Intrusion Detection System using AI and Machine Learning Algorithm

Syam Akhil Repalle[1], Venkata Ratnam Kolluru[2]

[1] Student, Department of Electronics and Communication Engineering, Koneru Lakshmaiah Educational Foundation, Andhra Pradesh, India
[2]Associate Professor, Department of Electronics and Computer Science, Koneru Lakshmaiah Educational Foundation, Andhra Pradesh, India

---

Abstract - *Secure automated threat detection and prevention is the more effective procedure to reduce the workload of analyst by scanning the network, server functions & then informs the analyst if any suspicious activity is detected in the network traffic. It monitors the system continuously and responds according to the threat environment. This response action varies from phase to phase. Here suspicious activities are detected by the help of an artificial intelligence which acts as a virtual analyst concurrently with network intrusion detection system to defend from the threat environment and taking appropriate measures with the permission of the analyst. In its final phase where packet analysis is carried out to surf for attack vectors and then categorize supervised and unsupervised data. Where the unsupervised data will be decoded or converted to supervised data with help of analyst feedback and then auto-update the algorithm (Virtual Analyst Algorithm). So that it evolves the algorithm (with Active Learning Mechanism) itself by time and become more efficient, strong. So it can able to defend form similar or same kind of attacks.*

Key Words: **Artificial Intelligence, Intrusion Detection System, Network Security, Machine Learning**

1. Introduction

There are many types of dangers on the internet, including malware and DDOS attacks. A network can be protected against such attacks using an intrusion detection system. An IDS system can detect intrusions and intrusion de-generates an alert when it detects an intrusion. This intrusion detection system in a network analyzes all traffic. For large datacenters this is a difficult task. There's an enormous amount of data through the network of a data center. Standard intrusion systems cannot then all traffic completely.

A way to fix this is by IP flows is regeneration of packet data. Using IP flows ensures that an intrusion detection system can check all traffic. Intrusion detection systems also require a lot of maintenance. This depends on course and also involves high cost. Sensitive data is also increasingly being stored digitally. All these new services could contain security flaws which could leak private data, such as passwords or other sensitive data. This means that security flaws become more and more important since they can cause so much damage. It is not just the leaking of sensitive data that is an issue, but also protecting a computer or network against malware is important.

Considering this, it becomes more important to be able to detect and prevent attacks on network systems. Intrusion detection systems are used for this purpose. An intrusion detection system can alert administrators of malicious behavior. In order to have good performance, most intrusion detection systems need a lot of manual maintenance. This thesis tries to find out whether an intrusion detection system can work out-of-the-box with an acceptable performance. This is done by using machine learning algorithms. These are algorithms which can learn and find patterns in input. Machine learning algorithms seem promising for the problem of automatic intrusion detection. This thesis tries to view therefore that an intrusion detection system out-of-the-box may have a good performance. This is done via machine learning algorithms. These are algorithms that can learn from data and patterns. This seems well applicable to the problem of intrusion detection, this will also view this thesis, as well as the algorithms may or may not work.

2. Attacks Classification

A useful classification is to first make a distinction between internal and external malicious behaviour. This makes it easier for humans to understand. The IDS itself can work with different kind of classifications. However, the IDS have to communicate with an administrator about the detections. A distinction between internal and external malicious behaviour is easier to understand. Every type of malicious behaviour is identified by different characteristics. Knowing these characteristics is useful to be able to tweak the IDS to make identification more effective.

2.1. External Abnormal Behavior

External abnormal behaviour consists of different kind of attacks on the systems. There are much different type of attacks. There are Physical attacks, Buffer overflows, Distributed Denial of Service, Brute- force attacks, Vulnerability scans and Man in the middle attacks.

2.2. Internal Abnormal Behavior

Internal abnormal behaviour can be called malware. There are several types of malware. There are four distinct categories of malware. There are Botnets, Viruses, Trojan Horses and Worms. Malware are actual programs that infect a system to execute a specific task. The task of the malware defines which category the malware belongs in.

2.3. Detection

An NIDS only monitors the network. As such not every attack can be detected by an NIDS. Only the attacks that actually use the network can be detected. Flow-based IDS have the additional constraint that they can only use flow data. This further limits the attacks that can be detected. The attacks that can generally be detected using flow-based network intrusion detection systems are DDOS, Vulnerability Scans, Worms and Botnets.

3. Intrusion Detection System

An intrusion detection system is a system which tries to determine whether a system is under attack, to detect intrusions within a system. Intrusion detection systems are often called IDS's. Intrusions can also be called attacks or anomalies. It does this by monitoring network or system activities. One way of categorizing IDS's is based on the method of detection intrusion.

