Machine Learning Approach for Network Intrusion detection in Normal and IoT Traffic SmartDS

Jeyaradnam Tharjalan
Department of information system - Engineering - Cyber Security.
Sri Lanka institute of information - technology.
Malabe, Sri Lanka.
Tharjalan@icloud.com

Anshajanth Yoganathan
Department of information system - Engineering - Cyber Security.
Sri Lanka institute of information - technology.
Malabe, Sri Lanka.
Aanshajanth1@gmail.com

Hamshika Kugaruban
Department of information system - Engineering - Cyber Security.
Sri Lanka institute of information - technology.
Malabe, Sri Lanka.
hamshirupan95@gmail.com

R.P.G.U.N.Alupotha
Department of information system - Engineering - Cyber Security.
Sri Lanka institute of information - technology.
Malabe, Sri Lanka.
un.alupotha1994@gmail.com

Lakmal Rupasinghe (Supervisor)
Senior Lecture and Program Coordinator – BSc and MSc
Department of information system – Engineering – Cyber Security. Sri Lanka institute of information technology.
Malabe, Sri Lanka.
lakmal.r@sliit.lk

Amila Nuwan (Co-Supervisor)
Lecture *Department of information system – Engineering – Cyber Security.*
Sri Lanka institute of information technology.
Malabe Sri Lanka.
amila.n@sliit.lk

***Abstract*— In today's world technological advancements are gradually increasing, so small scale to large scale companies use internet for business purposes. Other than that Internet of Thing (IoT) is a current trend in the world. Most companies try to improve their companies making them smart premises since that will give a lot of advantages; some are increasing customer satisfaction, saving time and many more. So, with the rise of these technological advancements, different types of security issues like DDOS, malware, virus, worms and many more issues occurs as well because of the software or device vulnerabilities. Those are harmful for organizations' sensitive data due to the violation of integrity, confidentiality and availability. Therefore, as a solution Intrusion Detection Systems (IDS) can be used to protect organizations' data from those attacks. However, there are certain issues with these existing IDSs and some of them are the accuracy of the results because most of the IDSs give false positive alerts thus, the administrators have to manually identify the real attacks which is a waste of time. Furthermore, those alerts not categorize according to their level and a major problem is IoT systems are more complex and use different types of devices which easily can have vulnerabilities. But using a normal IDS can't identify those vulnerabilities. So, as a solution we have developed an IDS using machine learning algorithms to train data and give an optimized result. Moreover, the implemented IDS be can used for both regular and IoT networks. This gives more accuracy than other IDSs and reduce the administrative work loads.**

Keywords—Security, Big-Data Analysis, Snort, Machine Learning, random forest, support vector

I. INTRODUCTION

In today's world use of technologies vary according to small to large scale businesses by changing their working environment with the help of new featured devices. As well as the current trend of the business world is working with Internet of Things (IoT). IoT networks are very complex rather than normal network. In company related devices they share their details with another employer, customer or third-party companies through network. All the above users can connect with the system through a wired or wireless network because of that they can easily send emails, making video conferences, accessing other resource devices like printers and many more. According to this nature, an intruder can easily attack on company network through vulnerabilities.

IoT networks uses more featured and new equipment other than regular networks. All the equipment like smart watches, tablets, fans, door locks, camera systems, fire alarm systems and many more which are used in day to day life can be manage through this technology. This is basically making connection between Internet enabled devices and unable devices those are called as nodes. An IoT gateway is situated in between those end devices and internet enabled devices. Through those end devices attacks can be passed as different types of payload to the system and easily crash the system. Several companies use Intuition Detection Systems (IDS) which is kind of a device or software application, which is used to secure their confidential details from unauthorized users. This analyze the network data traffics and identify, if it includes any malicious scripts or any other attacks which can break down the whole company working process. If it identifies those type of attacks it can generate alerts to an administrator or to a system log.

IDS can divide into two basic methods those are Network based IDS and Host based IDS. In Network based IDSs analyzing data traffics within the network or which comes from outside the network. If it identifies any malicious data packets it will generate alerts according to the severity of the situation. Those details include database inside the NIDS. In Host based IDSs works on single host or device within the network infrastructure. This only analyze data packets which regards to that device or host. This process takes a snapshot of an existing file system and match it with the previous snapshot.

Furthermore, IDSs can categorize according to the detection mechanism. Those methods are signature based and anomaly based. In signature based IDSs identifying the malicious attack by patterns. This can be only done for known attacks but hard to identify zero-day attacks. In anomaly based IDSs, it works if some abnormal behavior occurs within the network it will compare it with the normal behavior and send alerts to an administrator. There are some significant issues encountered with this, one of them is that most of above detection mechanisms can identify known or defined attack signatures or behaviors. If attacker sent a new attack which is not predefined in these scopes, then this can't identify it. The other issue is that the false positive rate is high because of that accuracy of the details is low and most probably severe damage can be happened to the network infrastructure.

Different types of attacks are capable of violating regular and IoT networks confidentiality, Integrity of information. In both cases those attacks may occur due to the device's vulnerabilities and poor security. In DDOS, one of the major attack type which comes on both network types. In simple definition of DDOS means attackers use multiple hosts to harm targeted system. Intruder tries to flood of traffic in destination point and finally slow or crash the whole system. This DOS attack concept is coming under DDOS attacks [1]. IoT based DDOS attacks can be separated according to three layers. Mainly RFID tags use to gather data without involving humans, this works via radio wave reader. Intruders try to jam those waves because of that wave reader is hard to understand the communication. As well as inject kill command to RFID tag due to that attacker can crash the system. Other than that, the attacker can use brute force attacks to crack passwords which use for both regular networks as well as IoT networks. Especially in IoT networks intruders try to crack passwords which are used in different tags.

In IoT uses sensor networks for communication, one mechanism named as ZigBee. It's low-cost and low power wireless communication mechanism. This can be break down "Hello Flood" attacks. Attack nodes sent hello flood request to the authorized nodes and crash the system security process [1]. Furthermore, some software attacks also can happen in IoT and normal networks. If intruder inject malicious script to the system, he/she can easily get system information. As well as Virus, Trojans, Worms etc. can expand through internet when downloading or visiting some sites through IoT devices or normal network. One attack type which comes for IoT network is node tampering [2] which means intruder tries to change the nodes and tries to gather sensitive information. Sometimes attackers tries to inject malicious codes in between nodes. When passing data through those nodes it generates altered information.

In regular networks attacks can be done through browsers. When an attacker breaches some website and inject some code into it if the normal user visit that website through web browser then those injected codes work on users' systems and crashes it. In IoT networks major problem is to mitigate above attacks as a solution some researchers' found method which is named as Cognitive Radio Sensor Networks (CRSN) but in later researches it is found another attack which come up to this as CSRF which is named as Primary User Emulation Attack (PUEA) [3]. This attack occurred when generating the cognitive cycles. Attacker work as a primary user and send untrusted signal because of that authorized users leave the spectrums.

