INDIAN INSTITUTE OF TECHNOLOGY KANPUR

CS 658 Semester 2021–2022-II: Project

Analysis of Machine Learning based Intrusion Detection System

TEAM - Trojan Hex

TEAM MEMBERS - Manish Yadav (20111031)
Shashwat Vaibhav (20111056)
Krishna Mohan Eaty (20111037)
Sharvari Oka (20111055)
Mallampet Adhi Lakshmi (20111003)
Siddhartha Bura (21111059)

KEYWORDS : Intrusion Detection systems, anomaly based IDS, machine learning, genetic algorithm, ID2T, tranalyzer

1 Introduction

Intrusions are a significant threat in cybersecurity, and intrusion detection datasets constitute a substantial source of research work done in this field. But there always exists some lack of publicly available realistic datasets that cover a variety of intrusive attacks. In this report, we worked on three publicly available datasets. We used various models to find out best-performing models using different performance metrics like precision, recall, accuracy, and F1 score. Apart from using publicly available datasets, we also used two tools named ID2T and Tranalyzer to generate our intrusion detection dataset. We used the publicly available pcap file of SCADA and injected attack packets using ID2T. After that, we used Tranalyzer to extract flow information from the pcap file, which has both normal and attack packets. The three publicly available datasets we worked on are 1) NSL-KDD dataset, 2)CSE-CIC-IDS2018 3) UNSW-NB15 dataset. We used various ML models like Decision Trees, Random forest, KNN, Naive bayes, SVC, etc. We also used Genetic algorithm and Neuro-Evolutionary techniques to find precision, recall, accuracy, and F1 score.

The methodologies which can be used for intrusion detection includes *signature based*,*anomaly based*, *stateful analysis*. These methodologies can be explained as follows:

1. **Signature based**:- A signature is simply a pattern that corresponds to a known attack. In signature-based detection, we compared the packets in the traffic with known attack signatures. Signature-based intrusion detection is also called knowledge-based or misuse detection.

2. **Anomaly based**:- An anomaly is a deviation from normal activity of a user, protocol, etc. The normal behavior of users can be detected by monitoring the regular activities of the user, and network packets (for traffic analysis) over a period of time. It is also known as behaviour-based detection.

3. **Stateful Protocol analysis**:- Stateful protocol analysis depends on vendor-developed standards for different variables in a protocol. It is also known as specification-based detection.

Signature-based intrusion detection system works on the known attacks. This IDS operation is based on a pre-programmed list of known threats and their indicators of compromise (IOCs). Whereas anomaly-based intrusion detection systems can detect unknown attacks by focusing on machine learning algorithms[1]. In this project, we injected the various types of attacks using ID2T tool. We used ML algorithms such as KNN, decision tree, random forest, genetic algorithm[2][3], and Neuroevolutionary approach for attack classification. We study standard ML-based models[4] like the KNN, SVC, Decision Tree, and genetic algorithm[5][6] to determine which model performs better for various performance metrics. We used these models on three standard datasets and the dataset generated by us using ID2T and Tranalyzer tools.

2 Data Generation

2.1 Open datasets

1. **NSL-KDD dataset:**(`https://www.unb.ca/cic/datasets/nsl.html`)
   KDD dataset is generated by subsampling the KDDCup99 dataset. It has various advantages compared to the KDD-Cup99 dataset[7], like removing redundant records and having a reasonable number of training and test data points. There are five classes in this dataset, out of which one is a normal class and four are attack classes. These attacks are Denial-of-service, Remote-to-local, User-to-root, and Probe. Each datapoint has 41 features, and the frequency of normal and intrusion classes in the NSL-KDD training dataset is 67343 and 58630, respectively[7].

2. **CSE CIC -2018**(`https://www.unb.ca/cic/datasets/ids-2018.html`):
   This dataset is generated in collaboration between the Communications Security Establishment and the Canadian Institute for Cybersecurity They used B-profiles and M-profiles to generate benign and attack traffic. The flow is extracted from it use CICFlowmeter. Each flow has 80 feature out of which one is label. They provide both csv and pcap file. There are 7 attack classes apart from a benign class. The attack list includes Brute-force, Botnet, DoS, DDoS, Web attacks, Heartbleed and infiltration of the network from inside.

