# Issues and Recent Advances in Machine Learning Techniques for Intrusion Detection Systems

**Anish Naik**, anish.r.naik@gmail.com (A paper written under the guidance of Prof. Raj Jain)

Download

## Abstract

As networks get more complex with the success of technologies such as cloud computing, virtualization, and IoT, the attack surface for cybercrimes continues to grow. There is a necessity for a line of defense that is both reactive and predictive. To bridge this gap, Intrusion Detection Systems (IDSes) have come to the forefront of academic and industry research. This paper presents a survey of machine learning applications for intrusion detection systems. In addition to discussing the current state of research for IDSes, the paper also discusses the overarching complications that plague the field today as well as the latest developments in trying to solve those problems. A special focus is given to discussing deep learning, computationally cheap, and distributed techniques to overcome the issues that shallow learning techniques present.

---

## Table of Contents

---

## 1 Introduction

The cybersecurity industry is one of the fastest growing today. As of 2019, the global security market is estimated to be worth over 140 billion dollars [IndustryARC19] and continues to grow at a breakneck pace. Trends such as cloud computing, virtualization, and IoT have made data the most lucrative asset, but also the most vulnerable. As networks get larger, the attack surface for hackers increases, making cyber risk a prevalent problem. Cyberattacks on companies, such as Equifax or Facebook, show just how vulnerable even the largest enterprises are [Bernard17, Isaac18]. Not only is the company's reputation at stake but also the sensitive and compromising information of millions of customers.

As hackers and attacks get sophisticated, the defense to prevent such attacks must as well. There are many common middlebox solutions that exist today. Applications such as firewalls and honeypots are the first line of defense against a cyber-attack. Firewalls act as a filter for network traffic - using a set of rules, a firewall will prevent hosts from outside the internal network to connect to a secure end system. Firewalls suffer from the ability to maintain state; a persistent attacker can easily bypass a firewall and gain entry into the enterprise network. Honeypots act like a trap for attackers - by advertising the possibility that it contains sensitive information, hackers will try to access the honeypot and are then blocked from the network. However, honeypots are only successful if they can bait the attacker. If the attacker realizes that the honeypot is trying to fool them, they can ignore it and continue to attack the network. Thus, a necessity was created for a system that can learn the structure of network data and differentiate normal from abnormal network traffic. To achieve this, the idea of an intrusion-detection system (IDS) was proposed.

An IDS monitors traffic as it travels through a network and raises alerts if it notices abnormal traffic entering the system. An IDS comes in two forms: host-based IDS (HIDS) and network-based IDS (NIDS). A HIDS is a software that runs on a host machine or on a centralized controller to monitor access to the file system, verify chains of system calls, or malicious changes to environment/system variables. On the other hand, a NIDS monitors traffic that travels through a network and usually runs on edge routers/switches. Both kinds of IDSes are used in practice. This paper mostly focuses on the latest research on developing NIDSes. Some discussion about HIDSes is provided, and the distinction is made wherever necessary.

The greatest challenge that network-based detection systems must overcome is trying to model the behavior of normal and abnormal traffic. There are over 100 different Internet Protocols and hundreds of different kinds of applications that the protocols are used for. Thus, how do we create a single, almighty IDS that can detect all kinds of attacks in an efficient, predictive manner but can also keep up with high network traffic and complex features? The answer proposed by the research community was to implement machine learning.

Machine learning (ML) has become the greatest trend in technology today - complex problems that don't seem to have a clear-cut answer can be realized using an ML model. The model will learn the necessary features of a dataset to make strong predictions. The most popular applications, like Facebook and Netflix, use ML to predict what advertisements to display and what TV shows a user might prefer. Today, cutting-edge research in IDSes employs a plethora of different ML techniques to see if they can be used in an IDS to secure enterprise networks. The rest of the paper is divided into the following sections: Section 2 will discuss the architecture of IDSes as well as a general overview on ML, Section 3 will present the application of ML techniques for IDSes through a survey of recent papers, Section 4 will provide the overarching issues and challenges that this research field faces, Sections 5-7 will discuss current solutions to the problems presented in Section 4, and Section 8 will provide general conclusions and discuss the future of the application of ML for intrusion detection systems. The novel contributions of this paper are presented in Sections 4-7. There are a lot of excellent surveys that are more thorough than this paper that discusses each machine learning technique and how they fare in understanding network traffic [Ahmad18, Boutaba18, Mishra19, Buczak18]. However, this paper tries to focus more on why these problems arise and what the research community is doing today to solve them.

---

## 2 Overview of Intrusion Detection Systems and Machine Learning

An Intrusion Detection System can come in three different forms: misuse, anomaly, and hybrid detection. The classical IDS, proposed in 1986, had two different components - one that implemented misuse-based detection and the other implemented anomaly-based detection [Kunal19]. Hybrid detection uses misuse and anomaly-based detection in tandem to reach its results. This paper mainly focuses on misuse and anomaly-based detection methods. Figure 1 shows the two-component IDS architecture that was originally proposed.

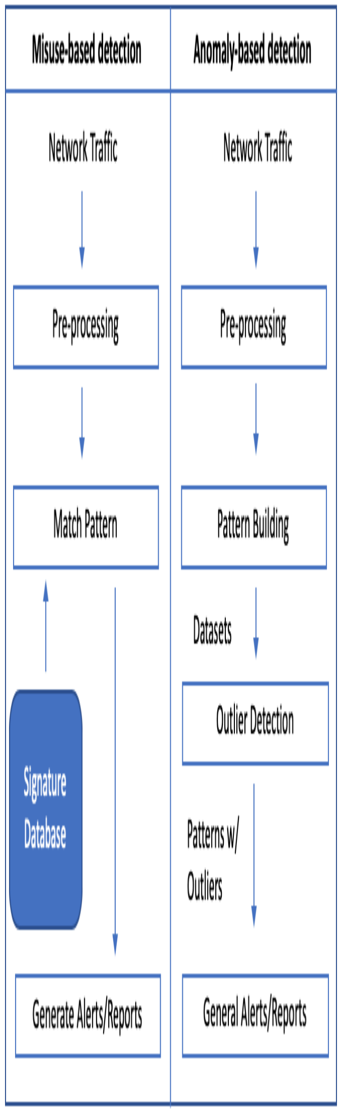

**Figure 1:** Architecture of an Intrusion Detection System [Kunal19]

## 2.1 Misuse-based Detection

Misuse-based detection can be broken into two categories: knowledge and ML-based. Knowledge-based detection commonly relies on a database of known attacks and their signatures, known as signature-based detection. The database could also store the state transitions of a system or the chain of system calls during an attack [Mishra19]. The latter is usually used in a HIDS to see if malware or a malicious file is trying to be downloaded on a host. In signature-based detection, the IDS monitors incoming packets and checks to see if any of the signature patterns in the database matches the incoming packet headers (see the left side of Figure 1). If there is a match, the system raises an alert for a potential attack.

Signature-based detection IDSes, such as SNORT, have gained commercial success due to its easy implementation. By employing a rule-based system, SNORT is able to detect known attacks with high accuracy and is able to work on cheap commodity hardware. SNORT is open-source and is considered to be the most popular signature-based detection system [Roesch99]. The system is able to work online and can detect potential attacks on-the-fly. Other popular signature-based IDSes are expensive and require powerful routers that can handle the heavy workload.

However, signature-based IDSes come with one main drawback. The system will fail to find variants or mutations of a known attack. Since the signatures of attacks change as they evolve, the IDS system won't be able to match the variant of a known attack and will fail to raise an alert [Kunal19]. Thus, the database of signatures must be constantly updated to keep up to date with evolving attacks. To deal with this, ML-based misuse detection was proposed. The benefit of using an ML model is that it can learn from the database of signatures and then predict the possibility of an evolving known attack. The training process allows a model to understand the general structure of a known attack - thus allowing it to predict possible variants.

Misuse-based detection, either knowledge-based or ML-based, fails to address the concern of dealing with unknown attacks. The database used for matching only contains information about known attacks, and thus unknown attacks will sneak through a network [Kunal19]. To better defend against unknown attacks, the idea of anomaly-based detection was proposed.

