 2019.

[28] Iqbal H Sarker, Yoosef B Abushark, Fawaz Alsolami, and Asif Irshad Khan. Intrudtree: a machine learning based cyber security intrusion detection model. *Symmetry*, 12(5):754, 2020.

[29] Shijoe Jose, D Malathi, Bharath Reddy, and Dorathi Jayaseeli. A survey on anomaly based host intrusion detection system. In *Journal of Physics: Conference Series*, volume 1000, page 012049. IOP Publishing, 2018.

[30] Markus Ring, Sarah Wunderlich, Deniz Scheuring, Dieter Landes, and Andreas Hotho. A survey of network-based intrusion detection data sets. *Computers & Security*, 86:147–167, 2019.

[31] Ansam Khraisat, Iqbal Gondal, Peter Vamplew, and Joarder Kamruzzaman. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1):1–22, 2019.

[32] Salvatore J Stolfo, Wei Fan, Wenke Lee, Andreas Prodromidis, and Philip K Chan. Cost-based modeling for fraud and intrusion detection: Results from the jam project. In *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*, volume 2, pages 130–144. IEEE, 2000.

[33] Jungsuk Song, Hiroki Takakura, and Yasuo Okabe. Description of kyoto university benchmark data. *Available at link: http://www. takakura. com/Kyoto_data/BenchmarkData-Description-v5. pdf [Accessed on 15 March 2016]*, 2006.

[34] DARPA2009. Darpa 2009 intrusion detection dataset. `http://www.darpa2009.netsec.colostate.edu/`, 2009. [online; accessed 12-April-2021].

[35] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*, pages 1–6. IEEE, 2009.

[36] Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, and Ali A Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *computers & security*, 31(3):357–374, 2012.

[37] Nour Moustafa and Jill 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)*, pages 1–6. IEEE, 2015.

[38] Waqas Haider, Jiankun Hu, Jill Slay, Benjamin P Turnbull, and Yi Xie. Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling. *Journal of Network and Computer Applications*, 87:185–192, 2017.

[39] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *ICISSp*, pages 108–116, 2018.

[40] MIT Lincoln Laboratory. 1998 darpa intrusion detection evaluation dataset. https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset, 1998. [online; accessed 12-April-2021].

[41] Steven A Hofmeyr, Stephanie Forrest, and Anil Somayaji. Intrusion detection using sequences of system calls. *Journal of computer security*, 6(3):151–180, 1998.

[42] Gideon Creech and Jiankun Hu. A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns. *IEEE Transactions on Computers*, 63(4):807–819, 2013.

[43] Dainius Čeponis and Nikolaj Goranin. Towards a robust method of dataset generation of malicious activity for anomaly-based hids training and presentation of awsctd dataset. *Baltic Journal of Modern Computing*, 6(3):217–234, 2018.

[44] Marc-Oliver Pahl and François-Xavier Aubet. All eyes on you: Distributed multi-dimensional iot microservice anomaly detection. In *2018 14th International Conference on Network and Service Management (CNSM)*, pages 72–80. IEEE, 2018.

[45] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In Sorelle A. Friedler and Christo Wilson, editors, *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, volume 81 of *Proceedings of Machine Learning Research*, pages 77–91, New York, NY, USA, 23–24 Feb 2018. PMLR.

[46] Curtis G Northcutt, Anish Athalye, and Jonas Mueller. Pervasive label errors in test sets destabilize machine learning benchmarks. *arXiv preprint arXiv:2103.14749*, 2021.

[47] Eitel J. M. Lauría and G. Tayi. Statistical machine learning for network intrusion detection: a data quality perspective. *International Journal of Services Sciences*, 1:179–195, 2008.

[48] Abhishek Divekar, Meet Parekh, Vaibhav Savla, R. Mishra, and M. Shirole. Benchmarking datasets for anomaly-based network intrusion detection: Kdd cup 99 alternatives. *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS)*, pages 1–8, 2018.

[49] Rakesh M. Verma, Victor Zeng, and Houtan Faridi. Data quality for security challenges: Case studies of phishing, malware and intrusion detection datasets. *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019.

[50] Wenfei Fan. Data quality: From theory to practice. *SIGMOD Rec.*, 44(3):7–18, December 2015.