3. **UNSW-NB15 2015**(`https://cloudstor.aarnet.edu.au/plus/index.php/s/2DhnLGDdEECo4ys`):
   UNSW-NB15 is a network intrusion dataset that was released in 2015. It contains 2540044 data points that contain realistic normal and attack data points. Each data point has 49 features that have both packet-based and flow-based statistical features. There are a total of 9 attack packets that include Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms signatures.

2.2 GNS Network Simulator

Building the own lab or renting services from vendors to make a network and collect traffic won't be feasible considering cost and time. So we moved to a Network simulation tool called GNS3. We have tried the data collection part in an emulated network using the GNS3 tool. The Free version of GNS3 allows for running a small topology consisting of only a few devices hosted on a laptop or cloud. Tools like EVE-NG and Cisco packet tracer are Alternatives to GNS3 but the free version of those tools also has very limited functionalities. In GNS3 users get easy to use interface that allows them to build complex labs consisting of supported Cisco switches and routers. The Network configuration process is simpler.
We use GNS3 to create a topology so that using Wireshark we can capture packets in the emulated network. We have done a toy network topology consisting of 4 nodes 1 switch then we ran Wireshark to capture packets in the emulated network.The **limitations** of GNS3 that we observe are as follows:

1. **Resource constraint:** GNS3 cab be used to create test size network topology which may have some nodes(devices), routers, switches,etc. But in the project we require the datasets which will include several attacks. Due to this resource constraint, we have not use GNS3 further. Due to this resource constraint, we have not use GNS3 further.

2. **Cisco images need to be supplied by user:** We need to download required images of router, virtual devices from Cisco.com, or purchase VIRL license, or copy from physical device.

2.3 ID2T

ID2T stands for Intrusion Detection Dataset Toolkit. The lack of adequate IDS datasets is always a problem for the scientific community as they require a variety of datasets to tackle new intrusive activity. ID2T can be used to inject syntactic packets in network datasets. We have used it to inject attack packets in the SCADA pcap file, which is downloaded from https://download.netresec.com/pcap/4sics-2015/4SICS-GeekLounge-151022.pcap . ID2T takes pcap as input, and analyzes and collects statistics from it. These statistics are stored in a local database. These can be used to define attack parameters for the injection of one or multiple attacks. It supports 13 attacks. It provides options to change various parameters before injecting attacks. These parameters can be source IP, destination IP, source Port, destination Port, attack duration, number of packets per second, etc. We have used it to inject six attack packets in the SCADA pcap file. After injecting the packet, it generates a new pcap file that has attack