## 2.2 Anomaly-based Detection

Anomaly-based detection was created to address the problem of the zero-day attack - a novel attack whose behavior or information is not stored in a database. Anomaly-based detection can be developed in three ways: ML techniques, a finite-state machine, or statistical techniques. Please refer to [Mishra19] for more information on the application of finite-state machines and statistical techniques for anomaly-based detection. The focus of this paper is on ML-based anomaly detection. For anomaly-based detection, the ML model doesn't learn through a database of labeled attacks with known patterns and signatures but rather uses features of network traffic flow such as source address, destination address, bytes per flow, source port, destination port, and much more to learn the general feature set of normal traffic [Mishra19]. These features are monitored over a period of time and are used as the dataset to train the ML model.

A common way of detecting an attack using an anomaly-based system is by using outlier detection (see the right side of Figure 1). If a strong probability distribution of what it means to be normal traffic is realized, an abnormal packet or flow can be flagged as an outlier, and an alert is raised. Anomaly-based detection suffers from a high false-positive rate since it is a difficult task to train a model that can accurately differentiate between abnormal and normal behavior. False-negatives are also common for the same reason [Kunal19]. This issue of modeling network traffic is one of the most difficult problems to solve since the number of features that a flow has is enormous, and sometimes the features that differentiate an attack from a normal flow are quite obscure. The next section discusses the taxonomy of different machine learning techniques and the common ones used for misuse and anomaly-based detection.

## 2.3 Machine Learning Overview

Machine learning can be broken down into shallow learning and deep learning. Shallow learning relies on a field expert to identify the relevant features for evaluation. Thus, among the features that are in a flow, the field expert must select the features they believe are the most relevant and use that to train the ML model [Apruzzese18]. Additionally, feature selection methods such as Principal Component Analysis (PCA) can be used. Typical examples of shallow learning are random forests, support vector machines (SVM), clustering, and many more. Deep learning, on the other hand, relies on the model selecting the features it deems to be the most important/valuable. The classic example of deep learning is a deep neural network (DNN) - a model that uses multiple layers of inputs and outputs to reach a conclusion [Apruzzese18]. Since feature selection has to be done a-priori for shallow learning, it is computationally less expensive but may give worse generalization bounds. On the other hand, deep learning requires a huge amount of data to reach a strong conclusion and is thus harder to implement, but the model learned may be better.

Learning can be broken down further - it can either be supervised or unsupervised. Figure 2 shows a taxonomy of different ML techniques. Supervised learning uses data that has been labeled/classified [Apruzzese18]. This means that each data point x has a result y that is known. In the case of signature-based systems, y is whether the data is normal or abnormal, and x is the signature for that packet/flow. Labeled data is useful for classification problems and is most commonly used for ML-based misuse detection [Apruzzese18]. A supervised ML model will use the labeled data to learn an algorithm that can detect known and variants of known attacks. On the other hand, unsupervised learning does not have labels for the data points and instead only has x values [Apruzzesse18]. This is useful for grouping data points that have similar features. Anomaly-based detection uses unsupervised learning to learn an ML model based on traffic flow data that has been captured. Anomaly-based detection should use unlabeled data because the model should learn the general structure and features of normal network traffic [Apruzzesse18]. Using labeled data for anomaly-based detection adds significant bias since a field expert has imposed that normal network traffic needs to look a certain way. This is not always the case, and [Buczak18] believes that anomaly-based detection can benefit from labeled data.

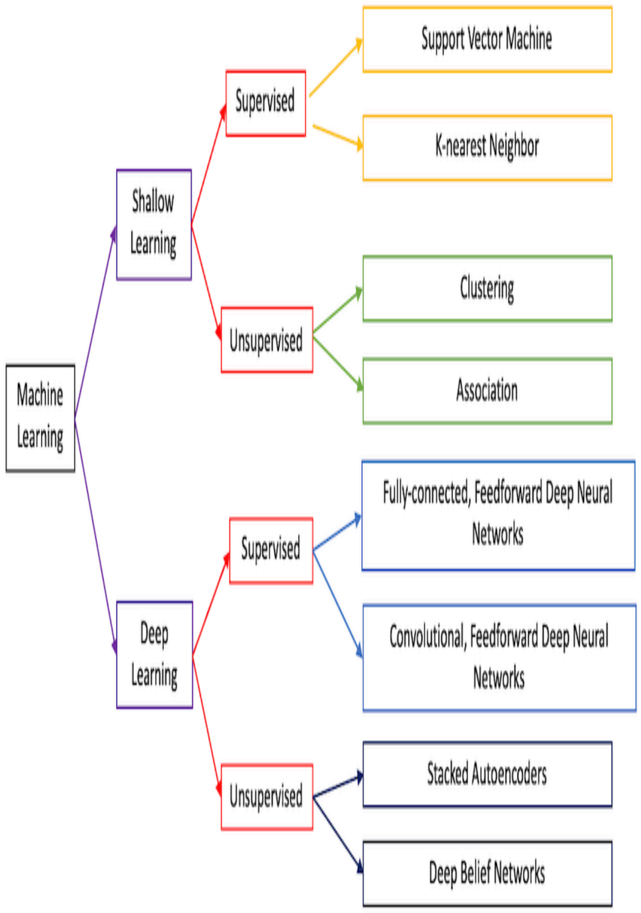

**Figure 2:** Taxonomy of Machine Learning Techniques [Apruzzese18]

The next section focuses on shallow learning models; this includes both supervised and unsupervised techniques. As the discussion will show, shallow models work fairly well for detecting different kinds of attacks, but no one single model or ensemble of models generalizes well for all the different kinds of attacks. The section also elaborates on the attacks of interest as well as the datasets that are most commonly used for training and testing these shallow learning models.

---

## 3 Shallow Learning for Intrusion Detection Systems

This section contains three main parts: the datasets used for testing and training the machine learning models, the types of attacks seen in these datasets, and a survey discussing how well different shallow learning ML models classify the different kinds of attacks presented. The three most important values for measuring the effectiveness of an ML model are precision, recall, and the F1-score. Precision measures the accuracy of the model or what percentage of the results are relevant. Recall measures the detection rate of a model or what portion of the relevant results were correctly classified by the model. The F1-score is the mean of recall and accuracy [Shung18]. The higher all these values are, the better the model.

## 3.1 Datasets Used for Training and Testing

An important factor of interest when it comes to training and testing an ML model is the dataset that is used. There are usually three types of datasets that ML models for IDSes are trained and tested on packet capture (pcap) data, NetFlow data, and public datasets [Buczak18]. A major problem we see in ML for cybersecurity research is the availability of data from a variety of popular vendors. Data from across vendor-specific applications are unique and are usually proprietary. Thus, it is difficult for the research community to obtain data that is true to what the face of networking and software is today. To circumvent this problem, the research world uses the three types of datasets mentioned above for training and testing.

### *Packet Capture Data*

Packet capture (pcap) data captures all the headers and payloads of a packet across the network stack. The IETF has defined 144 Internet Protocols such as UDP, TCP, ICMP, and many more [Buczak18]. Pcap data contains all the different features of a protocol or application. Packets that are created by an application travel through a network and can then be captured at switches or routers using applications such as Libpcap or WinPCap [Buczak18]. The pcap data can then be analyzed by software like Wireshark.

### *NetFlow Data*

NetFlow is a proprietary router/switch feature that was introduced by Cisco. As traffic enters and leaves a network, NetFlow is able to collect IP packets. NetFlow defines a network flow as a unidirectional sequence of packets with the same seven attributes: source IP address, destination IP address, source port, destination port, IP protocol, and IP type of service [Buczak18]. Additional features of importance can be derived, such as bytes per packet, packets per flow, and TCP flags. The main difference between NetFlow and pcap data is that NetFlow uses a definition of network flow to define a datapoint while pcap treats each packet independently.

### *Public Datasets*

Public datasets have become the most popular type of dataset to test ML models for IDSes. The Defense Advanced Research Projects Agency (DARPA) 1998 and 1999 datasets are the most extensively used in experiments and are the most cited [Buczak18]. The TCP/IP data was collected over the span of 9 weeks, of which 7 weeks were used for training while the remainder was used for testing. Following this development, the Knowledge Discovery and Data Mining 1999 (KDD'99) dataset was created, which was originally used for the KDD Cup Challenge. This dataset is based on the DARPA datasets but also contains features captured by pcap. This dataset has a total of 41 features - many are basic, but some are derived features. Although the KDD'99 dataset is the most popular, it has some serious limitations. First, the dataset contains a lot of redundant records (78% in training and 75% in test) [Buczak18]. Second, a lot of the data is synthesized instead of being actual data from a real network. This second issue is due to privacy concerns, as mentioned previously. To deal with the first issue, the NSL-KDD dataset was proposed, which contains only select records from the KDD'99 dataset. The DARPA and

KDD'99 datasets break down network traffic into distinct categories of attack types, which is discussed in the next section.

