Applied Machine Learning Intensive (AMLI), Summer 2021
Capstone Report

DETECTING ADVERSARIAL ATTACKS ON COMPUTER NETWORKS

July 30, 2021

Ron Merculief

Department of CSE

University of Alaska Anchorage

Abraham Mitchell

Claudia Mercado

Department of CSCE

University of Arkansas

1 Abstract

Detecting attacks on computer networks has been an issue for companies/organizations for years. No single model has been able to completely eliminate the threat despite numerous studies with high success rates. Our goal is to create a K-nearest neighbor (KNN) model that can accurately detect attacks on computer networks. Using a dataset collected from Los Alamos National Laboratory, the model was able to achieve a mean accuracy of 99.9%.

2 Introduction

As technology advances and organizations grow, the usefulness of computerization increases exponentially. Companies are reliant on vast computer networks to meet the great demand of customers. This reliance creates a vulnerability that must be minimized to prevent adversarial attacks. These models must also be ethically considerate. A model that is not socially beneficial, safe, and relatively unbiased should not be implemented. For this type of problem, a model needs to be able to protect organizations' computer networks, which will in turn protect the individuals who utilize these organizations. Certain steps, such as training models on multiple data-sets, investigating data before training, and actively watching for common problems such as over fitting, should all be taken to ensure safety and reduce bias.

This project aims to develop an advanced machine learning model that will detect these rare events while also upholding these ethical standards. Subsequently, creating this model, regardless of its effectiveness or lack thereof, will provide us with invaluable experience in machine learning that will improve our skills as computer scientists and machine learning specialists.

3 Literature Review

3.1 K-Nearest Neighbor

K-Nearest Neighbor (KNN) is an algorithm that was made popular by its simplicity, high maturity, and ability to handle noise [4] [5]. KNN works by classifying new objects based on the attributes and training data. To define the distance between two training objects and testing, the Euclidean distance formula is used [4].

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2} \tag{1}$$

In Okfalisa, et al (2014), an analysis of KNN for data classification was conducted using data from the Conditional Cash Transfer Implementation Unit (Unit Pelaksana Program Keluarga Harapan). The KNN model was able to achieve an average accuracy of 93.94%

with a high of 94.95% [4]. While the accuracy is high, the data set only contains 7395 records, which is very low when compared to the amount of data required to train rare event models.

One of the fallbacks of KNNs is that the accuracy of KNNs decreases when dealing with vastly large data sets. There have even been experiments that attempt to increase this accuracy for big data, such as the one conducted by Guo, et al (2003) [3] [6]. KNNs also require the choice of a k value. The value of k is crucial as k will directly affect the accuracy of the model. There are many ways to choose the best value for k, but this can be difficult and time consuming [6].

When it comes to rare event detection, KNNs have been used with a great deal of success. Hashmani, Manzoor Ahmed, et al (2018) used KNN to detect malicious URLs from a dataset containing 3.2 million features. The KNN method used was able to detect malicious URLs with a 98.67% accuracy [14]. The authors of [15] used two adaptations of KNN, Indexed Partial Distance Search kNearest Neighbor (IKPDS) and Partial Distance Search kNearest Neighbor (KDPS), to reduce the computation time of traditional KNNs while maintaining accuracy in detecting network intrusions. Each model used had an accuracy above 99% when classifying from the 12,597-feature dataset.

3.2 Support Vector Machines

Support Vector Machines (SVMs) are a set of related methods to be used in supervised learning and were designed specifically for binary classification. SVM classifiers separate data and utilize a maximum-margin hyperplane to measure the confidence of each classification [3] [7]. SVMs are frequently used because of their simple structure, high empirical performance, and the ability to learn with small datasets [3]. Bansal, Atul, et al (2012) attempts to classify the gender of humans with scans of the iris. The authors of [8] created several SVMs with different kernel functions, namely polynomial, Gaussian, and radial basis. The mean accuracy for the SVMs are 81.27%, 83.06%, and 80.97% respectfully, with only 300 iris scans in the dataset.

