Detection of IOT Botnet Attacks using Anomaly Detection

by Hemalatha Palanivelu, Lekha Vijayakumar

Abstract

In a IOT based network, the number of IOT devices getting compromised is propagating more due to the strong vulnerability existing in IOT endpoints. The most common infections which targets IOT endpoints are BASHLITE and Mirai Botnets. The botnets acquire the access of an IOT endpoint by brute force and the endpoint become a malware itself. The goal is to detect the infected endpoints by understanding the data flow behavior on the endpoints. We extract n features from the arriving packets and we train the data to develop a model to classify the data set. The aim of the project is to successfully differentiate between benign and malignant data in the data sets to detect the anomalies present in the data through anomaly detection method. Our evaluations were successfully able to detect malignant data by outlying the anomalies in the data set.

Acknowledgement

We would like to thank our Professor. Zhao, Juzi for encouraging, supporting and helping us complete the project successfully. We are thankful to our parents, teachers and friends for their invaluable support.

Table of Contents

List of Figures

List of Tables

1. Introduction (On IOT devices, attacks and detection techniques)

The Internet of Things means devices connected to the web which performs all actions without human intervention. IOT devices are utilized in number of applications but are prevalently more utilized in day by day computerization. The devices collect information from sensors at that point and either store the information locally or send the information to the server and prepare the information to send commands to other gadgets to encourage activity.

IOT gadgets are utilized in each conceivable space in the industry. Oftentimes, the IOT gadgets collect a tremendous sum of data which is handled by machine learning calculations to improve and construct them to modernize as per data growth. A huge need for protection of data arises as these gadgets collect and exchange data among them. The security of information is a bigger issue as the application of the technology include health tracking, observation of domestic environment and capturing imperative information through camera. While these gadgets collect delicate data, they also handle the information and, in some cases, clears out to form basic choices with this information. For instance, a pacemaker is a gadget responsible for preparing crucial information and takes straightforward activity on a human body.

As we see IOT gadgets collect more sensitive data and control bigger human into these gadgets without much exertion, they begun to get misused by the masses. Attackers filter the complete network and target the endpoints to construct gigantic botnets which leads to terabytes of malignant data being utilized by DDoS attacks. Taking these occurrences into consideration, the security community begun making mindfulness on how uncertain IOT gadgets are and why it is imperative to secure them.

Most IOT gadgets utilize Linux as their base working framework. Linux gives the adaptability of adjusting and compiling the working framework to have fair sufficient highlights to bolster a specific IOT gadget. The user too has the freedom to select any low power control. Given this, the most vulnerabilities found within the Linux framework are weak server-side controls, broken cryptography and Malignant usage of verification or authorization.

1.1 Common security issues of IOT

Malignant data are pernicious piece of computer program which is infected on the endpoint without the mindfulness of the user. Noxious program is designed with deliberation to attain assortment which involves taking vulnerable data to construct a huge botnet of target devices. *Mirai* and *BASHLITE* are common malwares among major IOT arranged malware which causes significant harm.

Mirai abuses basic infection in IOT gadgets like setting hard id and passwords for telnet. Mirai features a preloaded set of username and secret word patterns which it employments to constrain the device. Once Mirai effectively abuses a gadget it changes the device to a bot giving its control to the command and control server.

BASHLITE is also referred gafgyt is one of the popular malware prompting Linux based IOT devices to dispatch serious assaults. It is competent of propelling assaults of up to 400 Gbps. Most BASHLITE assaults are basic UDP, TCP surges and HTTP assaults. BASHLITE contaminate a IOT device by brute-forcing its telnet get to utilizing known default qualifications. The one most important perspective of BASHLITE is that malware payload conveyed in IOT gadgets has the BASHLITE’s C2Cs IP addresses hard-coded into it making easier to screen. Most of the infected gadgets are in Taiwan, Brazil and Columbia. The predecessor source code for Mirai is BASHLITE and is in constant competition for making IOT devices powerless.

1.2 Anatomy of IOT Botnets

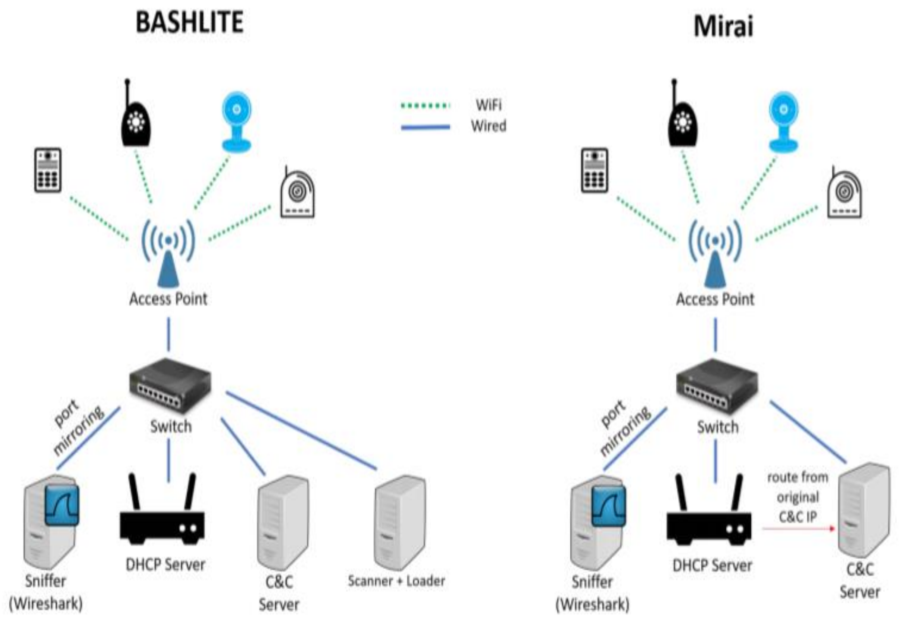


Figure 1 Generic Structure of IOT Botnet

Here, we analyze the structure of IOT Botnets and the mode of operations based on the open source released code of IOT Botnet infections. Specifically, BASHLITE and Mirai as well as based on the information from the collective invert building efforts made by numerous security analysts [4]. The code of IOT malware are generally composed in C dialect. For example, BASHLITE has been composed in C dialect. Mirai botnet employments both C and Google Go dialect. LightAidra/Aidra uses Python together with the C dialect. The source code for the bots is cross-compiled for numerous designs running the Linux working framework which is common working system among most of the IOT gadgets and implanted frameworks. The generic structure of the IOT botnet is outlined in the figure 1.1

