Journal of Mathematical Sciences: Advances and Applications
Volume 46, 2017, Pages 51-70
Available at http://scientificadvances.co.in
DOI: http://dx.doi.org/10.18642/jmsaa_7100121826

EVALUATING *K*-MEANS++ CLUSTERING FOR ANOMALY-BASED INTRUSION DETECTION SYSTEMS-FOCUS ON EXTERNAL THREATS

MATTHEW K. COUGHLAN[1], JOHN TUCKER[1], THOMAS NELSON[2] and BENJAMIN KLIMKOWSKI[3]

[1]United States Military Academy
West Point
NY 10996
USA

[2]Department of Mathematical Sciences
United States Military Academy
Building 601 (Thayer Hall)
Room 221A, Cullum Road
West Point
NY 10996
USA
e-mail: thomas.m.nelson@gmail.com

[3]Department of Electrical Engineering and Computer Sciences
United States Military Academy
West Point
NY 10996
USA

2010 Mathematics Subject Classification: 68U35.
Keywords and phrases: anomaly-based intrusion detection system, ssh brute force attack, *k*-means, cluster analysis.
Received May 1, 2017

Abstract

Intrusion detection systems (IDS) are systems used to defend a network against cyber attacks. Specifically, anomaly-based IDSs are systems that detect malicious activity on a network by identifying departures in network traffic from a previously established norm. For this project, we use a data set of network activity and assess the validity and effectiveness of *k*-means++ clustering at designating certain external traffic as malicious. We focus our detection efforts in on secure shell (SSH) brute force attacks. We evaluate the chosen clustering method by looking at benchmarks such as success/failure rates, false positive rates, and consistency across varying data.

1. Intrusion Detection Systems: A Background

A. Background

Pathan [6] discusses the evolution, implementation, and operation of intrusion detection systems (IDS). Intrusion detection systems have been around since the beginning of the internet. Although IDS saw vast improvement in the 1980s, it was not until the early 1990s that IDS found commercial significance with the invention of the automated security measurement system (ASIM) by the US Air Force. IDS are now one of the most popular methods of security in networking.

IDS can be classified into two main categories: misuse IDS (commonly referred to as signature-based IDS) and anomaly-based IDS. Misuse IDS identify potential attacks on a network by comparing the attack to known signatures which have been encoded into an expert database. Through the pattern matching method, misuse IDS match snippets of text or binary acquired from the malicious packet to formerly established signatures. Misuse IDS are susceptible to many limitations, including the ability of hackers to circumvent the system altogether by simply changing around certain pieces of their code such that no signature in the database matches the malicious code. The biggest limitation of misuse IDS is simply that they lack the ability to detect zero-day attacks on a network (attack for which there is no known signature). Anomaly-based IDS on the other hand are capable of identifying attacks that have never been detected before. Anomaly-based

IDS create models that represent the normal or expected behaviour of networks. Deviations from this expected behaviour raise an alarm. Both types of detection systems have four possible outcomes. A false negative is the most dangerous outcome in which the system fails to detect an actual attack. A true negative occurs when the system correctly dismisses network traffic as non-malicious. False positives are where the IDS misidentifies specific traffic as an intrusion. Finally, a true positive is the ideal situation in which an IDS correctly identifies an attack (Pathan [6]).

IDS architecture can be broken down into three main types: host-based IDS (HIDS), network-based IDS (NIDS), and hybrids. HIDS monitor behaviour on a specific machine and confront threats directed to the machine without having to capture traffic from the whole network. NIDS, on the other hand, monitor data on large pieces of the network. NIDS consist of a data collector, manager, and communication module that transmits/receives analysis results. Neither type of architecture is capable of monitoring encrypted traffic. Hybrid combines the advantages of HIDS and NIDS into one system. Generally, hybrids combine misuse and anomaly-based methods in order to maximize detection and minimize false positives. Hybrids also allow for centralized management of the network as well as each individual host. IDS architecture can be centralized (one manager responsible for event analysis, detection, classification, and system action), hierarchical (NIDS and HIDS managed separately), or distributed (multiple agents handle processing, analysis, and detection) (Pathan [6]).