Support Vector Machines suffer from high training times, which means using SVMs on big data will require a lot more time than other algorithms [3]. Experiments, such as the one described by Wang, Senzhang, et al (2014) have been conducted to try and reduce the high training time with minimal decrease in performance. The establishment of an effective SVM requires the definition of the proper SVM configuration. To do this, users must have extensive knowledge in the field as well as the time for multiple attempts of trail error [10].

SVMs are widely used for cyber-security and network protection. Hashmani, Manzoor Ahmed, et al (2018) used the same 3.2 million feature dataset that was used for the KNN model. The SMV model was able to detect malicious URLs with 98.67% accuracy [14]. Kajal, Abhishek, and Sunil Kumar Nandal (2020) used a hybrid Artificial Neural Network (ANN) SVM hybrid to improve upon SVM detection of Distributed Denial of Service (DDoS) attacks. The hybrid model performed extremely well, reporting a precision of 0.966 and a

detection accuracy of 98.247% from a KDD cup dataset containing millions of records [16].

3.3 Naive Bayes

Naive Bayes models are classifiers based on Bayes' Theorem, an equation that uses prior knowledge of relevant conditions to describe the probability of an event [11] [12]. The basic formula is:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2}$$

where P(A|B) is the probability of event A occurring, given event B has occurred, P(B|A) is the probability of event B occurring, given event A has occurred, P(A) is the probability of event A, and P(B) is the probability of event B [12]. Naive Bayes algorithms are popular because of their high performance with large datasets, quick training times, and because they are easy to build compared to other models [11].

The major drawback of Naive Bayes is that features are considered to be independent and do not affect one another. The likelihood of features being completely independent of each other in real world scenarios is very low, so this drawback could lead to bad predictions when using highly related data [3]. Another drawback of Naive Bayes is that smoothing methods must be used to prevent zero probabilities. Zero probabilities occur when a variable has a category in the test dataset but not in the train. The model will automatically assign a zero for the probability and will not make a prediction [13]. Smoothing methods are also highly complicated, and no single method will provide the best results for every model [3] [13].

Naive Bayes models are frequently used when creating Intrusion Detection Systems (IDS). Z. Muda, W. Yassin, M. N. Sulaiman and N. I. Udzir (2011) combine NaÃ¯ve Bayes classification with K-Means clustering to improve upon anomaly-based detection capabilities. This approach had an average detection rate higher than 99% with a false alarm rate under 0.5% [17]. Alqahtani H., et al (2020) used a NaÃ¯ve Bayes model to create an IDS and compared that model with other classification algorithms. The NaÃ¯ve Bayes model achieved an average accuracy of 91%, precision of 98%, recall of 91%, and f1-score of 95% [18].

3.4 Other Prevention Methods

Numerous non-machine-learning experiments have been conducted in efforts to prevent cyber-attacks. One such experiment, conducted by Kolev, Alexander, and Pavlina Nikolova (2020), attempts to create an effective cyber-security protection device for wireless networks using microcontroller devices. With the selected microcontroller devices, ESP-32 and ESP-07, the authors of [19] set up honeypot systems with two elements, a captor, and a decoy. The decoy attempts to use its information resources to attract unauthorized access

to the system. The captor monitors the attack, collects and analyzes data, and triggers alerts all while staying undetected from the intruder. The ESP-07 was unable to compile the code needed for the system, the ESP-32 was successful, and the authors intend to conduct more research to optimize the device configuration and software.