3.1. Host-Based Intrusion Detection System

Host-based intrusion detection systems are systems that monitor the device on which they are installed, or directly connected to. The way they monitor the system can range from monitoring the state of the main system through audit logs, to monitoring program execution. Since HIDS rely so much on audit logs, they can become limited by them. Another issue can be the sheer volume of the audit logs. Every monitored log needs to be parsed; this means that the HIDS can have a big impact on the performance of the host system if it is installed there.

Another disadvantage is that any vulnerability that causes the audit files to be changed, also impacts the integrity of the HIDS. If an audit file is changed, the HIDS cannot see and detect what truly happened.

3.2. Network-Based Intrusion Detection System

Network-based intrusion detection systems are placed at certain points within a network in order to monitor traffic from and to devices within the network. They operate on the same concept as wiretapping. They "tap" into a network and listen to all communication that happens. The intruder could try to minimize his network activity, but the risk is lower. NIDS are also more portable than HIDS. They monitor traffic over a network and are independent of the operating system they run on. The system can analyze the traffic using multiple techniques to determine whether the data is malicious. There are two different ways to analyze the network data. *Packet-based analysis* uses the entire packet including the headers and payload. An intrusion detection system that uses packet-based analysis is called a packet-based network intrusion detection system. The advantage of this type of analysis is that there is a lot of data to work with. Every single byte of the packet could be used to determine whether the packet is malicious or not. *Flow-based analysis* doesn't use individual packets but uses general aggregated data about network flows. An intrusion detection system that uses flow-based analysis is called a flow-based network intrusion detection system. A flow is defined as a single connection between the host and another device.

3.3. Intrusion Prevention System

An intrusion prevention system or IPS/IDPS is an intrusion detection system that also has to ability to prevent attacks. An IDS does not necessarily need to be able to detect attacks at the exact moment they occur, although it is preferred. An IPS needs to be able to detect attacks real-time since it also needs to be able to prevent these attacks. For network attacks these prevention actions could be closing the connection, blocking an IP or limiting the data throughput.

The change to requiring attacks to be detected at real time can severely impact the methods that are used to detect these attacks. For example, an IDS might give an alert even though the IDS is not certain that whatever it is alerting is actually an anomaly. An IPS needs to be certain before it can take action. Otherwise the IPS might take actions which the business employing the IPS does not want.

3.4. Detection

There are multiple different methods to detect intrusions. There are Signature based and Anomaly Based methods.

Signature based methods compare so called "signatures" with an existing database of signatures. A packet or flow record is decomposed into features that together construct a signature. If the signature of an incoming flow or packet matches with a signature in the database, it is flagged as malicious. Signature-based methods have little overhead in both computation and preprocessing as it only tries to match incoming signatures to known signatures in the database. Because it only compares signatures, it is easy to deploy within a network. The system does not need to learn what the traffic within a network looks like. Signature based methods are very effective against known attacks. New attacks cannot be detected unless the database is updated with new signatures. It is also possible for attackers to avoid being caught by signature based methods, only a slight modification of the "signature" is required in order to bypass the exact matching.

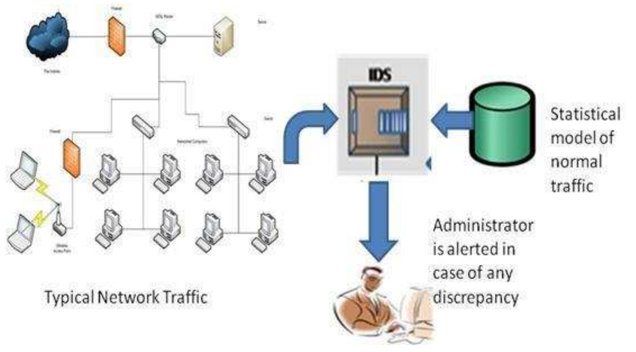

IDS
Statistical model of normal traffic
Administrator is alerted in case of any discrepancy
Typical Network Traffic

Fig-1: Signature Based IDS

Anomaly based methods; also called Behaviour based methods are methods in which the IDS try to model the behaviour of network traffic. When an incoming packet deviates from this model, it is flagged as malicious and an alert is sent. Because they use a statistical model of normal behaviour, they should be able to detect all deviations from this normal behaviour. As a result, new attacks that deviate too much from normal behaviour are detected as well.

Since a model of the network traffic needs to be created, the system cannot be deployed into a network and be expected to work. The system needs to learn the behaviour of the network traffic. Problems, such as generating a lot of false positive alarms, can arise when training data includes mistakes, such as misclassifications. Machine learning algorithms can be used as an anomaly based method. Machine learning techniques have the ability to learn from data and decide whether new data is malicious.