Once an intrusion is detected, the IDS has a few different post-detection options. The system can take an active approach, which is commonly referred to as an intrusion prevention system (IPS). The IPS sends packets to the site of the intrusion in order to interrupt and/or cease the connection. The IPS also interacts with the edge routers (routers standing between the network and the outside world) in order to establish rules regarding certain, potentially hazardous, source

addresses. In contrast to the active approach, the system can take a passive approach where it simply generates and sends notifications to the system operator indicating that there has been an intrusion (Pathan [6]).

In addition to the pattern-based method inherent in misuse IDS, IDS can take a few other approaches to detect intrusions. Most IDS analyze the header contents of common protocols: internet protocol (IP), transmission control protocol (TCP), and user datagram protocol (UDP). More advanced IDS go further, performing what is known as "protocol analysis". If anything about the traffic deviates from the standards established by the industry protocol, then the system raises an alarm. Protocol analysis facilitates detection of both known and unknown attacks; thus, it falls under the category of anomaly-based IDS since it looks for anomalies in the protocol standards. IDS can also use the statistical and probabilistic approach. Instead of rules or signatures, many IDS rely on Bayesian statistics, which are heavily centered on conditional probability. The Bayesian model is created using both real attack samples and false positive samples (the sample must be large). The statistical approach is generally implemented into an already existing misuse IDS in order to increase the system's accuracy. Finally, the neural network based approach is an approach that requires no user defined parameters. Instead, it emphasizes constant learning based on experience. Neural network based IDS are mostly research-based as of right now; commercial products are a rarity.

Bolzoni et al. [2] claim that anomaly-based intrusion detection systems are often inefficient when it comes to classifying the attacks they detect. Therefore, security teams or administrators have to manually process each alert generated by the IDS. There are two main challenges for security personnel. First, the most harmful attacks occur in stages and security teams must catch these attacks in the early stages. Second, there are a lot of alerts generated by BOTnets and automatic scanners that, although considered true positives, are not regarded as harmful. Therefore, there are many small attacks that a security team must sift

through in order to get to the main attacks. Whereas classification is trivial when it comes to signature-based IDS, anomaly-based classification is far more difficult. Bolzoni et al. present Pancea, a system that uses machine learning techniques to automatically and systematically classify attacks detected by the IDS. Pancea analyzes the payload of an attack, so an attack without a payload cannot be properly analyzed. Most attacks inject some kind of data in to their targeted systems, so this is not a major limitation.

The researchers use two algorithms in their experiments: support vector machines (SVM) and the RIPPER rule learner. These non-incremental algorithms iterate on samples several times to build a best classification model unless the training phase is started from scratch. SVM is a set of supervised learning methods used for classification. SVM uses "support vectors" and "margins" to classify data into different groups and then assigns new data to a specific group based on distance (Bolzoni et al. [2]). RIPPER is a rule induction algorithm that uses a set of IF-THEN rules. IF (condition), THEN (conclusion). Essentially, RIPPER takes already identified data, and builds an algorithm from the ground up that accurately classifies each data value. It heuristically adds one condition at a time until all of the data has been accurately classified. RIPPER then implements an optimization phase in order to simplify the set. Once the algorithm has performed on the teaching set, it is ready to be used on outside sources of intrusion. After implementing the algorithms, the study showed that the classification accuracy is almost always above 75%, with SVM outperforming RIPPER by a narrow margin in each trial (Bolzoni et al. [2]).

