Visualising Network Anomalies in an Unsupervised Manner Using Deep Network Autoencoders

Matthew Banton, Nathan Shone, William Hurst, Qi Shi
Department of Computer Science
Liverpool John Moores University
Liverpool, UK
e-mail: m.d.banton@2017.ljmu.ac.uk, {n.shone, w.hurst, q.shi}@ljmu.ac.uk

***Abstract*—As network data continues to grow in volume, it is important that network administrators have the tools to be able to identify anomalous network flows and malicious activity. However, it is just as important that tools allow the administrator to visualise this activity in relation to other benign activity. As such, this paper will propose a method to not only identify malicious activity, but also visualise the activity and how it relates to other network activity (both benign and malicious).**

Keywords-Autoencoder; Visualisation; k-NN; Deep Learning

I. INTRODUCTION

Computer networks are increasingly important to people's daily lives, and the rate of devices connecting to IP networks is increasing. The Cisco Visual Networking Index predicts that by 2021 there will be 3.5 networked devices per capita, up from 2.3 per capita in 2016 [1]. Most of this increase is coming from mobile devices and comes with a corresponding increase in the volume of data being used, with the amount of data in existence expected to increase to 44ZB by 2020 and 3.3ZB of data being transmitted across IP networks per year by 2021 [1].

Visualising this data presents a challenge for the network administrator, and visualising attacks presents an even bigger challenge. It is not enough to simply know an attack is happening, an administrator needs to know from where an attack is originating, what kind of attack it is, what its target is, and what other kind of systems may have been affected. The 2018 Cost of a Data Breach Study [2] found that companies that contained a breach within 30 days saved over $1 million compared to those that took over 30 days. By making data clearer, breaches can be contained faster, which can save companies money. The mean time to contain a breach was found to be 69 days.

Current systems allow the administrator to see an overview of a network and can include statistics and relevant details such as total network traffic, or even traffic over certain connections. However, these graphs typically lack a security view specifically, and are more focused on letting an administrator see which services and equipment may be under strain. Alternatively, security-focused services tend to provide anomaly detection and alert administrators to the presence of suspicious activity rather than providing clear visualisation combined with the normal network activity [3][4][7].

This paper will propose a method to visualise anomalous network activity. Anomalous activity will be detected using an Autoencoder network, which feeds into a k-NN (k-Nearest Neighbour) classifier. The results of this will be plotted onto a force-directed graph, which will highlight anomalous nodes clearly for the network administrator.

Deep learning can aid with this significantly. k-NN has been used in various methods of anomaly detection in the past [5], however accuracy has frequently been an issue as the noise of most network data means k-NN methods can fail to adequately classify data [6]. The Autoencoder deep network can aid with this by reducing the amount of noise in the data and increasing classification accuracy.

As such this paper shall propose a novel deep network to review network data and plot it in a form that allows a network administrator to see any anomalous activity, along with other relevant network details.

The rest of this paper is structured as follows. In Section 2, we will discuss other relevant work within machine learning and visualisation. In Section 3, we will discuss the methodology used within the experiment, including a brief description of unsupervised methods and why they are being used, as well as more detail about the Autoencoder and k-NN methods that will be used. Section 4 includes more detailed methodology and initial experimentation, including a detailed discussion of the structure of the model and rationale for any choices made, as well as initial results. Section 5 includes conclusions and future work.

II. RELATED WORK

Several tools already exist to allow administrators to view network activity, structure or alerts. However, many of these are unintuitive or omit important information. For instance, the OpenDaylight SDN (Software Defined Network) Controller [7] comes with a visualisation tool that allows the administrator to see a representation of all switches and nodes connected to the network, and how they are connected. It does not provide the administrator with any additional useful information, such as traffic volume over certain trunks, nor does it provide the administrator with any security alerts as no kind of Intrusion Detection System (IDS) is included. Alternatively, there are systems like Snort [8], which do not include any visualisation at all, simply alerting administrators that suspicious traffic has been detected. While its popularity speaks to its usefulness, it is purely text based, and sends alerts independent of other