Internet protocol version Low-power Wireless Personal Area Network protocol (6LoWPAN) is widely used as conversion between IEEE 802.15 and internet version 6 (IPv6) protocols. Normally Internet of thing devices are working on Lossy Network and low power with 6LoWPAN protocols (RPL). So, these devices are using a routing protocol which can deal with limited power, memory and etc. The RPL is based on packets forwarding to parents or downward to their children's and Destination Oriented Directed Acyclic Graph (DODAG). Whatever these routing protocols like RPL are not having good strength to avoid security threats. In this our research we are mainly focusing on three attacks which can be performed on RPL. They are Hello Flood, Sinkhole and Wormhole attacks. Thus, there are several researches that has been introduced to intrusion detection system for 6LoWPAN protocols. SVELTE is the first IDS for IoT and detect disturbance by analyzing the mapped data. These strategies are suitable in identifying particular routing attacks in which, SVELTE is created particularly to detect sinkhole attack though Pongle's attention on recognizing wormhole attack. In any case these two IDS are can be worthless against a difficult and extra destructive attack [1] [2].

II. RELATED WORKS

Intrusion-detection systems go for identifying attacks against computer systems as well as networks also. Usage of IDS are to detect attacks and other security violations and detect and deal with preambles to attack. Intrusion detection system implementation are having two type of methods. They are signature based and anomaly based [3].

A. Signature method

This method mainly focuses on a collection of signatures, each of which characterizes the profile of a known security threats. Signature based methods are easy to understand and develop once you figure out the sort of network behavior to be found out. These also have a disadvantage. Because signature based only detect the known attacks, but novel attacks cannot be detected [3].

B. Anomaly detection

Anomaly based IDS, monitors' network traffic and compares it against an established baseline of normal traffic profile. Anomaly based algorithm is based on the attacker's behaviors different from the normal users. Disadvantage of anomaly detection is the difficulty to define the rules. Because in anomaly detection each protocol must be implemented, defined as well as tested for accuracy [3].

C. Hybrid based IDS

It is a new generation IDS it contains signature detection and anomaly detection part within this, so it can overcome signature and anomaly detection problems. Hence, it can detect both cases either anomaly or signature attacks. PHAD and SNORT IDS are using anomaly detection technique to detect cyber-attacks [4].

Machine Learning Algorithm in IDS

Mostly machine learning algorithms are used for wireless network environment in network security. There are many Intrusion detection methods which are used in machine learning. Choosing reasonable features for ML algorithm is essential to separate among normal as well as anomaly behaviors. Utilized a few ML algorithms to break down features gathered from the network. At that point, huge features can filtered among from them which are related by their particular attacks. This component choice is performed to choose just imperative features just as to diminish the utilization of device resources. These critical features can be utilized as an IDS indicator to distinguish any attacks. The ML algorithm is connected to locate the new huge features and after that arranges the features as intrusion indicator for an IDS [5] [6] [7].

III. DATASETS PROCESSING AND ANALYSIS

A. *KDD Dataset*

In this project, the KDD dataset is mainly being used, the KDD datasets which is created by university of New Brunswick, Canada. This dataset is the updated version of the KDD'99 Datasets from DARPA Intrusion detection Evaluation program. KDD datasets is the one mostly used dataset for the machine learning based intrusion detection approach. Whatever the final results of the analysis that performed by Tavallaee et al. open up that the dataset is filled with unnecessary records that can easily cause a week evaluation of anomaly detections. The new updated NSD-KDD, lacking shortage noted in KDDCUP99 data is since it's proposed by Tavakaee et al. The NSL-KDD dataset utilized in this research and project that already consist 125,973 training sets and 22544, 42 testing sets along with the 42 attributes.

KDD datasets includes 21 type of attacks that are categorized into four main groups with contrasting number of examples and occurrences. The 79% of KDD dataset have DOS attacks type while the normal packets there are 19% and other attacks types can be 2% of existing data. Based on this data this KDD datasets come in to view as an unstable dataset but at the same time it concludes the biggest number of attributes (42).

Categories of Attack	Attack name	Number of instances
DOS	SMURF	2807886
	NEPTUNE	1072017
	Back	2203
	POD	264
	Teardrop	979
U2R	Buffer overflow	30
	Load Module	9
	PERL	3
	Rootkit	10
R2L	FTP Write	8
	Guess Passwd	53
	IMAP	12
	MulitHop	7
	PHF	4
	SPY	2
	Warez client	1020
	Warez Master	20
PROBE	IPSWEEP	12481
	NMAP	2316
	PORTSWEEP	10413
	SATAN	15892
normal		972781

Table 1 Details of Attacks inside the KDD DATASET.

These data attributes are classified as basic information that is gathered using any network connection established based on TCP/IP. The main advantage of this dataset is that it includes of 32 expert proposed attributes to find/detect DOS, PROBE, U2R and R2L Attacks.

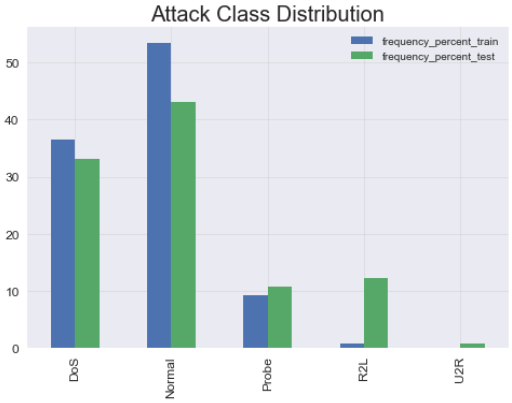


Figure 1 Attack Class Distribution performed in Jupyter notebook

B. *Overview of Classification Algorithms*

This Part will give some overview of different kind of classification machine learning algorithms to help the intrusion detection in network traffic. The outcome of continued development of technologies builds the requirement of the machine learning algorithms to create more efficient detection platform and to extract the large number of produced datasets. Generally, machine learning algorithms are defined as two categories. Those are supervised and unsupervised. Supervised learning study for prediction of the object class from pre-labeled objects (classified). Moreover, the unsupervised learning algorithms discover the neural collection of objects provided as unlabeled data.

In this project Following supervises alarms will be used. Because the KDD datasets consist the predefined classes.

- *Random Forest Classifier*

This is one of the categories of tree classification algorithms. The main achievement of this algorithm is to increase trees classifiers based on the idea of the forest. Random Forest have good accuracy rate and it can be developed to manage noise values of dataset. There is no re-alteration process among the characterization step. To execute this algorithm the quantity of trees inside the forest should be figured because every individual tree inside a forest predicts the expected output and after that the voting procedure is used to choose the expected yield that have the biggest number of votes.