Another example of non-machine-learning threat detection comes from N. Kumar Singh et al (2019). The authors of this experiment created algorithms that prevent Denial of Service (DOS) and Man in the Middle (MITM) attacks on smart grids. In DOS attacks, the client sends connection requests to the server without sending data and then closes the connection. This prevents meaningful connections from being made as the server is kept busy. To prevent DOS attacks, a variable that counts the number of times a connection without data is made. When the count exceeds a certain limit the IP address is blocked. MITM attacks happen when the communication link between connection in the smart grid. The attacker has access to the server and has the ability to alter messages between connections. To stop MITM attacks, the authors of [20] encrypt every transmission between the Remote Terminal Units (RTU) and the substation of the smart grid. If the encryption key from an IP address does not match the RTU, the transmission is considered a MITM attack, and the data is not accepted. The DOS model and the MITM model were both successful in preventing these attacks.

4 Data Analysis

Data was collected from the Los Alamos National Laboratory's corporate, internal computer network. The data is directly available from their website: $https://csr.lanl.gov/data/cyber1/$. This data was recorded over 58 days and was de-identified event data from 5 different networks. The majority of the data originated from individual users. Other data sources include centralized Active Directory domain controller servers, process start and stop events, logged Domain Name Service (DNS) lookups, network flow from various routers. A list of events created by the red team penetration testers defined bad behavior events over the period. Overall, the 12 gigabytes of compressed data gathered totals 1,648,275,307 events from 12,425 users, 17,684 computers, and 62,974 processes.

Well-known traffic was not de-identified such as users SYSTEM and Local Service, ports 80, 443, etc. All data points not listed before were de-identified for security purposes. The de-identified categories were users, computers, processes, ports, time, and other miscellaneous details. For time purposes, the epochs start at 1 with a resolution of 1 second. Failed authentication is logged if and only if they successfully authenticated at a previous time.

The data was downloaded and put into a pandas data frame. Threats were classified with a 1 and non-threats with a 0. 5% of data was used for training and 1% for testing. We originally planned to use 80% training and 20% testing, but the large size of the data set made this type of split cumbersome. We then gathered a list of unique categories for

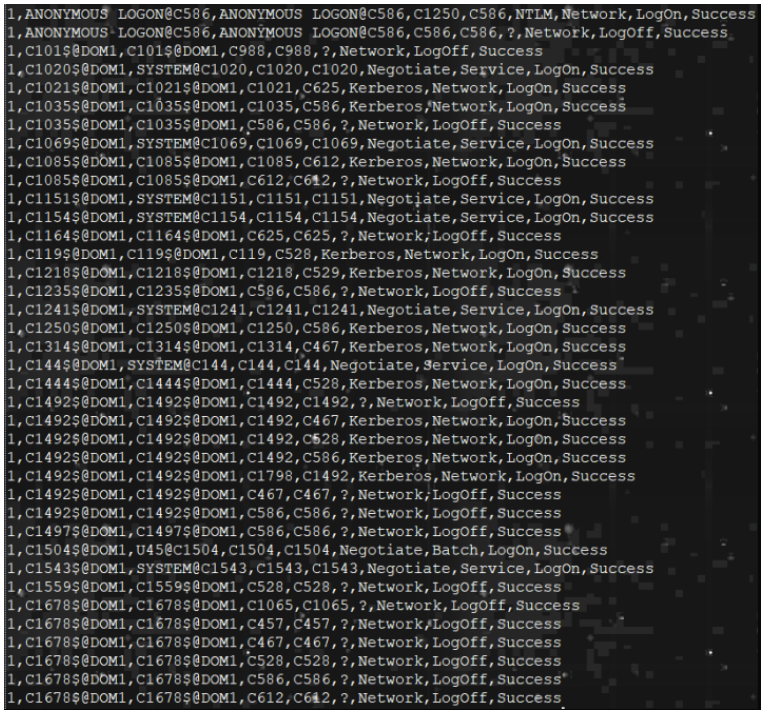

Figure 1: This data represents authentication events collected from individual Windows-based desktop computers, servers, and Active Directory servers.

each column in the auth.txt and generated a second enumerated list that we used as an ID. After, we made a dictionary with those two lists with the keys as the string input and the mappings as the ID output. We then ran each row from the auth.txt data frame, and input the data from the data frame into their respective dictionaries. Then, from the dictionaries we put the ID output divided by the size of their respective column dictionaries into a temporary data frame then appended a processed data frame. A bar graph, Figure 4, was used to determine source of attack. From this graph, 4 computers were identified as sources, with 1 producing the majority of threats. This finding helped us recognize a source of bias as the model is more likely to label anything from C17693 as a threat.