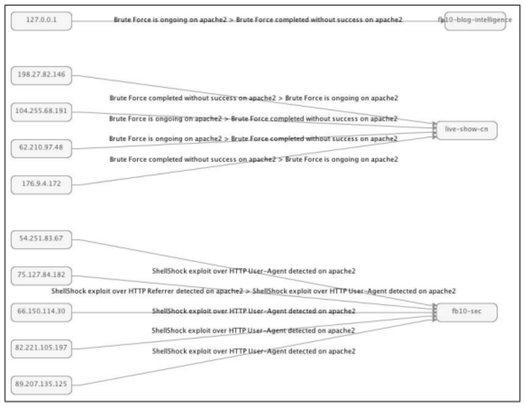
127.0.0.1
Brute Force is ongoing on apache2 > Brute Force completed without success on apache2
fb10-blog-intelligence
198.27.82.146
104.255.68.191
62.210.97.48
176.9.4.172
Brute Force completed without success on apache2 > Brute Force is ongoing on apache2
Brute Force is ongoing on apache2 > Brute Force completed without success on apache2
Brute Force is ongoing on apache2 > Brute Force completed without success on apache2
Brute Force completed without success on apache2 > Brute Force is ongoing on apache2
live-show-cn
54.251.83.67
75.127.84.182
66.150.114.30
82.221.105.197
89.207.135.125
ShellShock exploit over HTTP User-Agent detected on apache2
ShellShock exploit over HTTP Referrer detected on apache2 > ShellShock exploit over HTTP User-Agent detected on apache2
ShellShock exploit over HTTP User-Agent detected on apache2
ShellShock exploit over HTTP User-Agent detected on apache2
ShellShock exploit over HTTP User-Agent detected on apache2
fb10-sec

Figure 1. Line Graph model produced from SNORT output.

network conditions or traffic, making it easy for an administrator relying on it to miss other important information.

Researchers have tried to address this in several ways. For instance, in [9], the authors propose a system to visualise Snort logs by converting them into line graphs, showing the victims, attackers and types of attacks. This improves on the basic Snort logs, as it becomes easier to see which alerts are related (by host, or system) and which may be completely coincidental. However, it is dependent on the logs of Snort and therefore misses other potentially important information that may be needed (such as overall network load). In addition to this, on larger systems the graphs start to become harder to read as more and more data needs to be included within them, an example is shown in Figure 1.

Within [10] the authors use k-means clustering to group network data into groups (normal and anomalous), and then take the data from the anomalous cluster and use the same process to separate it into Transmission Control Protocol (TCP), User Datagram Protocol (UDP) and Internet Control Message Protocol (ICMP) traffic. After taking the cluster, which contains a mix of all three protocols they finally design a ruleset designed on this cluster and apply it to the testing set, which results in a detection rate of attacks above 80% for all five data types (normal, Denial of Service (DoS), Remote to Local (R2L), User to Root (U2R) and probe) except for U2R.

For unsupervised learning, the authors of [11] use Robust Autoencoders (RAE), which is an Autoencoder that splits the training data (X) into normal and outlier elements (L and S), such that $X = L+S$. The purpose of this is to avoid fitting anomalous or rare data, which should prevent underfitting of normal data, and help anomalous data be highlighted more easily. They find that RAEs are an effective way to reduce false positives and do have the benefit of not underfitting normal data. However, the approach was only used for port scan type attacks, and so may not scale as well when looking for other attack types.