- *Logistic Regression*

Logistic regression is utilized to represent data as well as to clarify the relation within individual dependent binary variable as well as one or extra ratio-level, ordinal, nominal independent variables, while choosing the model for the logistic regression examination, major thoughts are suitability of the model. Joining the logistic regression model to an independent variable will dependably build to the measure of change clarified in the logs and if any case, joining an ever-increasing number of factors to the model can bring about over fitting, which decreases the generalizability of the model past the information on which the model is a fit.

- *Naive Bayes Classifier*

Naïve Bayes are easy to train and powerful classifiers. The methodology is straightforward and the descriptor naive has been credited not on the grounds that these algorithms are restricted or less effective, but since of a central supposition about the causal elements that will be discussed in this research paper. A naive Bayes classifier is described along these lines since it depends on a naive condition, which suggests the restrictive independence of causes. This can appear to be hard to acknowledge in numerous settings where the likelihood of a specific feature is entirely associated with another.

Routing attacks in 6LoWPAN protocol

- *Hello Flood*

Few routing protocols in Wireless Sensor Networks expect nodes to communicate hello messages to report themselves to their next-door-neighbor. A node which gets such a message may accept, that it's inside a radio scope of the sender. This type of attack recognized with using the UDP packets. And the hello flood attacker may utilize a huge standard signal to allure different nodes to utilize its route and also, they are not a part of the next-door-neighbor either they are situated separately [9] [10].

- *Sinkhole*

The purpose of sinkhole attack is to manage packet traffic in a network. However, much as could reasonably be expected for a malicious node also. The attacker deceives the legitimate nodes to establish link with the malicious node by pretending having the optimal routes. This kind of attack make huge network traffic than another routing attack. It compromises the next-door-neighbor nodes with sending the fake routing data. Sinkhole attack starts to destroy the data packets. Additionally, this kind of attack is reached out to specific sending, HELLO flood, as well as black hole attack [1].

C. Training Models and Datasets Experiments

In this paper the implementation method that is used is: Environmental setup, Data Loading, Data preprocessing, Train the machine learning models, Evaluate Models, And Test Models. After Testing the Models, one Model will be chosen for the integration with the Normal network IDS. Machine learning model will be chosen based on its running time and the accuracy. This project was implemented using the python language. Python is the current trending language and open source general programing. It has the good documented machine learning module called scikit-learn with all the machine learning algorithms for the both supervised and the unsupervised learning.

For the IoT, the researchers' faced a hard time to collect IoT related attack data set so we used Cooja simulator with RPL attack Framework to simulate the attack and stored the traffic as PCAP file by using Wireshark tool. After that the researchers' wrote an own script for converting to CSV file from PCAP file. The CSV file contains 77. The below table 2 contains sample of dataset.

No.	Time	Src	Dst	Lngth	Info	TR	RR	TAT	RAT	TPC	RPC	TTT	TRT	DAO	DIS	DIO	Label
620.851	48.455665	94	893	76	1	0,197	0,197	197	197	0.012	0.012	0,00006	0,00005	197	0	0	1
620.852	48.455694	389	339	76	1	0,096	0,096	96	96	0,005	0,005	0,00005	0,00005	96	0	0	1
620.854	48.455762	158	620	76	1	0,196	0,166	196	166	0,01	0,008	0,00005	0,00005	166	0	30	1
620.855	48.455774	971	271	76	1	0,22	0,22	220	220	0,008	0,008	0,00004	0,00004	220	0	0	1
620.856	48.455779	994	331	76	1	0,227	0,227	227	227	0,008	0,009	0,00004	0,00004	227	0	0	1
620.857	48.455782	354	894	76	1	0,284	0,128	284	128	0,006	0,006	0,00005	0,00005	284	0	0	1
620.867	48.455816	565	9999	97	3	0,03	1,7	30	1698	0,002	0,097	0,00007	0,00006	0	0	30	1
620.858	48.455836	808	792	76	1	0,263	0,189	263	189	0,01	0,006	0,00004	0,00003	189	0	74	1
620.861	48.455991	691	134	76	1	0,19	0,19	190	190	0,012	0,012	0,00006	0,00006	190	0	0	1
620.874	48.456005	430	33	102	3	0,171	0,171	171	171	0,013	0,013	0,00006	0,00006	0	0	171	1

Table 2

But most of the columns are not related to attack prediction so we used BSF-CFS and GS-CFS algorithms to identify the most important feature. Based on that the researchers' have found five most significant features. Which are; destination context identifier, destination port, context identifier, pattern and next header. The below table 3 contains sample of final dataset.

	A	B	C	D	E	F
1	dst	cid	dci	next	pattern	class
2	33	33	0	33	33	normal
3	67	67	0	67	67	normal
4	54	54	0	54	54	normal
5	76	76	0	76	76	normal
6	72	72	0	72	72	normal
7	64	64	0	64	64	normal
8	35	35	0	35	35	normal
9	68	68	0	68	68	normal
10	48	48	0	48	48	normal
11	32	32	0	32	32	normal

Table 3

Initially for the training model and data preprocessing the Jupyter notebook was used. Jupyter notebook will give a platform to develop machine learning models using the python language. Notebook can be run in online host server as well as the local server using local host. Since all the python modules are needed for the further development the researchers' decided to develop using local server. The output file will be the Model.Ipynb file. Later it can be converted into pure python Language using command line.

- Import the Data and Perform Preprocessing.

Firstly, the dataset is needed to load into python jupyter Notebook Environment. The initial Task is mapping the various type attacks into 4 attack types. (Figure1)

- Exploratory Data Analysis

Fundamental exploratory data analyses were completed among different things to get the distinct measurements of the dataset, discover occurrences of missing qualities and repetitive features, investigate the data type and structure and examine the appropriation of assault class in the dataset.

- Standardization of Numerical Attributes.

The numerical highlights in the dataset were extricated and standardized to have zero mean and unit difference. This is a typical requirement for some machine learning algorithms actualized in Scikit-learn python module.

- Encoding the Categorical Attributes

The categorical features in the dataset were encoded to integers. This is likewise a typical requirement for some, machine learning algorithms actualized in Scikit-learn.

```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()

# extract categorical attributes from both training and test sets
cattrain = dfkdd_train.select_dtypes(include=['object']).copy()
cattest = dfkdd_test.select_dtypes(include=['object']).copy()

# encode the categorical attributes
traincat = cattrain.apply(encoder.fit_transform)
testcat = cattest.apply(encoder.fit_transform)

# separate target column from encoded data
enctrain = traincat.drop(['attack_class'], axis=1)
enctest = testcat.drop(['attack_class'], axis=1)

cat_Ytrain = traincat[['attack_class']].copy()
cat_Ytest = testcat[['attack_class']].copy()
```

Figure 2

- Data Sampling

The insufficient circulation of certain attack classes, for example, U2R and L2R in the dataset while others, for example, Normal, DoS and Probe are fundamentally spoken to inalienably prompt the circumstance of unevenness dataset. While this situation isn't sudden in data mining undertakings including distinguishing proof or characterization of cases of deviations from normal examples in a given dataset, research has appeared administered learning calculations are regularly one-sided against the objective class that is feebly spoken to in a given dataset.