## 3.2 Types of Attacks

The KDD'99 dataset focuses on four different kinds of attacks: resource depletion, scanning or probe attacks, remote to local (R2L) attacks, and user to root (U2R) attacks. Figure 3 shows the breakdown of the different kinds of attacks and specific tools or techniques that are used to deploy them. These four types of attacks do not exhaust the list of the types of attacks that can be seen on a network but are the most popular kinds of attacks. Understanding and classifying these attacks is already a challenge by itself and, if solved, will become the first huge milestone for an ML-based IDS.

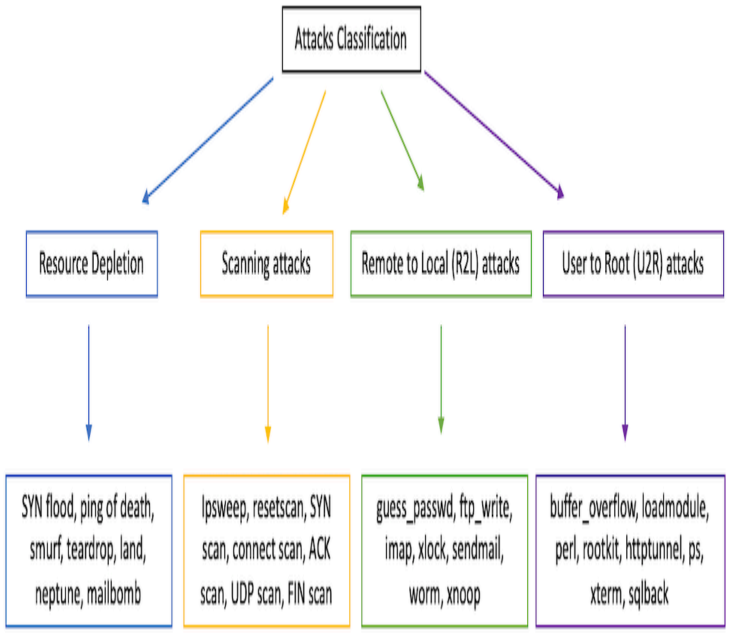
Attacks Classification
Resource Depletion
Scanning attacks
Remote to Local (R2L) attacks
User to Root (U2R) attacks
SYN flood, ping of death, smurf, teardrop, land, neptune, mailbomb
Ipsweep, resetscan, SYN scan, connect scan, ACK scan, UDP scan, FIN scan
guess_passwd, ftp_write, imap, xlock, sendmail, worm, xnoop
buffer_overflow, loadmodule, perl, rootkit, httptunnel, ps, xterm, sqlback

**Figure 3:** Taxonomy of the various attacks in the KDD'99 dataset [Mishra19]

### *Resource Depletion*

Resource depletion is an attack where users of an application or a network are unable to use the resources of that service. This is more commonly known as a Denial-of-Service (DoS) attack. An attacker can flood a network with requests and overload the bandwidth of the network. Since the network has to service the inundation of requests, normal users who make legitimate requests to the network are unable to access its services [Mishra19]. A more powerful version of a DoS attack is a Distributed Denial of Service. In this case, an attacker controls a number of machines (zombies) instead of just one and thus magnifies the bandwidth or resource depletion of a classical DoS attack.

#### *Scanning Attacks*

A scanning attack is usually the first step for an attacker to gain entry into a network. Attackers will send scanning attacks to understand the topology of a network or identify machines with vulnerable software running on them. Scanning attacks have become more sophisticated and have gone from just open scans, that a firewall can stop, to stealthy scans that obfuscate the attack more and can sneak through a firewall, revealing information about a machine's open ports and vulnerable software [Mishra19].

#### *Remote to Local Attacks*

Remote to Local attacks are used to gain local access to a machine in a network via a malicious remote host. This is usually done after a scanning attack. After scanning a machine for open ports, an attacker can send packets to those ports to try to obtain local access. This can happen in many ways, such as guessing the password of the machine, exploiting the FTP protocol, or using a backdoor to gain access to the machine [Mishra19].

#### *User to Root Attacks*

User to Root attacks are used to get root access to a machine by an unprivileged user. The classical form of this kind of attack is a buffer overflow attack. An unprivileged user can exploit a piece of vulnerable software to overwrite its stack with shellcode, which leads to the program opening a root shell for the attacker. This form of privilege escalation can help the attacker get access to sensitive information and allow them to wreak havoc on the machine or on the greater network. It is important to note that it is hard to differentiate between a U2R and R2L attack. For an attacker to perform a U2R attack, they must first gain local access to a machine - i.e., an R2L attack. Thus, the packet header/payload values for a U2R attack is quite similar to normal behavior since the malicious connection is already made [Mishra19]. This is an important fact and will play a big role in the conclusions that are drawn from the rest of the paper. The next subsection provides a brief survey how well different shallow learning techniques classify the attacks mentioned in this section.

### 3.3 Comparison of Shallow Learning Models

There are two facts that will be useful for the analysis provided in this section. First, as mentioned in Section 3.1, the KDD'99 dataset contains a total of 41 features. Second, machine learning models can be composed of a single classifier or an ensemble of classifiers. Since IDSes are focused on the multi-classification problem, an ML model learning to classify is called a classifier. The two distinctions presented provide us with four different categories of learning: single classifier with all features, single classifier with feature selection, multiple classifiers with all features, and multiple classifiers with feature selection. This is the same breakdown that [Mishra19] uses and is the most informative and easy to understand.

### *Single Classifier with all Features*

A single classifier with all features performed the worst out of all the categories. The benefit of a single classifier is that they are easy to interpret. However, using all the features leads to the possibility of overfitting as well as having bad computational time since a dataset with too many features makes learning the target function harder [Mishra19]. Classifiers that were discussed in the surveys researched included SVMs, Decision Trees (DT), Artificial Neural Networks (ANN), and many more. The multi-classification problem is hard for a single classifier to solve. Techniques such as SVMs and DTs, although suitable for multi-classification, do not perform well by themselves to learn a complex problem such as the structure of network traffic. Most of the papers surveyed used KDD'99 as their dataset of choice, and a few explored other public datasets such as the ISCX, UNSW, and ISOT datasets. Additionally, most papers were interested in creating a misuse-based detector, with a few exceptions attempting to develop an anomaly-based detector.

### *Single Classifier with Feature Selection*

To decrease the computational complexity of analyzing all features, techniques for feature selection can be used, such as Principal Component Analysis (PCA). Single classifiers with a limited feature set seem to perform better than a single classifier with all features. Since a fewer number of features are analyzed, the computational speed of the classifier is improved. However, the detection rate is marginally better than a single classifier with all features. It is important to note that features that are important for one kind of attack might not be the best feature set for another type of attack [Mishra19]. In general, single classifiers tend to be greedy and overfit the training data and thus have a hard time generalizing well to out-of-sample data points. This is evidenced by the fact that a lot of the single classifier models are able to classify a specific kind of attack well and not the others. As for the case with single classifiers with all features, the most popular dataset of choice was the KDD'99 dataset. A few papers used the NSL-KDD dataset, and [Sangkatsanee11] used a pcap library to obtain data for training and testing. Most papers were, again, interested in misuse-based detection.

### *Multiple Classifiers with all Features*

To improve on the shortcomings of one classifier, an ensemble of classifiers can be used. Using all the features, multiple classifiers do have better precision and recall compared to that of a single classifier. However, using an ensemble of classifiers is computationally more expensive and makes the system more complex. It is important to note that multiple classifiers with all features have a higher detection rate of U2R and R2L attacks than single classifiers, but it is still a poor to average detection rate [Mishra19]. The KDD'99 and DARPA were the most popular datasets of choice. An interesting takeaway was that a few papers presented hybrid detection techniques, a combination of the anomaly and misuse-based methods. This is doable since multiple classifiers are used in an ensemble. Thus, a potential ensemble could be a combination of clustering and SVM which would, ideally, lead to better results.