2. Clustering Methods

A. Cluster analysis

Leung and Leckie [4] provide a good overview of cluster analysis methods. Unsupervised anomaly detection makes two assumptions about the data: the majority of the network data is normal and some portion is

malicious; the attack traffic is statistically different from the normal traffic. The algorithm takes unlabelled data and attempts to find intrusions buried in the data. Many techniques involve the classification of threats first through clustering, then intrusion detection through the use of an algorithm.

Clustering is the method of grouping observations into meaningful subclasses. Members of a subclass are meant to be similar whereas members of different clusters should be noticeably different. Clustering methods can be useful in classifying large amounts of data. Once an observation is labelled, it is then grouped into a cluster with observations that share similar traits.

A few different clustering methods currently in use are partitioning methods, density-based methods, and grid-based methods. Given a database of n objects, a partitioning method constructs k partitions of data where each partition represents a cluster. Density-based methods are based on the assumption that clusters are dense regions in the data space that are separated by regions of lower density. Rather than data points being assigned to a cluster, a cluster is defined as a specific density or higher within a given radius. Grid-based methods divide the x-y grid into a finite number of cells. This approach has a very fast processing time. The algorithm in Leung's paper assumes that any point falling within the clusters is normal while all others are considered anomalies (Leung and Leckie [4]). In many cases, clustering algorithms will place anomalies in an entirely separate cluster of their own.

B. *K*-means and *k*-means++

Arthur and Vassilvitskii [1] provide insight into the method of k-means++. In the original k-means formulation, we are given an integer k and a set of n data points. The goal is to choose k centers in a way that minimizes φ, the sum of the squared distances between each point and its closest center. K-means involves choosing k centers (usually uniformly random) and assigning each data point to the closest center. The center is then recalculated to be the center of mass of the cluster of data points.

This process occurs over and over until the optimal clustering out of all k^n possibilities is achieved.

$$\varphi = \sum_{x \in X} \min_{c \in C} \|x - c\|^2,$$

where c is the mean of a specific cluster center, and x is the value of a given observation.

Arthur and Vassilvitskii propose a way of initializing k-means by choosing random starting centers with very specific probabilities – choose a point p as a center with probability proportional to p's contribution to the overall potential. In the original k-means, we choose initial centers uniformly at random. By recalculating the center of mass of each cluster during each iteration, φ is guaranteed to decrease with each iteration. The algorithm for k-means is:

1. Arbitrarily choose k initial centers $C = \{c_1, \ldots, c_k\}$.

2. For each $i \in \{1, \ldots, k\}$, set the cluster C_i to be the set of points in X that are closer to c_i than they are to c_j for all $j \neq 1$.

3. For each $i \in \{1, \ldots, k\}$, set c_i to be the center of mass of all points in C_i : $c_i = \frac{1}{|c_i|} \sum_{x \in C_i} x$.

4. Repeat Steps 2 and 3 until C no longer changes" (Arthur).

The k-means++ also begins with an arbitrary set of cluster centers; however, k-means++ employs a new algorithm:

Let $D(x)$ denote the shortest distance from a data point x to the closest center which we choose.

1a. Choose an initial center c_1 uniformly at random from X.

1b. Choose the next center c_i, selecting $c_i = x' \in X$ with probability $\frac{D(x')^2}{\sum_{a \in A} D(x)^2}$.

The probability term is known as "D^2 weighting".

1c. Repeat Step 1b until we have chosen a total of k centers.

2-4. Proceed with the standard k-means algorithm noted above (Arthur).

The researchers tested k-means and k-means++ on four datasets and tested $k = 10, 25,$ and 50. The k-means++ consistently outperformed k-means, both by achieving a lower potential value (φ), in some cases by several orders of magnitude, and also by having a faster running time. The D^2 seeding ran slightly slower than uniform seeding, but was still a faster algorithm since the search converged after fewer iterations. In contrast to k-means, k-means++ almost always perfectly clustered synthetic, pre-distributed datasets. K-means++ achieved a 10% accuracy improvement over k-means on the real-world datasets as well. K-means++ also achieved potentials that were 20 to 1000 times smaller than those achieved by k-means.