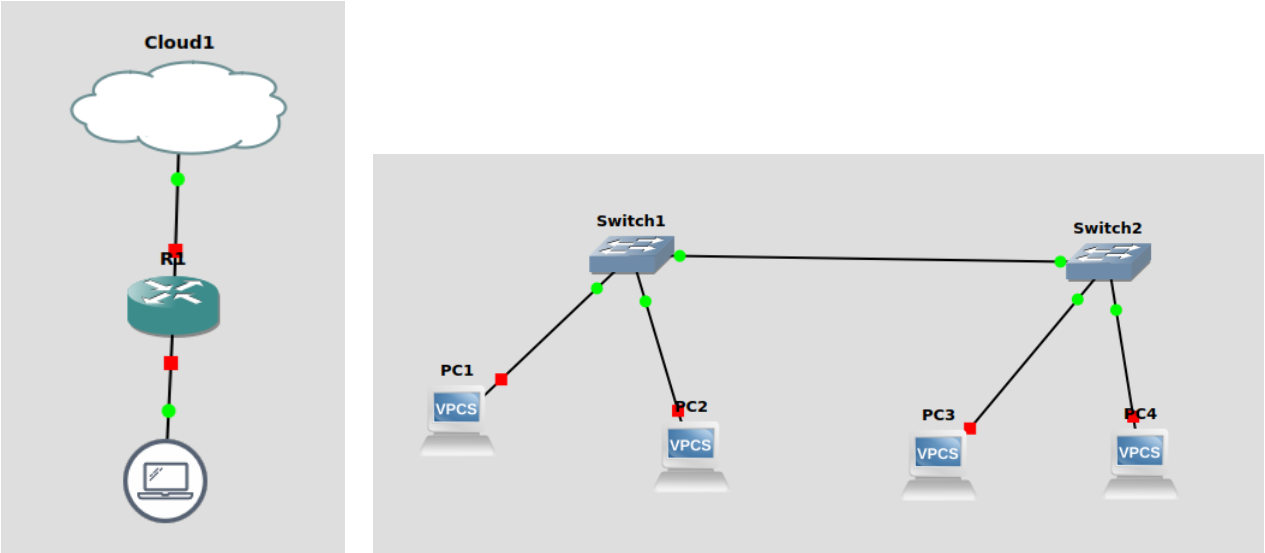

Figure 1: Topology

packets injected into it and another file(.xml format) which indicate the position of the first and last packet of attack injected.

Table 1:

List of attacks supported by ID2T	
DDoS Attack	Portscan Attack
EternalBlue Exploit	SMBLoris Attack
FTPWinaXe Exploit	SMBScan Attack
JoomlaRegPrivesc Exploit	SQLi Attack
MemcrashedSpoofer Attack	Sality Botnet
MS17Scan Attack	P2PBotnet

2.4 Tranalyzer

Tranalyzer is a lightweight flow generator and packet analyzer. It can be used to analyze ultra large packet dumps. A flow is identified by five tuples (source IP, destination IP, source Port, destination Port, Protocol). A flow is different from a connection. Generally, connection refers to TCP connection, whereas flow can also exist in UDP. We have worked on flow instead of the packet because in flow, we analyze some sets of packets and we label them as malicious or benign, whereas in per-packet analysis, labeling each packet is very difficult and also, it does not feel correct intuitively. In flow, we work on statistical features, while in per-packet analysis, it is challenging to create a large number of statistical features. We know that Machine learning algorithms work better when we use statistical features. Also, when we

Table 2:

List of attack we have injected using ID2T	
DDoS Attack	Portscan Attack
EternalBlue Exploit	MS17Scan
MemcrashedSpoofer Attack	SQLi Attack

work on flow then, the number of data points is less than when we work on packet as a flow is created by analyzing a group of similar packets. So it saves both time and computation. Tranalyser generates more than 100 features from pcap files, and it saves them in txt form. We have used python to generate a CSV file from a txt file which is then used in Machine learning models.