Random data sampling and cost sensitive methods are the certain approaches. These methods can be defined to address the problem of an unbalanced dataset. The main goal of sampling is moderation of an unbalanced dataset by some actions in order to give the balance version of distribution. Cost-sensitive learning centers on the imbalanced learning issue by utilizing unique cost lattices that portray the costs for misclassifying a specific data precedent instead of making balanced data distributions through various testing systems.

In this research a random oversampling technique was used to balance class distribution in the training dataset as shown in the figure3.

```
from imblearn.over_sampling import RandomOverSampler
from collections import Counter

# define columns and extract encoded train set for sampling
sc_traindf = dfkdd_train.select_dtypes(include=['float64','int64'])
refclasscol = pd.concat([sc_traindf, enctrain], axis=1).columns
refclass = np.concatenate((sc_train, enctrain.values), axis=1)
X = refclass

# reshape target column to 1D array shape
c, r = cat_Ytest.values.shape
y_test = cat_Ytest.values.reshape(c,)

c, r = cat_Ytrain.values.shape
y = cat_Ytrain.values.reshape(c,)

# apply the random over-sampling
ros = RandomOverSampler(random_state=42)
X_res, y_res = ros.fit_sample(X, y)
print('Original dataset shape {}'.format(Counter(y)))
print('Resampled dataset shape {}'.format(Counter(y_res)))
```

. Figure 3

- Feature Selection

In machine learning the key processing is the key part for the preprocessing step. That is involving the selection of those most important features as a subset of the native features based on the certain basis to degrees dimension in order to increase the efficiency and accuracy of the machine learning algorithms.

The vast majority of the data incorporates immaterial, redundant, or noisy features. Highlight determination decreases the number of features expels superfluous, redundant, or noisy features, and realizes discernable impacts on applications: accelerating a data mining algorithm, improving learning precision, and prompting better model intelligibility. There are two basic ways to deal with select or lessen the features; a wrapper utilizes the intended learning, algorithm itself to assess the value of features, and a filter evaluates features as per heuristics dependent on general qualities of the data. The Wrapper method is mostly used to improve feature subsets, but it executes slower than the filter method.

In this research project, the wrapper method was used with the Random Forest Classifier algorithm along with the function that will define feature importance was trained to extract feature importance from the training dataset. The second method implemented for the feature extraction is also involved using a recursive feature extraction that is also based on the random forest algorithm for take top 10 features for the good accuracy in the training model.

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier();

# fit random forest classifier on the training set
rfc.fit(X_res, y_res);
# extract important features
score = np.round(rfc.feature_importances_,3)
importances = pd.DataFrame({'feature':refclasscol,'importance':score})
importances = importances.sort_values('importance',ascending=False).set_index('feature')
# plot importances
plt.rcParams['figure.figsize'] = (11, 4)
importances.plot.bar();
```

Figure 4

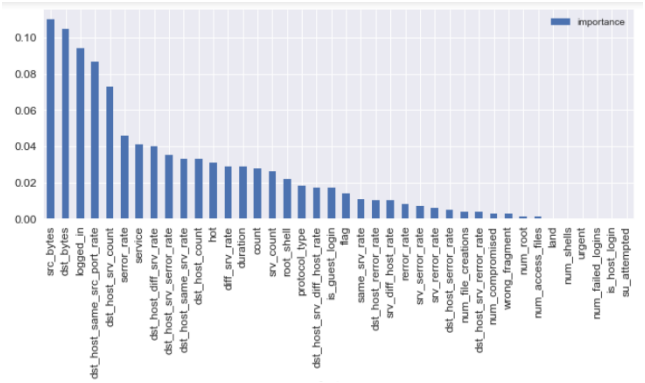


Figure 5 Output Top Features

For this project, the selected features are as follows;

```
In [20]:
from sklearn.feature_selection import RFE
import itertools
rfc = RandomForestClassifier()

# create the RFE model and select 10 attributes
rfe = RFE(rfc, n_features_to_select=10)
rfe = rfe.fit(X_res, y_res)

# summarize the selection of the attributes
feature_map = [(i, v) for i, v in itertools.zip_longest(rfe.get_support(), refclasscol)]
selected_features = [v for i, v in feature_map if i==True]

In [21]:
selected_features

Out[21]: ['src_bytes',
 'dst_bytes',
 'logged_in',
 'count',
 'srv_count',
 'dst_host_srv_count',
 'dst_host_diff_srv_rate',
 'dst_host_same_src_port_rate',
 'dst_host_serror_rate',
 'service']
```

Figure 6

- Data partition

After selection of pertinent features, the resampled dataset dependent on the selected features was apportioned into two-target classes for all the attack classes in the dataset to encourage binary classification. (Normal class & attack class).

Train Models

Classifier algorithms that are used to train the model for this project are: Support vector machine (SVM), Decision Tree, Random Forest, Naive Baye, Logistic Regression and the k-nearest Neighbor (KNN). An additional group voting Classifier is trained to combine the algorithms and average the forecast results from all the trained classifiers that are trained separately. The main plan behind the Voting Classifier is to merge different machine learning algorithms and utilize more vote or the average forecasted probabilities for the predicted class labels. Those classifiers can be helpful for the set of identical good functioning model in order to stable out the independent weakness.