There is an alternate supervised anomaly detection method called "K-means + C4.5" (Muniyandi et al. [5]). It combines k-means with the C4.5 decision tree. The k-means method is cascaded with the C4.5 by building decision trees using the instances in each k-means cluster. This alleviates two problems in k-means: forced assignment and class dominance. Forced assignment occurs when the k parameter is significantly less than the number of natural groupings within the training data. Class dominance occurs in a cluster when the training data has a large number of instances from one class and very few instances from the others. Cascading involves the selection phase and the classification phase. In the selection phase, the closest cluster to the observation is selected. In the selected cluster, the decision tree corresponding to that cluster is generated. In the classification phase, the test instance is classified as normal or anomaly (intrusion) using the decision tree result. The observation is included in the cluster with a label of either normal or anomaly.

The C4.5 algorithm is as follows:

Given a set *S* of cases, C4.5 grows an initial decision tree using a divide-and-conquer algorithm.

"1. If all the cases in *S* belong to the same class or *S* is small, the tree is a "leaf" (i.e., box) labelled with the most frequent class in *S*.

A. Otherwise, choose a test based on a single attribute with two or more outcomes (i.e., great than, less than). Make this test the root of the tree with one branch for each outcome of the test, partition *S* into corresponding subsets *S*1, *S*2, … according to the outcome for each case, and apply the same procedure recursively to each subset" (Muniyandi et al. [5]).

K-means + C4.5 does noticeably better than *k*-means alone when classifying a data set.

K-means clustering has two main shortcomings: number of clusters dependency and degeneracy. "Number of clusters dependency" simply means that the value of *k* is extremely important to the clustering result. The mathematical program R contains both methods of *k*-clustering and methods of determining the optimal number of *k*'s. Degeneracy means that the end result of the clustering may include some empty clusters which are useless. There are algorithms which can overcome the issue of degeneracy (Guan et al. [3]).

3. The Data Set

A. Data source

The effective comparison of various intrusion detection systems is hinged upon implementing the systems across a robust and realistic data set. Data sets are more often than not suboptimal for the needs of intrusion detection research. Most data sets are static datasets which means that they likely fail to reflect the most current and ever-changing state of intrusions. Static data sets are simply "outdated, unmodifiable,

inextensible, and irreproducible" (Shiravi). Intrusion detection systems must be rigorously tested, evaluated, and retested prior to their implementation into real-world applications. The best method for testing and evaluating an IDS is running it through a real network that is transmitting both normal and intrusive traffic. However, realistic data sets are extremely hard to come by, and most data sets that are readily available are over-specified to the IDS that they were originally created to test.

The Information Security Center for Excellence at the University of New Brunswick (UNB ISCX) used live network devices and workstations in order to create the UNB ISCX intrusion detection evaluation data set (Shiravi). Agents on the network imitated previously observed user behaviours while attack scenarios were executed to replicate real-world malicious traffic. The data set itself is comprised of multiple individual data files captured over the course of various days in 2010. The files are comprised of simulated network traffic, containing malicious HTTP, distributed denial of service (DDoS), and secure shell attacks as well as countless examples of normal, baseline traffic (Shiravi). The data file we use, captured on June 17, 2010, is 13 gigabytes of realistic SSH Brute Force data mixed with normal network traffic.

For various reasons, the UNB dataset is by far the best dataset to use for comparing different IDS-related clustering methods. First, as previously mentioned, the dataset contains realistic network traffic. No anomalies have been artificially implanted into the data, and the data contains no tags within the actual data that indicate the true normal or malicious nature of the traffic – had these identifiers been added to the raw data, then there is the possibility that inconsistencies would be introduced into the final dataset (Shiravi). Second, the dataset is externally labelled, which means that there is an identified distinction between the anomalous and normal traffic. After running a clustering algorithm, we are able to evaluate our results by comparing them to the actual labelled traffic. This greatly assists in the evaluation process and allows for more realistic benchmarks in IDS performance.