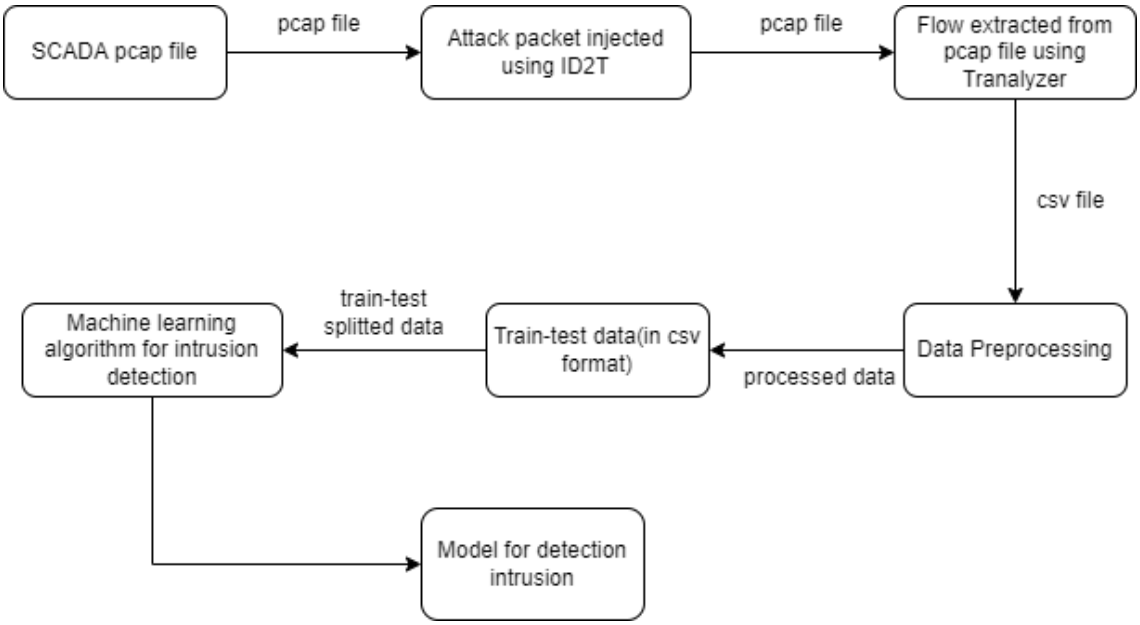

Figure 2: Flow diagram of using Machine Learning algorithms in SCADA traffic

3 Models Experimented

With availability of humongous data collected by machines in a network, it is very natural to experiment with models which draw inferences from data. Several models exist which have shown inspiring results when trained on large datasets. Although these models are powerful, they don't generalize well on attacks which were not available during the training period. For classification of network connections as *malicious* or *normal*, various classification Models like Decision trees, Random forest algorithm, Naive bayes etc. are faster while training and give optimal results. Other approaches like K-nearest neighbor, Support Vector Classifier etc. too give nearly good results but they take too much time while training as well as testing. Methods based on the ideas of evolutionary approach like genetic algorithm[8][9] also exist. We will take a brief look at their inner workings in the following subsections.

3.1 Standard Machine learning models

3.1.1 Decision Trees

Decision trees are tree like structures in which data arriving at each non-leaf node is classified by the node governing rules so that divided groups are as pure as possible. To assess the purity of a split, *information gain* or *entropy* is used. They are fast during training and even faster at test times. These models are extensively used in classification and regression problems. They can represent non-linear classification boundaries very well.

3.1.2 Random Forest

Derived from the ideas of *decision trees*, Random forests are nothing but ensembles of multiple decision trees. Each decision tree in the ensemble differs from every other tree in the sense that they use random subset of features for their operation. For classification tasks, predicted class is the one chosen by the most trees. Generally, random forests outperform decision trees.

3.1.3 K-Nearest Neighbors

As the name suggests, for a given test point, *k-closest* data points in the vector space define to which class it belongs. It is non-parametric model based on distance metric used for the vector space. It does not require any training as such but all the data points are required in the memory while making predictions. Also, this model suffers from longer test time duration as test point distance is calculated from each data point to arrive at the result.

3.1.4 Naive Bayes

It is a very simple *probabilistic classifier* based on the ideas of *Bayes' theorem* with very strong assumption that all the features are independent of each other, that's why the name *naive*. Since, not all the features are independent, results are not as good as shown by random forest and decision trees.

3.1.5 Support Vector Machines

These models differentiate classes by finding a $(p-1)$ dimensional hyperplane in the vector space with maximum margin, where each data point is p-dimensional vector. These models are expected to be slow at test time because computation with support vectors is done at test time similar to what we have seen already in *k-nearest neighbor*.

3.2 Genetic Algorithm

3.2.1 Introduction to Genetic Algorithm

Genetic Algorithm is a random search space method inspired by the process of natural selection, which belongs to the category of evolutionary algorithms. Genetic algorithms are used to generate solutions to optimization and search problems by relying on biologically inspired processes such as selection, crossover and mutation.[2]

Genetic Algorithm is a natural selection method for solving optimization problems . It is an iterative algorithm which modifies population in each iteration. In each iteration, the algorithm selects a pairs of parents which are to be crossed to produce offspring. Over successive iterations, an optimal solution is evolved from the population. [3]

3.2.2 Pseudocode

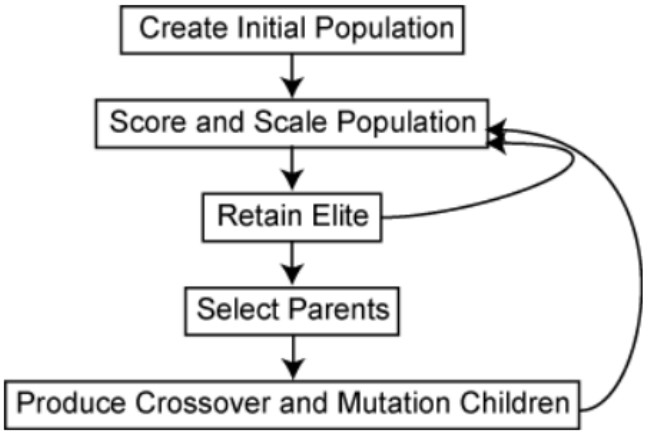

Figure 3: Flow Chart [3]