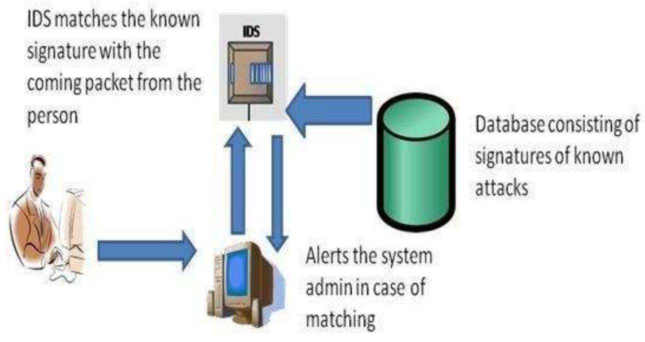

IDS matches the known signature with the coming packet from the person
IDS
Database consisting of signatures of known attacks
Alerts the system admin in case of matching

Fig-2: Anomaly Based IDS

4. Machine Learning

Machine learning is a subfield of Computer Science. It is a type of Artificial Intelligence which allows programs to learn and find patterns within data. Machine learning explores algorithms that can learn from and make predictions on data. These algorithms are called machine learning algorithms. A machine learning algorithm has to learn before it can be used to make predictions on data. Learning means that the algorithm has to be shown several examples of data and what the correct predictions for these examples would be. The amount of examples that have to be shown to the algorithm can be within the range of several thousands.

Once the machine learning algorithm has learned from the data, it can be used to make predictions on other data. For example, machine learning can be used in order to watch the heart rate of patients at a hospital. During the learning phase, the machine learning algorithm is shown the heart rate of a patient and the current time. After learning is done, the machine learning algorithm can predict what the heart rate of that patient should be based on the current time. This can be used to determine whether the patient's heart rate is normal by comparing the predicted heart rate and the real heart rate.

There are two classes of machine learning algorithms. There is supervised learning and unsupervised learning. Supervised learning is trained using labeled data. Unsupervised learning uses unlabeled data. The data used to train machine learning algorithms is called a training set.

4.1. Evaluating ML for an IDS

With a machine learning algorithm, performance can be measured using the F-score. However, for intrusion detection systems, this is not enough by itself. The F-score assumes that recall and precision has the same importance. This is not necessarily the case when evaluating intrusion detection systems. A false positive occurs when a sample is actually Normal but is classified as an Intrusion. A false negative occurs when a sample is actually an Intrusion but is classified as Normal. A false negative is bad, since it means that an Intrusion was not detected. But most IDS's are used in a layered approach. This means that if one layer does not detect an Intrusion, another layer might detect it.

The layered approach might also work completely different. Perhaps the first layer tries to detect as many anomalies as possible (and having a low recall) and then passing the data for which anomalies have been detected to other layers. This approach means that a low recall is not bad. The scoring used for IDS that uses machine learning is dependent on how the IDS is going to be used.

4.2. Using ML for IDS

Data has to be processed before it can be used within a machine learning algorithm. This means that features have to be chosen. Some features can be easy to find, other have to found by experimenting and running tests. Using all the features of a dataset does not necessarily guarantee the best performances from the IDS. It might increase the computational cost as well as the error rate of the system. This is because some features are redundant or are not useful for making a distinction between different classes.

5. Implementation

5.1. Technology Stack

The main library that was used is scikit-learn. Scikit-learn is a robust machine learning library for Python. It is build upon NumPy, SciPy, and matplotlib. It is also open source and commercially usable with the BSD license. learn. This library was chosen since the library offers the most important algorithms, the documentation. Scikit-learn also contain different methods to visualize machine learning algorithms such as a graph to show the learning curve. These can be a useful tool to evaluate the performance of machine learning algorithms. It also contains methods to calculate the F-score. This is useful since that means that mistakes when calculating the F-score are less likely to happen.

5.1.1. Program Execution

The implementation works in different steps. A JSON config file is used to define the elements that are used within the program. This contains the data to be used for learning, for checking, the machine learning algorithm, etc.. Once the config file has been read, the program can start the training phase. In this phase the specified algorithm is used and trained using the given data. Afterwards the prediction phase starts. This phase uses the prediction data and gathers all results. The structure of the program and the modules reflect these different phases.

