Evaluating Deep Autoencoder Neural Networks For Detecting Anomalous Lateral Movement In Computer Networks

Alex Aubrey
College of Computing and Technology, Lipscomb University
Nashville, TN, USA
aaubrey91@gmail.com

Stoney DeVille
College of Computing and Technology, Lipscomb University
Nashville, TN, USA
stoneydeville@gmail.com

Will Haight
College of Computing and Technology, Lipscomb University
Nashville, TN, USA
wthaightii@gmail.com

Ronald Holt
Pittsburgh, PA, USA
ronaldsholt@gmail.com

Todd Gary
College of Computing and Technology, Lipscomb University
Nashville, TN, USA
tgary@lipscomb.edu

Quingguo Wang
College of Computing and Technology, Lipscomb University
Nashville, TN, USA
qwang@lipscomb.edu

ABSTRACT

Cyber crimes are projected to cause damages of 2 trillion dollars annually worldwide by 2019 and will be over 6 trillion dollars by 2021. The purpose of this research is to use an autoencoding neural network to detect lateral movement, a common cyber attack technique involving network access from machine to machine. We construct features based on probabilities inherent to the data and use these to train the models. The several autoencoder architectures we evaluated on authentication events collected from the Los Alamos National Labs (LANL) dataset achieved a low false positive rate (fallout) of 0.0002 to 0.0004 and a robust true positive rate (recall) of 0.93. These results support the efficacy of autoencoders in anomaly detection.

KEYWORDS

Autoencoder, cybersecurity, lateral movement, unsupervised machine learning, authentication.

1 INTRODUCTION

Cyber crimes are projected to cause damages of 2 trillion dollars annually worldwide by 2019 [1] **and will be over 6 trillion dollars by 2021** [2, 3]. Organizations cannot only rely on inbound prevention tools for detecting and stopping cyber threats. Network defenders must go beyond just perimeter defenses in order to detect advanced adversaries. Lateral movement occurs on the inside of the organization's network subsequent to an attacker having already compromised a machine. This is an important step to detect in the cyber kill-chain because detecting and responding to lateral movement before an attacker reaches the objective can minimize the amount of damage incurred within the network [4].

The Lockheed Martin ***Cyber Kill Chain*** is an industry description of how nearly all sophisticated cyber-attacks are executed in enterprise networks. It is comprised of seven steps that express how attackers operate during their missions. First, the attacker performs ***reconnaissance*** on a target. The attacker must gather details about the target organization in order to know what or which individual to initially target. This may consist of gathering different email addresses found on social media sites or public data dumps available on the dark web. Secondly, the attacker must make their ***weapon*** to best suit their needs relative to this particular target based on findings during the reconnaissance phase. Thirdly, the attacker must ***deliver*** this weapon to the intended target. Continuing with the findings from the reconnaissance phase, the attacker may send a crafted email to a specific employee at the target organization [5]. Now comes the fourth stage, ***exploitation***. The victim clicks on the malicious email attachment or link and the attacker now exploits a vulnerability on a target computer in order to establish persistence and escalate privileges. The fifth stage is ***installing*** the weapon developed in stage two. Once this weapon is installed on the victim host, the attacker must establish communication with the compromised host (stage six) in order to ***send commands*** to perform the last stage entitled ***actions on objective*** [6].

The attacker has now established an entryway into the organization enabling execution on the motive behind targeting this particular enterprise. For example, if the motive behind the attack was to steal credit card numbers, the attacker may need to compromise a point-of-sale (POS) system. However, the initial point-of-compromise (the victim that clicked on the phishing email) is not the POS system. The attacker must now navigate ***laterally*** by moving from host to host in the environment until the POS systems are reached in order to plant the malware. This is called ***lateral movement***. According to MITRE, "Lateral movement consists of techniques that enable an adversary to access and control remote systems on a network and could, but does not necessarily, include execution of tools on remote systems" [7]. In this example, lateral movement would consist of the techniques used by the attacker to navigate from the initial point of compromise to the objective.

Using the Target™ breach as an example from 2013, attackers first gained access to a third party who had access to Target's business sub-network [8]. The attackers then, while performing lateral movement, were eventually able to access the point of sale (POS) terminals and plant malware leading to the breach. If lateral movement had been detected before the attackers gained access to the POS systems, the result of the compromise could have been different. Our intention for this research is to develop a new technique to more accurately detect lateral movement.