5 Project Implementation

Before we began to work on our project, we had to decide who would be responsible for what parts of the project. Ron took responsibility of the notebook because he has the most coding experience in the group. Claudia and Abraham would split the responsibilities of the required writings and the presentation design.

To decide on a plan for our model, we first did extensive research into machine learning

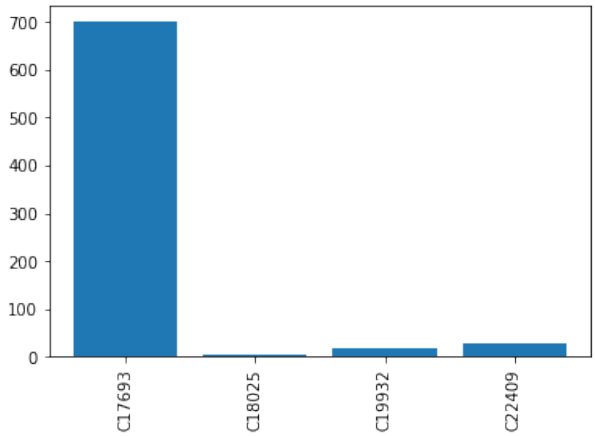

Figure 2: This data presents specific events taken from the authentication data that present known redteam compromise events. These may be used as ground truth of bad behavior that is different from normal user and computer activity.

models for rare event detection. We decided on a KNN model of the simplicity of KNN models and because other experiments relating to rare event detection have been very successful using KNN models. Our KNN model utilizes Euclidean, the default metric from the SKlearn library. Our model uses 5 neighbors, which is also the default from the SKlearn library. We kept these at the default value to maintain the simplicity of our model. We set verbose to 0 to prevent high run times. 5 cross validation folds are used in an attempt to get a higher accuracy while using a small percent of the data. Almost all of our problems in the project stem from the enormous size of our data set. Simply downloading the data required the deletion of files from our PC's hard drive and an increase in the size of the page file system. Another problem was the slow run-time of train_test_split on our data. The machine used has 16 gigabytes of RAM, an AMD Ryzen 7 3750H CPU, and a GeForce GTX 1660Ti graphics card but still struggled to split the data. This actually led to a nearly complete halt in progress for two days while a different solution was being created. Eventually, we decided on an alternative splitting method that used a much smaller percentage of the data set than we previously intended to use. This alternative method consisted of gathering the threats and labeling the data, indexing the threats and putting them in a list, and sampling the list to be stratified. Finally, 5% of the main data was split for training and 1% for testing.