5.1.2. Structure

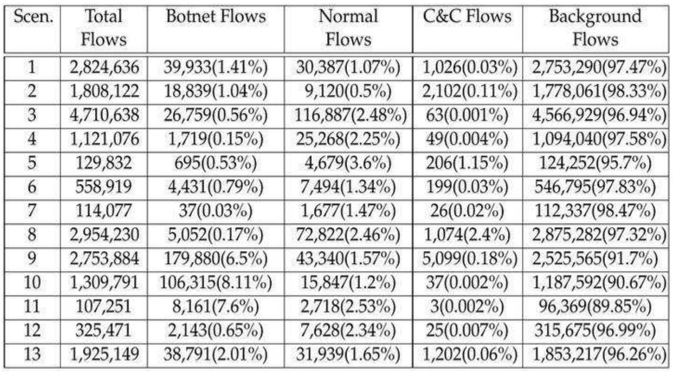
The implementation is build to be modular. The first module is the machine learning module. This module contains all machine learning algorithms that can be used. There is also a feature module. This module contains the available classes that can be used to extract features from the flows. A loader module contains all classes required to load the data from the different datasets.

A training module contains the different classes used for training. These classes use a loader class and pass the data to the machine learning algorithm. They define which data is supposed to be used (for example, using only abnormal behaviour and leaving out the normal behaviour). Finally there is a results module. This module receives all the output from the machine learning algorithms and has to log these or visualize them.

5.2. Datasets

In order to test the implementation and the algorithms, different datasets were used. Each dataset is used to test a different aspect of the machine learning algorithms. First, a subset of a dataset has to be chosen to be fed to the machine learning algorithms for learning. Afterwards, using the method, the algorithm is tested using another subset of the same dataset.

In the next step, the algorithms are tested using real-world data that is labeled. Finally, in the fourth step, the algorithms are tested using raw, unlabeled real-world data. This is to make sure that the algorithm performs well on unprocessed real-world data. Several datasets have been used to test the machine learning algorithms.

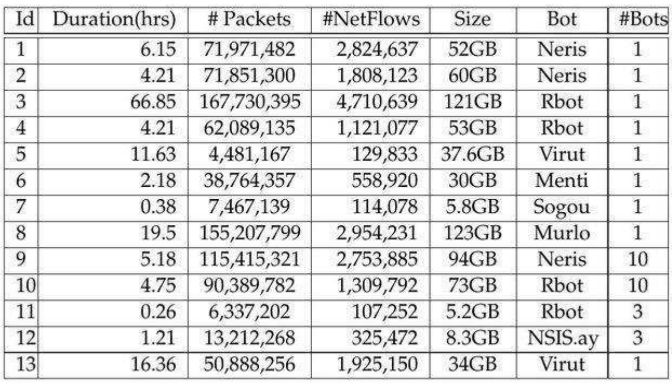


Scen.	Total Flows	Botnet Flows	Normal Flows	C&C Flows	Background Flows
1	2,824,636	39,933(1.41%)	30,387(1.07%)	1,026(0.03%)	2,753,290(97.47%)
2	1,808,122	18,839(1.04%)	9,120(0.5%)	2,102(0.11%)	1,778,061(98.33%)
3	4,710,638	26,759(0.56%)	116,887(2.48%)	63(0.001%)	4,566,929(96.94%)
4	1,121,076	1,719(0.15%)	25,268(2.25%)	49(0.004%)	1,094,040(97.58%)
5	129,832	695(0.53%)	4,679(3.6%)	206(1.15%)	124,252(95.7%)
6	558,919	4,431(0.79%)	7,494(1.34%)	199(0.03%)	546,795(97.83%)
7	114,077	37(0.03%)	1,677(1.47%)	26(0.02%)	112,337(98.47%)
8	2,954,230	5,052(0.17%)	72,822(2.46%)	1,074(2.4%)	2,875,282(97.32%)
9	2,753,884	179,880(6.5%)	43,340(1.57%)	5,099(0.18%)	2,525,565(91.7%)
10	1,309,791	106,315(8.11%)	15,847(1.2%)	37(0.002%)	1,187,592(90.67%)
11	107,251	8,161(7.6%)	2,718(2.53%)	3(0.002%)	96,369(89.85%)
12	325,471	2,143(0.65%)	7,628(2.34%)	25(0.007%)	315,675(96.99%)
13	1,925,149	38,791(2.01%)	31,939(1.65%)	1,202(0.06%)	1,853,217(96.26%)

Fig-3: Distribution of labels in CTU 13 Dataset.

The CTU-13 dataset has been used for steps one to three for the testing of the machine learning algorithms. This is a labeled dataset. It contains botnet behaviour, normal and background traffic. The data was captured in the CTU University, Czech Republic, in 2011. It consists of thirteen different captures, each of which runs a different botnet malware. Figure 4 shows the amount of data within each capture. Note that the captured data is only from a couple hours. The flows within the dataset contain extra information. Each capture contains only a small amount of botnet samples. Most flows are background flows.

This is expected of botnet behaviour since it does not generate an large amount of network traffic. Each flow is labeled with its exact source. This could be Google analytics, Google webmail or a windows update. The flows within the dataset only contain the regular information that is found within net flow. The abnormal behaviour within this dataset is internal abnormal behaviour. In the evaluation chapter, this dataset is called the CTU dataset.