Various approaches have been used to detect lateral movement. Example log sources include Windows Event Logs, NetFlow data, or

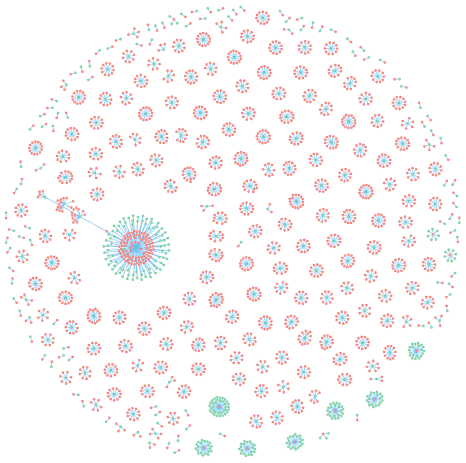


Figure 1: A network graph example featuring a subset of LANL source, user and host to destination, user and host.

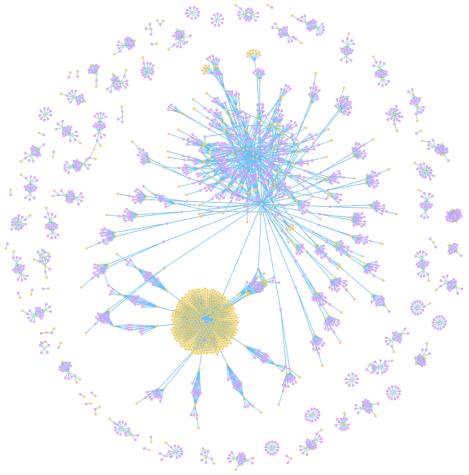


Figure 2: A network graph example featuring a subset of LANL source and destination user to source and destination host.

authentication data [9]. Traditional and non-traditional signature-based approaches seek to address this issue [10]. There has also been other research in cybersecurity for detecting lateral movement using graph models [4]. Autonomous machine learning techniques are needed to efficiently detect anomalous activity. This paper takes a new approach to improve false positive rate, true positive rate, and to decrease detection time by using an autonomous autoencoder for anomaly detection rather than manual detection.

2 MATERIALS AND METHODS

2.1 Data Collection

The LANL dataset is 73 gigabytes and contains 58 consecutive days of de-identified event data collected from five sources within the LANL network. It represents 1,648,275,307 events in total for 12,425 users, 17,684 computers, and 62,974 processes [11]. There is a distinguished subset of *redteam* data which represents lateral movement. This forms the core of a larger dataset used to validate the autoencoders.

This dataset represents authentication events between computers on a network. Each event consists of a source computer, source user, destination computer, destination user and a timestamp.

2.2 Data Preparation and Processing

To preserve the included lateral movement, we constructed test data sets by including first the red team data. To this we added all rows from the full dataset which mentioned any computer included with the red team data. Beyond that, we included 160 random source computers, as well as all computers which connected with those according to the data. This was our **developmental data set (DDS)**. We also constructed a **larger dataset (LD)** by including all rows from the full data set mentioning any computer or user included with the red team data, as well as all users associated with the 160 randomly chosen computers mentioned above.

Features of interest include time, source user, destination user, source computer, and destination computer. An `is_malicious` feature indicates malicious red team data. The raw dataset used for this research is structured as in **Table 1**. **Figure 1** shows a graph of network connections in which a source user-source computer (green node) connects to a destination user-destination computer (red node) along a blue edge. **Figure 2** is another graph depicting the relationship between users and computers: a source user-destination user (yellow node) connects to a source computer-destination computer (pink node) along a blue edge.

2.3 Probability Features

The first features created were based on the frequency of a given condition. Four features were created using this method.

We adopt the convention that if $\mathcal{S}$ is a (finite) set, then $n(\mathcal{S})$ denotes the cardinality of $\mathcal{S}$, i.e., the count of distinct elements of $\mathcal{S}$. Let X denote the original set of data as indicated in **Table 1**, i.e. a two-dimensional array with columns `timestamp`, `src_user`, `dest_user`, `src_comp`, `dest_comp`, and `is_malicious`. For our purposes, it is necessary to think of X as a set whose elements are the rows of the table; so, using the cardinality notation, $n(X)$ indicates the number of rows in our table. Let $\mathcal{U}$ and C denote the sets of all users and computers listed in our network data. Define distinguished subsets $\mathcal{U}_{\mathcal{S}}$ and $\mathcal{U}_{\mathcal{D}}$ to be the set of all source users and destination users in our data; likewise let $C_{\mathcal{S}}$ and $C_{\mathcal{D}}$ denote the sets of all source computers and destination computers.