***Multiple Classifiers with Feature Selection***

Analyzing a limited number of features allows for a less complex system, compared to multiple classifiers using the entire feature set. In some cases, the detection rate is also better. Specifically, the detection of R2L and U2R is better than the rate realized with multiple classifiers on the entire feature set. The computational time to train the model is still quite high, and so is the time to classify a record as normal or malicious. To improve upon this, the training process can be done in parallel such that each ML module is trained at the same time instead of in a serial fashion. Also, it is important to note that there is a myriad of combinations that can be used for an ensemble classifier. To figure out the best combination, techniques such as cross-validation can be used to identify the best ensemble classifier for a specific kind of attack [Mishra19]. Similar to the case with multiple classifiers using all features, KDD'99 was the most popular dataset of choice, and hybrid detection techniques were also explored.

See Table 1 to identify which kind of ML model/ensemble performs best against each kind of attack. Each cell contains the ML models used, the number of features evaluated, and the detection rate. It is evident that there is no single solution for all attacks or an ML model that works best for any single attack.

**Table 1:** ML techniques used, number of features, and detection rate for each type of attack [Mishra19]

| | **Single Classifier with all Features** | **Single Classifier with Feature Selection** | **Multiple Classifiers with all Features** | **Multiple Classifiers with Feature Selection** |
|---|---|---|---|---|
| DoS | Decision Tree, 41, **97.24%** | CANN, 6, **99.99%** | Ensemble of ANN, SVM, and MARS, 41, **99.97%** | SVM, DT, and SA, 23, **100%** |
| Scanning | Naive Bayes, 41, **88.83%** | CANN, 6, **99.99%** | ANN with Elman Network, 41, **100%** | Subspace Clustering, DBSCAN, and EAR, 9, **100%** |
| User to Root | Fuzzy Association, 41, **68.60%** | MMIFS with SVM, 8, **30.70%** | Multi-NN, 41, **99.7%** (only for guess_passwd) | FNT, PSO, and GA, 12, **99.7%** |
| Root to Local | Neural Network, 41, **26.68%** | MMIFS with SVM, 8, **84.85%** | Ensemble of ANN, SVM, and MARS, 41, **98%** | FNT, PSO, and GA, 12, **99.709** |

## 4 Overarching Themes

Based on the discussion provided in Section 3, there is still a lot of work to be done. There is no one solution that can solve all the cyber-attacks that a network can face. However, as two decades of research shows, there is a lot of potential in the field of machine learning to change the face of cybersecurity today. This section will discuss the overarching problems that research in ML for IDSes is facing today.

### 4.1 Issues with DARPA and KDD'99 Datasets

Of the 39 papers that were analyzed in [Buczak18], 28 of them used DARPA 1998, DARPA 1999, DARPA 2000, or KDD'99. The remainder used a combination of NetFlow, tcpdump, DNS, or SSH commands data. The problem with the datasets used for training and analysis is two-fold: (1) the DARPA and KDD'99 datasets are close to 2 decades old, and (2) the data is synthesized. As networking continues to evolve through virtualization, cloud computing, and the growth of IoT, a dataset that was developed in the late 1990s is not a representative model of what networking is today. Second, the data in the DARPA and KDD'99 datasets is synthesized traffic data generated by a small number of machines that created planned attacks [Buczak18]. This is far from how real traffic looks. Unfortunately, obtaining real traffic from large enterprises is a huge privacy issue. Releasing traffic information is equivalent to showing a company's proprietary protocols as well as network topology.

The reason a huge majority of papers today use datasets such as the KDD'99 or DARPA datasets is because they are public and are useful to compare results to previous papers. The only way two ML models can be compared is if the same data is used for training and testing. Unfortunately, a lot of research papers use a subset of the larger KDD'99 or DARPA datasets to train and test - the subsets used could be different across papers, and thus comparing the performance of the models becomes even harder [Buczak18]. Thus, it is important to take all the results presented in Table 1 with a grain of salt - comparing the different models is useful to learn general trends, but the comparisons don't hold as much as weight as a variety of research papers claim. Aside from the two problems mentioned in this section about public datasets, they also suffer from data imbalance and data scarcity.

### 4.2 Data Imbalance, Data Scarcity, and Labeled Data

Data imbalance and data scarcity are issues that plague the IDS research field. Public datasets such as KDD'99 and have millions of records, but approximately only 1/10,000 data points is an attack [Amit19]. This level of imbalance leads to inflated measures of precision - since most of the data is considered normal, precision, by itself, is not a useful statistic [Kayacik05]. The imbalance of the datasets also leads to a data scarcity problem. Due to this, it is harder for a classifier to learn about the general behavior of normal data since there aren't enough malicious data points to make a strong distinction.

Another issue that IDS research faces is the lack of labeled data. However, labeling data is a difficult task given that terabytes of data are produced a day and is mostly proprietary [Buczak18]. Labeling data, today, is a manual process done by a field expert who must go through thousands of records and label each one. This is one of the huge reasons why the DARPA and KDD'99 datasets are the most popular choices for ML-based IDS research. Misuse-based detection requires labeled data for training, and anomaly-based detection can heavily benefit from it [Buczak18, Ahmad18]. Creating a dataset that is labeled and is a true representation of networks and attacks today is the most crucial step that needs to be taken to further this field of research. Data imbalance and data scarcity also lead to some kinds of attacks being less represented than others. Specifically, low-frequency attacks such as U2R and R2L.

## 4.3 Low-frequency Attacks

U2R and R2L attacks are the most difficult attacks to detect. This is for many reasons. First, R2L and U2R are the least represented attacks in the DARPA and KDD'99 datasets. Table 2 shows the breakdown of the different kinds of traffic represented in the NSL-KDD dataset - which is the more refined version of the KDD'99 dataset [Vinaykumar18]. Additionally, [Kayacik05] also analyzed the feature relevance of the 41 features presented in the KDD'99 dataset. Using the information gain of each feature, a common metric used in Decision Trees, for each attack, the research team found out that only one or two features were valuable to distinguish different kinds of U2R and R2L attacks. To compare, there were 10-12 features that were considered relevant to discriminate different DoS and scanning attacks. Additionally, the information gain from the one or two features that distinguished R2L and U2R attacks were as small as 0.0001, meaning their discriminatory power was small. This makes it harder, if not impossible, for a model to understand the general structure of low-frequency attacks using KDD'99 or NSL-KDD.

**Table 2:** NSL-KDD Dataset Breakdown [Vinaykumar18]

| Type of Attack | Training Set | Testing Set |
|---|---|---|
| Normal | 67343 | 9711 |
| DoS | 45927 | 7458 |
| Scanning | 11656 | 2421 |
| User to Root | 52 | 67 |
| Root to Local | 995 | 2887 |

Second, the feature statistics between a normal and a malicious connection attempting to perform a U2R or R2L attack are quite similar. Third, R2L and U2R attacks look quite similar and so it is hard to even differentiate between the two. As mentioned previously in Section 3.2, a U2R attack can only be done after an R2L attack since an unprivileged user cannot try to gain privileged access to a machine until they have made a local connection. Fourth, the feature subset that the KDD'99 dataset uses to identify a U2R attack may not always be how it is performed. For example, an attacker can complete a U2R attack in a single connection (by loading a malicious kernel module) while the KDD'99 dataset measures features such as num_failed_logins. If num_failed_logins is high, an alert is raised. However, in the case of loading a malicious kernel module, this feature may be equal to 0 or 1, at which point no alerts are raised [Mishra19]. This conclusion is corroborated by the research conducted by [Kayacik05]. Detecting U2R/R2L attacks require the IDS to analyze the 'content' or payload of a packet - something that the feature set in datasets like KDD'99 cannot describe [Sabhnani04]. Thus, a HIDS might be a more effective tool to detect U2R and R2L attacks. Since a HIDS can analyze a chain of system calls or changes in system state, it can have a better chance of detecting low-frequency attacks. This potential solution is explored in Section 6.1. The next subsection presents two issues that many shallow learning techniques face: the time required for re-training and classifying an incoming packet.