Id	Duration(hrs)	# Packets	#NetFlows	Size	Bot	#Bots
1	6.15	71,971,482	2,824,637	52GB	Neris	1
2	4.21	71,851,300	1,808,123	60GB	Neris	1
3	66.85	167,730,395	4,710,639	121GB	Rbot	1
4	4.21	62,089,135	1,121,077	53GB	Rbot	1
5	11.63	4,481,167	129,833	37.6GB	Virut	1
6	2.18	38,764,357	558,920	30GB	Menti	1
7	0.38	7,467,139	114,078	5.8GB	Sogou	1
8	19.5	155,207,799	2,954,231	123GB	Murlo	1
9	5.18	115,415,321	2,753,885	94GB	Neris	10
10	4.75	90,389,782	1,309,792	73GB	Rbot	10
11	0.26	6,337,202	107,252	5.2GB	Rbot	3
12	1.21	13,212,268	325,472	8.3GB	NSIS.ay	3
13	16.36	50,888,256	1,925,150	34GB	Virut	1

Fig-4: Amount of data and botnet type for each capture.

5.3. Algorithm selection

Both supervised and unsupervised algorithms have been used. The algorithms are the most common algorithms. Before more complex algorithms such as deep neural networks should be used, the more common and general algorithms should be tested.

5.3.1. Unsupervised learning

K-Means clustering is used in order to test whether results can be found using clustering algorithms. K-means is a simple clustering algorithm and already gives an indication whether a problem can be solved using clustering, or whether clustering offers no advantage. However, no method was found to verify whether to clusters that the K-means algorithm made were correct.

One-class Support Vector machines are used in an attempt to use binary classification. They are quite fast in execution. They were used to find out whether it is a viable technique to preprocess incoming data and check whether a One-class Support Vector machine finds it to be abnormal behaviour before passing it to other algorithms.

5.3.2. Supervised learning

Support vector machines have been used in the implementation. It is a popular algorithm and can do both linear and non-linear classification which makes it a promising choice to test in the implementation.

K-nearest Neighbors was the most promising algorithm. This algorithm is used extensively through throughout the implementation and the tests. The fact that the classification happens on basis of the different neighbors instead of trying to make a classifier seemed to fit the feature data better.

Through the study of different machine learning algorithms, decision tree algorithms and Bayesian algorithms have also been discussed. They seemed less promising for the problem of intrusion detection. The difference between a normal flow and an abnormal flow is very slight and it seemed that these algorithms would make more mistakes. They are still used in the implementation to find out whether this assumption is correct or not.

6. Literature Review

MIT is also working on methods to use machine learning to defend against cyber attacks. In their paper "AI^2: Training a big data machine to defend", they present a new method. Their system has four components. A big data processing system, an outlier detection engine, a mechanism to obtain feedback from security analysts, and a supervised learning module.

Their system tries to combine the expertise of security experts, and the speed and ability to detect new attacks of machine learning. More specifically, they use unsupervised machine learning. They preferred to use unsupervised machine learning since labeled data is rare and attacks constantly evolve. In the system they generate their own labels and use a supervised learning algorithm with theselabels. The big data processing system is a system that can extract features of different entities from raw data. The outlier detection engine is a system that uses unsupervised learning. It uses the features that have been found in the big data processing system. They use three different methods, density, matrix decomposition, or replicator neural networks. The output of this unsupervised system is processed and shown to a security analyst. The security analyst can verify or refute the output. The feedback is fed to a supervised learning algorithm. The supervised learning algorithm learns a model that can use this feedback to better predict whether any new event is normal or abnormal. With more feedback, the system becomes more and more correct.

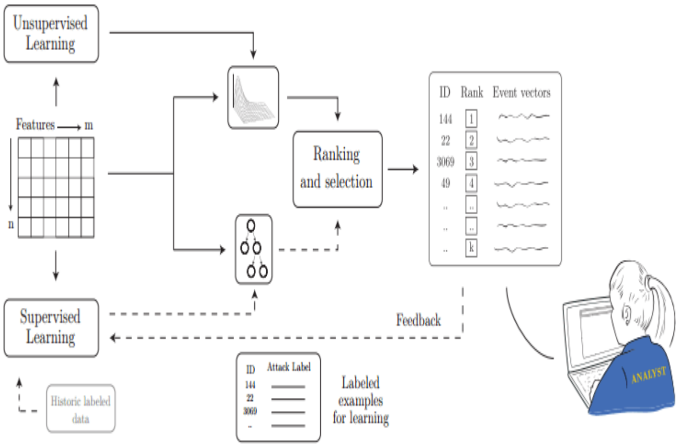
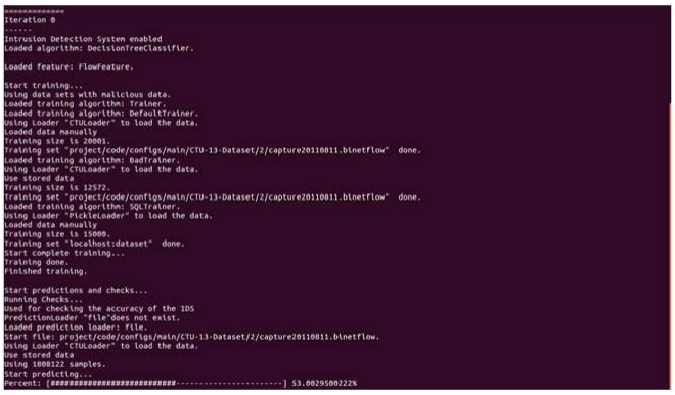


