BENCHMARKS FOR EVALUATING ANOMALY-BASED INTRUSION DETECTION SOLUTIONS

Nicholas J. Miller and Mehrdad Aliasgari

Department of Computer Engineering and Computer Science, California State University, Long Beach, Long Beach 90840, USA.

ABSTRACT

Anomaly-based Intrusion Detection Systems (IDS) have gained increased popularity over time. There are many proposed anomaly-based systems using different Machine Learning (ML) algorithms and techniques, however there is no standard benchmark to compare them based on quantifiable measures. In this paper, we propose a benchmark that measures both accuracy and performance to produce objective metrics that can be used in the evaluation of each algorithm implementation. We then use this benchmark to compare accuracy as well as the performance of four different Anomaly-based IDS solutions based on various ML algorithms. The algorithms include Naive Bayes, Support Vector Machines, Neural Networks, and K-means Clustering. The benchmark evaluation is performed on the popular NSL-KDD dataset. The experimental results show the differences in accuracy and performance between these Anomaly-based IDS solutions on the dataset. The results also demonstrate how this benchmark can be used to create useful metrics for such comparisons.

KEYWORDS

Anomaly-based Detection, Intrusion Detection, Benchmarks

1. INTRODUCTION

Cyber security attacks are the most profitable they have ever been. In ransomware attacks alone, more than five billion dollars were lost in 2017, a 15,000% increase from 2015 [1]. Equifax, one of the three credit bureaus in the United States, lost the personal data for at least 145 million Americans, with more than 209,000 credit card numbers also lost [2]. As more and more data becomes Internet-facing, more lax and underfunded security systems are being exposed. Vulnerabilities in network defenses, inadequate security policies, and lack of cybersecurity education mean these adversaries have multiple avenues to attack most networks. By connecting to the Internet, any network is taking on the risk of exposing itself to infiltration by nefarious actors. Malware like ransomware is hard to defend against, as newer iterations are released constantly, resulting in "zero-day" exploits that standard anti-viruses are not ready to catch [3].

An Intrusion Detection System (IDS) is a system that monitors network traffic and flags potential intrusions by adversarial actors. There are two ways that an IDS can detect these intrusions, by using signatures or by detecting anomalies. Signature-based systems compare network traffic to the "signatures" of recognized attacks, if a pattern lines up with one of these signatures, it is flagged as an attack. These systems are familiar to most users, they are anti-viruses that scan systems and look for known threats. Signature detection accuracy is reliably high on known attacks, however they can only detect attacks after they have been discovered. With the current Internet landscape, catching zero-day exploits is a priority and it is not reasonable to wait until an attack signature has been discovered and uploaded [3]. Anomaly-based systems, however, use Machine Learning (ML) to classify network traffic as either normal or anomalous. Anomaly-based IDSs are trained on labeled network data and learn patterns in the data to enable them to classify new novel data. They focus on learning "normal" patterns, and everything that is not

recognized is anomalous. This allows the classifiers to identify new attacks, since they flag anything abnormal [4].

In an anomaly-based system, a classifier is trained that can read input data and determine a label, in this case normal or anomalous. This classifier is trained using a dataset that is similar to what it will be monitoring, and this dataset must contain both normal and anomalous data. This allows the classifier to recognize patterns in the data, and with enough examples it can classify novel data based on what it learned in training. The strength of the classifier is tested by getting its accuracy on a testing dataset, which should look similar to the training set but contain new data. Different algorithms approach this in unique ways, however the base concepts of training and classification are the same [5].

The primary advantage of anomaly-based IDSs is that, trained correctly, they should be able to recognize never-before-seen attacks (zero-days) as they happen. The problem is there is a lot less confidence in their labels, since they are essentially guessing from patterns in the data. If the training set is not sufficient, or the underlying classifier is implemented incorrectly, the results can be severely unreliable [6]. Another issue is the vast selection of algorithms makes it hard to determine which is actually the most appropriate to use.

There are myriad papers about different algorithms and their success in anomaly-based IDSs. However, each paper trains and tests their algorithm differently. Different papers will use their own datasets, and if they do use the same dataset they often use different subsets as the training and testing sets. This makes comparisons meaningless, the accuracy of each could be explained by factors outside of the algorithms implementation itself (such as the difficulty of the test set) [5].