The third advantage to using the UNB dataset lies in how much detail the dataset captures. The data includes all of the traffic that occurs on the network over the course of one 30-minute period. It excludes no network interactions. Similarly, for every interaction, the dataset includes information about the traffic's payload data. This allows for a more robust analysis of the data that is passing through the network (Shiravi). The final advantage of the UNB dataset is that it contains a diverse set of intrusion scenarios. Cyber-attacks have evolved into different classes of size, type, and complexity. Some attacks vary in frequency while others may follow a multi-stage algorithm in order to undermine a network with exacting precision over time. The UNB dataset takes the variance of attacks into account and includes multiple types of intrusions, making the dataset more realistic and giving the IDS methods a real-world test of effectiveness (Shiravi).

B. The attack: Brute force SSH

Secure shell (SSH) is a protocol used for remote login which allows a user to securely operate a remote host over an unsecure network. A brute force attack occurs when a system such as a BOTnet tries to crack a password by simply testing a high volume of randomly generated combinations of characters or words (Shiravi). As password length increases, the average amount of time to find the correct password increases exponentially.

Brute force attacks are a very common type of intrusion since they can break into accounts with weak username-password combinations. The dataset that we are using contains various instances of brute force attacks which pursued an SSH account on the simulated UNB network. The attacks used a dictionary style method; the protocol ran through a 5000 word dictionary that contains entries of varying lengths and characters. The overall attack ran for 30 minutes and was able to acquire the account credentials of the network admin. The attacker then used the account credentials to remotely log into the server (Shiravi).

C. Initial data analysis

Each protocol executed in the network is defined in the dataset by a range of variables, including the source and destination IP addresses, the start and end times, the total source and destination bytes, the total source and destination packets, and the protocol type. For analysis purposes, we took a small slice of the data; specifically, of the nearly 700,000 unique observations from June 17, 2010 we choose a random sample of 61,832 which allowed us to more efficiently carry out some preliminary work with the data. Of those protocols, 61,651 were normal traffic and 181 were malicious attacks. Using the external labels provided by UNB, we saw that all 181 attacks were an SSH protocol, though there were 27 other SSH attempts that were simply normal traffic. All SSH traffic was sent to the destination IP of 192.168.5.122 on port 22 and all malicious SSH attempts originated at IP 131.202.243.90.

Comparison of IP and port destination frequencies did not immediately show any significant results. However, one of the data variables that immediately jumped out at us was the difference in time duration between the different protocols. The time duration of all protocols ranged from less than 1 second to nearly 14 hours; however, for SSH specifically, the time durations ranged from 2 seconds to just under 10 minutes. Of all 181 SSH attacks, 149 were either 3 seconds or 4 seconds in length, whereas normal SSH traffic generally took at least 10 seconds to complete. Given that over 80% of the observed malicious SSH traffic falls within the 3-4 second range, it appears that time duration may play a significant role in identifying SSH attacks when compared to normal SSH traffic. One possible explanation for this discrepancy in timing between attacks and normal traffic is that the benign traffic is usually initiated by a human user whereas the attacks are generally BOTnets executing commands at a much higher rate than the average person ever could.

4. Preliminary Results

A. Initial efforts

The purpose of our project changed direction a few times from the start. At first, we intended to compare the effectiveness of different clustering methods in detecting SSH brute force attacks. However, we ran into some issues when trying to analyze the initial data set. The first issue was that we did not have enough SSH data. Given that only 208 of the 61,832 protocols were SSH, we simply could not make any major inferences on the nature of the SSH traffic. Taking into account the variables of packet size, number of packets, and time duration of each protocol and applying the method of *k*-means proved to be useless. We used the method of linear discriminant analysis, a technique that incorporates pattern recognition and machine learning to create a linear algorithm which separates data into different classes, in order to illuminate the underlying issues with our data set. The linear model that we created accurately classified 12 out of the 181 attacks correctly. The other 169 attacks were classified as "normal". Likewise, 373 of the 61,651 normal traffic protocols were classified as "attack", essentially drowning out the 12 that we had correctly assigned as attacks. The relatively low amount of SSH in the larger dataset meant that clustering methods would have low false positive rates, but also low accuracy rates. Clearly, using the dataset in its entirety was not going to work for clustering purposes.