The IOT botnets comprises for the most part of the two essential and four additional components, namely

(1) Bots or specialists or the end zombie IOT gadgets that perform DDoS assaults on command
(2) Command-and-control servers are utilized to control the bots
(3) Scanners are utilized to check for powerless IOT devices
(4) Announcing server utilized to gather the comes about or scan reports from bots or outside scanner
(5) Loaders utilized to log on to defenseless IOT gadgets and instruct them to download malware
(6) Malware conveyance server is the area where malware code is put away to be downloaded by tainted IOT device.

IOT malwares are getting to be progressively versatile and sophisticated with numerous modern highlights like IPv6 support, sophisticated communication strategies between bots and C2C's. The IOT scene is additionally developing huge with the decision of numerous customer electronic producers to deliver more Internet empowered shopper gadgets. The battle for dominance amongst diverse IOT malwares for the IOT genuine estate is a commonly watched characteristic of IOT malwares. For example, Carna, Darlloz and Wifatch evacuated LightAidra, Mirai expelled BASHLITE and small known Anime malware. This forceful behavior is to maximize the assault potential of the botnet gadgets and to avoid comparable evacuation attempts from other malwares.

1.3 Detecting Infected IOT's

Botnets have utilized unreliable consumer IOT gadgets to conduct conveyed refusal of benefit(DDoS) attacks on basic Web foundation. This motivates the improvement of unused procedures to consequently detect consumer IOT assault activity. In this paper, we demonstrate that informing IOT-specific arrange behaviors can result in high precision DDoS detection in IOT organize activity with assortment of machine learning algorithms, counting neural systems. The research on detection analysis also predict low-cost machine learning calculations to automatically identify neighborhood IOT gadget sources of DDoS attacks and activity data that is malignant.

For identifying assaults emerging from IOT bots, we propose a different approach based on network which employments deep learning techniques to perform irregularity discovery. Particularly, we extract factual highlights which capture behavioral snapshots of benign IOT activity and prepare a profound autoencoder (one for each gadget) to memorize the IOT's typical behaviors. The advantage of utilizing deep autoencoders, is their capacity to memorize complex systems, e.g. of different gadget functionalities. This comes about in an anomaly detection with barely any wrong cautions. We observationally show that the autoencoder's false caution rate is significantly better than other calculations utilized for anomaly detection [13].

2. Machine Learning

Machine learning is a technique of programming the system to work without human interactions and explicitly programmed. There are number of proposals which choose Machine learning technique for various type of attack detection. Machine learning has been through a great level of growth among many organizations aiming to build up their network using Internet of Things. Numerous companies are setting and assigning IOT in their network. As a result, these organizations produce enormous sum of information on a day by day premise and they suffer to efficiently store, analyze and utilize such sensitive information. On study of analysis, data analytics and machine learning prove to be a promising technique to detect and prevent attacks.

2.1 Data analytics

Data analytics is an important requirement before applying machine learning techniques to any raw data. Before knowing the value of machine learning, on a high level, it is significantly important to know the importance of data analytics. Practically, machine learning takes in lots of data and gives useful insights on the information. Data analysis is the initial processing step to take which greatly help in improving the design, cutting the cost and creating a better experience for the customers and for the developers to create a better business model. Data analytics is the one of the best technique at clarifying information. With the help of data sets obtained, we create reports or models of what happened in the past or what is happening in the present and draw valuable amount of information to apply and observe the behavior of a certain activity. Data analytics can help quantify and track objectives which embraces more brilliant choice making and implies for measuring success over time.

Collection of data sets from IOT devices for analysis is a very useful technique to start the detection. As the data models obtained from the devices are mostly static and restricted to less change. When in IOT, it is more critical to distinguish correlation between handful of sensor inputs and outside variables that are changing rapidly producing millions of data sets.

The data sets are obtained from the devices after a deep research about which values to consider for the analysis. For instance, a research on cancer analysis come up with many measures on patient's cell nucleus features calculated for each cell. The features calculated from the data sets play an important part in improving the accuracy level of the prediction. The more the important features calculated will result a precise prediction of abnormalities. So, that's why data analysis is an important step before applying machine learning techniques.

2.2 Machine learning algorithms

Machine learning algorithms makes a process function more independently. The algorithms are of different types: (1) supervised learning which include parametric and non-parametric algorithm, support vector machine, kernels and neural networks (2) Unsupervised learning which includes dimensionality reduction, clustering , recommender systems and deep learning. In general, there cannot be any comparison between the algorithms, as they produce accurate results on different criteria depending on where they are applied.

Majority of the practical applications uses Supervised Learning. Supervised learning has functions for input and output variable and the algorithm learns to create mapping function to match the expected outputs. The algorithm is called supervised learning since the algorithm learn from the dataset as a part of the learning process.

$Y = f(X)$

We already have the expected answers and rectified by the instructor. The training or learning stops when the answers meet the satisfactory level of execution. Supervised learning problems can be categorized mainly to two types: Regression and classification.

(1) Classification method is applied when the output must be marked to a category such as benign or malignant.
(2) Regression method is applied when the output must be marked to a real value such as weight, size or correlation values.

Unsupervised learning has only input variables and no corresponding expected output variables. The objective of unsupervised learning is to show the basic pattern in the data to fetch more information and predict output behavior. They can be grouped into two main types: Clustering and Association.