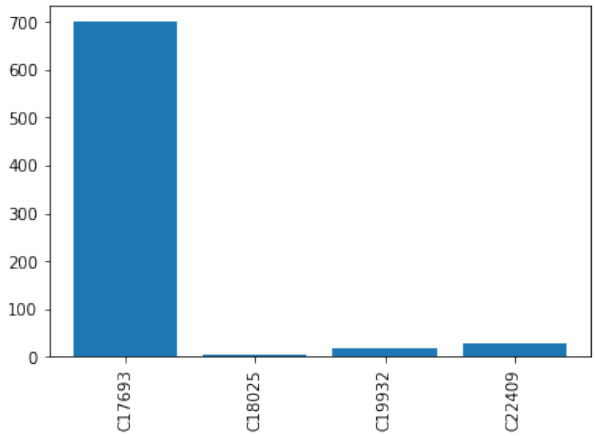

Figure 3: IDs of computers producing threats

6 Experimental Results

The data set is over twelve gigabytes in size and contains an uneven mix of threats and non-threats to computer networks with less than 1% of data classified as a threat. Because of the rarity of threats to computer networks, a large data set is required to be able to train the model. A threat column classifier was created to identify threats with a 1 and non-threats with a 0. Code was written to iterate through each element of the data-frame counting network threats (redteam_df). When an element from redteam_df matches an element from the data-frame containing both threats and non-threats (aut_df), the element is set to threat = 1 and appended into the threats data-frame. The data set was split into 5% train and 1% test sets of data. The model has a k value of 5. This value was chosen because it is the default k-value from the SKlearn library. The model uses the standard euclidean metric. A mean accuracy of 99.9% was achieved after running the model.

	time	source_user@domain	destination_user@domain	source_computer	destination_computer	authentication_type	logon_type	authentication_orientation	success/failure	threat
0	1	0.124328	0.101610	0.098829	0.825606	0.827586	0.4	0.285714	1	0
0	1	0.124328	0.101610	0.838016	0.825606	0.000000	0.4	0.142857	1	0
0	1	0.155612	0.127167	0.995379	0.984838	0.000000	0.4	0.142857	1	0
0	1	0.156630	0.560423	0.007948	0.008304	0.862069	0.8	0.285714	1	0
0	1	0.156717	0.128070	0.008318	0.842403	0.103448	0.4	0.285714	1	0

Figure 4: Output of code showing if element is a 0 or 1.

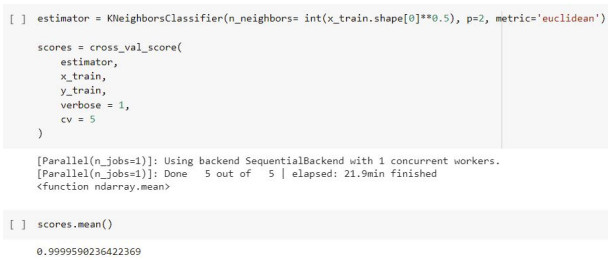

Figure 5: KNN model with mean accuracy shown.

7 Conclusion

A KNN model was built with the goal of identifying adversarial attacks on computer networks. The proposed approach was trained and tested with data collected from the Los Alamos National Laboratory's corporate, internal computer network.The KNN model was able to achieve a mean accuracy of 99.9%. Our model adds further credibility to KNNs as a method of threat detection and shows that KNNs can be utilized even when dealing with extremely large data sets. With more time, we could have found ways to reduce run time and improve the accuracy of the model. Changes to the k value, cross validation layers, or a switch from the euclidean metric are examples of some possible modifications.

8 Acknowledgment

We would like to give a special thanks to Dr. Rainwater for mentoring us while we worked on our project. We would also like to thank all the various professors and TAs that have taught/assisted us throughout the duration of the program.

References

[1]"The Largest Data Breaches in U.S. History" Spanning, 5 Nov. 2020, spanning.com/resources/industry-research/largest-data-breaches-us-history/.

[2] MFoxCNBC. "10 Ways Companies Get Hacked." CNBC, CNBC, 10 July 2012, www.cnbc.com/2012/07/06/10-Ways-Companies-Get-Hacked.html.

[3] Oladimeji, Tolulope O., et al. "Insider Threat Detection Using Binary Classification Algorithms." IOP Conference Series: Materials Science and Engineering, vol. 1107, no. 1, 2021, p. 012031., doi:10.1088/1757-899x/1107/1/012031.

[4] Okfalisa, et al. "Comparative Analysis of k-Nearest Neighbor and Modified k-Nearest Neighbor Algorithm for Data Classification." IEEE Xplore, 8 Feb. 2008, ieeexplore.ieee.org/abstract/document/8285514.