Even once we focused in on the SSH traffic by itself, clustering methods could not properly parse out malicious SSH attempts from the benign attempts because, of the 208 SSH protocols in the sample, 181 of them were malicious. In reality, there should be far fewer instances of attack than normal traffic in any given type of protocol. Again, we were able to illuminate the issues with our dataset by running a linear discriminant analysis on the SSH attempts alone. Of the 181 attacks, we accurately predicted that 179 of them were in fact attacks. However, of the 27 remaining normal protocols, our model assessed 22 of them as

being attacks. Thus, our model had a generally high accuracy rate but a high false positive rate as well.

Subsection A in Section 5 describes the methods by which we were able to correct both the issue of low accuracy and that of high false positive rates. We focused solely on SSH traffic (disregarding other protocol types) and added more normal SSH data so that there was far more normal traffic in our dataset than there was attack data. Additionally, we made slight changes to the types of variables across which we carried out the *k*-means clustering.

5. Final Results

A. Dataset modification and variables

As stated above, one of the main issues we encountered in our preliminary attempts to analyze the network traffic for SSH attacks was that there were simply not enough SSH attacks to make any significant claims about the data. Thus, the first step we took to improve our predictive capability was to expand the data set to one that contained far more normal SSH data than it did attack data. In any network, it is not unreasonable to assume that normal SSH protocol will outnumber any malicious traffic in the network. Likewise, it is likely that an IDS searching for an SSH attack will analyze only SSH data, meaning that we could essentially ignore other types of protocols and just build a dataset that consisted of only SSH traffic. In order to build a dataset with more SSH, we used the other six days of data in the UNB ISCX dataset. Whereas the SSH Brute Force data was compiled on June 17, 2010 the dataset contains other network traffic that was analyzed from June 11 to June 16, 2010 including denial of service attacks and days of normal activity. Each of these six days contain plenty of benign SSH data that we used in order to establish a baseline for normal SSH. In total, our new dataset contained a total of 14,449 instances of SSH being used. Of those 14,449 observations, there were still only 181 that were attacks. This

ratio of normal to malicious is far more realistic than our original dataset.

Another issue we faced when analyzing our original data was that the variables on which we were conducting the *k*-means clustering, and verifying by linear discriminant analysis, were not ideal indicators of a brute force attack. Although timing effects were definitely an important factor in identifying attacks, we needed to rework the way in which we considered the time. In the other words, rather than looking at the duration of time that a given protocol is open, we instead considered the inter-packet timing for a given IP address. For instance, in the case of our attacker's IP (131.202.243.90), we looked at the timing between SSH packets sent to the network with the assumption that small time differences between the end and start times of successive packets sent from a single IP would indicate a brute force SSH attack.

In addition to inter-packet timing, we also considered the average packet size, the average number of packets that an IP address sends, and the average number of packets an address receives over the course of a given period of time. For all four variables, we analyzed them over intervals of 1 second, 2 seconds, 5 seconds, 10 seconds, 30 seconds, and 60 seconds. By adjusting the time parameter and observing the resulting sensitivity of the detection methods, we intended to determine what time interval best assists an IDS in correctly separating SSH attacks from normal network traffic. The reason for testing different time intervals of consolidation and analysis of traffic data is twofold. In almost any situation, a low time interval is ideal since we hope to identify and block attacks as soon as possible – the difference between 2 and 60 seconds can be drastic when it comes to protecting a network. However, 2 seconds of data provides far less information to an IDS than 60 seconds, so there are benefits to allowing the detector to gather more data before making a conclusion about a given protocol or piece of traffic. Thus, we tested our detection methods on six different intervals. An example of our data output for the 10 second interval is shown in Table 1.