Fig-5: The structure of AI^2 system.

7. Execution of Source Code and Results



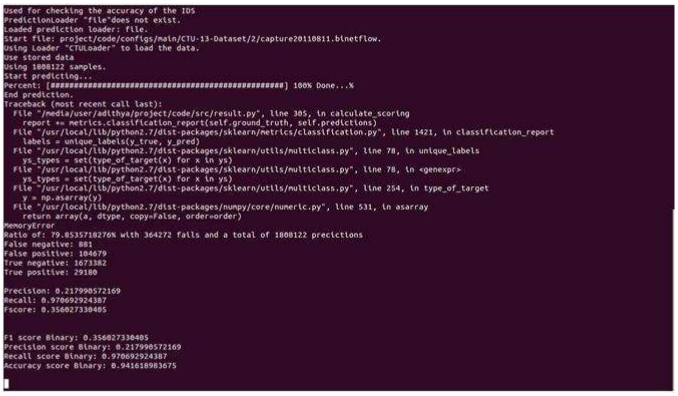
```
=============
Iteration 0
------
Intrusion Detection System enabled
Loaded algorithm: DecisionTreeClassifier.

Loaded feature: FlowFeature.

Start training...
Using data sets with malicious data.
Loaded training algorithm: Trainer.
Loaded training algorithm: DefaultTrainer.
Using Loader "CTULoader" to load the data.
Loaded data manually
Training size is 20001.
Training set "project/code/configs/main/CTU-13-Dataset/2/capture20110811.binetflow"  done.
Loaded training algorithm: BadTrainer.
Using Loader "CTULoader" to load the data.
Use stored data
Training size is 12572.
Training set "project/code/configs/main/CTU-13-Dataset/2/capture20110811.binetflow"  done.
Loaded training algorithm: SQLTrainer.
Using Loader "PickleLoader" to load the data.
Loaded data manually
Training size is 15000.
Training set "localhost:dataset"  done.
Start complete training...
Training done.
Finished training.

Start predictions and checks...
Running Checks...
Used for checking the accuracy of the IDS
PredictionLoader "file"does not exist.
Loaded prediction loader: file.
Start file: project/code/configs/main/CTU-13-Dataset/2/capture20110811.binetflow.
Using Loader "CTULoader" to load the data.
Use stored data
Using 1808122 samples.
Start predicting...
Percent: [###########################-----------------------] 53.0029500222%
```