Along with the inconsistencies in datasets, there are also no uniform metrics being used. While most papers will try to emphasize the âAIJaccuracyâAI of their algorithm, this word could mean different things depending on the method. Some papers describe the accuracy on a small test set, or the average accuracy across many test sets and on specific attacks, and some even describe the accuracy as the percentage correct when classifying the training set itself [7, 8, 9, 10]. Along with these issues, a singular accuracy score is not enough to determine the strength of an algorithm, particularly of this nature. It is important to see a confusion matrix, since we need to know if it is more likely to have false negatives than false positives, or to see the ratio of true positives to true negatives, and to calculate other metrics [11]. Accuracy based on attack type is crucial as well, since a classifier may get one hundred percent accuracy with one attack but completely fail with another [11].

The goal of this work is to address the above issues by creating a singular benchmark that will serve as a basis to compare different algorithms. By creating a benchmark that trains and tests on the same data, and collects the same metrics, this should ensure that each algorithm can be compared objectively and provide meaningful results. The benchmark should be consistent, and capture useful information about each algorithm. It should also be extensible, and benchmarking new algorithms should be straightforward and easily allow for more future comparisons. To demonstrate this, four different ML algorithms are benchmarked on the NSL-KDD dataset, a large dataset used in the evaluation of anomaly-based IDSs.

The remainder of this paper is organized as follows: Previous Work contains a survey of related work in anomaly-based IDS research. Materials and Methods include information about the dataset, algorithms and proposed work for the benchmark. Results and Discussion describes the observations of the benchmark. Finally, the Conclusion is a summary of the findings in this work.

2. PREVIOUS WORK

The impetus for this paper comes from a survey on current machine learning algorithms used in anomaly-based IDSs [5]. This survey gives an overview of each of the common algorithms and cites several papers that implement each. For future work, the authors pointed out that comparing these algorithms methodically is not possible right now, as "If one were to compare the accuracy of several ML/DM methods, those methods should be trained on exactly the same training data and tested on exactly the same testing data. Unfortunately, even in the studies that used the same data set (e.g., KDD 1999), when they compared their results with the best methods from the KDD Cup (and usually claimed their results were better), they did so in an imperfect fashion–they used a subset of the KDD data set, but not necessarily the same subset that the other method used. Therefore, the accuracy of these results is not comparable" [5].

There are several other surveys that are relevant to this paper. A survey by Dhanabal and Shantharajah [12] implements several algorithms on the NSL-KDD dataset, however the focus of that survey was testing the strength of the dataset, not the algorithms. The authors used WEKA to test three different classifiers on the dataset and noted how well the dataset performed. They found that using all forty-one features was not ideal for training and classification, however they stated that it was a clear improvement over KDD'99 [12].

The survey by Garcia-Teodoro et al. [4] raises similar concerns about the ability to benchmark and assess anomaly IDS algorithms. The authors surveyed common algorithms and pointed out the issues with comparing their efficacy. The authors note that a primary issue in benchmarking is the lack of an effective test bed. They reiterated common issues with KDD'99, and mentioned the incredible difficulty to create and share a modern dataset of similar size [4].

Shiravi et al. [6] set out to address the issue of the lack of datasets in network intrusion detection research. They state that current data sets for testing are kept secret for privacy concerns, or are out of date or poorly made. They proposed a set of guidelines to ensure that datasets used in testing are valid, and also created a method to generate different types of traffic that fell under these guidelines [6].

Damopoulos et al. [13] recognized the importance of IDSs for mobile devices and tailored profiles for each user in order to create an effective anomaly-based IDS for misuse. They benchmarked several algorithms on a dataset of iPhone activity that they constructed in-house and recorded their results. The authors found that they could detect misuse with a high degree of accuracy. They also gathered several useful metrics for each algorithm used in their mobile IDS. Their primary focus was creating an IDS that would work with their dataset and targeted misuse specifically, however this paper is proposing a benchmark that can be used in future research [13].

From these surveys and research, we were able to find papers that proposed implementations for specific algorithms in anomaly-based IDS solutions. There are many published research papers in this area, but we selected strong papers that were clear about their implementation on NSL-KDD or KDD'99, and recreated them for the experiment. We chose one paper to represent each of the most common algorithms to demonstrate the benchmark. Background on the specific algorithms implemented is provided in the next section.

Mukherjee and Sharma [10] implement Naive Bayes on NSL-KDD and use feature selection algorithms to reduce the number of input features. This simplifies the model and makes it more understandable, and they posit it does not reduce the accuracy. Using the Feature-Vitality Based Reduction Method, they reduced the features to twenty- five. The authors use confusion matrices and accuracy to demonstrate the performance of their classifier, which they claim has ninety-

eight percent accuracy on the testing set. Crucially, they only used a subset of the NSL-KDD training and testing sets [10], while this benchmark will use the entire dataset.