Potluri, *et al.* [12] evaluate Stacked Autoencoders and Deep Belief Networks (DBN) as feature reducers, with Softmax Regression and Support Vector Machine classifiers. They found that the stacked Autoencoders achieve higher

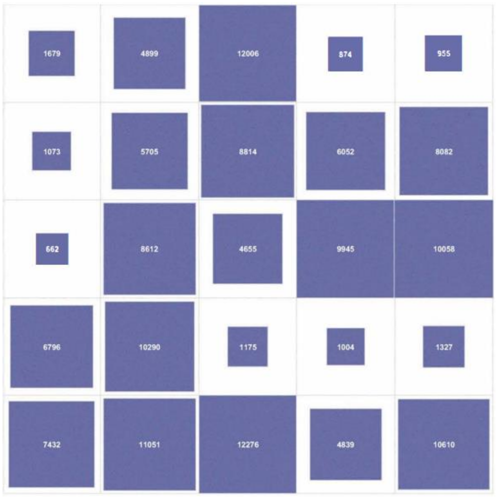

Figure 2. Input data hits for the 5x5 SOM.

levels of accuracy than the DBN when classifying fewer classes, whilst DBN and Soft-max Regression achieved higher accuracy with more classes.

Alom and Taha [13] look at attack detection using Auto Encoders and Restricted Boltzmann Machine (RBM) for feature extraction and dimensionality reduction, and combine it with k-means clustering for classification. They show that the combination can produce an accuracy of 92.12% with 9 features, or 90.86% with only 3 features. This compares to using k-means alone with 39 features for 87.72% accuracy or an Extreme Learning algorithm (again with 39 features) gaining an accuracy of 89.17%. This shows the potential of Autoencoders for categorising sets with extremely limited data sets, something that becomes more important in SDN environments with limited flow features.

Palomo *et al.* [14] decide to use a self-organising map to group and highlight network data. Using real network data captured from four subnets of a university network, they create a dataset from 150,871 samples, where 1 sample is a single packet. Each sample consisted of nine features, namely IP source address, IP destination address, protocol, source port, destination port, date, time, packet length and delta time. They find that self-organising maps can be an effective way to group similar network data, highlighting suspicious network data clusters. While the clusters do represent distinct network activity, the size of each node is determined by the amount of traffic that cluster exhibits. This means that the comparatively small amounts of anomalous network activity could be confused with other benign network protocols that generate low volumes of traffic if an administrator were to not examine the details more closely. For example, in Figure 2 nodes 18, 19 and 20 look very similar, however node 19 represents benign Address Resolution Protocol (ARP) traffic while nodes 18 and 20 represent suspicious activity from Russia and Italy.

III. PROPOSED METHODOLOGY

As we have seen, visualisation of network states is an ongoing research area, with many papers and projects proposing different ways to visualise the current state and data being transmitted. However, research into visualising network anomalies has not kept pace with this work. This paper intends to propose a method to both detect and visualise network anomalies, making it easier to see what kind of attacks network administrators are dealing with, thus allowing them to react quicker.

Our system proposes using an Autoencoder deep network to reduce the features of the SDN, and then using k-Nearest Neighbour to sort the resulting data. As has been shown by [15] and [16], reducing noise in data can improve accuracy for k-NN, and other shallow learning methods, and this is an important stage in our model.

The k-NN can then classify the reduced data into groups, allowing related data to be grouped together. This is then placed into a force directed graph, which will show the administrator the related flows in a clear and concise manner.

While other researchers have proposed similar unsupervised models to this, results are only ever given in a table or using a ROC curve. The method proposed allows the administrator to quickly see which flows are malicious, and which ones are benign.

A. *Unsupervised Methods*

The use of both Autoencoders and k-NN means that the system is unsupervised. Unsupervised machine learning has benefits in not needing labelled data to train the models. Labelled data within network environments can be difficult to access, and typically requires skilled administrators or other Network Intrusion Detection System (NIDS) to label the data appropriately. As such, many researchers have proposed that using unsupervised methods is more appropriate for intrusion detection. Unsupervised methods tend to have lower accuracy and more false positives than supervised methods, as shown by [17] and [18]. However, this is not necessarily always the case, as shown by [11] and [13], where the authors show an unsupervised method using recurrent Autoencoders can be effective when attempting to detect port scans and show that they can gain lower false positives.