(1) Clustering method is a process of finding the different groupings within the information. For instance, gathering clients by obtaining behavior.
(2) Association is a process of finding rules that portray expansive amount of your information.

2.3 Gaussian Distribution model

In our project, we use classification and clustering methods to group the data either as benign or malignant. Clustering method performs various steps for grouping to predict the results. The data set is divided into training data and test data. The method uses the Gaussian distribution to model the data. As per Gaussian distribution, say X € R, if X is a distributed gaussian with mean μ and variance σ. Gaussian Distribution formula is given as follows,

$$\mathbf{P\ (X: \mu, \sigma^2) = 1/\sqrt{2\pi\sigma}\ \ exp\ (-(X-\mu)^2/2\sigma^2)}$$

Given the real data set, we must estimate gaussian parameters from a multivariate set of data. Gaussian Mixture Modes are the most used in unsupervised learning or clustering methods. They are said to be the most advanced type of model. Gaussian Mixture models are calculated to have multivariate gaussian distribution character with parameters mean vector μ and variance-covariance matrix ∑ . The probability density function for multivariate distribution is written as:

$$p(x) = \frac{1}{(2\pi)^{k/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)'\Sigma^{-1}(x-\mu)\right)$$

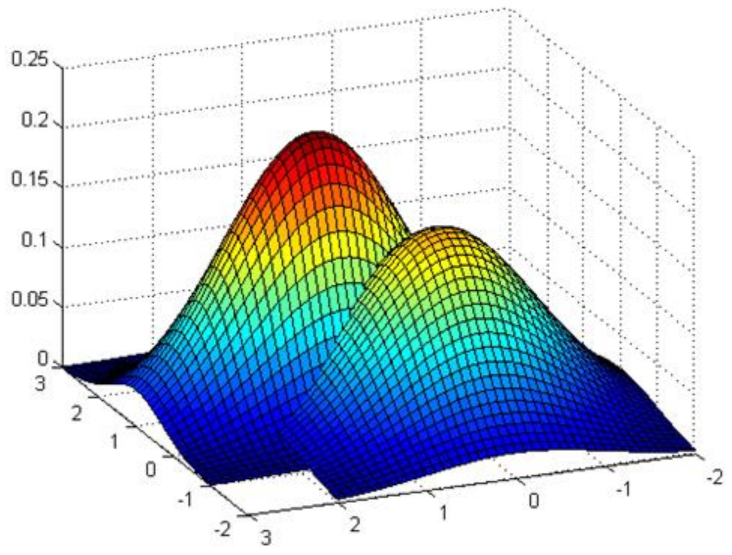


Figure 2.1 Gaussian Mixture Model

The classification method performs the following basic steps: Once the model is trained using the Gaussian model on the training set, we apply to the test datasets to get the ultimate results. The testing technique has the following steps in classification method:

(1) Isolate the flows into time windows

(2) On each time window, aggregate the flows in according to aggregation windows

(3) Utilize the Gaussian model to predict the group to which a flow belongs

(4) If the output is predicted as pernicious, it is assigned the malignant group and label as anomaly

(5) From each test data set, we compute error measurements of the calculation and predict the accuracy of the method.

From the steps, we predict the existence of malignant data indicating the presence of botnets in the machine learning technique.

3. Feature Extraction

Before applying the method of finding abnormality and the techniques for finding accuracy, it is vital to understand the data set, their feature set and the correlation between them to bring up better results. The data set has data obtained from 9 IOT devices categorized into separate files of benign and malignant data. The data set is separated into two parts: First part is the training data set with two-third of the benign data and Second part is remaining one-third of benign data plus the malicious data.

Initially, the packets coming from the source must be send to a packet capturer and to a packet parser and then sent to feature extractor step as in the order as follows:

The arriving packets pass through this process initially.

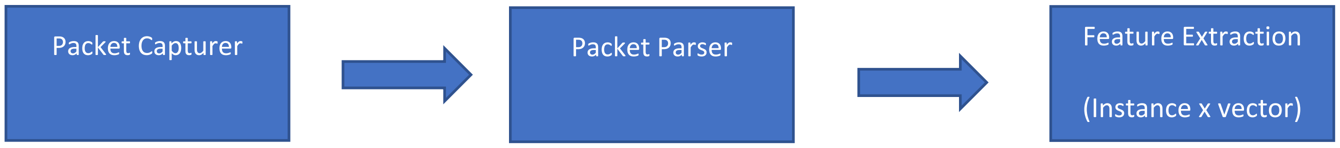


Figure 3.1 Flowgraph of traffic flows

Feature Extraction is responsible for extracting n features from the arriving packets to create the instance x vector. The features of x vector describe a packet and the network channel where it came from. It uses incremental statistics to perform feature extraction. We arrive at deriving 115 features for each device with five window sizes. We aggregate the stream from five different headers and fetch important values which are needed for accurate detection. The following is as shown in the table:

Value	**Statistic**	**Aggregated from**	**Number of Features**
Packet Size (Incoming packets)	Mean, Variance	Source IP, Source MAC-IP combination	8
Packet Count	Number	Source IP, Source MAC-IP combination, Channel and Socket	4
Packet Jitter	Mean, Variance, Number	Channel	3
Packet Size (Incoming and outgoing packets)	Magnitude, Radius, Covariance and Correlation Coefficient	Channel, Socket	8

Table 3.2 Extracted features

As from the table, we have four values and four stream aggregation parameters, we totally have 23 features for each window size. We are using five temporal sizes such as L5, L3, L1, L0.1 and L0.01 which denotes 500ms, 100ms, 1.5 s, 10 secs and 1 min. On total, we have 115 features. The statistics extracted from the packet stream are as follows:

a) Weight: The weight of the stream means the number of items observed in the recent capture. [3]

b) Mean and Variance: The calculated mean of the number of items observed in a stream [3]

c) Radius: The root squared sum of the two stream's variances [3]

d) Magnitude: The root squared sum of the two-stream's means [3]

e) Covariance: An approximated covariance between two streams [3]

f) Correlation Coefficient: An approximated correlation coefficient between two streams [3]

These statistics are performed over a dynamic number of data streams. The whole feature extraction process uses Dynamic incremental statistics phenomenon to extract these features.