Novikov et al. [9] use several neural networks to create an anomaly-based IDS on the KDD'99 dataset. Although they used the KDD'99 dataset rather than NSL-KDD, they preprocessed the data in a similar fashion to reduce duplicate records. The authors trained the classifiers in different ways, focusing on subsets of attacks at first and then eventually testing on unseen attacks. For data preprocessing, they removed duplicate rows and rebalanced the attacks, making the data more like NSL-KDD. They found that both the Multi-layer Perceptron and Radial Basis Function network performed well under these conditions, and compared their results to the previous cup winners [9].

For clustering, Kumar et al. [14] use the entire set of input features for NSL-KDD, however numbers are normalized to a range of zero to one. Using a technique called Simple k-means clustering with four clusters, the centroids of each cluster were calculated. The authors used a twenty percent subset of the training and testing sets from NSL-KDD, and found that the clusters accurately grouped records into attacks and normal behavior [14].

Calix and Sankaran [15] used feature reduction and several different kernels to experiment with support vector machines (SVMs) on the NSL-KDD dataset. The authors found that using a reduced feature set of nineteen features gathered from the information gain feature ranking algorithm negligibly reduced accuracy. Feature reduction for SVMs is critical because the SVM algorithm often requires a large memory overhead [16]. They also found that out of all the kernels, Radial Basis Function (RBF) had the highest accuracy and was the quickest, however they only used a subset of NSL-KDD for training and testing [15].

There are much more research work that describes new ways to create anomaly-based IDS solutions [17]. However, the aforementioned research papers are strong candidates and represent a broad spectrum of different types of ML algorithms used in such IDS systems.

3. MATERIALS AND METHODS

In order to create a benchmark, a uniform dataset for training and testing is necessary. As mentioned earlier, one of the biggest obstacles to comparing algorithms is the inconsistency in data. A single dataset must be used, and it needs to be a dataset that accurately represents real-world implementations [4, 5].

The KDD Cup 99 dataset (KDD'99) is a widely used dataset for benchmarking IDS solutions [18]. It contains network traffic generated by the MIT Lincoln Lab in 1998 and was used in the 1999 KDD Competition as a dataset for building an IDS [19]. The data was generated in 1998 and is unlikely that training an IDS on this data would be useful with modern attacks. However, it accurately represents the scale and attack variety that is necessary for a strong IDS. With its size and reputation, it is an almost ideal dataset for a benchmark. According to Lazarevic et al. [11] and McHugh [19] the dataset contains several issues that must be addressed, such as many duplicate rows and skewed proportions for attacks. The size of the dataset also means that a subset of the training and testing dataset must be used; incorporating all of the data is too impractical for current hardware systems [20].

The NSL-KDD dataset is a subset of the KDD'99 dataset [21]. It was created to solve some of the inherent issues with the former dataset, primarily by drastically decreasing the number of duplicate rows. The NSL-KDD dataset is significantly smaller than KDD'99, which means that it is not necessary to partition the data into smaller testing and training sets as they are already small enough to be handled by most classifiers. This is essential for the issue this paper seeks to

resolve, since we can use the entire training and testing sets for NSL-KDD, ensuring that the classifiers are seeing the same data [20]. The NSL-KDD dataset has become a standard of its own since its release, and has been used in many papers in place of the former dataset [21].

The NSL-KDD dataset is still not an accurate real-world dataset, since it is a subset of the KDD'99 data. Any classifier trained on this dataset and then used in a real network is unlikely to perform well. This does not change the fact that it is still an effective benchmark for IDSs. Any anomaly-based IDS should be trained on network traffic from the network in which it will be used, since every network is going to have different patterns [22]. Also, the NSL-KDD dataset still has similar data to a real network and classifiers will face the same issues (different attack types, never-before-seen attacks), and is the most complete dataset of this type publicly available, which is why it is so commonly used [23].

This paper will use the NSL-KDD dataset, however the benchmark would work with any dataset that conforms to these rules. There are newer datasets containing more modern attacks, such as the UNSW-NB15 dataset generated for the Australian Centre for CyberSecurity [24]. This dataset contains nine attack types and has a training set with one hundred seventy-five thousand records and a testing set with eighty- two thousand records. There is also the HTTP CSIC dataset generated for the CSIC (Spanish Research National Council) in 2010 as a response to some of the criticisms of KDD'99 [25]. The dataset contains HTTP requests, thirty-six thousand of which are normal requests and more than twenty-five thousand that are anomalous [25]. Both of these datasets may be more applicable for certain use cases, however they are not as ubiquitous as NSL-KDD. For the demonstration of the benchmark, NSL-KDD is ideal since there are many papers that describe their implementations on this dataset specifically.