### 4.4 Re-training and Online Capabilities

ML models, especially complex ones like an ensemble or neural networks, require a long time to train as well as a long time to classify an out-of-sample record as anomalous or normal. ML is quite successful when the ML model is trained once, and then it continues to work as necessary. However, network data is quite dynamic and changes every day. This begs the question, how often should an ML-based IDS be re-trained? If the database of collected network flows is updated every day and the ML model must be re-trained each time, the amount of time that the ML model is online and can classify real-time incoming packets is quite small and makes having an ML-based IDS impractical. Most of the research surveyed in other papers, complete the learning offline and are then deployed [Mishra19, Buczak18]. But, if the data changes every day or if a new novel attack is learned, fast incremental learning or other online methods must be employed to make ML for IDSes a viable solution [Mishra19]. Additionally, most of the papers that are surveyed don't actually implement a real IDS and test it - instead the ML model is tested using a dataset [Ahmad18]. Thus, the hypothetical ML-based IDSes proposed in Section 3 fail to answer the questions: is the model feasible to use in a real intrusion detection system, and can it deal with data in real-time? Re-training and online capabilities are vital issues that must be answered for ML-based IDSes to become a part of industry today.

Section 4 presents a variety of issues that the ML-based IDS research field faces today. These include (1) old and non-representative datasets, (2) data imbalance, data scarcity, and lack of labeled data, (3) low detection rates of low-frequency attacks, and (4) the large amount of time required for re-training and classifying packets in real-time for ML models. The next section shows the latest research in trying to solve (1) and (2).

---

## 5 Solutions to Data Scarcity, Data Imbalance, and Labeled Data

Research in trying to generate a dataset that is not scarce, properly balanced, and labeled is crucial. Imbalanced datasets are a problem with a lot of the public datasets - most of the data points in the set are normal while a small number are considered attacks. Thus, models that use these datasets tend to see very high levels of accuracy, as explained in Section 4.1 and 4.2 and supported by [Kayacik05]. An imbalanced dataset also indirectly leads to data scarcity. In the case of DARPA and KDD'99, the imbalance of normal to abnormal data also results in a small number of data points that are labeled as abnormal. Finally, as mentioned in Section 4.2, having real traffic data that is labeled is still an open question and needs to be solved to implement powerful misuse-based IDSes. This section presents two papers: a paper that aims to generate new representative labeled datasets and a paper that finds a novel way to mitigate the effects of data scarcity and data imbalance.

## 5.1 Towards New Representative Datasets

Palo Alto Networks collaborated with Shodan to create new datasets that can be used for intrusion detection analysis. These datasets include system/network level data and also deal with the problem of the lack of labels. A dataset is provided for malware, host similarity, and network traffic or connections. The malware and host similarity datasets are labeled and can be useful for building HIDSes. The network traffic dataset is not labeled but can be used to learn the structure of stealth port scan attacks, which is a form of a scanning attack. The network traffic dataset can be used to build NIDSes. The main benefit of these datasets is that they have the proper ratio of imbalance and also consist of real network traffic, instead of synthesized traffic [Amit19].

There a variety of other datasets also presented in the paper such as the Android malware, n-grams, and bind shell datasets. The Android malware dataset can be used to build mobile IDSes. The n-grams dataset can be used to learn a model that can distinguish between benign and malicious files. Finally, the bind shell dataset could be used to better model R2L attacks [Amit19]. Although this paper shows the transition towards creating new datasets, there is a lot of room for improvement.

The paper also discusses the privacy concern for companies. By using anonymization, older data, and removing compromising information that is proprietary to the company, the privacy issue can be mitigated. This would, unfortunately, require a lot of work from the company. However, contributing companies can only benefit from a world of experts using their data to tackle cyber problems [Amit19]. The next subsection discusses a paper that deals with data imbalance and data scarcity in a unique way.

## 5.2 Data Augmentation

A novel way to mitigate the effects of data imbalance and data scarcity in the public datasets is by generating synthetic intrusion data. [Zhang19] does just this by creating a Data Augmentation (DA) module that is able to use a small dataset of real intrusion samples to create a larger dataset of augmented intrusion samples. The DA module contains two parts: (1) a Poisson-Gamma joint probabilistic model (PGM) and (2) a Deep Generative Neural Network (DGNN).

The PGM uses statistical techniques to try to figure out the distribution of a given dataset. Thus, given a dataset of x data points, the PGM tries to derive P(x). This is a major problem in network data since we don't know the underlying distribution. Once P(x) is resolved, the PGM component creates synthesized intrusion samples [Zhang19].

DGNNs are neural networks that perform adversarial learning. Adversarial learning is the process in which a machine learning model is fooled by being given spoofed input. Thus, the proposed DGNN has two components: G-net and the D-net. Both, the G-net and the D-net are Deep Neural Networks (DNN). The G-net tries to generate malicious inputs, which in this case are intrusion samples, while the D-net tries to identify the malicious inputs from the real inputs. As the model continues to run, G-net will generate better malicious inputs as D-net gets better at identifying them [Zhang19]. This battle allows the DGNN to create samples that, although augmented, comes close to how real intrusions look.

Once the PGM generates the synthetic data, the DGNN is fed both the synthetic and real intrusion data. The DGNN, through adversarial learning, will converge to create augmented intrusion data, which is close to real intrusion data. This augmented data is now mixed in with normal data. The benefit of the DA module is that now the ML model, either shallow or deep, has a dataset of an equal number of normal and malicious data points [Zhang19]. Figure 4 shows a high-level framework of the procedure described. In the paper, this DA module was used as an enhancement step before training an SVM and a DNN.

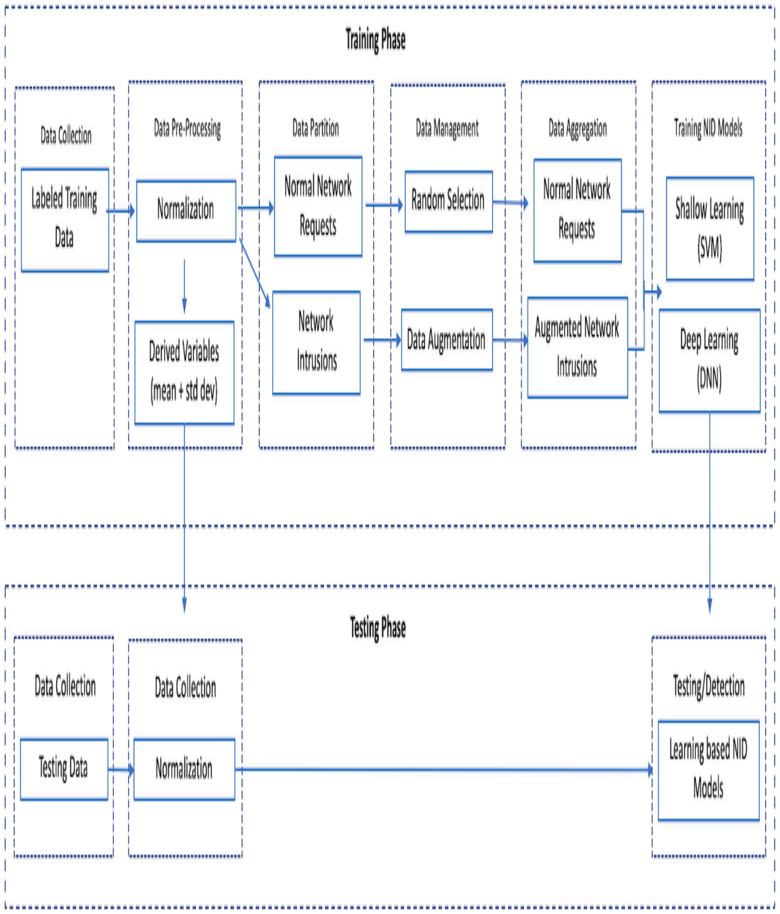

**Figure 4:** Proposed enhanced-NID framework using a Data Augmentation module [Zhang19]

Using this DA-enhanced IDS system, the research team found that the detection rate, precision, and F1-score is better than using classical ML techniques without the DA module. This is true for all kinds of attacks in the KDD'99 dataset (DoS, scanning, U2R, and R2L). Additionally, the DA-enhanced IDS with DNN as the learning model outperformed the DA-enhanced IDS with SVM - except for R2L attacks [Zhang19]. This paper shows a unique way to deal with data scarcity and data imbalance.

Section 5 discusses two ways that the latest research has tried to tackle issues of data scarcity, data imbalance, and labeled data. The paper presented by Palo Alto Networks and Shodan helps bridge this gap by providing a myriad of different datasets to better model malware, mobile malware, network traffic, and R2L attacks. On the other hand, [Zhang19] found a novel way to mitigate the issues of data scarcity and data imbalance by using a data augmentation module that is capable of generating augmented intrusion samples. The next section focuses on understanding how to better model and detect low-frequency attacks.