3.1 Dynamic Incremental Statistics

The process of feature extraction uses a framework for high speed extraction of the recent behavior of packets. To make the process dynamic, it uses damped window sizes which has many advantages. Using damped window sizes means small memory footprint. The damped incremental statistics uses no memory to store information and just process the recent data. Damped window also means the extracted information at a step are temporal. The incremental statistics is removed when dampening weight equals zero. Therefore, this method has O (1) complexity and it used hash tables to store temporally.

Incremental statistics uses a tuple IS. Let S be a sequence of observed packet sizes. S = {x1, x2, x3….} be an unlabeled data stream and x_i € R. The mean, variance and standard deviation of S updated incrementally using the tuple IS.

IS = (N, LS, SS)

where N = number of instances seen so far, LS = linear sum of instances seen so far, SS = squared sum of instances seen so far. Every time, incremental statistics uses this tuple to calculate a value. And after when a new instance occurs, new instance x_i should be inserted. The tuple for inserting new instance is IS = (N+1, LS+ X_i , SS+ X_i^2).

The statistics for any given time is calculated by following basic formulas:

Mean $\mu = LS / N$

Variance $\sigma_s{}^2 = |\ (SS/N) - (LS/N)^2\ |$

σ_s = standard deviation $\sqrt{\sigma_s{}^2}$

Incremental statistics uses these formulas to calculate the IS tuple. There are some disadvantages with incremental statistics. The approach has small memory requirements and time complexity O(n). Sliding window approach is not considering amount of time covered by the window. These shortcomings are reduced by using damped incremental statistics.

Damped incremental statistics uses window sizes where weight of older values is exponentially decreased over time. To make the process to behave in a damped manner, we have a delay factor function. Let d be the decay function, $d\lambda(t) = 2\text{^} -\lambda t$

where λ is the delay factor

t is the time elapsed since last observation from stream S_i

The tuple for the damped incremental statistics is **$I\ S_i, \lambda = (w, LS, SS, SR_{ij}, T_{last})$**

Where w is the current weight, T_{last} is the timestamp of the last update of $I\ S_i, \lambda$ and SR_{ij} is the total sum of residual multiplications or products between streams i and j. Therefore, the algorithm for inserting a new value into a damped incremental statistic as follows:

Update $(IS_i, \lambda, X_{cur}, t_{cur}, r_j)$

(1) Computing delay factor by updating the delay function $\gamma = (t_{cur} - t_{last})$

(2) Processing delay $IS_i,\lambda = (\gamma w, \gamma LS, \gamma SS, \gamma SR, T_{cur})$

(3) Inserting the new update, $I\ S_i, \lambda = (w+1, LS+X_{cur}, SS+X_i^2, SR_{ij} + r_i r_j, T_{cur})$

(4) Return I S_i λ

From all these process, we extract the features and final vector x vector is derived and passed to the next step.

4. Artificial Neural Networks

The Neural Networks play an important role in understanding non-linear complex relations in huge datasets. The model involves establishing a relation between input features and thus once trained can recreate the output approximate to the input. In our project which involves Network Intrusion Detection Systems, the neural network provides the promising solution with their capacity to learn from traffic patterns and differentiate the Malignant data from the Benign data.

We are using an ensemble of neural networks that are the Autoencoders. This model is called Kitsune which is a plug and play NIDS for Anomaly Detection. The Kitsune's essential algorithm is the Kitnet that differentiates the normal data traffic from the malicious traffic. This NIDS if deployed at multiple points in a network can monitor the entire network for malicious activities. Thus a distributed deployment is the best strategy for better scaling across a number of routers and gateways in a network. Thus the Autoencoders are better than the other Machine Learning algorithms at detecting non-linear and complex patterns in the dataset.

The procedure involves training an Autoencoder with Benign data to make the model learn from the normal traffic patterns and establish relations between in a dataset. Then train the ANN to automatically classify the Malicious data instances from the Benign ones. And now we can deploy the trained model at any network organization's NIDS. we can now execute the trained model in a new data traffic.

The main disadvantages of the approach include,

- Offline Processing since it is not feasible to train the model in a network where the labelled instances are not available.
- Supervised Learning, which makes the model very expensive as it involves a lot of work and time.
- High Complexity of ANN computations, since the complexity of the ANN increases with increasing number of neurons in the Neural Network.

To avoid the above problems, we can train and execute the model with an instance that can will be later discarded. Thus, reducing the problem of memory usage. And we use Unlabeled data and the model is fed with all the required metadata information.

Thus, our project involves a model named Kitsune that is Unsupervised, online and memory efficient. The terminology, Kitsune is from a Japanese belief which is a fox like creature that has a number of tails and can change its appearance to align to any form. Similarly, our model Kitsune is an ensemble of Autoencoders which can be trained to reconstruct complex patterns in network traffic. The performance of the model increases with training over time.

Every instance of Training data is fed as input to an individual Autoencoder which maps the features of the instances to neurons in its neural network. It then tries to reconstruct the instance features. And computes the RMSE score based on the proportion of the correctness of its reconstruction to the actual output. These RMSE’s from the ensemble of trained Neural Networks are fed as input to an autoencoder. Thus, while training the Kitsune model only one instance is stored in memory at a time. The one main parameter of the Kitsune is the maximum number of inputs to any autoencoder in the ensemble. We use an ensemble of autoencoders instead of a single autoencoder for increased efficiency and less noise.