B. *Autoencoders*

An Autoencoder is a neural network designed to learn the features of a set of data. Within an Autoencoder the desired output is the input itself. So for input I and output O, $I = O$. However, there are also one or more hidden layers that are smaller than the input, forcing the network to encode a representation of the input which can then be decoded into the output.

This forces the model to learn a representation of the input data that it can use to attempt to recreate the eventual output. The goal of this is to reduce noise or unneeded features in an automatic manner. This has shown to be effective within network security, as network data tends to include a lot of noise that is not relevant to classifying the data [11] [19]. Within this context, noise refers to data that is unimportant to the overall network wellbeing. Within larger or more complete datasets this is often low level network admin data (e.g., Dynamic Host Configuration Protocol (DHCP) joins and parts), but within higher levels this can persist with for example, benign retransmissions of data.

C. *k-Nearest Neighbor*

As noted, the purpose of the Autoencoders is to reduce the noise within the data, not to classify any data. To classify the data we will use a k-NN algorithm. This unsupervised algorithm classifies data based upon a plurality vote of its nearest neighbours as to which class it belongs in.

D. *Plotting the Graph*

The output of the model is a list of x, y coordinates for each neighbour, on each flow (so for 5 neighbours, each flow will have 5 sets of x, y coordinates). From here, each coordinate can have its results averaged (creating an average x, y coordinate for each flow) and these are added as nodes to the graph. The final step is to create the links. The same averages are run, however each time a node is created with the same x, y coordinate as another flow, a link is made between them. The result is a graph that joins similar flows together, while dissimilar malicious flows will be separate, making them easier to identify.

IV. Initial Experimentation

In this section, we will describe the experiments undertaken for this research, whilst detailing more about the proposed method and reasoning behind the choices made.

A. *The Dataset*

The dataset used for this research is the real network data from the University of Twente [20]. The dataset consists of connection monitoring for multiple SSH servers, which is organised into flows matching the IP Flow Information Export (IPFIX) [21] standard. The data is not labelled, however, with the use of the unsupervised learning technique proposed this is not a problem. The dataset consists of network flows recorded on four routers and includes Date first seen, Duration, Protocol, Source IP Address and Port, Destination IP Address and Port, Number of Packets, Bytes and Number of flows. Also included is the logs from the SSH servers, which allows us to create a dataset of mixed log and flow data. Pre-processing was performed in order to convert text data (protocol, date) into a numerical form. Due to the size of the dataset, a 5% subset was used. This was split into training and testing subsets, with approximately 66.66% training to 33.33% testing, in order to make twice as much training as testing data. Simple random sampling was used to select the 5% of the dataset to be used, as well as to determine whether the record would be part of the training or testing datasets, and this was accomplished using the Python random function. The dataset consists of primarily benign data, however slightly less than 2% are malicious flows that include brute force or dictionary attacks. The dataset is therefore not evenly distributed, but is a fair representation of what other SSH server network traffic may look like.

B. The Model

As stated, we took 66.66% split of the dataset and used it to train the Autoencoder part of the network, whilst leaving the remaining data for testing. The output from the Autoencoder is fed to the k-NN algorithm. Again, k-NN classifies the data it receives based upon a plurality vote of its nearest neighbours. A representation of the model is shown in Figure 3.

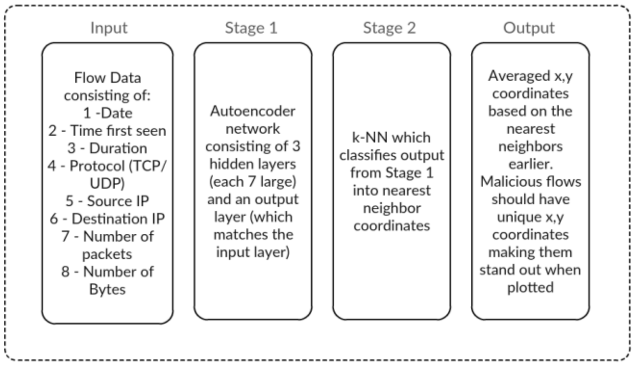
Input
Stage 1
Stage 2
Output
Flow Data consisting of: 1 -Date 2 - Time first seen 3 - Duration 4 - Protocol (TCP/UDP) 5 - Source IP 6 - Destination IP 7 - Number of packets 8 - Number of Bytes
Autoencoder network consisting of 3 hidden layers (each 7 large) and an output layer (which matches the input layer)
k-NN which classifies output from Stage 1 into nearest neighbor coordinates
Averaged x,y coordinates based on the nearest neighbors earlier. Malicious flows should have unique x,y coordinates making them stand out when plotted