Table 1.

Observation	Average packet size (kB)	Average time between packets (sec)	Number of packets: In	Number of packets: Out	Label
1	1171.455253	0.004327131	25089	119867	Normal
2	1272.184655	0.010430248	28829	227757	Normal
3	16	3	45769	8079294	Attack
4	1126.032628	0.004413063	17314	51485	Normal
5	1048.536585	0.004786979	9787	24618	Normal
6	16	3	45732	7935573	Attack
7	28.8	3.333333333	46954	11031565	Attack
8	24	4.5	45450	7123587	Attack
9	21.33333333	6	45766	8031384	Attack
10	8.404040404	0.005102041	408	2740	Normal
11	1227.797918	0.006131208	19822	69718	Normal
12	1184.873988	0.006952491	26542	145376	Normal
13	837.6363636	0.007283321	12087	32111	Normal
14	1125.333333	0	80717	13785576	Normal
15	1301.5	0	79493	10016355	Normal
16	48	0	79463	9847358	Normal
17	152	0	79086	8556277	Normal
18	854.1886792	0	31883	407330	Normal
19	937.3913043	0	30467	325172	Normal

Each 10 second interval is labelled as 'Normal' or 'Attack' for reference after clustering.

B. Results

We ran the six different interval data sets through a *k*-means++ algorithm in R in order to parse out potential attacks into their own cluster. The *k*-means++ algorithm created two separate clusters. In order to determine which cluster represented the attack cluster in each scenario, we looked at its mean average packet size, which is shown below for each detection time interval:

Table 2. Average packet size (kB) for both clusters in each of the six observation scenarios. The attack cluster is selected based on average packet size

Observation time interval (sec)	Cluster 1 average packet size (kB)	Cluster 2 average packet size (kB)	Attack cluster
1	105.67	78.04	2
2	108.88	77.13	2
5	111.18	80.94	2
10	117.75	83.19	2
30	77.87	124.41	1
60	79.39	132.27	2

The attack cluster was selected based on the average packet size of each cluster. On average, the payload data in the normal traffic was around 119kB whereas the payload data in the attack traffic was 61kB. Thus, in each scenario, we choose the cluster with the significantly smaller average packet size as the attack cluster.

The final results for the ability of the *k*-means++ algorithm to detect SSH brute force attacks are shown below in Table 3 and Table 4:

Table 3. Final results for the ability of the *k*-means++ algorithm to identify SSH attacks at varying time intervals of observation

Observation time interval (sec)	Normal observation correctly identified	Normal observation identified as attacks	Attacks identified correctly	Attacks identified as normal
1	2388	1474	859	1731
2	1542	898	505	997
5	933	460	253	473
10	511	258	146	263
30	200	112	60	104
60	118	67	34	59

Table 4. Statistics showing the accuracy, false positive, and false negative rates of the *k*-means++ algorithm at varying time intervals of observation

Observation time interval (sec)	Accuracy rate	False positive rate	False negative rate
1	0.503254805	0.38166753	0.668339768
2	0.519279554	0.368032787	0.663781625
5	0.559697971	0.330222541	0.651515152
10	0.557724958	0.33550065	0.643031785
30	0.546218487	0.358974359	0.634146341
60	0.54676259	0.362162162	0.634408602

Our results show that, in general, *k*-means++ is not a very reliable method of brute force detection when using these variables to create clusters. Looking at intervals of 5 seconds appears to result in the highest accuracy rate and the lowest false positive rate. However, the 5 second interval also produces the 3rd highest false negative rate. Across all three metrics (accuracy, false positives, and false negatives), it appears that where an observation interval succeeds in one metric, it fails in another.