THE DATASET

NSL-KDD is a large dataset of network traffic, and is already partitioned into a training set and testing set. There are forty-one attributes describing network traffic. Each record is also labeled with an attack that falls into one of five categories, the number for each is recorded in Table 1 [26].

1. Normal: Normal network traffic is anything that is not classified as an attack. This is one of two labels in a binary classification (normal/anomalous).
2. Probe: Probing attacks involve some sort of surveillance or method to gain information about a network, particularly for circumventing security controls. For example, port scanning. Attacks in the dataset include: ipsweep, mscan, nmap, portsweep, saint, satan.
3. DoS: Denial-of-Service attacks involve exhausting the resources of a network, often through aflood of meaningless requests. Attacks in the dataset include: apache, back, land, mailbomb, Neptune, pod, processtable, smurf, teardrop, upstorm.
4. User to Root (U2R): These attacks involve an adversary with a normal user account somehow
   gaining root privileges by exploiting vulnerabilities. Attacks in the dataset include: buffer_overflow, load module, perl, rootkit, ps, sqlattack, xterm.
5. Remote to Local (R2L): These attacks consist of a remote attacker somehow getting access to a local machine on the network. Attacks in the dataset include: ftp_write, guess_password, imap, multihop, named, phf, send mail, snmpgetattack, snmpguess, warezmaster, worm, xlock, xsnoop, http-tunnel.

Dataset	Normal	Probe	DoS	U2R	R2L
Training	67343	11656	45927	52	995
Testing	9710	2421	7458	200	2754

Table 1: Number of records in the NSL-KDD dataset, split between the five categories of attacks.

3.1 ANOMALY DETECTION BACKGROUND

While there are many different machine learning algorithms used in IDSs, this paper will implement four in order to demonstrate the benchmark. This is not an exhaustive list of algorithms, and simply provides a brief introduction to each concept.

NAIVE BAYES

Naive Bayes classifiers are a subset of Bayesian classifiers, which are statistical classifiers that rely on Bayes Theorem [10]. The classifier simply picks the label with the highest probability, given the input features. The naive portion of the classifier is that it assumes a strong independence between attributes. Essentially it assumes the probabilities for each of the input features are independent of each other. Intuitively this is not usually the case, however in practice the assumption greatly reduces computation without sacrificing performance [5, 10].

NEURAL NETWORKS

Neural networks are classifiers inspired by the brain. They use connected layers of neurons that "fire" when thresholds are met. There is usually an input layer (of features) and an output layer (the result/classification), with one or more hidden layers of neurons between them. Neural networks are trained in cycles, sometimes referred to as "epochs", where each neuron's activation function is tuned to increase accuracy. Neural networks are often considered "black boxes", since the individual neurons are tuned without any input from the user [27].

K-MEANS CLUSTERING

K-means clustering is a technique that clusters data points based on their similarity. It is an unsupervised learning technique, meaning the data does not have labels during training. There are many different techniques for clustering, but in k-means clustering a mean vector is calculated for each cluster [28]. The total number of clusters can be set beforehand, and the goal is to maximize the similarity between vectors inside each cluster, while minimizing similarities across clusters [28].

SUPPORT VECTOR MACHINES

Support Vector Machines (SVMs) are vector-space classifiers that aim to find a hyperplane to separate classes, using support vectors. In the vector-space model, each training record is interpreted as an n-dimensional vector, where n is the number of input features. The support vectors are training vectors close to the border between the classes and are used to create a hyperplane that maximizes the distance between the training vectors and itself [16].

The surface itself is calculated using some sort of kernel function, which defines the shape of the plane. The most common kernels are linear, polynomial, and Radial Basis Function (RBF). Tuning the parameters of SVMs is critical, and bad performance for this classifier can often be explained by poor parameter choices [16].

3.2 PROPOSED WORK

This work focuses on benchmarking IDS approaches for use with mobile devices. A new and common use case for IDS solutions is an implementation on smartphones. An IDS running in this environment must have high performance. More metrics are necessary, since not only will the algorithm need to be accurate, it must also be efficient [13]. The benchmark itself will capture information for both of these aspects of the IDS systems.

In order to create the benchmark, it is necessary to gather the metrics that are valuable and comparable between algorithms. The next step is to find and implement strong research work that created anomaly-based IDS solutions for NSL-KDD. Once the metrics and algorithms are gathered, the benchmark is developed and each algorithm is tested with it.

