CSCI-735 Project Phase 1: Conventional IDS Design

Parijat Kawale
pk7145@rit.edu
Rochester Institute of Technology
Rochester, New York, USA

Archit Joshi
aj6082@rit.edu
Rochester Institute of Technology
Rochester, New York, USA

ACM Reference Format:
Parijat Kawale and Archit Joshi. 2023. CSCI-735 Project Phase 1: Conventional IDS Design. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 EXECUTIVE SUMMARY

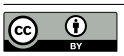
The current cybersecurity infrastructure relies highly on the utilization of Intrusion Detection Systems (IDS), which monitor a network or systems for malicious activities or policy violations. While IDS has little say in remedying these violations it's a good start to make the user aware of any impending threats. In this project phase, we have researched, learned, and worked towards implementing an intrusion detection system (IDS) that would be able to detect a possible intrusion in the network landscape. The final result obtained based on the custom anomaly-based IDS is a 50% accuracy in attack detection, while the custom misuse-based IDS resulted in 96% accuracy in attack detection with the least number of false positives.

For the current phase of our project, we have used the KDD Dataset[5] which was retrieved from Kaggle[1]. The KDD dataset contains samples for the following attacks - snmpgetattack, named, xlock, smurf, ipsweep, multihop, xsnoop, sendmail, guess_passwd, saint, buffer_overflow, portsweep, pod, apache2, phf, udpstorm, warezmaster, perl, satan, xterm, mscan, processtable, ps, nmap, rootkit, neptune, loadmodule, back, httptunnel, worm, mailbomb, ftp_write, teardrop, land, sqlattack, snmpguess

In the initial approach, we worked on setting up two different IDS using SNORT[3] and Suricata[4] as the tools for an anomaly-based and misuse-based IDS. However, while working and researching these tools, we were unable to accommodate them to our needs with stale data or older data as these tools are generally used in live networks. The reason for this will be discussed in upcoming sections. So, we built a custom anomaly-based intrusion detection system based on the Isolation Forest machine learning model and a custom misuse-based intrusion detection system based on the Support Vector Machine model. The usage of custom IDS resulted in better visualization of the data and understanding of behaviors of certain parameters that would determine a specific class of attack.

2 SPECIFICATION

The initial expectation during implementation was to use an existing IDS tool such as SNORT[3] or Suricata[4] to develop two different IDS i.e. anomaly-based IDS and misuse-based IDS.



Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

However, based on the extensive research on both tools, we came to the conclusion that these tools do not work well with stale data and are more accustomed to working with live monitoring of the network and/or the system. We initially tried to convert the existing dataset into a more acceptable format of these tools-Packet Capture (PCAP) format. However, these generated PCAP files did not contain all the information required by both SNORT and Suricata to appropriately detect the type of attack. Hence, based on these conclusive results, we developed a custom anomaly-based IDS and misuse-based IDS in python using scikit-learn packages. In order to develop a misuse-based IDS, we studied some of the AI and Machine Learning algorithms such as Support Vector Machine in order to create a classification model that would determine if a packet is safe or not after analyzing its structure.

Next, for an anomaly-based IDS we used the Isolation Forest model which resulted in better detection of anomalies present in the data. The reason for choosing the Support Vector Machine was due to the better accuracy and lower amount of false positives. However, the reason for the usage of the Isolation Forest model is to detect and extract anomalies, which can be done better using this model rather than a simple Random Forest.

We use scikit-learn's train_test_split package to divide the KDD data into test and training sets with a ratio of 60:40. The final model was then tested on the testing set for each IDS to classify different types of attacks and detect anomalies.

3 METHODS AND TECHNIQUES

3.1 Data preparation

In order to use the KDD dataset[5] for the custom IDS tool, we had to preprocess the data to clean it so that it could be used to develop the models. Initially, we normalized the data and scaled it in the range [0,1]. This was achieved using the MinMaxScaler function in the sklearn.preprocessing[2] module. Using the scaler ensured that all the different outputs have an equal weightage as well as improved the convergence of the model based on the data.