# 6 Improving Detection Rates of Low-frequency Attacks

There are two main ways that the latest research has dealt with a low detection rate of low-frequency attacks: a HIDS that monitors system calls and state changes and the use of deep learning. As elaborated in Section 4.3, it is hard to detect U2R and R2L attacks because the content of a packet must be analyzed, which is not provided by the feature set in popular datasets like KDD'99. A HIDS can help solve this problem by monitoring the activity of a host to see how the system reacts to malicious packets. For example, if a chain of system calls is executed that would give root access to an illegitimate user or delete sensitive information, an alert is raised to notify the administrator of a potential attack. Section 6.1 discusses a paper that develops a HIDS for a cloud environment to detect host-level attacks such as R2L and U2R attacks.

Deep learning can also be used. A typical example of deep learning is a Deep Neural Network (DNN), which, on a high level, is a more complex Artificial Neural Network (ANN). The main difference is that a DNN tries to progressively obtain lower-level features of a dataset at every level of the neural network. This is a general trait of deep learning methods where the featurse set is something that is learned, instead of being pre-specified. This helps to solve the problem that many shallow learning methods face: a feature set that works well for one kind of attack might not work well for another. Deep learning is better at finding the optimal feature set and thus lends itself well to learning a multi-classification problem. This, in turn, allows it to better classify low-frequency attacks such as U2R and R2L attacks [Rawat19, Martin-Lopez20]. Section 6.2 presents a paper that uses a DNN in a computationally cheap fashion for better detection rates.

## 6.1 Program-Behavior Analysis

U2R and R2L attacks are difficult to analyze at a network level, given the public datasets that are commonly used. However, a HIDS lends itself well to this problem since it can be used to analyze the state of a system and monitor if illegitimate users try to access an end system or manipulate sensitive information on it. [Mishra16] present a novel solution that can be implemented in a cloud environment to monitor activities on Virtual Machines (VMs) running in that cloud. The paper proposes an approach named Malicious System Call Sequence Detection (MSCSD). This approach analyzes a program's behavior at runtime to see if there is a sequence of system calls that can be malicious.

During the training phase, user programs such as sendmail, login, xlock, Perl script run and the chain of system calls are captured using the strace tool. The system calls are then converted into an array of n-grams where each n-gram is a unique sequence of system calls of size n. The matrix of n-gram arrays for each running program in different execution scenarios is stored as a baseline database and is used to train the classifier [Mishra19]. The classifier of choice for the paper was a Decision Tree. The learned classifier is called the Decision Model.

Using the baseline database, the Decision Model is able to learn what the chain of system calls should look like at runtime for all the different applications measured. During the testing phase, the University of New Mexico (UNM) dataset, a dataset filled with U2R and R2L attacks, is used to provide system call traces for applications such as sendmail, xlock, etc. The Decision Model is then responsible for classifying the data points in the UNM dataset as malicious or benign. The main contribution of this paper was the MSCSD approach, which uses novel pre-processing and storage of application runtime data to allow a machine learning model to understand the system call patterns of privileged applications running on VMs [Mishra16]. Figure 5 shows the framework of the MSCSD-enhanced HIDS. The detection rate of MSCSD-enhanced DT was between 72%-99% for the different kinds of R2L and U2R attacks in the UNM dataset [Mishra16]. The next subsection discusses a deep learning solution for improved detection of R2L and U2R attacks.

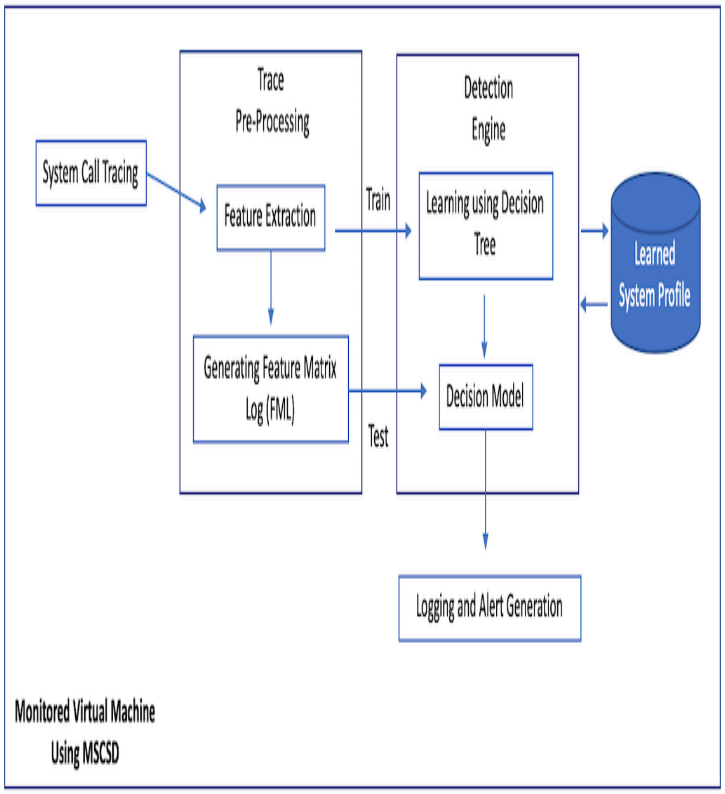

**Figure 5:** Proposed framework of the MSCSD-IDS [Mishra16]

## 6.2 Long Short-Term Memory Network

A paper submitted recently describes a deep learning model based on the concept of the Long Short-Term Memory (LSTM) network. The model tries to tackle three main issues: the multi-classification problem for shallow ML models is difficult, low-frequency attacks are hard to classify, and it is hard to model sequential samples based on temporal features between different connections [Yuan18].

The model goes through a few main steps (see Figure 6):

1. The NSL-KDD training dataset is preprocessed and normalized to mitigate the effect of very large feature values.
2. PCA is used to reduce the dimensionality of the feature set.
3. The LSTM model is then trained on the PCA dataset.
4. The LSTM model is then tested on the test NSL-KDD dataset.

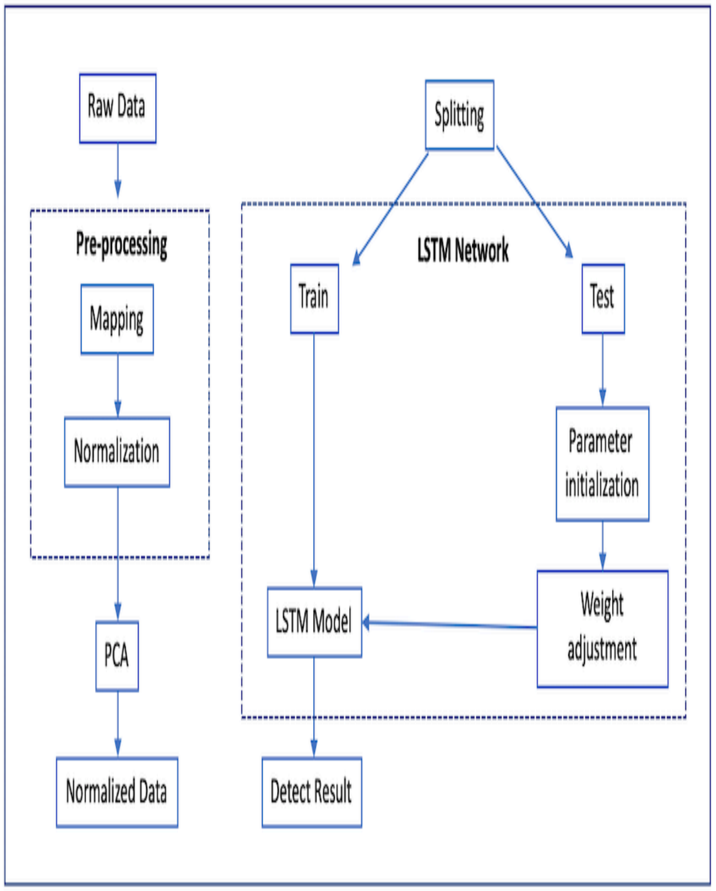

**Figure 6:** Proposed framework of the LSTM network [Yuan18]

The LSTM network is based on the Recurrent Neural Network (RNN) model. The RNN model is based on using knowledge from previous data points - thus, the decision on the current input is based on previous data. RNNS are useful for analyzing network flows since they can analyze temporal characteristics of the traffic. However, the main problem with the RNN model is that the amount of previous data that can be used for future analysis is unbounded and can lead to degradation in performance. To tackle this, the LSTM network is proposed where the amount of previous knowledge that can be used is filtered. If incoming data is considered useless, it is discarded and won't be used in future analysis. Thus, the long-term dependence issue of the RNN is solved [Yuan18].