Pseudo code for GA:

1. Initialize the population P by randomly selecting individuals form search space S.
2. Evaluate the fitness $f(x_i)$ for each individual in P
3. Repeat (until stopping condition satisfied)
    - Selection - according to the fitness value individuals are selected
    - Crossover - according to predetermined crossover probability, crossover the selected individuals
    - Mutation - according to mutation probability, newly generated in individuals are mutated P_{new}
    - Update - $P \leftarrow P_{new}$.
    - Evaluate - compute the fitness $f(x_i)$ of each individual in P
4. Return the most fitted individuals from P

Figure 4: Genetic Algorithm Pseudo Code

3.2.3 Phases of Genetic Algorithm

Initial Population : The initial set of individuals are randomly selected as a population. For feature selection, we randomly selected the features for generating initial population. If a bit is set in a particular index, it means that particular feature is selected in the individual.

Fitness Function : The fitness function is a evaluation metric chosen by the user to evaluate how good the individual is. Based on the data, domain knowledge, experience, the users can select their own fitness function. But, the selection of the fitness function very important to make the model converge to a optimal solution in less time. We have used cohen kappa score as evaluation for feature selection.

Selection : Fittest individuals are the selected based on the optimal fitness function value in a population. The selection phase is responsible to select the fittest individuals and let them pass their genes to the next generation.

Crossover : Crossover is a process where the parents(fittest individuals) have to be mated with each other to produce a offspring. Offspring are created by exchanging the genes(features) of parents among themselves.

Mutation : Mutation is the part of the Genetic Algorithm which is related to the **exploration** of the search space. Mutation is crucial to the convergence of the algorithm while crossover is not. Mutation can also be defined as a small random flip in the individual, to modify individual.

Termination Criterion : The termination condition in general emphasizes on whether there is significant difference in offspring from the previous generation.We can put the criteria as some max number of iterations, if it is sufficient enough to make the algorithm converge. We can also put a criterion on expected optimal value of fitness function value if the user is satisfied with particular level of optimum.

3.2.4 Advantages of Genetic Algorithms

- It reduces the complexity of search space to some extent.
- Unlike other algorithms it does not require the derivative information.
- It is very effective than traditional methods.
- Not only one optimal solution, but a group of good solutions can be obtained.
- It can also deal with large search space and huge amount of parameters.

3.2.5 Limitations

- It is not preferable to apply genetic algorithm on all kinds of problems, the problems which requires derivative information may be solved more easily with other methods rather solving with genetic algorithm.
- Fitness function evaluation is an expensive task
- There is no gauranteee that the solution is optimal
- Hardest part of genetic algorithm is to properly implement the method with a good fitness function.
- The quality and time taken to arrive at a solution may also depends on the initial population, which was chosen randomly.

3.3 Neuro-Evolutionary Method

3.3.1 Model Approach

The genetic algorithm is a feature selection method in which the main is to get the set of features that account for in the final model. Another model which is based on the idea of genetic algorithms is Neuro-Evolutionary model. The idea of this algorithm is to learn the architecture of the neural network. The main parameters that decide a neural network architecture are the number of nodes, the number of layers, and the type of activation function. So the aim is to learn these parameters, build a neural net, and evaluate the performance of the cross-validation dataset. A set of iterations are performed to come to final parameters, and then the model is ready to be tested on the test data.

3.3.2 Iteration Process

Like a genetic algorithm, we have an initial population where each member of the population holds information about the number of nodes, number of layers, and type of activation function. Let's say we have N members in the initial population. The members of the initial population are randomly initialized. Using each member parameter values, a neural net is built and is evaluated on a cross-validation dataset. Here, the evaluation function is also called the fitness function. In the algorithm, we used accuracy as the dataset is not much biased. Incase if the dataset is biased then metrics like F1-score should be adopted. Once all the members are evaluated, we select top N/2 members who performance is good and use these to create the next population. The rest are discarded. The following population set is created by using two techniques, cross-over and mutations. The size of the population remains the identical across iterations. One-half of the population is created using cross-over. In cross-over, two parents are randomly selected from the retained members of the previous populations and are crossed to get a child. The crossing is randomly selecting parameters from either of the parents. The other half is created by using mutations. In this child, a member is built by randomly initializing the parameters. So the next population size is the same as the previous one. Fig. 5 is the pictorial representation of the complete process. The new population is again evaluated using the Fitness function. This process continues till the model converges, i.e. the fitness metric across two iterations does not change by much.