After normalizing the data, we encode the training data using one-hot encoding technique. This encoding converts the output data, which is in a non-numerical format to a numerical format, that is, assigning a number to each type of categorical attribute. This is used to ensure that the machine learning model can understand and treat different classes of output based on the different output numbers assigned. The total data length resulting after the preprocessing is a packet record of 311029 packets in total, each containing 42 different attributes.

3.2 Misuse based IDS

As discussed in the previous sections, the usage of existing tools such as SNORT, and Suricata were not useful to operate on stale

data to detect attacks. Hence, to create a misuse-based IDS that can detect any misuse of the data in the network or the misuse of the network resources, we developed a custom classifier that can detect these attacks.

This classifier detects misuse of the system using a Support Vector Machine classifier model at its core to distinguish between normal use and misuse of the system. The Support Vector Machine Classifier uses the RBF kernel, which implies that the decision boundary between the classes will be nonlinear in nature. SVM classifier also contains a regularization parameter that controls the trade-off between maximizing the margin between classes and minimizing classification errors.

3.3 Anamoly based IDS

Similar to the misuse-based IDS, the anomaly-based IDS was developed using the Isolation Forest machine learning model. The Isolation Forest model is an approach to detect and identify anomalies and outliers in the dataset that would be considered unconventional user behavior when compared to other process usages.

This unsupervised learning model is based on contamination of the rate present in the dataset. The contamination rate essentially tells the model to look out for a certain number of anomalies that are present. For example, if the contamination rate is set to 0.5 then it would mean that the model should expect approximately 50% of the dataset to contain anomalies in it.

4 IMPLEMENTATION

The Python code is dependent on the scikit-learn library for its machine-learning models.

4.1 Structure

The overall structure is a directory named "CSCI-735-main". Inside this directory, we have a python3 file that contains the custom models defined in their respective functions and a data folder that contains the actual data from the dataset.

4.2 Software and Hardware Requirements

- Cores: 4
- RAM: minimum 4GB
- Python3 installed
- Machine learning libraries: scikit-learn, numpy, pandas

4.3 Limitations

The machine should have at least four cores for the software to run properly. Please note that running the machine learning model takes a considerable amount of time.

4.4 User guide

Please note this user guide assumes that you have satisfied the software and hardware requirements mentioned in section 4.2

(1) Download the "CSCI-735-main.zip"
(2) Extract the zip file.
(3) Inside the directory, we have the data folder and and python file.
(4) Run the python file inside the directory location using: *python3 idsClassifier.py*
(5) The program will generate heatmap for the confusion matrix after each classifier finishes running. Thus please close the heatmap so that the program resumes execution.

5 TESTS

In order to conduct testing on the model developed, we tried and tested different strategies to choose the best strategy from all of them. Firstly, we tried different split strategies such as dividing the dataset into 60% training data and 40% testing data or 70% training data and 30% testing data and so on. These splits were done to choose the best split of the data to get the maximum accuracy and have the maximum amount of data to train the model. The splits also allow in determining how well the model would be towards generalization. A higher proportion for the test set accompanied with a higher accuracy suggests that the model can handle unseen data efficiently.

Along with splitting the dataset into training and testing, we tuned the hyperparameters for each of the models - c value and gamma for the Support Vector Machine classifier, and the contamination factor for Isolation Forest Model - to achieve the highest accuracy possible based on the dataset.

A combination of splitting the dataset into training and testing data and tuning the hyperparameters to achieve the best results resulted in the misuse-based IDS having an accuracy of attack detection to be 96% and anomaly-based IDS having an accuracy of 80% . The anomaly-based IDS uses the Isolation Forest model which has an underlying assumption that at least 50% of the dataset will be normal data. However, in our case, this is not true as the dataset contains about 70% of attack data and 30% of normal data. This is a reason why the accuracy of anomaly-based IDS is lower than that of misuse-based IDS.