The LSTM network results show that it is better at solving the multi-classification problem, compared to other classifiers such as SVM, RF, and K-nearest neighbors. It is able to achieve a higher detection rate for all kinds of attacks (DoS, scanning, U2R, and R2L) compared to the classic shallow ML models. It is important to note that the detection rate for U2R attacks is still mediocre (50.7%), while the detection rate for R2L is better than average (80.2%) [Yuan18]. Also, the LSTM network generalizes well to out-of-sample data points and is able to predict attacks that were not part of the training dataset. This is because of two reasons. First, the LSTM network is able to learn temporal patterns in the dataset which regular ML models are unable to do [Yuan18]. Second, the LSTM network is an anomaly-based detector and was thus able to learn a strong target function that was able to find unknown anomalies that weren't present in the training dataset.

In this section, we presented two novel ways of detecting U2R and R2L attacks. Since these kinds of attacks depend on the payload or content of a packet/network flow, it is essential to find novel techniques that can properly classify them. This section provides two possible solutions: a HIDS that monitors system call sequences and fast incremental deep learning. Monitoring system calls allow a HIDS to realize when an attacker is attempting privilege escalation or trying to access sensitive information. Deep learning can also be used since it has better generalization properties and is better at learning a multi-classification problem. However, it is important to note that, regardless of the technique, identifying low-frequency attacks is very difficult since they look so much like regular traffic [Mishra16]. The next section shows novel ways to create an online ML-based IDS.

---

# 7 Online Learning

Two issues with a lot of the recent research on IDSes is that the ML models of interest are not implemented in a real IDS and also how long re-training and maintaining the IDS takes. Most papers discuss the efficacy of their ML model using offline learning where the model is trained offline and is then tested on the test dataset (the ML model does not ever encounter real traffic data). This results in two problems: (1) the model must be re-trained to deal with new data or attacks, and (2) the classification task must be very fast to deal with the speed of incoming traffic. Online learning is a solution that tries to solve the above problems. To be an online solution, the ML model must be able to learn on-the-fly and also be able to classify an incoming packet quickly. The two papers discussed in this section provide novel solutions for online learning - one uses computationally cheap techniques (Section 7.1), and the other uses distributed computing (Section 7.2).

## 7.1 Kitsune

Kitsune is a lightweight NIDS solution that uses an ANN to perform online processing. ANNs are computationally expensive since learning a large feature set requires a lot of time, which is not feasible in an online setting. To improve this, Kitsune breaks apart a single ANN into an ensemble of smaller autoencoders. An autoencoder, given an input x, tries to recreate the same x. Autoencoders are useful to understand the representation of the data provided - thus, the algorithm is unsupervised and does not need labels to perform classification. Kitsune provides three main benefits: it can perform online processing, it is an unsupervised learning method, and it has low complexity - since the training and classification are done by an ensemble of autoencoders, each with low complexity [Mirsky18].

Kitsune is lightweight enough that it does not need to be run on state-of-the-art machines - the experiments ran by the team were on a Raspberry Pi with a single core. Kitsune was able to achieve this by limiting each autoencoder to having a maximum of three layers and a small number of neurons [Mirsky18]. Therefore, Kitsune takes away the complexity that is inherent in deep learning algorithms by spreading out the work between many autoencoders, each with a small computational footprint. More specifically, the framework breaks down a traffic instance x into a representative set of vectors v. Each vector in v is sent to a specific autoencoder for processing. After processing, the instance x is discarded. Each autoencoder is only responsible for learning a small subset of the features, making it computationally cheaper than a single ANN [Mirsky18]. Figure 7 shows the framework of Kitsune.

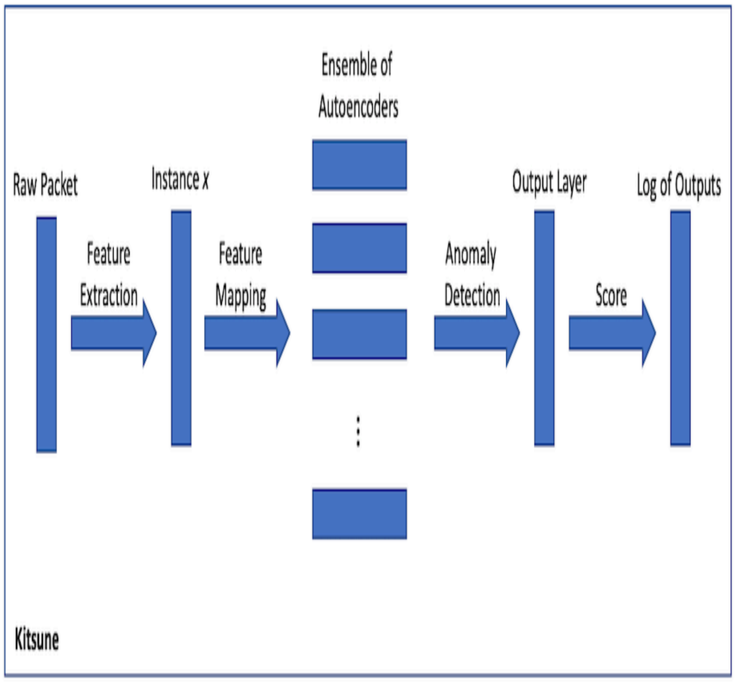

**Figure 7:** Proposed framework of Kitsune [Mirsky18]

Kitsune has a training and an execute mode. In training mode, each packet is processed by the ensemble of autoencoders, but there is no output. In execute mode, each packet is processed, and an output is given to notify the system if the packet is anomalous or not. The team does point out that Kitsune is vulnerable during the training phase. During the training phase, Kitsune treats all packets as benign. Thus, a preexisting attacker will be able to circumvent Kitsune and attack the network. Kitsune is also vulnerable to DoS attacks [Mirsky18].

Kitsune was tested in a simulated environment with real IP cameras, IoT devices, and PCs. Attacks such as scanning, DoS, man-in-the-middle, and botnet malware were performed. Kitsune was compared to a baseline online NIDS and offline algorithms such as Isolation Forests and Gaussian Mixture Models. Kitsune outperformed the online model and was competitive against offline algorithms, which will naturally perform better since they have more time to learn [Mirsky18]. Kitsune, however, was able to achieve these results on a Raspberry Pi with a single core. The next subsection presents an IDS that is run in a distributed system to spread the workload.

### 7.2 Scale-hybrid-IDS-AlertNet

Scale-hybrid-IDS-AlertNet is a hybrid IDS (both a HIDS and NIDS) that runs on a distributed system of commodity hardware to process network data in real-time [Vinaykumar18]. This result is achieved by successfully dealing with two challenges. First, most of the papers discussed in this survey have tried to create an ML model that can be used only in a NIDS or HIDS. The proposed architecture is able to deal with both host and network data in real-time. This means that the system can analyze system call traces, packet data, and much more. Second, a DNN suffers from the necessity to have an immense amount of data to realize low error rates and good generalization. However, processing large amounts of data requires a lot of time - which is not feasible for online systems. To deal with this, the architecture is run on a distributed system of commodity hardware - hardware that is easily accessible and relatively cheap. The distributed structure of the system is proprietary and is not discussed in the paper. However, it uses software that handles big data, such as Apache Spark and Hadoop [Vinaykumar18].

The proposed system is able to outperform classical ML algorithms, and also has a strong detection rate of attacks such as U2R and R2L. The better U2R and R2L detection rate can be attributed to the fact that the hybrid IDS can also analyze system call traces. The improved performance is an expected result since it is known that DNNs will perform better compared to classical ML algorithms, given enough data. However, this result is realized in real-time for both system and network-level data in a distributed manner [Vinaykumar18]. This paper hints at the possibility that a lot of the issues that machine learning detection systems face can be mitigated through multiprocessing and big data tools. Multiprocessing helps increase the computational speed of training or classification, and the big data tools can help manage the influx of data in a high-speed network.