Fig-6: Source Code Execution

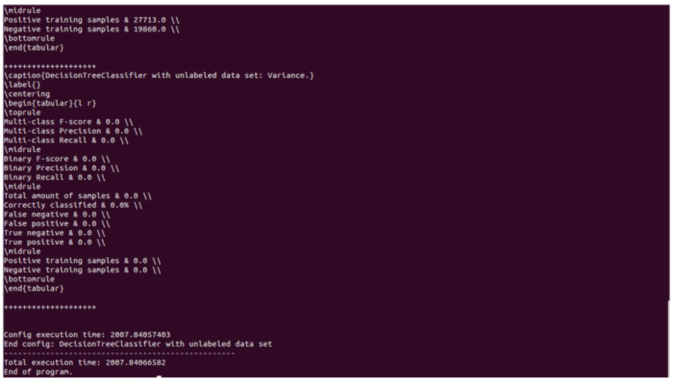


```
Used for checking the accuracy of the IDS
PredictionLoader "file"does not exist.
Loaded prediction loader: file.
Start file: project/code/configs/main/CTU-13-Dataset/2/capture20110811.binetflow.
Using Loader "CTULoader" to load the data.
Use stored data
Using 1808122 samples.
Start predicting...
Percent: [##################################################] 100% Done...%
End prediction.
Traceback (most recent call last):
  File "/media/user/adithya/project/code/src/result.py", line 305, in calculate_scoring
    report += metrics.classification_report(self.ground_truth, self.predictions)
  File "/usr/local/lib/python2.7/dist-packages/sklearn/metrics/classification.py", line 1421, in classification_report
    labels = unique_labels(y_true, y_pred)
  File "/usr/local/lib/python2.7/dist-packages/sklearn/utils/multiclass.py", line 78, in unique_labels
    ys_types = set(type_of_target(x) for x in ys)
  File "/usr/local/lib/python2.7/dist-packages/sklearn/utils/multiclass.py", line 78, in <genexpr>
    ys_types = set(type_of_target(x) for x in ys)
  File "/usr/local/lib/python2.7/dist-packages/sklearn/utils/multiclass.py", line 254, in type_of_target
    y = np.asarray(y)
  File "/usr/local/lib/python2.7/dist-packages/numpy/core/numeric.py", line 531, in asarray
    return array(a, dtype, copy=False, order=order)
MemoryError
Ratio of: 79.8535718276% with 364272 fails and a total of 1808122 precictions
False negative: 881
False positive: 104679
True negative: 1673382
True positive: 29180

Precision: 0.217990572169
Recall: 0.970692924387
Fscore: 0.356027330405


F1 score Binary: 0.356027330405
Precision score Binary: 0.217990572169
Recall score Binary: 0.970692924387
Accuracy score Binary: 0.941618983675
```

Fig-7: Binary Values



```
\midrule
Positive training samples & 27713.0 \\
Negative training samples & 19860.0 \\
\bottomrule
\end{tabular}

++++++++++++++++++++
\caption{DecisionTreeClassifier with unlabeled data set: Variance.}
\label{}
\centering
\begin{tabular}{l r}
\toprule
Multi-class F-score & 0.0 \\
Multi-class Precision & 0.0 \\
Multi-class Recall & 0.0 \\
\midrule
Binary F-score & 0.0 \\
Binary Precision & 0.0 \\
Binary Recall & 0.0 \\
\midrule
Total amount of samples & 0.0 \\
Correctly classified & 0.0% \\
False negative & 0.0 \\
False positive & 0.0 \\
True negative & 0.0 \\
True positive & 0.0 \\
\midrule
Positive training samples & 0.0 \\
Negative training samples & 0.0 \\
\bottomrule
\end{tabular}

++++++++++++++++++++


Config execution time: 2007.84057403
End config: DecisionTreeClassifier with unlabeled data set
--------------------------------------------------
Total execution time: 2007.84066582
End of program.
```

Fig-8: F-Score

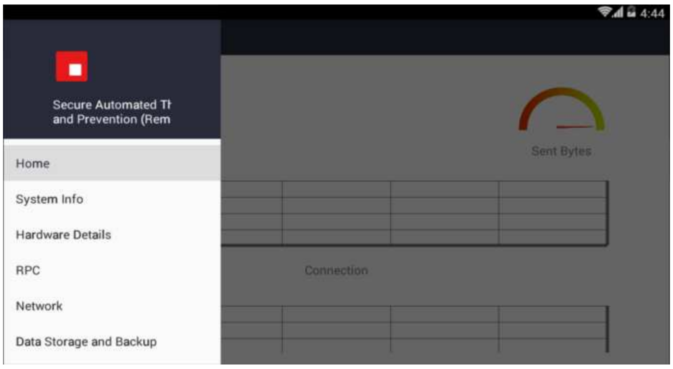
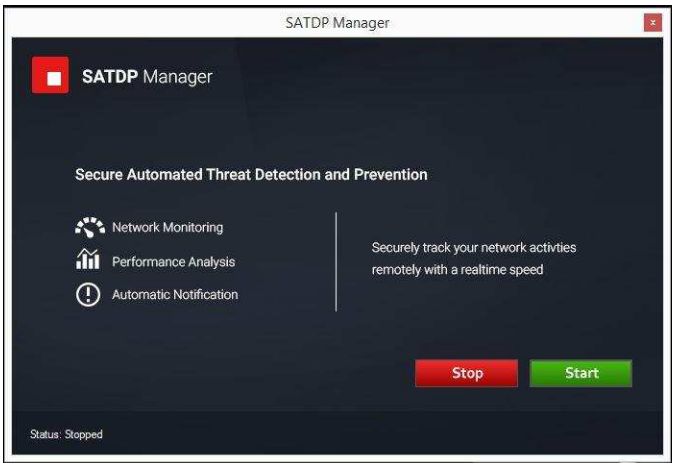