6 RESULTS

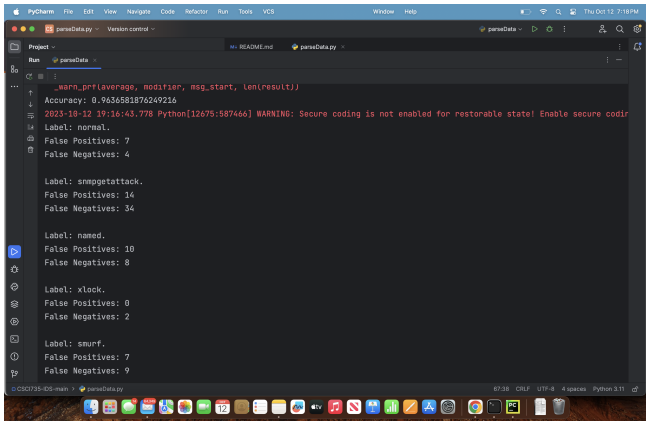


Figure 1: Console output from Misuse-based IDS using SVM model

As seen in Figure 1, the accuracy for the SVM model for misuse-based IDS is 96.36% accuracy. The false positives and false negatives

for most attacks remain on the lower end which suggests that the classifier is correctly identifying the intrusions from the data. The entire output for the entire program can be seen in the console once the program is run. For simplicity, we have also included a "misuse_results.txt" file which contains the program results.

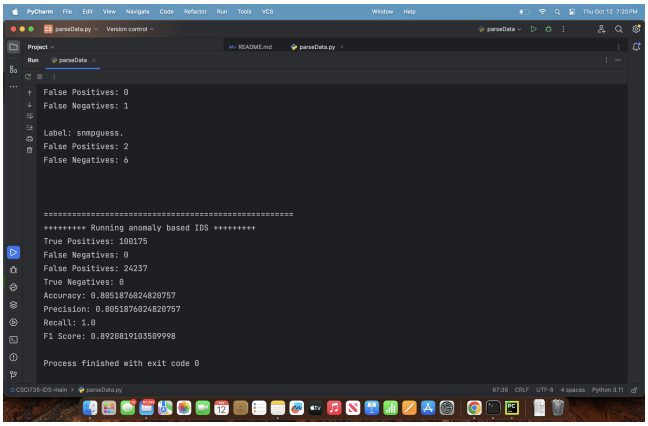


Figure 2: Console output from Anomaly-based IDS using Isolation Forest Model

As seen in Figure 2, the accuracy for the Isolation forest model is 80.51%. As this model is classified on a binary scale - anomalous or normal, we do not have false positive and false negative ratios for each label. We were able to successfully check for such instances in our model up to 80 times out of 100.

7 DEVELOPMENT PROCESS

The process of development started with firstly understanding the data and its attributes. The dataset used for our project is the raw KDD dataset[5]. This involved reviewing basic network concepts such as the packets structure which would allow us to understand what each attribute meant. Once the dataset was available and loaded into a pandas dataframe it was easy to access for further operations.

The next step in the process was to explore and attempt to detect any attacks that present the data using the tools that are readily available in the market. The two tools chosen between me and my partner were SNORT[3] and Suricata[4]. After deciding on the tools, we worked on various ways to manipulate the data such that it would be accepted by these tools for intrusion detection. However, after spending about a week to a week and a half, we came to the conclusion that it would not be possible to use the current dataset and run it on SNORT or Suricata, as these tools are based on live network monitoring and had difficulties in reading the 42 attributes which would be included in the packet's payload. Then, we decided to develop a custom anomaly-based IDS and a custom misuse-based IDS using the scikit-learn library available in the Python programming language.

We choose the Support Vector Machine classifier and Isolation Forest Model as the machine learning algorithms to be used by misuse-based IDS and anomaly-based IDS respectively. We then pre-processed the data and developed to models to be able to detect intrusions to the system. The rest of the development was focused on data cleaning and hyperparameter tuning.

REFERENCES

[1] [n. d.]. Kaggle. https://www.kaggle.com/
[2] [n. d.]. Scikit-Learn. https://scikit-learn.org/stable/
[3] [n. d.]. SNORT IDS tool. https://www.snort.org/
[4] [n. d.]. Suricata IDS tool. https://suricata.io/
[5] Steven Huang. [n. d.]. KDD Cup 1999 Data. https://www.kaggle.com/datasets/galaxyh/kdd-cup-1999-data