The feature extraction framework is used for automatically extracting the necessary features from the network traffic. This framework works on a small memory space since the statistics extracted from the traffic are updated incrementally over damped windows are the memory footprint is erased subsequently.

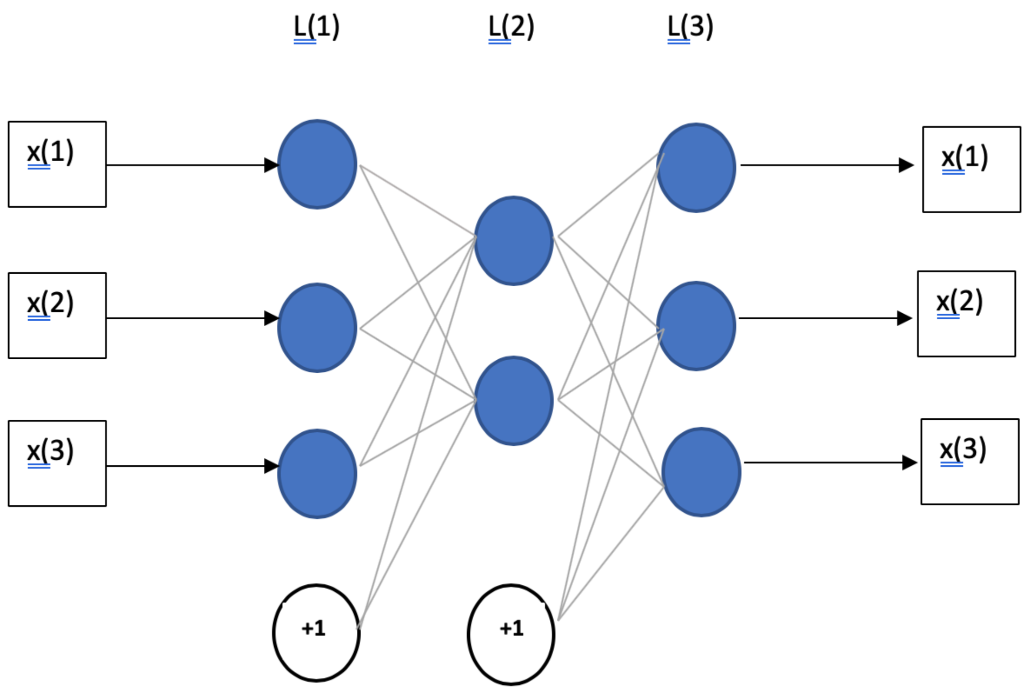
5. Anomaly Detection model:

In this paper we have implemented an anomaly detection model where we an Autoencoder was implemented for every device for the purpose of Feature Mapping. The Feature Mapping is a procedure involving creating a smaller set of features from the dataset that was extracted after Feature Extraction. The feature mapping involves creating k smaller instances for every layer of the Anomaly Detector.

The previously done work on NIDS uses Artificial Neural Networks where multiple ANN classifiers were used and each one of them were trained to detect a attack. Another work on NIDS involved an hierarchical method where a packet when passed through the anomaly detector if flagged as anomaly is then evaluated by a set of ANN classifiers that detects a specific attack type. But these models are not very efficient since they are Supervised or requires a big memory which is not feasible in simple network gateways.

Another implementation of the Autoencoders was done with an ‘ensemble of deep neural networks that can address the problem of object tracking online’. The model used SDAE autoencoder where each layer of the Autoencoder model serves as different feature spaces which is transformed to a deep neural network for the raw image data which is used as binary classifier. The classifiers and not the anomaly detectors were used to detect the anomalies. But the model required labelled data which is not very efficient.

The autoencoders are also used to detect anomalies in Power grids which were done offline and is not applicable for Network Intrusion Detection Systems. The architecture is lightweight and scalable as an ensemble. The feature extraction model in this architecture is being used in our proposed model. In the proposed model, Autoencoders are the building block of the Kitsune model. To introduce Autoencoders and they work, we will refer to the example image illustrated below. The figure depicts an Autoencoder with a single compression layer.



The ANN depicted above is made of layers of neurons that are connected to one another sequentially by the synapses. The weight associated with the synapses defines the relations between features learned by the model. If there are l(i) layers in the Artificial Neural Network, and if there are |l(i)| neurons in it, and if the total number of layers in the network is L, then the connection between the two layers can be denoted as the ||l(i)|| * || l(i+1) matrix

W(i) and ||l(i+1)|| dimensional bias vector b(i). The parameters in general are represented as a tuple. It is Ɵ = (W, b). W and b represent the weight associated with that particular layer.

In general, the entire neural network can be represented into two distinct parts ie., the Visible layer and the Hidden layer. The Visible layers are the input and the output layer. The Input layer takes in the array of vectors as inputs for every input instance and a bias variable is initialized with a constant value 1. The input vector generalized as the x vector is a vector formed of numerical features that describes the x at the time. It is usually normalized to be in range of [-1, +1]. The features when fed into this layer is passed along to the subsequent layers where the weights of the link are applied to the features and computations are done at the nodes. Subsequently computations are done along the forward propagation and are added up along the way and reaches the Output layer.

6. Execution of an ANN

The second hidden layer l(2) is activated once it receives the output from the first hidden layer l(1) which is weighed by a factor of W(1). Now The output l(2) weighed with W(2) activates the l(3) and this continues till the final layer in the network is activated. The process is called as *Forward-Propagation*. If a(i) is ||l(i)|| vector of outputs from layer l(i). Now to get a(i+1),

a(i+1) = f (W(i) . a(i) + b(i)) , where x is the vector of inputs at the instance

Where f is the neuron's activation function. One of the common activation function is the **Sigmoid function,**

f(x) = 1 / (1 + e ^ x) , x is the vector of inputs at the instance

The Output layer after all the computations produces y' vector, and is represented as

hƟ(x) = y' , x and y' are the input and output vectors respectively.