Figure 3. A representation of the model being used.

1) The Autoencoder

The Autoencoder network uses a size 8-7-7-7-8 in order to reduce noise within the data, where the sizes 8 are the input and output respectively, and the 7-7-7 is the middle hidden layers. This is based off similar model shapes in [15], who use a similar structure, in addition to our own testing where we found adding more layers leads to overfitting on the majority class. This structure should allow the model to remove noise without losing valuable data. Additionally, 500 epochs were chosen to train, along with a batch size of 200 and a learning rate of 0.01. These values were based off [11] and [22] who choose very similar values for their models, however as these works were not using the same datasets further optimisation could be done. With additional testing on the dataset itself, a batch size of 200 and 500 epochs showed no signs of overfitting, so these values were chosen as final values.

2) k-NN

After training within the Autoencoder network the output was given to the k-NN algorithm to train and classify. The dataset consists of both anomalous “brute force” access attempts and regular access attempts (with occasional legitimate access attempts rejected due to admin errors). As such, the k-NN model should produce two primary clusters, one of legitimate access attempts and one (significantly smaller) of illegitimate access attempts. The model was set to give the five closest neighbours to the input. A larger value could have been set; however, this would increase the amount of time it takes to process the model, and would increase the complexity of the final graph. There is a possibility that having too many neighbours would produce a graph that does not show the different outputs as the similarities of the benign and malicious data would work to pull them together.

3) Tools Used

GPU-based Tensorflow running on an NVIDIA RTX2080 Ti 11GB GPU was used to construct the Autoencoder, with the results of the Autoencoder going to train the k-NN. The k-NN was coded using sklearn, and processed on an i9-9900 CPU. The output of the k-NN was a text file with the node and nearest neighbors to it. This is converted into a JSON file and then imported into the NVD3 generated graph.

Finally, some areas of the model have not been fully optimised and further accuracy could be gained as a result of further optimisation. In particular, the number of epochs and learning rate were chosen based upon common values in other works, which used different datasets. Some testing was done to ensure these values were still relevant, but could still be optimised further. The number of epochs could likely be increased above 500, without signs of overfitting, but this will come with a corresponding increase in the amount of time to train and test the model. As has been noted in Sections 1 and 2, time is an important aspect in intrusion detection, and the quicker intrusions can be detected and contained, the more money can be saved.

C. Results

In Figure 4 we can see the result of the graph that came from the k-NN without the use of the Autoencoder network. As can be seen, anomalous results are not as easy to identify, they are separate nodes, however due to the loose clustering of benign flows, and it could be easy to miss malicious flows. Figure 5 shows the results from the model using the Autoencoder network to reduce the noise. The malicious flows are more notable due to the aggressive clustering of benign flows, and a busy administrator would be able to note them and gain useful and additional information from the suspicious flows. In Figure 5 benign nodes have clearly joined up, while the malicious nodes are separate, while in Figure 4 the benign nodes are not as joined up, making them harder to identify at a glance. Shown is the result for flows found within a 2 hour period on 1st Feb, as the NVD3 process used had trouble managing more flows than this.