Now, for a particular user a in $\mathcal{U}$, let XU_a denote the subset of X where a is listed as the source user, and let XU^a denote the subset of X where a is listed as the destination user (think of these as a selection of particular rows from X). Similarly, for a particular computer k in C, let XC_k denote the subset of X where k is listed as the source computer, and let XC^k denote the subset of X where k is listed as the destination computer.

Assuming that $a \in \mathcal{U}_{\mathcal{S}}$, $b \in \mathcal{U}_{\mathcal{D}}$, $k \in C_{\mathcal{S}}$, and $j \in C_{\mathcal{D}}$, we calculate the following conditional probabilities:

	time	src_user	dest_user	src_comp	dest_comp	is_malicious
0	1	U6@DOM1	U6@DOM1	C606	C1065	0
1	1	U6@DOM1	U6@DOM1	C606	C529	0
2	1	U6@DOM1	U6@DOM1	C606	C529	0
3	1	U14@DOM2	U6@DOM1	C616	C29	1

Table 1: Example of LANL source data.

(1) The conditional probability that b is the destination user, given that a is the source user

$$\mathcal{P}(XU^b|XU_a) = \frac{n(XU^b \cap XU_a)}{n(XU_a)},$$

(2) The conditional probability that j is the destination computer, given that k is the source computer

$$\mathcal{P}(XC^j|XC_k) = \frac{n(XC^j \cap XC_k)}{n(XC_k)},$$

(3) The conditional probability that k is the source computer, given that a is the source user

$$\mathcal{P}(XC_k|XU_a) = \frac{n(XC_k \cap XU_a)}{n(XU_a)}, \text{ and}$$

(4) The conditional probability that j is the destination computer, given that b is the destination user.

$$\mathcal{P}(XC^j|XU^b) = \frac{n(XC^j \cap XU^b)}{n(XU^b)}.$$

These four conditional probabilities constitute engineered features used as input to the models as indicated in **Table 2**.

2.4 Hourly Metrics

We partitioned our data into one hour bins for capturing sequences of events changing through time. For each row of the table, find the quotient of the timestamp and 3600 (count of seconds in one-hour) and apply a ceiling function to round up to the next whole number. In **Table 1**, the time column indicates the index of the hour in which the event took place.

We create the first of two features by counting how many distinct destination hosts to which a given source host has authenticated, and then grouped by each hour bin. We then do the same for users by computing how many destination users each source has use to authenticate, and then also grouped by each hour bin. After this process is completed, the final data is structured as indicated in **Table 2**.

3 METHODS

3.1 Autoencoders

Autoencoders (AEs) have been proven to perform well in different types of anomaly detection problems. AE architecture and function is based on a classical unsupervised model called the encoder-decoder function [12–14]. However, AEs can be considered to be semi-supervised learning due to the involvement of reconstruction error scoring [13].

The basic anatomy of an AE has an encoder function f and a decoder function g. If X represents input data, $Y = f(X)$ is the encoded data, and $Z = g(Y)$ is the decoded data. A well trained autoencoder should closely approximate the identity function on the input data, i.e., the **reconstruction** $g(f(X))$ should be nearly the same as the original input data X. The encoder will learn important features based on the input, but the reconstructed data will not perfectly match the input. This helps the autoencoder to learn generalized mappings of the original features [15–17].

We developed four autoencoder models to evaluate in our research. The first model was a *vanilla* autoencoder (V-AE) consisting of only three layers and configured as 6-2-6; the second model (D-AE) was a deeper version consisting of five layers 6-3-2-3-6; the third model featured an asymmetric (or “staggered”) encoder-decoder structure (S-DAE), layered 6-3-2-3-4-5-6; finally, the fourth model used an architecture which was the reverse of the third, i.e., layered 6-5-4-3-2-3-6 (RS-DAE).

We used Keras with TensorFlow to build and optimize our AE models utilizing Python. We utilized a hybrid architecture of hyperbolic tangent (tanh) activation layers and rectified linear unit (ReLU) activation layers [18]. The first layer was regularized with an ℓ^1 penalty of 10^{-5} for each model.