The crucial objective of this benchmark is to gather metrics regarding accuracy that can be used to compare different algorithms. Accuracy over the entire testing dataset is the most used metric in IDS proposals. Along with this, confusion matrices are commonly gathered and can be used to extract even more metrics, such as Positive Predicted Value (Precision) and Recall [5]. Confusion matrices contain four elements: True Positives (TP) which are rows correctly classified as anomalous in this dataset, True Negatives (TN) are correctly classified as normal, and False Positives (FP) and False Negatives (FN), the former being incorrectly classified as anomalous and the latter classified incorrectly as normal. Precision is $TP = (TP + FP)$, which is the proportion of correctly classified anomalous data to the total amount of data classified as anomalous. This is different from accuracy since it does not count the number correct or incorrect for normal data classification. The recall is $TP = (TP + FN)$, which is the ratio of correctly classified anomalous data to the total number of anomalous rows in the dataset [5]. Each of these metrics gives a different perspective on the strength of the algorithm.

Along with these general metrics, we can gather accuracy for each of the five categories of attacks. This paints a more complete picture of the performance of the algorithms, and shows where potential weak points are in each. These metrics must be gathered using the exact same set of testing data. This ensures each algorithm is on an equal playing field and no external factors affect their performance. For the accuracy metrics, statistics were gathered over fifty passes on the testing data. As most of the algorithms are deterministic [5], only neural networks should see any variance between passes.

The secondary focus of the benchmark involves time and power consumption. The CPU time and power consumption of an algorithm are important to know when using these algorithms in resource-starved devices. For each of the standard actions (data preprocessing, training, testing, classifying), the benchmark captures the average CPU time and power consumption over one hundred passes. While CPU time will differ between devices, the proportional time should be the same. Power consumption is difficult to track since it is subject to many external factors. We relied on Operating System's power utilities to measure power consumption.

4. RESULTS AND DISCUSSION

The benchmark is conducted on an Asus Chromebook C300. It has an Intel Celeron N2830 2.16 GHz Processor and four gigabytes of RAM [29]. In comparison, a Samsung Galaxy S7 has a Snapdragon 820 processor which has a similar clock rate and cache size [30] along with four gigabytes of RAM and no dedicated GPU. While the iPhone X is faster with its A11 bionic chipset and dedicated GPU [31], the Chromebook is similar in power to the vast majority of mobile phones currently in use. While IDS systems written specifically for mobile should be more efficient since they can use native code and be customized for the exact architecture of the device, the relative speed of each algorithm should be consistent.

The benchmark is written in Python and uses pandas [32] for data manipulation and scikit-learn [33] for machine learning algorithm implementations. Time is tracked using Python's internal library to track CPU time, and power consumption was tracked using Ubuntu's reported battery charge levels. The Chromebook itself is running the benchmark in Ubuntu 14.04 [34]. Each algorithm was implemented as closely as possible to their respective papers, and the source code is available online [35].

ACCURACY

The accuracy metrics were gathered on the test set, with the average accuracy over fifty passes recorded. All but the neural network are deterministic, so any variance inaccuracy is most likely the result of floating point precision issues [36]. From Table 2 we can see that the overall accuracy is very similar between the algorithms. The neural network performed the strongest with almost 78% accuracy, and k-means performed the worst with 74% accuracy. The categorical accuracy is interesting, since the only category the neural network performed best in was R2L, which is where every algorithm failed. Probe and DoS had high accuracy for most algorithms, while U2R and R2L were much more difficult to detect. K-means performed the best on normal, with 99% accuracy.

Algorithm	Accuracy (Total)	Probe	DoS	U2R	R2L	Normal
Naive Bayes	0.753626	0.827757	0.775275	0.645	0.028655	0.926364
Neural Network	0.778019	0.861016	0.774462	0.5302	0.132055	0.948373
SVM	0.769064	0.938868	0.760794	0.39	0.107843	0.928424
K-means	0.740367	0.741875	0.687318	0.5601	0.012338	0.990937

Table 2: Accuracy results from testing. The mean accuracy was measured over 50 passes, however the only algorithm with variance is the neural network.

The confusion matrices in Table 3 tell a similar story, the neural network had the highest TP score with the highest TP + TN score. K-means had the highest TN score but struggled with TP, meaning it had a lot of false negatives. Clearly k-means leans towards classifying most behavior as normal, whereas the rest of the algorithms had a similar FP measure. From the confusion matrices we can derive the metrics discussed earlier, precision and recall. These are commonly used in the evaluation of ML algorithms, and from the results we see that K-means actually has more than 98% precision since it has very few false positives. The neural network has the second highest with more than 94% precision, due to its strong TP number. For recall, though, k-means performs poorly with barely 55% recall while the rest are between 62% and 65% recall. What this shows is that k-means are the least likely to flag normal data as anomalous, its precision is high and positive predictions can be trusted. On the other hand, its recall is the lowest which means it will let more attacks through without catching them compared to the other algorithms.