[51] Alan F. Karr, Ashish P. Sanil, and David L. Banks. Data quality: A statistical perspective. *Statistical Methodology*, 3(2):137–173, 2006.

[52] Haihua Chen, Jiangping Chen, and Junhua Ding. Data evaluation and enhancement for quality improvement of machine learning. In *2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS)*, pages 13–13, 2020.

[53] Chieh-Han Wu and Yang Song. Robust and distributed web-scale near-dup document conflation in microsoft academic service. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 2606–2611. IEEE, 2015.

[54] José A Sáez, Bartosz Krawczyk, and Michał Woźniak. On the influence of class noise in medical data classification: Treatment using noise filtering methods. *Applied Artificial Intelligence*, 30(6):590–609, 2016.

[55] Gabriel Maciá-Fernández, José Camacho, Roberto Magán-Carrión, Pedro García-Teodoro, and Roberto Therón. Ugr '16: A new dataset for the evaluation of cyclostationarity-based network idss. *Computers & Security*, 73:411–424, 2018.

[56] Richard Y Wang and Diane M Strong. Beyond accuracy: What data quality means to data consumers. *Journal of management information systems*, 12(4):5–33, 1996.

[57] Carlo Batini, Cinzia Cappiello, Chiara Francalanci, and Andrea Maurino. Methodologies for data quality assessment and improvement. *ACM computing surveys (CSUR)*, 41(3):1–52, 2009.

[58] Wenfei Fan. Data quality: From theory to practice. *Acm Sigmod Record*, 44(3):7–18, 2015.

[59] Haihua Chen, Gaohui Cao, Jiangping Chen, and Junhua Ding. A practical framework for evaluating the quality of knowledge graph. In *China Conference on Knowledge Graph and Semantic Computing*, pages 111–122. Springer, 2019.

[60] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision support systems*, 43(2):618–644, 2007.

[61] Amira Bradai and Hossam Afifi. Game theoretic framework for reputation-based distributed intrusion detection. In *2013 International Conference on Social Computing*, pages 558–563. IEEE, 2013.

[62] Ranjit Panigrahi and Samarjeet Borah. A detailed analysis of cicids2017 dataset for designing intrusion detection systems. *International Journal of Engineering & Technology*, 7(3.24):479–482, 2018.

[63] Kathy Lee, Sadid A Hasan, Oladimeji Farri, Alok Choudhary, and Ankit Agrawal. Medical concept normalization for online user-generated texts. In *2017 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 462–469. IEEE, 2017.

[64] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[65] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

A Appendix A: Abbreviations of the machine learning models

Logistic regression (LR), support vector machine (SVM), decision tree (DT), random forest (RF), artificial neural network (ANN), isolation forest (IF), k nearest neighbor (KNN), scaled convex hull (SCH), histogram-based outlier score (HBOS), cluster-based local outlier factor (CBLOF), naïve bayes (NB), averaged one dependence estimator (AODE), radial basis function network (RBFN), multi-layer perceptron (MLP), deep neural networks (DNN), convolutional neural network (CNN), long short-term memory (LSTM), recurrent neural network (RNN), gated recurrent unit (GRU), softmax regression (SR), hidden markov model (HMM), variational autoencoder (VAE), sequence covering for intrusion detection (SC4ID), Intrusion Detection Tree (IntruDTree).

B Appendix B: Detail descriptions of the datsets for IDS.

Kyoto 2006. The Kyoto 2006+ dataset built on the 3 years of real network traffic data from Kyoto University servers and honeypots. Kyoto 2006 dataset has 93,076,270 sessions (50,033,015 normal sessions, 42,617,536 known attack sessions and 425,719 unknown attack sessions) of honeypot data. Instants are labelled as normal (no attack), attack (known attack) and unknown attack. It consists of 24 features including (“duration”, “service”,“source bytes”, “destination bytes”, “count”, “same srv rate”, “serror rate”, “srv serror rate”, “dst host count”, “dst host srv count”, “dst host same src port rate”, “dst host serror rate”, “dst host srv serror rate”, and “flag”, “IDS detection”, “malware detection”, “ashula detection”, “label”, “source IP address”, “source port number”, “destination IP address”, “destination port number”, “start time”, and “duration”).