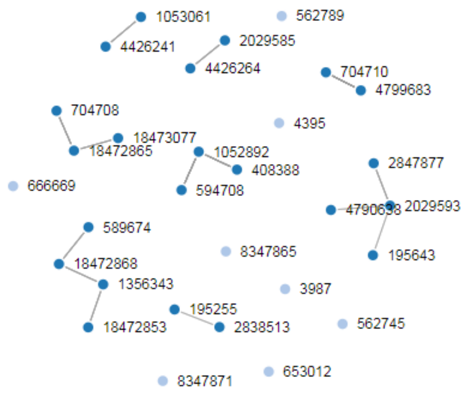

Figure 4. Shallow learning graph.

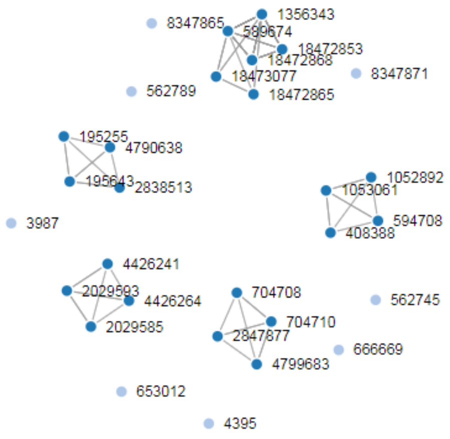

Figure 5. Deep learning graph.

V. Conclusions and Further Work

In this Section, we will provide further discussion on the results, as well as outlining the future direction of our work.

A. *Conclusions*

As Figure 5 shows, Autoencoder networks are an effective method to reduce noise in network data, and highlight anomalies that can then be detected by a shallow learning algorithm, in this case k-NN. We have shown that by mapping the output of the k-NN onto a force directed graph, we can more easily visualise the malicious flows, and how they interact with other flows. Additionally, this graph allows mouse rollover to give more information about the flow, allowing the administrator to quickly identify malicious flows, flows that are related to it, and gain additional information, such as IP address of the source and destination, ports and volume of data. Force graphs are clearly an effective method of conveying this information to the administrator in a clear and concise manner, which if implemented in a production environment could reduce the risk of administrator error and speed up response time. Figure 4 shows the equivalent shallow learning graph.

B. *Further Research*

Other types of unsupervised learning could be used for classification and noise reduction. We chose k-NN as the classifier due to its speed and ability to gain accurate results. However, k-means could offer similar benefits and results, and has been shown to be effective when paired with models that reduce the dimensionality of the data it is classifying.

Unsupervised versions of DBN could also be effective. When being used to classify data in a supervised manner, DBNs have been shown to produce highly accurate results on common datasets, but DBNs do not have to be trained in a supervised manner. Using a RBM network instead of Autoencoder might result in higher accuracy, although again the amount of time to train and test the network would need to be considered.

This paper has focused on unsupervised methods, and it is the authors' belief that unsupervised machine learning is preferable over supervised methods, simply because of the difficulty in obtaining fully labelled datasets for production environments. Obtaining labelled data typically requires highly skilled administrators to manually review the training data, and label accordingly. This is a time consuming and, given the expertise required, often expensive process, that is still prone to error. However, often supervised methods do provide higher accuracy, and in a world where some breaches are detected over a year from the initial attack (and, it must be assumed, some are never detected) it could be argued that the extra cost of supervised methods is worth the extra accuracy. With this in mind, methods such as Support Vector Machines or Softmax classifiers could be considered, especially Softmax where the output probabilities could be mapped directly onto the force directed graph.

REFERENCES

[1] Cisco, “Cisco Visual Networking Index : Forecast and Methodology , 2016 – 2021 Investor relations Search jobs,” p. 2021, 2017.