Algorithm	True Positives	True Negatives	False Positives	False Negatives
Naive Bayes	7994	8995	715	4839
Neural Network	8330	9209	501	4503
SVM	8322	9015	695	4511
K-means	7068	9622	88	5765

Table 3: Confusion matrices recorded from experiments. Mean numbers are collected over 50 passes, however only the neural network had any variance.

POWER

The benchmark also collects metrics regarding power consumption, both in CPU time and straight battery use. The benchmark collected the mean CPU time over one hundred cycles for each function of the classifier, and a single battery charge loss for the entire duration of each test. In Figure 1 we see that there is a vast gap in training time between the algorithms. There is almost a 2000% increase in training time between Naive Bayes and SVM, while the neural network and k-means are about the same duration and twice as fast as SVM.

We see a similar story with classifying the testing dataset Table 4. Naive Bayes is clearly the fastest here, however for data preprocessing SVM is actually faster, and when classifying one row

it is almost as fast. These are both of the algorithms that were implemented using heavy feature reduction, which is a potential reason for these big speed increases.

Algorithm	Preprocessing Training Data (s)	Training Classifier (s)	Preprocessing Testing Data (s)	Classifying Test Dataset (s)	Classifying One Row (s)
Naive Bayes	1.81429	1.808617	0.362088	0.348027	1.251e 4
Neural Network	3.34136	18.01759	0.642061	0.511775	2.758e 4
SVM	1.67410	36.65068	0.353161	0.463680	1.323e 4
K-means	3.29604	18.40322	2.50727	0.725168	5.788e 4

Table 4: Average CPU time over 100 passes for each of the main actions in loading and training the classifiers.

BATTERY

When looking at battery consumption for each of these activities in Figure 2, we see that they more or less line up with the CPU time to train. K-means consumes the most battery on preprocessing the test dataset and classifying it, which is also where it is slowest compared to the other algorithms. SVM clearly uses a lot of battery to train the classifier, which is unsurprising given the training time. The battery metrics are not reliable enough to draw conclusions from, since the battery consumption is hard to track and subject to so many external factors. However, these results highlight a concern that must be addressed for mobile devices. If the battery consumption of an algorithm is too great, it is not a good candidate for use on mobile. From our results in this setting, it is clear that Naive Bayes creates the least battery drain.

DISCUSSION

The results of this benchmark paint an objective picture of the relative strengths of the above algorithm's implementations on the NSL-KDD dataset. The neural network is clearly the most accurate on this data with 78% accuracy. However, if CPU time is a concern, the Naive Bayes implementation is magnitudes faster than the other algorithms, with some reduction in accuracy at 75%. K-means is unlikely to accidentally flag normal traffic as anomalous with 98% precision, however it has a much lower categorical accuracy than the other algorithms and a recall of only 55%.

Figure 1: Average training time with marked standard deviation for each algorithm, collected over 100 passes.

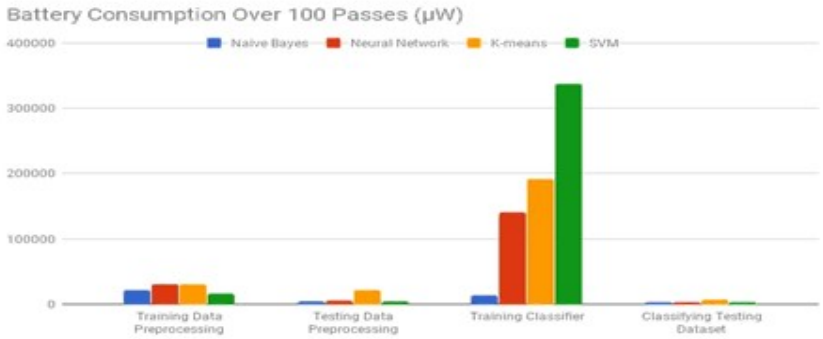

Figure 2: Total battery consumption over 100 passes for each of the main actions in loading and training the classifiers. Classifying one row had negligible battery impact for each algorithm.

Naive Bayes is by far the fastest algorithm. For data preprocessing and training, it only takes on average about 3.6 seconds, whereas k-means and the neural network take more than 21 seconds, and SVM takes over 38 seconds. Since it is only 3% less accurate than the best implementation, it appears to be the best choice for an anomaly-based IDS on a mobile device. A case can be made for the neural network, which is clearly slower for training but is less than a second slower for loading and testing the test dataset. In battery consumption, the differences are minimal outside of training. If training is infrequent, the boost in accuracy is potentially a good trade and makes the neural network the best candidate.