```
from sklearn.svm import SVC
from sklearn.naive_bayes import BernoulliNB
from sklearn import tree
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import VotingClassifier

# Train KNeighborsClassifier Model
KNN_Classifier = KNeighborsClassifier(n_jobs=-1)
KNN_Classifier.fit(X_train, Y_train);

# Train LogisticRegression Model
LGR_Classifier = LogisticRegression(n_jobs=-1, random_state=0)
LGR_Classifier.fit(X_train, Y_train);

# Train Gaussian Naive Baye Model
BNB_Classifier = BernoulliNB()
BNB_Classifier.fit(X_train, Y_train)

# Train Decision Tree Model
DTC_Classifier = tree.DecisionTreeClassifier(criterion='entropy', random_state=0)
DTC_Classifier.fit(X_train, Y_train);

 Train RandomForestClassifier Model
RF_Classifier = RandomForestClassifier(criterion='entropy', n_jobs=-1, random_state=0)
RF_Classifier.fit(X_train, Y_train);

 Train SVM Model
SVC_Classifier = SVC(random_state=0)
SVC_Classifier.fit(X_train, Y_train)

Train Ensemble Model (This method combines all the individual models above except RandomForest)
combined_model = [('Naive Baye Classifier', BNB_Classifier),
                  ('Decision Tree Classifier', DTC_Classifier),
                  ('KNeighborsClassifier', KNN_Classifier),
                  ('LogisticRegression', LGR_Classifier)
                 ]
VotingClassifier =  VotingClassifier(estimators = combined_model,voting = 'soft', n_jobs=-1)
VotingClassifier.fit(X_train, Y_train);
```

Figure 7

For the IoT environment we have trained 6 classification machine learning algorithms with our dataset and have compared the output of all the models. That information is listed in below table 4 and figure 8.

Algorithm	Output accuracy
Logistic Regression	0.941133
Leaner Discriminant	0.920074
KNeighbors Classifier	0.930542
Decision Tree Classifier	0.986084
GaussianNB	0.972424
SVM	0.885468
Random Forest	0.992980

Table 4

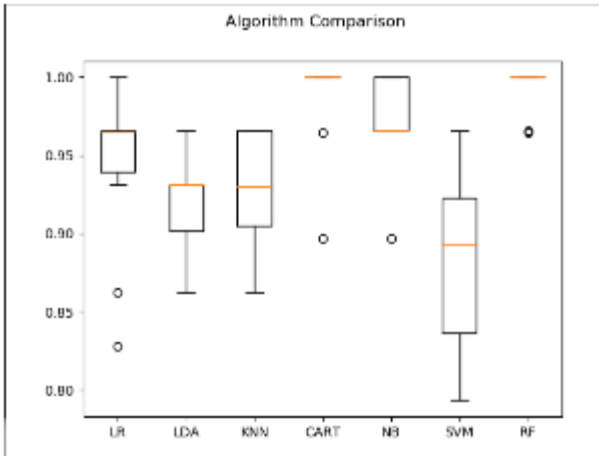


Figure 8

IV. ENVIRONMENT SETUP AND INTEGRATION

Initially, for the machine learning model training, Jupyter notebook environment was created. After finishing the Training and Evaluation processes, the final notebook file gets saved in local server in .ipynb format.

After the Evaluation identifies the classifiers for the project, since it's a python project, it'll need to install additional python modules.

- Step 1: Create a python virtual environment
- Step 2: Install the required python modules
- Step 3: Convert ipynb Notebook file to raw python script.
- Step 4: To create train models or see the accuracy, now can run the raw python file in command line using python command.
- Step 5: Integrate machine learning model to the developed IDS.

A. Methodology

Developed a simple Intrusion Detection system that is written using a python script. This program will monitor to protect from the common attacks on the network for the IPv4, such as finding if an attacker is sending unjustified traffic. This IDS script can be un-poison the sufferer and the default gateway by forwarding out defensive traffic with their original address.

Dependencies that are used to develop this IDS are;

1. Python 3
2. Pyshark python module

Pyshark will be used to capture the traffic. Then the Traffic data will be saved in the log file. In the particular time frame the log pcap file will be converted in to a csv file and be saved it in the test data directory. An automated script will run and get those csv into train model and then do the testing and print the accuracy as well as the results whether there is an attack or not.

Since this project is dealing with both normal and IoT network in the IDS, there will be a script that classify whether the traffic is IoT traffic or normal traffic based on the particular attributes.

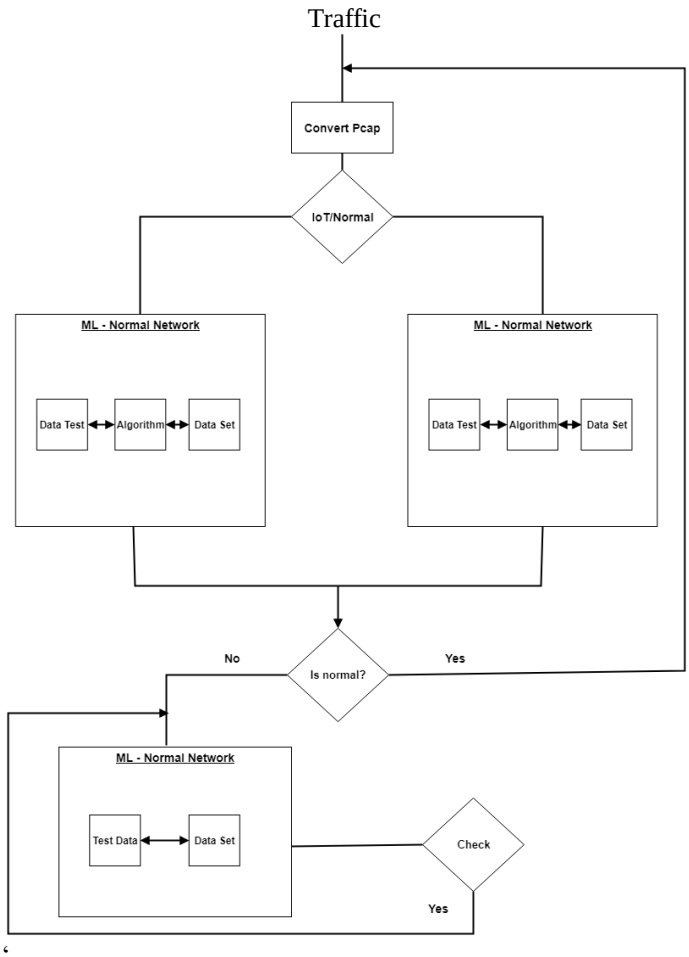


Figure 9

```
[**] [129:4:1] TCP Timestamp is outside of PAWS window [**]
[Priority: 3]
11/08-09:45:54.394828 7.204.241.161:25 -> 10.1.60.203:50176
TCP TTL:64 TOS:0x0 ID:1283 IpLen:20 DgmLen:40 DF
*****R** Seq: 0xE06DEE6F  Ack: 0x0  Win: 0x0  TcpLen: 20