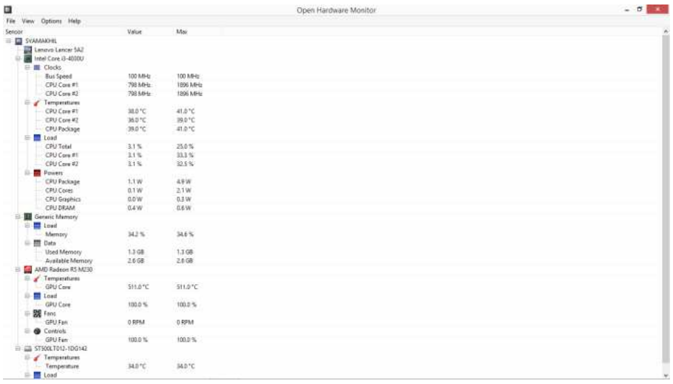
Fig-9: Remote Android App



Securely track your network activties remotely with a realtime speed

Status: Stopped

Fig-10: Desktop Application



Sensor	Value	Max
SYAMAKHIL		
Lenovo Lancer 5A2		
Intel Core i3-4030U		
Clocks		
Bus Speed	100 MHz	100 MHz
CPU Core #1	798 MHz	1896 MHz
CPU Core #2	798 MHz	1896 MHz
Temperatures		
CPU Core #1	38.0 °C	41.0 °C
CPU Core #2	36.0 °C	39.0 °C
CPU Package	39.0 °C	41.0 °C
Load		
CPU Total	3.1 %	25.0 %
CPU Core #1	3.1 %	33.3 %
CPU Core #2	3.1 %	32.5 %
Powers		
CPU Package	1.1 W	4.9 W
CPU Cores	0.1 W	2.1 W
CPU Graphics	0.0 W	0.3 W
CPU DRAM	0.4 W	0.6 W
Generic Memory		
Load		
Memory	34.2 %	34.6 %
Data		
Used Memory	1.3 GB	1.3 GB
Available Memory	2.6 GB	2.8 GB
AMD Radeon R5 M230		
Temperatures		
GPU Core	511.0 °C	511.0 °C
Load		
GPU Core	100.0 %	100.0 %
Fans		
GPU Fan	0 RPM	0 RPM
Controls		
GPU Fan	100.0 %	100.0 %
ST500LT012-1DG142		
Temperatures		
Temperature	34.0 °C	34.0 °C
Load		

Fig-11: Hardware Monitoring Application

8. Conclusion

This thesis has given an overview of machine learning algorithms and has shown how they can be used in an intrusion detection system. Not all machine learning algorithms work as good. The biggest problem that was discovered during the thesis was finding good labeled datasets which could be used to train the machine learning algorithms. If a good training dataset is used to train a machine learning algorithm, it can be used to create an intrusion detection system which offers acceptable performance out-of-the-box. A lot depends on the quality of the training dataset. If the training dataset does not contain enough samples of the different intrusions, the machine learning algorithm will exhibit a large amount of false positives and false negatives. K-Nearest Neighbors performed the best. It has good results in both the evaluation and the real-life scenario. When using an algorithm such as K-Nearest Neighbors close attention needs to be paid to what value of *k* is chosen and which distance metric is used. Unsupervised learning algorithms do not work well out-of-the-box. They need a lot of manual interference before they are viable to be used for intrusion detection.

9. References

[1] AI2 : Training a big data machine to defend- Big Data Security on Cloud (Big Data Security), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS), 2016 IEEE 2nd International Conference.

[2] Intrusion Detection Based On Artificial Intelligence Technique –International Journal of Computer Science Trends and Technology (IJCST) – Volume 2 Issue 4, July-Aug 2014

[3] Application of Artificial Intelligence in Network Intrusion Detection -A Succinct Review, World Applied Programming, Vol (2), No (3), March 2012. 158-166

[4] Intrusion Detection System (IDS): Anomaly Detection Using Outlier Detection Approach International

Conference on Intelligent Computing, Communication & Convergence (ICCC-2014)

[5] D. Ten, S. Manickam, S. Ramadass, and H. A. Bazar, “Study on Advanced Visualization Tools In Network Monitoring Platform,” in Third UKSim European Symposium on Computer Modeling and Simulation, EMS ‘09’, Minden Penang, Malaysia, December 2009.

[6] L. Chang, W.L. Chan, J. Chang, P. Ting, M. Netrakanti, “A network status monitoring system using personal computer,” presented at IEEE Global Telecommunications Conference, August 2002.

BIOGRAPHIES

Syam Akhil Repalle
Student, Department of Electronics and Communication Engineering, Koneru Lakshmaiah Educational Foundation, Andhra Pradesh, India

Venkata Ratnam Kolluru
Associate Professor, Department of Electronics and Computer Science, Koneru Lakshmaiah Educational Foundation, Andhra Pradesh, India