5. CONCLUSIONS

In this work, we developed and implemented a benchmark on a uniform dataset to evaluate accuracy as well as the performance of various Anomaly-based IDS solutions. This benchmark has the potential to provide objective quantitative judgments on different algorithms and their implementations. By using this strategy, it is possible to make informed decisions on the best possible algorithm for the situation.

Based on our observations, we can conclude that for performance, Naive Bayes is significantly faster at training than all of the algorithms, resulting in a much lower total CPU usage and battery consumption. The neural network had the highest accuracy, and is the best contender when performance is less of an issue. While specific use cases may value metrics differently, this benchmark provides definitive numbers that can be used to compare these algorithms.

In the results of our experiment, we saw the strengths and weaknesses of each algorithm we implemented for use in anomaly-based IDS solution. As mentioned earlier, there are many other algorithms and countless implementations available, but by using this benchmark it is possible to compare each of them and find the one that is truly the best suited for use in a given setting.

A goal of this work is to make this benchmark a standard method for comparing algorithms in the future. New algorithms can use this benchmark and compare their results with the ones in this work, which will add to a growing set of comparisons. Any proposed algorithms that were implemented on NSL-KDD can now be benchmarked and meaningful statistics can be generated. Future solutions that incorporate NSL-KDD can utilize this benchmark when generating their results as well, and they can compare their observations to the ones shown here.

The source code for the benchmark and each algorithm implementation are available on GitHub [35], and it can be downloaded and adapted to any use case. Future work could include implementations of more ML algorithms for this benchmark. In addition, the methods used in this benchmark can be utilized to evaluate such ML algorithms in other applications in the future.

ACKNOWLEDGEMENTS AND DATA AVAILABILITY

This work was performed at the Department of Computer Engineering and Computer Science, California State University, Long Beach. The research was conducted as a part of Nicholas J. Miller's Master thesis under the supervision of Dr. Mehrdad Aliasgari [37]. As stated before, the source code used for the benchmark and the experimental results have been deposited on GitHub [35].

REFERENCES

[1] S. Morgan, "Ransomware Damage 2015," 2017, [Online]. Available: report-2017-5-billion/.Costs 5 Billion in 2017, Up from 350 Million in https://cybersecurityventures.com/ransomware-damage-

[2] V. Charypar, "The End of the Cloud is Coming," 2017, [Online]. Available:https://venturebeat.com/2017/11/04/the-end-of-the-cloud-is-coming/.

[3] N. Scaife, H. Carter, P. Traynor, and K. R. Butler, "Cryptolock (and Drop it): Stopping Ran-somware Attacks on User Data," in Proc. Intl. Conf. on Distributed Computing Systems, 2016, pp. 303–312.

[4] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-Based Net-work Intrusion Detection: Techniques, Systems and Challenges," Computers & Security, vol. 28, no. 1-2, pp. 18–28, 2009.

[5] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," IEEE Communications Surveys & Tutorials, vol. 18, no. 2, pp. 1153–1176, 2016.

[6] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward Developing a Systematic Ap-proach to Generate Benchmark Datasets for Intrusion Detection," Computers & Security, vol. 31, no. 3, pp. 357–374, 2012.

[7] D. H. Deshmukh, T. Ghorpade, and P. Padiya, "Intrusion Detection System by Improved Pre-processing Methods and Naïve Bayes Classifier Using NSL-KDD 99 Dataset," in Proc. Intl. Conf. on Electronics and Communication Systems, 2014, pp. 1–7.

[8] M. S. Pervez and D. M. Farid, "Feature Selection and Intrusion Classification in NSL-KDD Cup 99 Dataset Employing SVMs," in Proc. Intl. Conf. on Software, Knowledge, Information Manage-ment and Applications, 2014, pp. 1–6.

[9] D. Novikov, R. V. Yampolskiy, and L. Reznik, "Traffic Analysis Based Identification of Attacks." IJCSA, vol. 5, no. 2, pp. 69–88, 2008.

[10] S. Mukherjee and N. Sharma, "Intrusion Detection Using Naive Bayes Classifier with Feature Reduction," Procedia Technology, vol. 4, pp. 119–128, 2012.

[11] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava, "A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection," in Proc. Intl. Conf. on Data Mining, 2003, pp. 25–36.

[12] L. Dhanabal and S. Shantharajah, "A Study on NSL-KDD Dataset for Intrusion Detection Sys-tem Based on Classification Algorithms," International Journal of Advanced Research in Computer and Communication Engineering, vol. 4, no. 6, pp. 446–452, 2015.