[**] [129:4:1] TCP Timestamp is outside of PAWS window [**]
[Priority: 3]
11/08-09:45:54.395078 7.204.241.161:25 -> 10.1.60.203:50176
TCP TTL:64 TOS:0x0 ID:1284 IpLen:20 DgmLen:40 DF
*****R** Seq: 0xE06DEE6F  Ack: 0x0  Win: 0x0  TcpLen: 20
```

Figure 10

Intrusion detection system for 6LoWPAN based network.

Now a days IoT devices are becoming more popular and at the same time increasing cyber-attack percentage against IoT devices is also increasing. So, Intrusion Detection System for 6LoWPAN become a more essential need. IoT attack has main two type, which are inside attack and outside attack. Botnet attack and ferment attack and many attacks are in outside attack. In inside attack there are several attacks such as compromised or malicious node and etc. Anyhow, this research covers only 6LoWPAN protocol-based attacks. Based on that we took 3 main attack as our research scope. Those attacks are Hello Flood, Wormhole and Sinkhole.

6LoWPAN Background

6LoWPAN is a network communicating protocol for low-power wireless network. Also, it is a subnet of IPv6 network and part of IP based on infrastructures. It uses IPv6 packet to route in 6LoWPAN network. In this network all the device nodes are connected to internet through the Border Router. This protocol is based on tree approach. The root node sends DODAG information Object (DIO) message to the other node. After receiving the DIO message, the other node is going to calculate rank value of it. If the rank value is less than sender rank, the sender is going to become a parent or vice versa accordingly. Border router only one solution to access outside and inside of the network. So IoT senior networks are globally identified by IP address so, this method is helped to access these networks through the internet [1][8].

V. EXPERIMENTS OF MACHINE LEARNING CLASSIFIERS

Evaluation of the models based on the ten-fold cross validation method. The method is k-fold cross validation that involve the generation of validation set out of training dataset to access the trained model prior that is unprotected to test data.

In the K-fold cross validation, the training dataset is arbitrarily partitioned into k equal sized subsamples. Out of the k subsamples, a solitary subsample is held as the approval information for testing the model, and the rest of the k − 1 subsamples are utilized as training information. The cross-approval process is then rehashed k times (the folds), with each of the k subsamples utilized precisely once as the approval information. The k results from the folds.

```
models = []
#models.append(('SVM Classifier', SVC_Classifier))
models.append(('Naive Baye Classifier', BNB_Classifier))
models.append(('Decision Tree Classifier', DTC_Classifier))
#models.append(('RandomForest Classifier', RF_Classifier))
models.append(('KNeighborsClassifier', KNN_Classifier))
models.append(('LogisticRegression', LGR_Classifier))
#models.append(('VotingClassifier', VotingClassifier))

for i, v in models:
    scores = cross_val_score(v, X_train, Y_train, cv=10)
    accuracy = metrics.accuracy_score(Y_train, v.predict(X_train))
    confusion_matrix = metrics.confusion_matrix(Y_train, v.predict(X_train))
    classification = metrics.classification_report(Y_train, v.predict(X_train))
    print()
    print('============================== {} {} Model Evaluation ==============================='.format(grpclas
    print()
    print ("Cross Validation Mean Score:" "\n", scores.mean())
    print()
    print ("Model Accuracy:" "\n", accuracy)
    print()
    print("Confusion matrix:" "\n", confusion_matrix)
    print()
    print("Classification report:" "\n", classification)
    print()
```

Figure 11

Model Evaluate Results for each classifiers

```
============================== Normal_DoS LogisticRegression Model Evaluation

Cross Validation Mean Score:
 0.980822083372

Model Accuracy:
 0.980777512139

Confusion matrix:
 [[65527  1816]
 [  773 66570]]

Classification report:
             precision    recall  f1-score   support

        0.0       0.99      0.97      0.98     67343
        1.0       0.97      0.99      0.98     67343

avg / total       0.98      0.98      0.98    134686
```

```
============================== Normal_DoS Naive Baye Classifier Model Evaluation

Cross Validation Mean Score:
 0.973776072139

Model Accuracy:
 0.973768617377

Confusion matrix:
 [[65346  1997]
 [ 1536 65807]]

Classification report:
             precision    recall  f1-score   support

        0.0       0.98      0.97      0.97     67343
        1.0       0.97      0.98      0.97     67343

avg / total       0.97      0.97      0.97    134686
```

```
============================== Normal_DoS KNeighborsClassifier Model Evaluation

Cross Validation Mean Score:
 0.996569816347

Model Accuracy:
 0.99775774765

Confusion matrix:
 [[67287    56]
 [  246 67097]]

Classification report:
             precision    recall  f1-score   support

        0.0       1.00      1.00      1.00     67343
        1.0       1.00      1.00      1.00     67343

avg / total       1.00      1.00      1.00    134686
```

```
============================== Normal_DoS Decision Tree Classifier Model Evaluation

Cross Validation Mean Score:
 0.999769836897

Model Accuracy:
 0.999948027263

Confusion matrix:
 [[67343     0]
 [    7 67336]]

Classification report:
             precision    recall  f1-score   support

        0.0       1.00      1.00      1.00     67343
        1.0       1.00      1.00      1.00     67343

avg / total       1.00      1.00      1.00    134686
```

Figure 12

Test Model Performance Analysis

The trained models are utilized to categorize the labels in the test dataset those are collected from the HTTP traffic and convert it in to a csv format. Those are the test datasets that are not imported to algorithms before. The performance metrics such as efficiency accuracy, confusion matrix and classification report will be generated for the trained algorithms.

```
============================== Normal_DoS LogisticRegression Model Test Results
========

Model Accuracy:
 0.8338866561826548

Confusion matrix:
 [[6023 1435]
 [1417 8294]]

Classification report:
              precision    recall  f1-score   support

         0.0       0.81      0.81      0.81      7458
         1.0       0.85      0.85      0.85      9711

   micro avg       0.83      0.83      0.83     17169
   macro avg       0.83      0.83      0.83     17169
weighted avg       0.83      0.83      0.83     17169
============================== Normal_DoS KNeighborsClassifier Model Test Results
==========

Model Accuracy:
 0.8634748674937387

Confusion matrix:
 [[5760 1698]
 [ 646 9065]]

Classification report:
              precision    recall  f1-score   support

         0.0       0.90      0.77      0.83      7458
         1.0       0.84      0.93      0.89      9711

   micro avg       0.86      0.86      0.86     17169
   macro avg       0.87      0.85      0.86     17169
weighted avg       0.87      0.86      0.86     17169
============================== Normal_DoS Decision Tree Classifier Model Test Results
==============

Model Accuracy:
 0.3922185334032267 7

Confusion matrix:
 [[6576  882]
 [9553  158]]

Classification report:
              precision    recall  f1-score   support

         0.0       0.41      0.88      0.56      7458
         1.0       0.15      0.02      0.03      9711

   micro avg       0.39      0.39      0.39     17169
   macro avg       0.28      0.45      0.29     17169
weighted avg       0.26      0.39      0.26     17169
```

```
============================== Normal_DoS Naive Baye Classifier Model Test Results
===========

Model Accuracy:
 0.8337119226512901

Confusion matrix:
 [[5447 2011]
 [ 844 8867]]