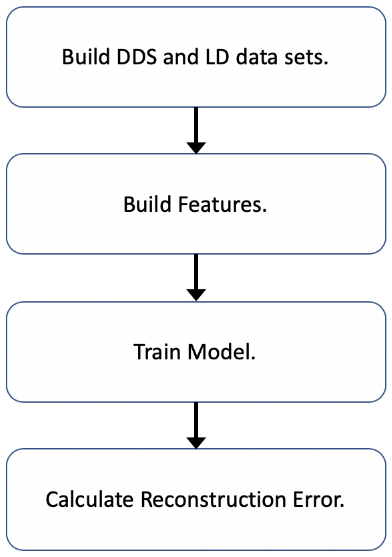


Figure 3: Data flow pipeline.

$\mathcal{P}$(dest_user| src_user)	$\mathcal{P}$(dest_comp| src_comp)	$\mathcal{P}$(src_comp| src_user)	$\mathcal{P}$(dest_comp| dest_user)	dc_dest_comp _by_src_comp	dc_dest_comp _by_src_user	is_malicious
0.96	0.0027	1.0	0.029	6	3	0
0.96	0.0336	1.0	0.171	6	3	0
0.96	0.0336	1.0	0.171	6	3	0
0.96	0.0060	1.0	0.043	6	3	0
0.96	0.0016	1.0	0.029	6	3	0

Table 2: Calculated features as input data for the autoencoder.

Model	FPR	Precision	Recall
V-AE	0.0055	0.02	0.79
D-AE	0.0085	0.03	0.81
S-DAE	0.0095	0.03	0.81
RS-DAE	0.2038	0.00	0.00

Table 3: The preliminary analysis of the sub-sampled data set.

3.2 Technology Stack for Research

3.3 Model Preprocessing and Training

We scaled our features with the `MinMaxScaler` function in Scikit-Learn [16, 18, 19]. Our training and test data were split 90 / 10 using a random seed with the `Preprocessing` module from Scikit-Learn. Each model fit the training data over 5 epochs with a batch size of 1024. Our model was compiled with the Adam optimizer and we used mean squared error (MSE) to calculate the loss [17]. We assessed our training by plotting the minimization of our objective function during training vs. epochs [16, 18] .

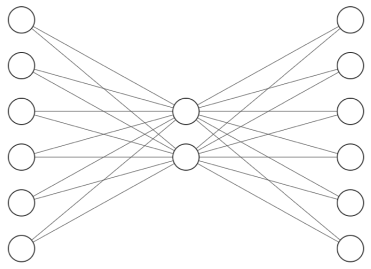
3.4 Reconstruction Error

We used mean squared error (MSE) to calculate the error. To do this, after training, we ran the `predict` function from our fitted models on our test data. The predictions were compared back to the test data as by our MSE scoring formula. In the case of our design, a high reconstruction error would indicate and be treated as an anomaly. To evaluate the reconstruction errors, we defined a manual threshold so that if any reconstructed value exceeded the set threshold it would be labeled as an anomaly. This threshold was manually optimized and set for all models. Subsequently, we used our binary labeled array to calculate a confusion matrix as well as our other key metrics of interest. We also visually plotted out each MSE vs. data point index to visually assess the scoring. We repeated this processes throughout cross validation.

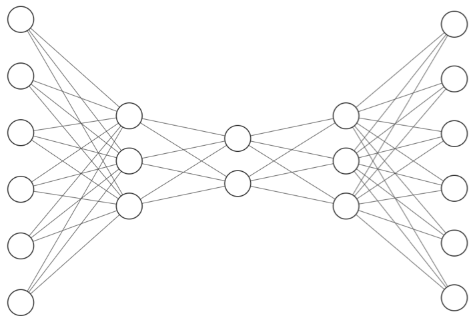
4 RESULTS

From the analysis, we conducted a stratified 10-fold cross-validation for the models on a developmental data set using `model selection` from Scikit-Learn. Based on that, we assembled an aggregate confusion matrix, and analyzed several metrics including the recall, false positive rate, precision and accuracy. We calculated the above metrics with the `classification report` and `confusion matrix` functions of Scikit-Learn [16].