[13] D. Damopoulos, S. A. Menesidou, G. Kambourakis, M. Papadaki, N. Clarke, and S. Gritzalis, "Evaluation of Anomaly-based IDS for Mobile Devices Using Machine Learning Classifiers," Se-curity and Communication Networks, vol. 5, no. 1, pp. 3–14, 2012.

[14] V. Kumar, H. Chauhan, and D. Panwar, "K-Means Clustering Approach to Analyze NSL-KDD Intrusion Detection Dataset," International Journal of Soft Computing and Engineering, vol. 3, no. 4, pp. 1–4, 2013.

[15] R. A. Calix and R. Sankaran, "Feature Ranking and Support Vector Machines Classification Anal-ysis of the NSL-KDD Intrusion Detection Corpus." in FLAIRS Conference, 2013, pp. 292–295.

[16] R. Jain and N. Abouzakhar, "A Comparative Study of Hidden Markov Model and Support Vec-tor Machine in Anomaly Intrusion Detection," Journal of Internet Technology and Secured Trans-actions (JITST), vol. 2, no. 1/2, p. 3, 2013.

[17] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A Deep Learning Approach for Network Intrusion Detection System," in Proc. 9th EAI Intl. Conf. on Bio-inspired Information and Communications Technologies, 2016, pp. 21–26.

[18] J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan, "Cost-based Modeling and Evaluation for Data Mining with Application to Fraud and Intrusion Detection," Results from the JAM Project by Salvatore, pp. 1–15, 2000.

[19] J. McHugh, "Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory," ACM Transactions on Information and System Security (TISSEC), vol. 3, no. 4, pp. 262–294, 2000.

[20] University of New Brunswick, "NSL-KDD Dataset," [Online]. Available: http://www.unb.ca/cic/research/datasets/nsl.html.

[21] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set," in IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009, pp. 1–6.

[22] R. A. Maxion and K. M. Tan, "Benchmarking Anomaly-Based Detection Systems," in Proc. Intl.Conf. on Dependable Systems and Networks, 2000, pp. 623–630.

[23] S. Revathi and A. Malathi, "A Detailed Analysis on NSL-KDD Dataset Using Various Machine Learning Techniques for Intrusion Detection," International Journal of Engineering Research and Technology. ESRSA Publications, no. 12, pp. 1848–1853, 2013.

[24] N. Moustafa and J. Slay, "The UNSW-NB15 Data Set Description," 2016, [Online]. Available:https://www.unsw.adfa.edu.au/australian-centre-for-cyber-security/cybersecurity/ADFA-NB15-Datasets/.

[25] C. T. Giménez, A. P. Villegas, and G. Á. Marañón, "HTTP DATASET CSIC 2010," 2012, [On-line]. Available: http://www.isi.csic.es/dataset/.

[26] P. G. Jeya, M. Ravichandran, and C. Ravichandran, "Efficient Classifier for R2L and U2R At-tacks," International Journal of Computer Applications, vol. 45, no. 21, p. 29, 2012.

[27] University of Wisconsin-Madison, "A Basic Introduction To Neural Networks," [Online]. Avail-able: http://pages.cs.wisc.edu/ bolo/shipyard/neural/local.html.

[28] R. Xu and D. Wunsch, "Survey of Clustering Algorithms," IEEE Transactions on Neural Networks, vol. 16, no. 3, pp. 645–678, 2005.

[29] ASUS, "ASUS Chromebook C300 | Laptops," [Online]. Available:https://www.asus.com/us/Laptops/ASUS_Chromebook_C300/specifications/.

[30] K. Hinum, "Qualcomm Snapdragon 820 MSM8996 SoC," 2015, [Online]. Available:https://www.notebookcheck.net/Qualcomm-Snapdragon-820-MSM8996-SoC.156150.0.html.

[31] Apple, "iPhone X - Technical Specifications," [Online]. Available: https://www.apple.com/iphone-x/specs/.

[32] Pandas, "Python Data Analysis Library," [Online]. Available: https://pandas.pydata.org/.

[33] Scikit-learn, "Machine Learning in Python," [Online]. Available: http://scikit-learn.org/stable/.

[34] Canonical, "Ubuntu," 2018, [Online]. Available: https://www.ubuntu.com/.

[35] N. Miller, "Anomaly IDS Benchmark," 2018, [Online]. Available:https://github.com/AnomalyIDSBench mark/.

[36] D. Goldberg, "What Every Computer Scientist Should Know About Floating-point Arithmetic," ACM Computing Surveys (CSUR), vol. 23, no. 1, pp. 5–48, 1991.

[37] N. J. Miller, "Benchmarks for evaluating anomaly-based intrusion detection solutions," Master Thesis, California State University, Long Beach, 2018. [Online]. Available: https: //search.proquest.com/docview/2040857774