This is the execution part of the ANN which is followed by the Training of the ANN where the model is made to learn from a dataset and understand the patterns in the dataset and to reconstruct the output.

7. Training of an ANN

The model implemented in the execution part of the ANN is now trained to learn from patterns from the training dataset. The Training set is composed of several instances with x vectors having a number of features and their respective outputs are y vectors. During the process of training the network, the weights of the Artificial Neural Network are tuned such that hΘ(x) = y'. This process of training the ANN that the optimum weights W and b are found for a given set of (X, Y). This technique is referred to as the back-propagation Algorithm.

This process of fine tuning the Optimum weights for every neuron of the ANN model is done by propagating the error between the predicted value of y vector and the expected value of y vector from the output to each neuron. During this process the error value between the neurons actual and the expected activation values are stored.

For some execution of hΘ, let A be the activation of every neuron and Delta be the activation errors of all the neurons. The Learning rate for such a network is $l \in (0, 1]$. Using the Gradient Descent algorithm, we will now incrementally optimize the weights of W and b. Thus, implementing the *back-propagation algorithm* in the Neural Network.

The Gradient Descent (GD) updates the value of weights according to the total errors of all instances of X and hence this technique is called as *Batch-Training*. The other approach is called the Stochastic-Gradient-Descent (SGD) where the weights are updated based on the error value noted in every instance. But Gradient Descent converges much better than the other method, even though the other method converges much faster.

In our model we are using the SGD with max_iter = 1. While in the Training phase of the algorithm, every time a new instance arrives we perform a single iteration with the Back-propagation algorithm and discard the instance and wait for the next input instance. This frees up the memory footprint making the model more cost efficient and online.

8. Autoencoders:

The Autoencoder is an ANN that is trained to reconstruct its input at the output ie., X = Y. It works on the function, $h\Theta(x) \approx x$ vector. Thus, it is understood that the autoencoder is trying to make a relation between the input features in the instances. The main constraint in the model is the limit the of neurons in the inner layers of the model. The path makes the model to learn complex ‘encodings(compressions) and the decoding’s of the input x’ . Same weights can be used for encoding and decoding the autoencoder model if the model is symmetric in layer sizes thus reducing the computations done during the training process.

There are various applications of autoencoders like filtering out noise from Images, generating new contents. An autoencoder trained on dataset can reconstruct its test instances that is new from the same distribution of trained dataset. If a test data does not belong to the relations learned during the training then the reconstructed output will have an high error, indicating the anomaly. This error in reconstruction is measured with RMSE value, which can be generalized as,

RMSE (x vector, y vector) = (Σ for i=1 to n, (x(i) - y(i))^2 / n)^½

, where n is the Dimensions of the input vector x.

Initialize **Φ = 1**, to be the anomaly threshold and **β ∈ [1,** ∞) be some sensitivity parameter, now to implement the autoencoder we do the following steps,

1. Training mode:
    - Involves training the autoencoder model with our normal (clean) training data.

      For every instance of x(i) in our training set X:

      a) we Execute: **s = RMSE (x , hΘ(x))** ,where x is the instance vector of the training set X

      b) Update: **If (s ≥ Φ) then Φ ← s**

      c) Train, Update **Θ** by learning from x(i)

2) Execution mode:
    - Involves working with unknown instances of x vector

      a) Execute: **s = RMSE (x , hΘ(x))**

      b) Verdict: **If (s ≥ Φ) then Alert**

8.1 Complexity of Autoencoders

This is the Kitsune procedure for reconstructing input instances at the output using the Autoencoders. [2] The complexity of the activation layer is O(l(i). l(i+1)) ^2. Thus, the complexity of an Artificial Neural Network is totally dependent on the number of layers in the model and the number of neurons in every layer of the model. The complexity of training the Artificial Neural Network is about twice the complexity of executing an ANN because of the back-propagation involved in the training process. Having many hidden layers in ANN helps the model to learn complex relations between the features however they are computationally expensive to be trained and executed. Therefore, the Kit NET model of Autoencoders is therefore has a limit of three layers and at most of seven visible neurons.

This NIDS model we build is composed of four major components. They are

1. Packet-Preprocessing Framework
2. The Feature Extractor
3. The Feature Mapper
4. The core Anomaly Detection Algorithm

The KitNET model has an input parameter *m*, which is the max number of inputs for each of the Autoencoder. Thus there is a tradeoff between the runtime cost and Detection. The model as mentioned before in the paper is a ‘Plug and Play NIDS’, entirely based on the Artificial Neural Networks. The goal of the model is to take a traffic instance at input and detect the instances that are abnormal. This is done by 1) Monitoring the statistical patterns in the Benign (Normal) data traffic 2) Detection of Anomalous patterns using an collection of Autoencoders. Every Autoencoder is designed to detect anomalies related to a specific aspect of the networks behaviour. Since the model is designed to work on small network routers and the memory footprint of our Kitsune model is small, and the computational complexity is also less.

8.2 Packet-Preprocessing Framework

This part of the Kitsune framework is composed of *Packet Capture* and *Packet Parser*. The libraries used to acquire the raw packets are the Wireshark API’S: NFQueue, afpacket, tshark. This raw data packet captured in pcap files are forwarded to the Packet Parser that parses important meta information required by the Feature Extractor. It parses around 100 statistical features from the data which form the instances x vextor . The Libraries used for Packet Parser include, Packet++ and tshark.

8.3 Feature Extractor:

This component of the framework is made to extract the required n features (115) that are the arriving data traffic to make the x vector. The features extracted here contains all the information about a packet like the channel used, MAC-IP port used and so on. The features extracted defines the packets contents and need for the packet in the network.

Examples: Consider a single tcp syn packet in a network. This packet could be a single packet trying to get a connection with a server which would make it a Benign packet, or it could be one in a million of similar packets that is being sent to cause a DoS attack. Another example is a video from a Surveillance camera. Even if the contents of the data traffic are legitimate, If you analyze the packet and realize there is a consistent rise in the Jitter, it means that the traffic is constantly sniffed with a ‘man-in-the-middle attack’. Thus statistical features can help us detect anomalies.