One possible explanation for the shortcomings of the *k*-means++ algorithm is that we did not include enough variables in the clustering algorithm. Although we choose the four variables that we believed would have the largest impact on the clustering, we could have tried to include other variables, such as specific IP data. Incorporating the quantity and frequency of packets being sent to and from specific IP address may have shed more light on which packets were part of an SSH attack. Another more likely explanation is that anomaly-based intrusion detection often fails when it comes to detecting subtle differences in network traffic. In other words, had the SSH brute force attack caused a more noticeable shift in the dynamics of the network, the *k*-means++ algorithm would have had a much easier time labelling the attack as such. However, where the SSH attack was rather inconspicuous in nature (likely an intention of the attacker), the *k*-means++ algorithm had difficulties.

Anomaly-based detection systems work well when data is truly anomalous and work best when they are paired with other systems such as a signature-based IDS.

C. Future work

Our work here lays a ground work for future exploration of the *k*-means++ algorithm and its ability to detect SSH brute force attacks. By looking at the data through the lens of different time intervals, we note that cluster analysis can produce varying levels of output. Future work can expand on our work in a few ways.

First, the *k*-means++ algorithm should be paired up with signature-based capabilities in the future. Given that real world data does not often give clear-cut evidence to what is normal data and what is an attack, *k*-means++ will more than likely fall short of being able to make highly accurate predictions. In order to detect more inconspicuous attacks, a signature-based method that takes into account known vulnerabilities and attacks should be coupled with the ability of *k*-means++ to detect changes in the dynamic of the network traffic.

In order to enact a combined signature-based and anomaly-based system, future research would need to take into account the data that the packets are carrying, something that we did not do since the payload data in our sample was encrypted. Analyzing packet headers through protocol analysis as well as understanding a packet's entire payload would greatly increase the chance that an improved IDS could detect an attack and would greatly reduce the false negative rate of the system.

Another expansion of our results should look at the practicality of different detection time intervals. If, for instance, it only takes an attacker 30 seconds to brute force a given password, then using observation intervals of 60 seconds is useless since it allows an attacker to conduct a full attack before we even begin to notice it. Dictionary style attacks further complicate this issue, especially since users rarely employ enough randomness in their password generation to beat these attacks. Therefore, future works should look to combine functional IDS with real-world attacker tendencies.

References

[1] David Arthur and Sergei Vassilvitskii, *k*-means++: The advantages of careful seeding, Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms Society for Industrial and Applied Mathematics, 2007.

[2] Damiano Bolzoni et al., Pancea: Automating attack classification for anomaly-based network intrusion detection systems, Recent Advances in Intrusion Detection (2009), 1-20.

[3] Yu Guan, Ali-Akbar Ghorbani and Nabil Belacel, *Y*-means: A clustering method for intrusion detection, 2003.

[4] Kingsly Leung and Christopher Leckie, Unsupervised anomaly detection in network intrusion detection using clusters, Proceedings of the Twenty-eighth Australasian conference on Computer Science, Volume 38, Australian Computer Society, Inc., 2005.

[5] Amuthan Prabakar Muniyandi, R. Rajeswari and R. Rajaram, Network anomaly detection by cascading *k*-means clustering and C4.5 decision tree algorithm, Procedia Engineering 30 (2012), 174-182.

[6] Al-Sakib Khan Pathan, Chapter 2 Network Traffic Monitoring and Analysis. The State of the Art in Intrusion Prevention and Detection, Boca Raton FL: CRC Press, 2014, 23-46.

[7] Steven L. Scott, A Bayesian paradigm for designing intrusion detection systems, Computational Statistics & Data Analysis 45(1) (2004), 69-83.

[8] Ali Shiravi et al., Toward developing a systematic approach to generate benchmark datasets for intrusion detection, Computers & Security 31(3) (2012), 357-374.

■