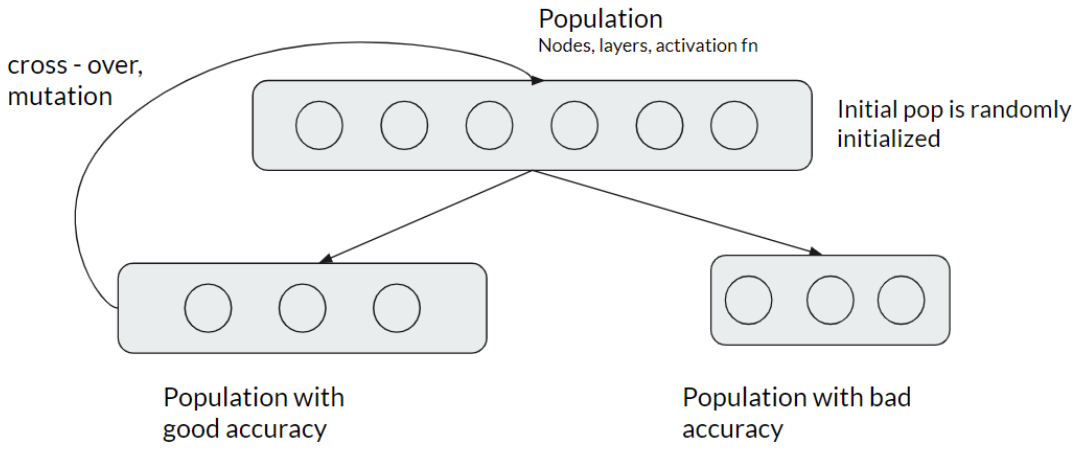

Figure 5: Representation of Neuro Evolutionary Algorithm

3.3.3 Limitations

Some limitations of using methods based on the genetic algorithm are

Models	Accuracy %	Recall %	Precision %	F1-Score %
		NSL KDD		
Naive Bayes	28	32	13	13
KNN	77	62	72	62
Decision Trees	70	46	46	46
Random Forest	71	48	63	47
Genetic Algo + Random Forest	82	62	71	66
Neuroevolutionary	78	66	61	63
		UNSW 2015		
Naive Bayes	70	69	71	76
KNN	78	76	81	82
Decision Trees	86	85	87	85
Random Forest	82	80	87	81
Support Vector Machine	71	70	72	70
		CSE CIC 2018		
Naive Bayes	24	57	31	30
Decision Trees	99	95	95	95
Random Forest	98	89	97	92
		SCADA		
Naive Bayes	49	69	39	33
KNN	99	59	59	59
Decision Trees	99	78	78	78
Random Forest	79	78	80	79

Table 3: Results of various ML models on different datasets

1. Iterations take time to complete since the time taken by every iteration depends on the population size. Iterations with a larger population size will take more time than traditional ML methods.

2. Genetic algorithms do not scale well with complexity or feature size. Consider two datasets, one with 100 features and the other with 1000 features. We need to use a larger population to cover all possible combinations for bigger feature space compared to smaller feature space. This increases time for each iteration, as a larger population will take more time to get evaluated on a cross-validation dataset.

3. Choice of the fitness function is an important aspect. The fitness functions should be chosen so that it requires less time to evaluate population members, and it also depends on the skewness of the dataset towards some labels.

4 Results and Discussion

Results derived from various models is populated in Table 4. As we can see from the table, *Genetic algorithm* gives encouraging result on the *NSL-KDD* dataset. Since it is a bit heavy and requires significant computational power, we could not test it on remaining datasets. Another fact that can be derived from the tables is that *Naive Bayes'* performs very poorly on NSL-KDD dataset. The reason behind it can be attributed to the fact that it makes strong assumption that the features are independent of each other. In the tables, we have even reported F1-score, Recall and Precision along with the accuracy. Accuracy is never a good performance measure as it does takes into account class imbalance. If a dataset is skewed, accuracy might be not that useful metric. In that case, F1-score, recall and precision are good measures to consider.