[2] P. Institute, “2018 Cost of a Data Breach Study: Global Interview,” no. July, p. 15, 2018

[3] “SIEM | LogRhythm.” [Online]. Available: https://logrhythm.com/products/siem/. [Accessed: 30-May-2019]

[4] W. C. Lin, S. W. Ke, and C. F. Tsai, “CANN: An intrusion detection system based on combining cluster centers and nearest neighbors,” Knowledge-Based Syst., vol. 78, no. 1, 2015

[5] B. Xu, S. Chen, H. Zhang, and T. Wu, “Incremental k-NN SVM method in intrusion detection,” Proc. IEEE Int. Conf. Softw. Eng. Serv. Sci. ICSESS, vol. 2017-Novem, pp. 712–717, 2018

[6] T. O. Project, “OpenDaylight,” 2019. [Online]. Available: https://www.opendaylight.org/. [Accessed: 30-May-2019].

[7] Cisco, “Snort.” [Online]. Available: https://www.snort.org/. [Accessed: 30-May-2019].

[8] Cisco, “Snort.” [Online]. Available: https://www.snort.org/. [Accessed: 30-May-2019]

[9] A. Azodi, F. Cheng, and C. Meinel, “Towards better attack path visualizations based on deep normalization of host/network IDS alerts,” Proc. - Int. Conf. Adv. Inf. Netw. Appl. AINA, vol. 2016–May, pp. 1064–1071, 2016.

[10] B. Langthasa, B. Acharya, and S. Sarmah, “Classification of network traffic in LAN,” 2015 Int. Conf. Electron. Des. Comput. Networks Autom. Verif., pp. 92–99, 2015.

[11] G. Kotani and Y. Sekiya, “Unsupervised Scanning Behavior Detection Based on Distribution of Network Traffic Features Using Robust Autoencoders,” 2018 IEEE Int. Conf. Data Min. Work., pp. 35–38, 2018.

[12] S. Potluri, N. F. Henry, and C. Diedrich, “Evaluation of hybrid deep learning techniques for ensuring security in networked control systems,” 2017 22nd IEEE Int. Conf. Emerg. Technol. Fact. Autom., pp. 1–8, 2017.

[13] M. Z. Alom and T. M. Taha, “Network Intrusion Detection for Cyber Security using Unsupervised Deep Learning Approaches,” in 2017 IEEE National Aerospace and Electronics Conference (NAECON), 2017, vol. 2017–June, pp. 2379–2027.

[14] E. J. Palomo, J. North, D. Elizondo, R. M. Luque, and T. Watson, “Visualisation of network forensics traffic data with a self-organising map for qualitative features,” Proc. Int. Jt. Conf. Neural Networks, pp. 1740–1747, 2011.

[15] P. Vincent and H. Larochelle, “Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion Pierre-Antoine Manzagol,” J. Mach. Learn. Res., vol. 11, pp. 3371–3408, 2010.

[16] G. E. Hinton and R. R. Salakhutdinov, “Reducing the Dimensionality of Data with Neural Networks,” vol. 313, no. July, pp. 504–508, 2006.

[17] P. Laskov, P. Dussel, C. Schafer, and K. Rieck, Learning intrusion detection: Supervised or unsupervised?, vol. 3617 of Le. Springer Berlin / Heidelberg, 2005.

[18] R. C. Staudemeyer, “Applying long short-term memory recurrent neural networks to intrusion detection,” Sacj, no. 56, pp. 136–154, 2015.

[19] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie, “High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning,” Pattern Recognit., vol. 58, 2016.

[20] R. Hofstede, L. Hendriks, A. Sperotto, and A. Pras, “SSH Compromise Detection using NetFlow/IPFIX,” ACM SIGCOMM Comput. Commun. Rev., vol. 44, no. 5, pp. 20–26, 2014.

[21] R. Hofstede *et al*., “Flow monitoring explained: From packet capture to data analysis with NetFlow and IPFIX,” IEEE Commun. Surv. Tutorials, vol. 16, no. 4, pp. 2037–2064, 2014

[22] Y. Chuan-long, Z. Yue-fei, F. Jin-long, and H. Xin-zheng, “A Deep Learning Approach for Intrusion Detection using Recurrent Neural Networks,” IEEE Access, vol. 3536, no. c, pp. 1–1, 2017.