ISCK. ISCX dataset consists of the 7 days of network activity (normal and malicious) from Information Security Center of Excellence (ISCX) at the University of New Brunswick. It contains labeled network traces, including full packet payloads in pcap (packet capture file) format. It has six general categories including E-mail, Instant Messaging, Streaming, File Transfer, VoIP, and P2P. Inside the categories, there are 14 different applications. The two general profiles used in the ISCX dataset are (1) profiles attempt to describe an attack scenario in an unambiguous manner and (2) profiles encapsulate extracted mathematical distributions or behaviors of certain entities. The malicious activity includes (1) Infiltrating the network from inside, (2) HTTP Denial of Service, (3) Distributed Denial of Service, and (4) Brute Force SSH.

UNSW-NB15. UNSW-NB15 is a very new dataset. This data set consists of 49 features of network traffic using the flow based between hosts. It has nine different modern attack types and new patterns of normal traffic. The full dataset consists of 25,400,443 records and contains ten classifications, one typical and the nine attacks: nonexclusive, misuses, fuzzers, DoS, observation, examination, secondary passage, shellcode, and worms. The UNSW-NB15 data set main characteristics are a hybrid of the real modern normal behaviors and the synthetical attack activities.

DARPA 2009. The DARPA 2009 dataset is created with synthesized HTTP, SMTP, and DNS background data traffic to emulate traffic between a /16 local subnet that goes through a cisco router to the Internet. This dataset contains modern attack vectors. It has been captured in 10 days of the year 2009. DARPA 2009 has a variety of security events and attack types that describes the modern style of attacks including different distributed denial of service (DDoS) attacks and worms. The dataset consists of 7000 pcap files with around 6.5TByte of total size.

CICIDS2017. CICIDS2017 consists of labeled network flows, including full packet payloads in pcap format. It is a five days attack and general traffic data distributed among eight files of Canadian Institute of Cybersecurity. The dataset contains 3119345 instances and 83 features containing 15 class labels, one normal and 14 attack. There is a total of 100748 records current and a total of 43 variables. The executed attacks include Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet and DDoS. The dataset is labeled and imbalanced with no redundant instances. This dataset is also suffer from high class imbalance.

NGIDS-DS. NGIDS-DS dataset was generated based on the modern computing infrastructures in 2016 by UNSW at the Australian Defense Force Academy. It reflects the latest characteristics and realistic performance of recent attacks. NGIDS-DS contains ground-truth.csv; 99 csv files of host logs; NGIDS.pcap of the network packets; feature-descr.csv; and readme.txt. Thus, it is useful to perform either HIDS or NIDS, or a combined analysis. NGIDS-DS contains 99 host log files with csv format. The dataset has normal and malicious activities, including system calls, timestamp, event ID, process ID, path, attack category, attack subcategory, and label ("1" is marked for an attack and "0" is for a normal activity). The dataset contains 313,926 records with 7 features for ground-truth cs; 90,054,160 records with 9 features for the 99 csv files of host logs; and 1,094,231 records with 18 features for NGIDS.pcap. The malicious activities include Exploits; DoS; Worms; Generic; Reconnaissance; Shellcode; and Backdoors. However, there are fewer attack than normal records, with a ratio of 1:90 which make the dataset imbalance.

DARPA 98/99. DARPA dataset is a network traffic and audit logs collected files specify sessions on a simulation network at MIT Lincoln Laboratory as an attempt to identify attack sessions in the midst of normal activities. The DARPA 98 dataset contains seven days traces and DARPA 99 contains five weeks of network traffic. The total collected data contains more than 200 instances of network-based attack types, classified as normal or as one of the 39 attack, embedded into background traffic like air force base local area network. The training set used for this paper contains 22 different attack types where the test set contains approximately 114 instances of 37 different attacks. Also, 17 attacks are new and not part of the training set and the other two attacks only appear in the training data. Each line in a list file corresponds to a separate session where each session corresponds to an individual TCP/IP connection between two computers. There are 11 columns in list files, the first nine columns provide information identifying the TCP/IP connection. The performance of intrusion detection systems will be evaluated using scores assigned to the tenth column. Each Score is associated with an attack type in columns eleven. The score and attack name columns will be provided already filled in for sample and training data. Attacks include instances where a remote user illegally obtains local user-level privileges or local root-level privileges on a target machine and also instances where a remote user surveys a potential target for weaknesses or searches for potential targets. Attacks in the sample data include guess, ping-sweep, port-scan, phf, rlogin, rsh, and rcp.