5 Future Work

In the current implementation, CSE CIC-2018 dataset contains large amount of data, hence it is not feasible to ML based algorithms on entire dataset as it requires lot of compute power. With availability of limited compute power, we were able to implement limited number of models. The idea is to look at the performance of other models as well. The result of genetic algorithm and neuroevolutionary on NSL KDD dataset are encouraging and are better compared to standard ML models. So it is worth try these two models on other datasets as well. A web interface will be implemented which takes pcap file as input and uses tranalyzer to create flows and using one of genetic algorithm or neuroevolutionary methods we can detect attacks present in the network.

6 Conclusion

In this project, we worked on an anomaly-based intrusion detection[1]. There are various standard datasets available for IDS, but there always is a need for new and realistic data to analyze intrusive activities. We used a tool named ID2T[4] to inject a synthetic attack packet in normal traffic to generate our own dataset. ID2T takes pcap as input and injects various attack packets based on user input, and it outputs a pcap file. This pcap contains normal as well as a attack packets. Apart from giving a pcap file, it also outputs an XML file that includes the location of the first and last attack packet for each attack. In this project, we have worked on flow analysis instead of the packet as it saves both time and computation. Also, working on flow is easy as we can get statistical features from each flow. We have used the Tranalyzer tool to extract flow from a pcap file. It generates more than 100 features. Apart from our own dataset, we also worked on three standard datasets, which are 1) NSL-KDD, 2)CSE-CIC-2018 3)UNSW-NB 2015. We used various standard machine learning models on this dataset like KNN, Decision Trees, Random Forest, SVC, Naive Bayes, etc. Apart from these models, we have also worked on the Genetic Algorithm and Neuro-Evolutionary Method. We were able to run the Genetic and Neuro-Evolutionary Method on the NSL-KDD dataset. The result of these two algorithms is pretty encouraging. Due to time and computational limits, we could not apply these two models to other datasets. Still, the result from the NSL-KDD dataset is quite satisfactory, and it encourages us to use these models in other datasets.

7 Individual Contribution

Work	Manish	Krishna	Shashwat	Sharvari	Lakshmi	Siddhartha
Literature Survey	✓	✓	✓	✓	✓	✓
Data Generation	✓			✓	✓	
ID2T & Tranalyzer tool	✓				✓	
Naive ,KNN Bayes	✓	✓	✓	✓	✓	✓
Random Forest,Decision Tree		✓	✓	✓		✓
Genetic Algorithm		✓	✓		✓	✓
SCADA	✓	✓	✓	✓		✓
PPT	✓	✓	✓	✓	✓	✓
Total	16.67%	16.67%	16.67%	16.67%	16.67%	16.67%

References

[1] T. Kaur and S. Kaur, "Comparative analysis of anomaly based and signaturecom based intrusion detection systems using phad and snort," 2013.

[2] "Genetic Algorithm Wikipedia." https://en.wikipedia.org/wiki/Genetic_algorithm.

[3] "Genetic Algorithm Mathworks." https://www.mathworks.com/help/gads/what-is-the-genetic-algorithm.html#:~:text=The%20genetic%20algorithm%20is%20a,a%20population%20of%20individual%20solutions.

[4] M. Sabhnani and G. Serpen, "Application of machine learning algorithms to kdd intrusion detection dataset within misuse detection context.," pp. 209–215, 01 2003.

[5] "Mutation Genetic Algorithm Tutorialspoint." https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_mutation.htm.

[6] "Genetic Algorithm Tutorialspoint Introduction." https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_introduction.htm.

[7] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1–6, 2009.

[8] K. S. Desale and R. Ade, "Genetic algorithm based feature selection approach for effective intrusion detection system," in *2015 International Conference on Computer Communication and Informatics (ICCCI)*, pp. 1–6, 2015.

[9] A. Goyal and C. Kumar, "Ga-nids: A genetic algorithm based network intrusion detection system," 01 2007.