Classification report:
              precision    recall  f1-score   support

         0.0       0.87      0.73      0.79      7458
         1.0       0.82      0.91      0.86      9711

   micro avg       0.83      0.83      0.83     17169
   macro avg       0.84      0.82      0.83     17169
weighted avg       0.84      0.83      0.83     17169
```

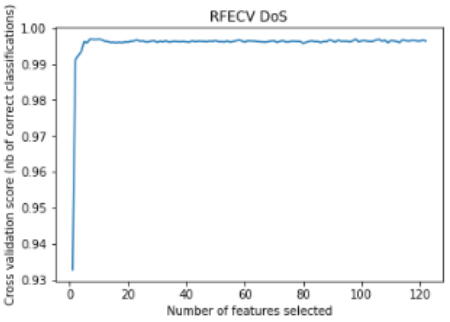
Summary of Confusion Matrix

A. Illustration with RFECV in Decision Tree Classifiers

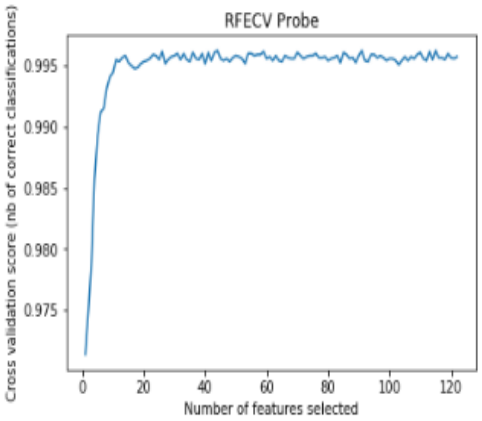
Creating the RFE object and do the computing for the cross validated score. The efficiency and accuracy of scoring is corresponding to the number of correct classifications.

1. RFCV for DoS Attack

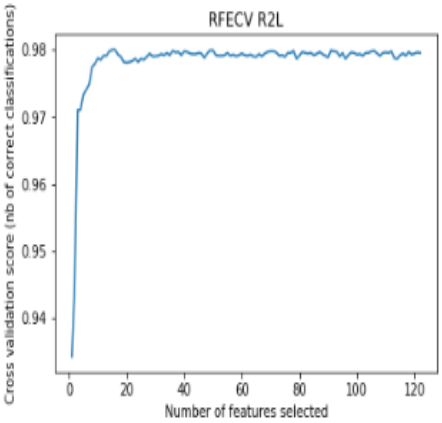
```
Automatically created module for IPython interactive environment
```



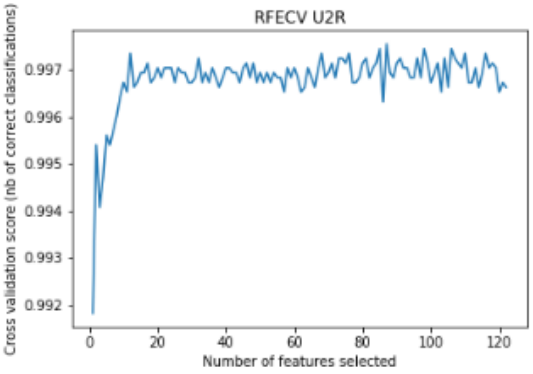
2. RFCV for PROB Attack



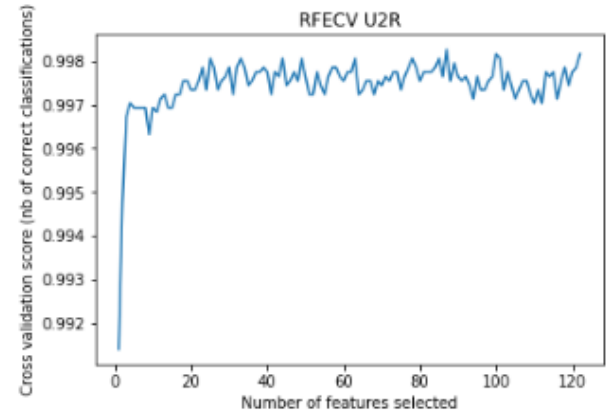
3. RFCV for R2L Attack

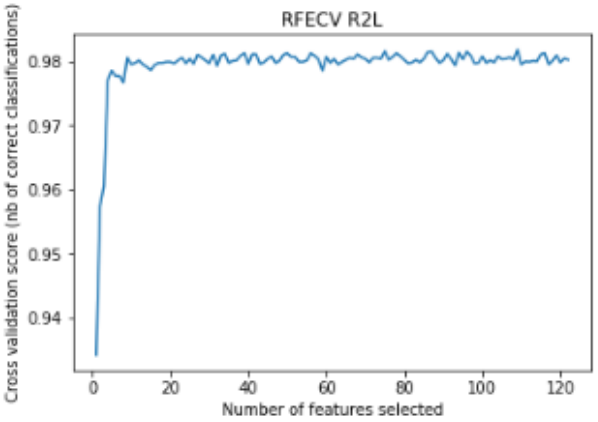


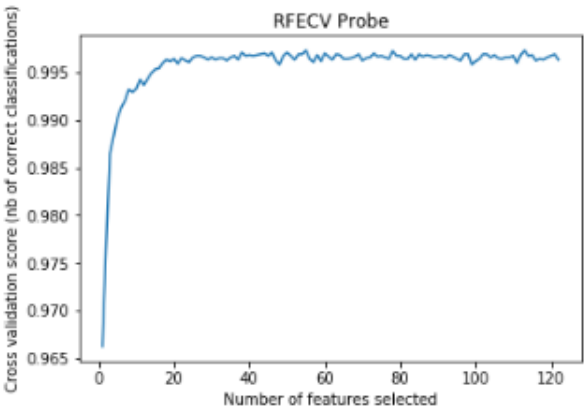
4. RFCV for U2R Attack

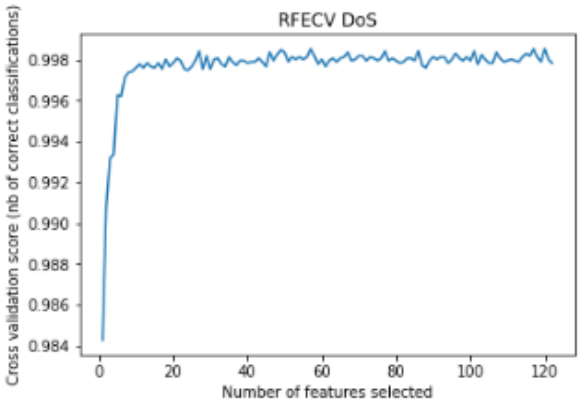


B. Illustration with RFECV in Random Forest Classifiers









Execution Time for the attack Classifiers

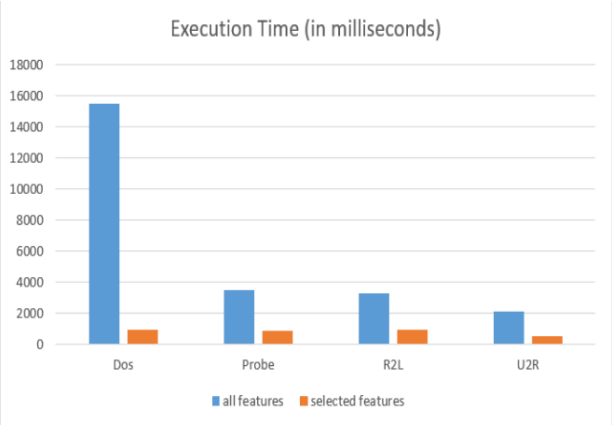


Table 5: Build Time taken for the Claffifiers on Each sub datasets.

Attack Classes	With all Features	With selected Features
Dos	15.5 seconds	0.959 seconds
Probe	3.48 seconds	0.868 seconds
R2L	3.31seconds	0.929 seconds
U2R	2.10 seconds	0.530 seconds

Results and Discussion

Based on the results that are obtained above, all the trained models are evaluated with the average 99% on half a training set when the model's performance on testing data that indicates the average of greater than 80% over the 4 main attacks group that are identified. Normal_U2R attack's test accuracy in all the models are greater than 90%. But when it is reviewed, an overview of performance as shown through the confusion matrix indicated that the 'U2R' detection charge changed into very terrible throughout Decision Tree, Random Forest, KNN and Logistic Regression fashions. The attack magnificence 'R2L' detection rate is likewise no longer far from being negative as 'Normal' and other instructions with higher distribution were given more attention of the fashions to the detriment of the low distribution instructions. The results typically confirm that sampling may additionally enhance model education accuracy. However, a negative detection fee in detecting U2R and R2L minority attacks are inevitable because of their massive bias available within the dataset.

Conclusion

A fact mining venture was accomplished to build models for classifying community intrusions, the usage of NSL-KDD dataset. Classifier algorithms which include Support Vector Machine (SVM), Naive Baye, Decision Tree, K-Nearest Neighbor, Logistic Regression and Random Forest have been educated, evaluated and tested to determine each model performance. Additionally, an ensemble Voting Classifier became also skilled to combine and common the person version performances. A key lesson learned in this undertaking is that, for class project, accuracy is not a correct degree of a model performance in which there is an imbalance dataset. Accuracy value can be excessive for the version however, the elegance with decreased samples may not be correctly classified. Metrics consisting of Confusion Matrix is more realistic in comparing classifier version's overall performance than accuracy measure. Also, finding the Attack in real time with the good accuracy will help for many system admins. False positives will be reduced.

ACKNOWLEDGMENT