---

## 8 Summary

As cyber-attacks continue to become more sophisticated, it is important to create a line of defense that is capable of identifying and reacting to them. Intrusion Detection Systems have come to the forefront of security research as one of the possible solutions. Machine learning has become the main approach to achieve this since it allows the system to understand the structure of network traffic and make novel predictions about potential attacks entering a network or machine. This survey provides a systematic look at the current face of applying ML for IDSes.

The novel contributions of this paper are as follows: (1) a breakdown of the current issues that plague this research field today and (2) the latest solutions to counter and mitigate these issues.

The paper begins with a high-level introduction to IDSes, machine learning, datasets used for evaluation, and attacks of interest. Then, a brief discussion is provided to present a variety of shallow learning techniques that were used for IDSes and their performance. The three takeaways were that (1) no single shallow learning technique is able to solve the multi-classification problem and properly identify all the different kinds of attacks, especially low-frequency attacks, (2) the datasets used for evaluation are afflicted with problems of data scarcity, data imbalance, and require labels and (3) none of the techniques employed are feasible to be used in a real IDS due to long training time and inability to provide online capabilities.

The latest solutions to these problems are then presented. Deep learning has now come to the forefront of research since it is capable of solving the multi-classification problem and has better generalization. Additionally, deep learning, if deployed in a distributed or computationally cheap fashion, can be used for real intrusion systems and realize great success. The necessity to find representative datasets is still an open question, but the datasets provided by Palo Alto Networks and Shodan is a step in the right direction. A novel way of mitigating the effects of data scarcity and imbalance through the use of deep adversarial learning is also presented.

Creating a reactive defense system is essential to solve the cyber crisis that we are in today. Current solutions like firewalls and honeypots, although useful, are insufficient to deal with sophisticated attackers. Dealing with the current issues for ML-based IDSes and finding novel ways of applying deep learning and distributed computing will be the necessary stepping-stones before a machine learning-based intrusion detection system can be deployed in the industry today.

---

## 9 References

1. [IndustryArc19] IndustryARC, "Cyber Security Market - Forecast (2019-2024)", IndustryARC, 2019, https://www.industryarc.com/Report/15646/cyber-security-market.html
2. Bernard17] Tara Bernard, "Equifax Says Cyberattack May Have Affected 143 Million in the US", https://www.nytimes.com/2017/09/07/business/equifax-cyberattack.html, this article discusses the impact the Equifax hack had on 143 million users
3. [Isaac18] Mike Isaac, "Facebook Security Breach Exposes Accounts of 50 Million Users", https://www.nytimes.com/2018/09/28/technology/facebook-hack-data-breach.html, this article explains the impact the Facebook breach had on 50 million users
4. [Ahmad18] Bilal Ahmad, "Role of Machine Learning and Data Mining in Internet Security: Standing State with Future Directions," Journal of Computer Networks and Communications, 2018, pp. 11, https://www.hindawi.com/journals/jcnc/2018/6383145/
5. [Boutaba18] Raouf Boutaba, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," Journal of Internet

Services and Applications, 2018, pp. 99, https://jisajournal.springeropen.com/articles/10.1186/s13174-018-0087-2

6. [Mishra19] Preeti Mishra, "A Detailed Investigation and Analysis of Using Machine Learning Techniques for Intrusion Detection," IEEE Communications Surveys & Tutorials, 2019, pp. 43, https://ieeexplore.ieee.org/document/8386762
7. [Roesch99] Martin Roesch, "SNORT - Lightweight Intrusion Detection for Networks," 13th Systems Administration Conference, 1999, pp. 11, https://www.usenix.org/legacy/event/lisa99/full_papers/roesch/roesch.pdf
8. [Buczak18] Anna Buczak, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," IEEE Communications Surveys & Tutorials, 2018, pp. 24, https://ieeexplore.ieee.org/document/7307098
9. [Kunal19] Kunal, "Machine Learning Approach to IDS: A Comprehensive Review," 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA), 2019, pp. 5, https://ieeexplore.ieee.org/document/8822120
10. [Apruzzese18] Giovanni Apruzzese, "On the Effectiveness of Machine and Deep Learning for Cyber Security," 2018 10th International Conference on Cyber Conflict, 2018, pp. 20, https://ieeexplore.ieee.org/document/8405026
11. [Shung18] Koo Ping Shung, "Accuracy, Precision, Recall or F1?", https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9, this article presents the differences between precision, recall, and F1-score
12. [Sangkatsanee11] Phurivit Sangkatsanee, "Practical real-time intrusion detection using machine learning approaches," Computer Communications, 2011, pp. 8, https://www.sciencedirect.com/science/article/pii/S014036641100209X
13. [Amit19] Idan Amit, "Machine Learning in Cyber-Security: Problems, Challenges, and Data Sets," The AAAI-19 Workshop on Engineering Dependable and Secure Machine Learning Systems, 2019, pp. 8, https://arxiv.org/abs/1812.07858
14. [Kayacik05] Gunes Kayacik, "Selecting Features for Intrusion Detection: A Feature Relevance Analysis on KDD 99 Intrusion Detection Datasets," Third Annual Conference on Privacy, Security and Trust, 2005, pp. 6, https://www.researchgate.net/publication/220919984_Selecting_Features_for_Intrusion_Detection_A_Feature_Relevance_Analysis_on_KDD_99
15. [Sabhnani04] Maheshkumar Sabhnani, "Why Machine Learning Algorithms Fail in Misuse Detection on KDD Intrusion Detection Data Set," Intelligent Data Analysis, 2004, http://www.cs.cmu.edu/~sabhnani/PDF/ida04.pdf
16. [Zhang19] He Zhang, "Deep Adverserial Learning in Intrusion Detection: A Data Augmentation Enhanced Framework," 2019, pp. 10, https://arxiv.org/pdf/1901.07949v3.pdf
17. [Rawat19] Shisrut Rawat, "Intrusion detection systems using classical machine learning techniques versus integrated unsupervised feature learning and deep neural network," 2019, pp. 9, https://arxiv.org/pdf/1910.01114.pdf
18. [Martin-Lopez20] Manuel Martin-Lopez, "Application of deep reinforcement learning to intrusion detection for supervised problems," Expert Systems with Applications, 2020, pp. 24, https://www.sciencedirect.com/science/article/pii/S0957417419306815
19. [Mishra16] Preeti Mishra, "Securing Virtual Machines from Anomalies Using Program-Behavior Analysis in Cloud Environment," 2016 IEEE 18th International Conference on

High Performance Computing and Communications, 2016, pp. 8, https://ieeexplore.ieee.org/document/7828482
20. [Yuan18] Quizhuang Yuan, "Aligning Network Traffic for Serial Consistency and Anomalies with a Customized LSTM Model," IET Intelligent Transport Systems, 2018, pp. 5, https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8706275
21. [Mirsky18] Yisroel Mirsky, "Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection," Network and Distributed Systems Security Symposium 2018, 2018, pp. 15, https://arxiv.org/pdf/1802.09089.pdf
22. [Vinaykumar19] R. Vinaykumar, "Deep Learning Approach for Intelligent Intrusion Detection System," IEE Access, 2019, pp. 26, https://ieeexplore.ieee.org/document/8681044

---

## 10 Acronyms

- IDSes: Intrusion Detection Systems
- IDS: Intrusion Detection System
- HIDS: Host Intrusion Detection System
- NIDS: Network Intrusion Detection System
- ML: Machine Learning
- PCA: Principal Component Analysis
- SVM: Support Vector Machine
- DNN: Deep Neural Network
- Pcap: Packet Capture
- DARPA: Defense Advanced Research Projects Agency
- KDD'99: Knowledge Discovery and Data Mining 1999
- R2L: Root to Local
- U2R: User to Root
- DoS: Denial of Service
- DT: Decision Tree
- ANN: Artificial Neural Network
- DA: Data Augmentation
- PGM: Poisson-Gamma joint probabilistic model
- DGNN: Deep Generative Neural Network
- MSCSD: Malicious System Call Sequence Detection
- VMs: Virtual Machines
- LSTM: Long Short-Term Memory

---

Last Modified: December 10, 2019
This and other papers on latest advances in computer networking are available on line at
http://www.cse.wustl.edu/~jain/cse570-19/index.html
Back to Raj Jain's Home Page