KDD99. To select appropriate system features from audit data to build models for intrusion detection, Lee and Stolfo developed framework to extract features from DARPA 98/99 datasets and then the dataset called KDD99. The KDD99 is the most widely dataset used for IDS. It continues TCP attributes but no information about IP addresses. KDD99 provides labeled data for intrusion detection and contains 41 features (excluding the labels) and five classes, namely Normal, DoS, Probe, remote-to-local (R2L), and user-to-root (U2R). The KDD99 contains 494,021 and 311,029 records in the training and testing sets. The classes in the training and testing sets of the KDD99 are imbalanced.

NSL-KDD. NSL-KDD is a revised cleaned-up version of the KDD'99 from the University of New Brunswick. The NSL-KDD dataset was created to improve upon the shortcomings of the KDD99 dataset. KDD99s record redundancy hinders an algorithms ability to learn by causing a bias against infrequent records and, in turn, overlooking harmful attacks. This issue was resolve with the removal of duplicate records in both the training and testing sets. Therefoe, NSL-KDD is comprised of four sub data sets: KDDTest+, KDDTest-21, KDDTrain+, KDDTrain+_20Percent, although KDDTest-21 and KDDTrain+_20Percent are subsets of the KDDTrain+ and KDDTest+. The trainind dataset from kd99 contained 78% redundant instants while testing dataset 75% redundant instants. After removing the redundant records the NSL-KDD consists of 1,152,281 distinct records from KDD99 dataset; And contains 41 features per record. The data set contains 43 features per record, with 41 of the features referring to the traffic input itself and the last two are labels normal or attack and Score.

UNM Dataset. University of New Mexico (UNM) uses strace to capture system call sequences from 10 different programs: synthetic sendmail, synthetic ftp, synthetic lpr, live lpr, xlock, live named, login and ps, inetd, sendmail and stide. Each of them contains normal and intrusion data, except sendmail dataset. Therefore, we do use sendmail in our experiment. Some of them are live data, and some are synthetic data. Live data were recorded during normal usage, whereas synthetic data were collected using scripts that simulate both normal and anomalous behaviors. However, these datasets are out of date as they were collected around 1996. Each dataset includes sequential traces of multiple processes from the start until the end. Since processes run simultaneously, the sequence trace from a specific process ID (PID) can be interrupted by the traces of other PIDs. The data format from UNM contains 2 integer columns: PID and System Call number. Additionally, there is no time frame nor user ID in the data as they have already been processed;

however, the sequences of system calls are listed in order. Since different programs with different versions were used to generate the data, most dataset has their own mapping file. A mapping file contains a list of system call name in order, in which the index (starting from 0) corresponds to the system call number in the dataset. Due to a large number of data files, we were able to map the system call numbers to their names in some files.

MIT dataset. MIT dataset is also included in the UNM data repository. In 1997, they have collected live normal data for lpr for 2 weeks using 77 hosts. A synthetic attack script was written to exploit vulnerabilities from the older versions of lpr. MIT data also has the same format as UNM data, where each file contains 2 numeric columns: PID and System Call number.

ADFA dataset. In 2013, the University of New South Wales at the Australian Defense Force Academy developed 2 new datasets that are ADFA-LD and ADFA-WD. The first one includes host-based and network-based intrusion data from a Linux operating system version 11.04, whereas the second one includes data from a Window operating system. The host-based dataset contains 3 files: training, validating and testing set. Since it is designed for an anomaly-based intrusion detection, the training and validating sets were collected during normal activities, where there were only normal sequences. On the other hand, testing set contains intrusion sequences from various attacks including password brute force, web-based attacks and remote exploits. Any sequences that are not recognized during training phase will be flagged as an intrusion sequence. These attacks are conducted by penetration testers and hackers, and it represents current cyberattack strategies. There are 6 types of simulated attacks, and the testing set contains 10 attacks per type. The main difference between datasets from ADFA versus from UNM is the data format. The ADFA dataset contains only traces of system call number from different programs. Additionally, there is no PID nor time stamp for each system call in ADFA data.