These studies changed into support with the aid of Sri Lanka Institute of information technology. We thank our colleagues from SLIIT who provided perception and know-how that greatly assisted the research, despite the fact that they'll not trust all of the interpretations/conclusions of this paper.

We thank Prabath Lakmal Rupasinghe, Program Coordinator-MSc Cyber Security Degree Program at SLIIT for helping with machine techniques and guidance and Amila Nuwan Senaratne Lecturer at SLIIT for providing remarks that significantly improved the manuscript.

We might also like to reveal our gratitude to all sharing their pearls of understanding with us for the duration of the direction of this research, and we thank three "nameless" reviewers for their so-called insights. We also are immensely grateful to other friends for their remarks on an in advance version of the manuscript, despite the fact that any mistakes are our own and have to no longer tarnish the reputations of those esteemed people.

REFERENCES

[1] P. Pongle and G. Chavan, ''Real time intrusion and wormhole attack detection in Internet of Things,'' Int. J. Comput. Appl., vol. 121, no. 9, pp. 1–9, 2015. [Online]. Available: https://doi.org/10.5120/21565-4589

[2] S. Raza, L. Wallgren, and T. Voigt, ''SVELTE: Real-time intrusion detection in the Internet of Things,'' Ad Hoc Netw., vol. 11, no. 8, pp. 2661–2674, 2013. [Online]. Available: https://doi.org/10.1016/j.adhoc.2013.04.014

[3] T. Sherasiya, H. Upadhyay, and H. B. Patel, ''A survey: Intrusion detection system for Internet of Things,'' Int. J. Comput. Sci. Eng., vol. 5, no. 2, pp. 91–98, 2016. [Online]. Available: http://www.iaset.us/view_archives.php?year=2016&id=14&jtype=2&page=2

[4] M. A. Aydın, A. H. Zaim, and K. G. Ceylan, ''A hybrid intrusion detection system design for computer network security,'' Comput. Elect. Eng., vol. 35, no. 3, pp. 517–526, 2009. [Online]. Available: https://doi.org/10.1016/j.compeleceng.2008.12.005

[5] S. Shamshirband, A. Amini, N. B. Anuar, M. L. M. Kiah, Y. W. Teh, and S. Furnell, ''D-FICCA: A density-based fuzzy imperialist competitive clustering algorithm for intrusion detection in wireless sensor networks,'' Meas., J. Int. Meas. Confederation, vol. 55, pp. 212–226, Sep. 2014. [Online]. Available: https://doi.org/10.1016/j.measurement.2014.04.034

[6] S. Shamshirband et al., ''Co-FAIS: Cooperative fuzzy artificial immune system for detecting intrusion in wireless sensor networks,'' J. Netw. Comput. Appl., vol. 42, pp. 102–117, Jun. 2014. [Online]. Available: https://doi.org/10.1016/j.jnca.2014.03.012

[7] C. Kolias, G. Kambourakis, A. Stavrou, and S. Gritzalis, ''Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset,'' IEEE Commun. Surveys Tuts., vol. 18, no. 1, pp. 184–208, 1st. Quart., 2016. [Online]. Available: https://doi.org/10.1109/COMST. 2015.2402161

[8] Z. Shelby and C. Bormann, 6LoWPAN: The Wireless Embedded Internet, vol. 43. West Sussex, U.K.: Wiley, 2011.

[9] V. P. Singh, A. S. A. Ukey, and S. Jain, ''Signal strength based hello flood attack detection and prevention in wireless sensor networks,'' Int. J. Comput. Appl., vol. 62, no. 15, pp. 1–6, 2013. [Online]. Available: https://doi.org/10.5120/10153-4987

[10] K. Grgic, D. Zagar, and V. K. Cik, ''System for malicious node detection in IPv6-based wireless sensor networks,'' J. Sensors, vol. 2016, 2016, Art. no. 6206353. [Online]. Available: https://doi.org/10.1155/ 2016/6206353

[11] G.-H. Lai, ''Detection of wormhole attacks on IPv6 mobility-based wireless sensor network,'' EURASIP J. Wireless Commun. Netw., vol. 2016, Dec. 2016, Art. no. 274. [Online]. Available: https://doi.org/10.1186/ s13638-016-0776-0