Here the incremental statistics are maintained over a damped window. The model maintains 1D and 2D incremental statistics of a connection. The 2D statistics captures the relation in incremental statistics between the sending and receiving traffic. The incremental statistics computed from the traffic streams are,

1D: Weight, Mean, Standard deviation

2D: Magnitude, Radius, Approximate Covariance, Correlation Coefficient

The features extracted from packets sender and the traffic between the packets sender and receiver are SrcMAC-IP (packets source MAC and IP address), SrcIP (source IP address), Channel used at the sender and receiver, TCP/ UDP Socket of the packets sender and receiver. The following statistics are extracted from the features which is a total of 23 features extracted from a single time frame of λ

. The window sizes are 100ms, 500ms, 1.5sec, 10sec and 1min, thus making a total of 115 features.

TYPE	**STATISTIC**	**NOTATION**	**FORMULA**
ID	Weight	w	w
	Mean	μ_{si}	LS / w
	Standard Deviation	σ_{si}	(|SS/w - (LS / w) |)^1/2
2D	Magnitude	$\|\| S_{i,} S_j \|\|$	(μ_{si}^2 + μ_{sj}^2)^1/2
	Radius	$R_{si, sj}$	(σ_{si}^2 + σ_{si}^2)^1/2
	Approx Covariance	$Cov_{si, sj}$	SR_{ij} / (w_i + w_j)
	Correlation Coefficient	$P_{si, sj}$	$Cov_{si, sj}$ / ($\sigma_{si}\sigma_{sj}$)

The Feature Mapper and the Anomaly Detector runs on two different mode of operation: 1)Train-mode 2) Execution-mode. The Train mode is converted to Execution mode after a time limit. The components of Train mode constantly updates its variables with the training inputs but does’nt generate any output. But the components of the Execution mode do not update its variables but produces the output.

8.4 Feature Mapper:

This component of the framework is responsible for mapping the n feature of the x vector to k different smaller ‘sub instances’, with one ‘sub instance’ for every encoder in the ensemble layer of Anomaly detector.

Let v denote the set of k different sub instances, v = {v1, v2, v3, …, vk}where vk is a sub-instance vector.

- **Train-mode:** The model learns a featuremap, from the input x vector. This map groups the features to a *maximum size of m* for each set v. Once the mapping is complete, the map is passed along to the Anomaly Detector. Now the AD uses this map to build the ensemble architecture where each set of m, forms the input to an individual Autoencoder in the ensemble.
- **Execution-mode:** The learned mapping is used to create a collection of small instances of v vector from x vector. It is then passed along to the respective Autoencoders in the ensemble layer of the Anomaly Detector.

Conditions for Feature Mapping, f(x) = v:

1. Every sub-instance vector of v should not have features more than m. This reduces complexity of the ensemble.
2. Map all the n features of the x instance only once to the features of v
3. Contain subspaces of X well enough for normal behavior such that it can easily detect anomalies that are in the defined features.
4. one instance at a time is stored in the memory footprint.

[2] To maintain these rules, we map the function by clustering the features into k different groups no larger than m by doing the agglomerative hierarchical clustering on the incrementally updated statistical data. The steps done in the feature mapping after receiving an instance x can be put as,

- During the Training mode, incrementally updating the summary statistics with features of the x instance.
- Once the Train mode ends, performing the hierarchical clustering on the statistics to form the function f
- During the Execution mode, performing the function, f(x) = v and then passing the v to the Anomaly Detector.

8. 5 Agglomerative Hierarchical Clustering

To make sure that the features that were grouped capture the normal behaviour during the process of clustering, we calculate another parameter called the ‘correlation distance between the two vectors (u, v)’, dcorr

dcorr = (u - u’) . (v - v’) / [|| (u - u’)^2 || || (v - v’)^2]

Where u’ and v’ are the mean of the elements in the u vector and v vector and u . v is the dot product between the vectors. The correlation distance between the features can be used to establish a relation between the features and can also be used for the purpose of clustering. Let n(t) be the number of instances, c vector be the n dimensional vector containing the Linear sum of each features values, c(r) denote the vector containing summed residuals of each feature, c(rs) denote the vector containing the summed squared residuals of each feature. Then the Correlation matrix (C) of order n*n and the Correlation Distance (D) between each feature of X is,

$$[C_{i,j}] = \sum_{t=0}^{n_t} \left(\left(x_t^{(i)} - \frac{c^{(i)}}{n_t} \right) \left(x_t^{(j)} - \frac{c^{(j)}}{n_t} \right) \right)$$

$$D = [D_{i,j}] = 1 - \frac{C_{i,j}}{\sqrt{c_{rs}^{(i)}} \sqrt{c_{rs}^{(j)}}}$$

The algorithm begins with n clusters and performs Agglomerative Hierarchical Clustering on D to find the relation established using a function f. Basically each cluster is represented as a point represented by D and then starts searching for nearby clusters and joins their associated clusters together. This process is repeated until all clusters are joined together as one big cluster containing n points. Now the D matrix becomes small. This process makes sure that all the clusters have features less than m and all the k groups have strong inter-correlation making sure that not any group has more than m features. The groupings are saved to make use for performing the mapping f. This sort of hierarchical clustering cannot be large datasets because of the complexity involved.

8.6 KitNET Anomaly Detector

- **Train-mode:** It takes the v to train the respective Autoencoder in the ensemble layer. Now the RMSE is calculated during the forward-propagation and it is used in training the output layer. The Largest value of RMSE obtained during the training is stored as ϕ which is used for evaluations during the Execution phase.
- **Execution-mode:** The v vectors derived from the execution phase of the Feature Mapper is executed at the respective autoencoders of the ensemble. Now the RMSE is calculated and,

  If **(RMSE ≥ Φβ) then an Alert is logged** along with the packet details.