[5] Li, Wenchao, et al. "A New Intrusion Detection System Based on KNN Classification Algorithm in Wireless Sensor Network." Journal of Electrical and Computer Engineering, Hindawi, 30 June 2014, www.hindawi.com/journals/jece/2014/240217/.

[6] Guo G., Wang H., Bell D., Bi Y., Greer K. (2003) KNN Model-Based Approach in Classification. In: Meersman R., Tari Z., Schmidt D.C. (eds) On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE. OTM 2003. Lecture Notes in Computer Science, vol 2888. Springer, Berlin, Heidelberg.

[7] Shmilovici, Armin. "Support Vector Machines." Springer Link, Ben-Gurion University, Beer-Sheva, Israel, 7 July 2010.

[8] Bansal, Atul, et al. "SVM Based Gender Classification Using Iris Images." IEEE Xplore, 6 Dec. 2012.

[9] Wang, Senzhang, et al. "Training Data Reduction to Speed up SVM Training." Applied Intelligence, vol. 41, no. 2, 2014, pp. 405-420.

[10] Sabar, Nasser R., et al. "A Bi-Objective Hyper-Heuristic Support Vector Machines for Big Data Cyber-Security." IEEE Access, vol. 6, 15 Mar. 2018, pp. 10421-10431., doi:10.1109/access.2018.2801792.

[11] Gupta, Govind P., and Manish Kulariya. “A Framework for Fast and Efficient Cyber Security Network Intrusion Detection Using Apache Spark.” Procedia Computer Science, Elsevier, 12 Aug. 2016, www.sciencedirect.com/science/article/pii/S1877050916314806.

[12] “Bayes’ Theorem - Definition, Formula, and Example.” Corporate Finance Institute, 8 July 2021, corporatefinanceinstitute.com/resources/knowledge/other/bayes-theorem/.

[13] Aggarwal, Shruti, and Devinder Kaur. “Enhanced Smoothing Methods Using Naive Bayes Classifier for Better Spam Classification.” International Journal of Engineering Research & Technology, IJERT-International Journal of Engineering Research & Technology, 30 Sept. 2013.

[14] Hashmani, Manzoor Ahmed, et al. “An Ensemble Approach to Big Data Security (Cyber Security).” International Journal of Advanced Computer Science and Applications, vol. 9, no. 9.

[15] Brao, Bobba, and Kailasam Swathi. “Fast KNN Classifiers for Network Intrusion Detection System.” Indian Journal of Science and Technology, vol. 10, no. 14, Apr. 2017, pp. 1-10.

[16] Kajal, Abhishek, and Sunil Kumar Nandal. “A Hybrid Approach For Cyber Security: Improved Intrusion Detection System Using Ann-Svm.” Indian Journal of Computer Science and Engineering, vol. 11, no. 4, 2020, pp. 412-425.

[17] Z. Muda, W. Yassin, M. N. Sulaiman and N. I. Udzir, “Intrusion detection based on K-Means clustering and NaÃ¯ve Bayes classification,“ 2011 7th International Conference on Information Technology in Asia, 2011, pp. 1-6.

[18] Alqahtani H., Sarker I.H., Kalim A., Minhaz Hossain S.M., Ikhlaq S., Hossain S. (2020) Cyber Intrusion Detection Using Machine Learning Classification Techniques. In: Chaubey N., Parikh S., Amin K. (eds) Computing Science, Communication and Security. COMS2 2020. Communications in Computer and Information Science, vol 1235. Springer, Singapore.

[19] Kolev, Alexander, and Pavlina Nikolova. “Instrumental Equipment for Cyberattack Prevention.“ Information & Security: An International Journal 47, no. 3 (2020): 285-299.

[20] N. Kumar Singh et al., “Identification and Prevention of Cyber Attack in Smart Grid Communication Network,“ 2019 International Conference on Information and Communications Technology (ICOIACT), 2019, pp. 5-10.