The Anomaly Detector is made of 2 layers of Autoencoders namely 'the Ensemble Layer and the Output Layer'. The Ensemble layer is formed of a set of k three layered autoencoders mapped to their respective instances of v. This layer measures the abnormality of each instance of v. Then the training mode the autoencoders now understands the normal patterns of their instances. During the Training and Execution mode, every autoencoder sends out their reconstructed RMSE as its output to the Output layer. Based on the RMSE values received by the Autoencoder, it then tries to establish an Anomaly score considering the normal data patterns and the naturally occurring noise in the traffic.

Operation of Ensemble layer and the Output layer for Anomaly Detection:

We are now going to discuss about how the model will work on the Ensemble and Output layers of the network.

a. Initialization: Once the Anomaly Detector receives the set of mapped x of v from the Feature Mapper, and now the Anomaly Detector begins to initialize the KitNET architecture using the v instances as the blueprint. Consider an entire autoencoder denoted as θ and if L (1) represent the Ensemble and L(2) represent the Output layers the model. [2] Then the $L(1) = \{\theta(1), \theta(2),\ldots, \theta(k)\}$, and the autoencoder represented at i , θ(i) that belongs to the **L(1) having a dimension, dim(v(i)).** If there are three layers of neurons in both of the input layers and output layers , and $[\beta \cdot \dim(v(i))]$ number of neurons at the inner layers where $\beta \in (0,1]$. The L(2) layer is defined by a single autoencoder θ(0) which has k input and output neurons and | k . β | inner neurons. The L(1) and L(2) are not related with weighted synapses. The output normalized value of RMSE error values from each of the respective encoders in the L(1) are 0 to 1 i.e.., ensemble layer. Signaling the aggregated errors from the entire autoencoders instead of the individual neutrals of the autoencoders reduces the models. [2]Then, the weights of all the autoencoders, θ(i) are initialized from the uniform distribution with random values

$$\mathcal{U}\left(\frac{-1}{\mathbf{dim}(\vec{v_i})}, \frac{1}{\mathbf{dim}(\vec{v_i})}\right).$$

b. Training-mode: The Kit NET signals the RMSE error of reconstruction between the two layers (Ensemble, Output) of the autoencoder. The Kit NET is trained using Stochastic Gradient Descent (SDE) using the observed instance v only once. This algorithm is the *back-propagation algorithm*. During this training we need to normalize the features in range of 0 → 1. To do this, we will need the maximum and minimum value observed for each of the input features. These values are updated during the training mode. Furthermore, the model is trained only with the Training (normal) data that do not have anomalies to establish a range of the normal anomaly score for benign instances.

c. Execute-mode: At this stage the Kit NET does not perform any updating of the internal parameters. It performs a *forward propagation* through the whole network and this outputs only the L (2)'s reconstructed RMSE values. These error values represent the x abnormalities with respect to the relationship between the parameters in v. For example, consider two different autoencoders from the ensemble layer L(1) ,θ(i) and θ(j). If the RMSE of θ(i) and θ(j) correlate at the Training mode, then a lack of this correlation during the Execution mode is considered to be an Anomaly. Since such relationships between the autoencoders are learned during the training process, we could easily detect anomalies using the L(2) error of reconstruction of the L(1)'s RMSE value.

d. Anomaly-scoring: The Output generated at the Kit NET is the RMSE anomaly score, $s \in [0, \infty)$. The larger the value of s, the larger is the anomaly. To establish a threshold for Anomaly score, we define the parameter **Anomaly score cutoff threshold Φ.** We have to come up with the value of Φ to establish a cutoff and to flag anomalies above the value. This value varies with our application. There are several methods to determine the best value of Φ. One way is to set it as the maximum score seen during the training mode. An another way is to plot the RMSE outputs to a log of the normal or any other distributions and raise a flag whenever there is a value of s that has very low probability of occurring.

9. Summary and Conclusion

In summation, we applied the machine learning and deep learning techniques to efficiently perform evaluations on the data set. We were able to successfully train the Autoencoder to detect anomalies based on the anomaly score estimate based on the RMSE reconstruction error from the Ensemble layer of the Anomaly Detector. Since the Output layer learns the patterns and relations between the input features in the training data, the reconstruction error of the Ensemble layer's RMSE is able to reflect the anomalies in the data and thus we were able to differentiate malignant instances of data from the benign instances. The algorithm was able to successfully detect anomalies in the cross validation and test data set.

10. References:

[1] Incremental learning of gestures by imitation in a humanoid robot by Sylvain Calinon and Aude Billard in International Conference on Human-robot interaction 2017.

[2] Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection in Network and Distributed System Security Symposium, California by Mirsky, Doitshman, Elovici and A. Shabtai 2018.

[3] 'N-BaIoT: Network-based Detection of IoT Botnet Attacks Using Deep Autoencoders in IEEE Pervasive Computing 2018 by Meidan, Bohadana, Mathov, Mirsky, Breitenbacher.

[4] A review of machine learning based anomaly detection techniques in 2013 by Harjinder Kaur, Gurpreet Singh and Jaspreet Minhas.

[5] Machine Learning course by Professor Andrew Ng

[6] Neural Networks: Tricks of the trade in Stochastic gradient descent tricks 2012

[7] An empirical analysis of cyber security incidents at a large organization by Marshall Kuypers, Thomas Mailart and Elizabeth cornell in Department of Management Science and Engineering, Berkeley, 2016

[8] Evaluation of anomaly-based ids for mobile devices using machine learning classifiers in security and communication networks by Dimitrios Damopoulos, Sofia Menesidou and Maria Papadaki 2012

[9] An efficient intrusion detection system based on support vector machines and feature removal method in Expert Systems with Applications 2012

[10] Neural Network Design by Demuth, Beale, Jess and Hagan 2014