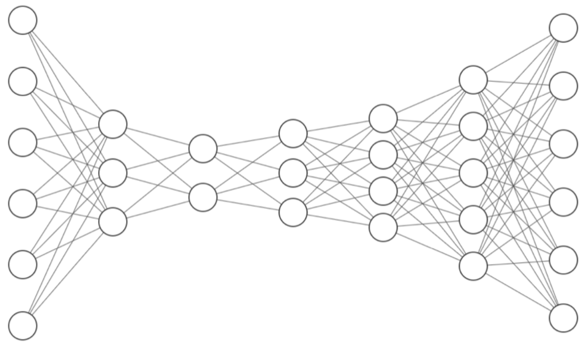
In the preliminary results, listed in **Table 3**, all AE models performed well except for RS-DAE, which had a high false positive rate.



(a) V-AE (6-2-6).



(b) D-AE (6-3-2-3-6).



(c) S-DAE (6-3-2-3-4-5-6).

Figure 4: The three autoencoders we tested.

Although the accuracy of RS-DAE was 0.99, the recall, precision, and fall-out were zero. Consequently, we rejected this model for its poor performance on the developmental data. The architecture of the remaining models can be seen in **Figure 4**. However, the V-AE, D-AE, and S-DAE performed with high accuracy and a false positive rate of 0.0055, 0.0085, and 0.0096 respectively. The respective recall values were 0.79, 0.81, and 0.81.

We moved on to evaluate our models on the larger dataset in which the same metrics were calculated under the same cross validation design. We found that our models improved in false positive rate and recall, but the change in precision remained negligible. The V-AE achieved a false positive rate of 0.0004 and a recall of 0.92. The D-AE and the S-DAE obtained a false positive rate of 0.0002. However, the S-DAE recall was slightly higher at 0.94 as compared to 0.93 from the D-AE model. When we used larger data set LD we saw that our autoencoders' false positive rate and recall not only maintained performance, but increased overall. These results are depicted in **Figure 5**. Although our precision was low, based on the other metrics, we considered our results to meet or exceed the performance of other autonomous detection methods conducted on the LANL dataset [20, 21].

5 RELATED WORK

Unsupervised machine learning has been used to detect insider threats using deep and recurrent neural networks [22]. Machine learning has also been used to detect attacks on web applications [23].

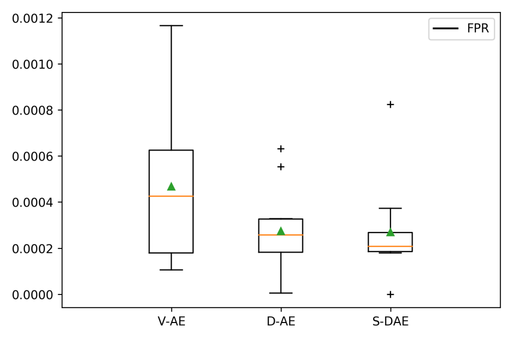
Other research has been conducted on the LANL dataset for anomaly detection protocols. A rule based visualization pattern discovery tool, APT-Hunter, conducted an experiment with a false positive rate of 0.00005 (0.005%). Their model performed better than our AE models. However, this platform required an analyst to manually look through patterns to identify potential threats, and was not automated to detect malicious events [24]. Another statistical approach devised by Heard et al. was used on the LANL dataset to determine anomalies based on source computers only [25]. More recently Sanders et al. [21] provided a methodology using unsupervised ensemble approaches. Sanders et al. reached an overall average true positive rate of detection of 0.89 with varying ranges of false positive rate performance. We have seen other unsupervised approaches in Chapter 9 of *Data Science for Cyber-Security* relative to classifying red team events based on various spectral and normalized Laplacian embedding techniques on the LANL dataset [20].

Overall, to the best of our knowledge, we have not seen published identical approaches in which an AE was used specifically for detecting malicious events in the LANL dataset.

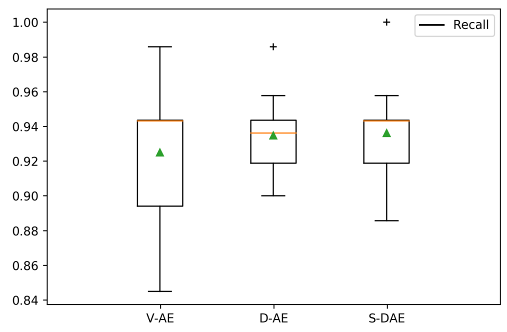
6 LIMITATIONS AND FUTURE WORK

Several determining factors restricted our team in this study.

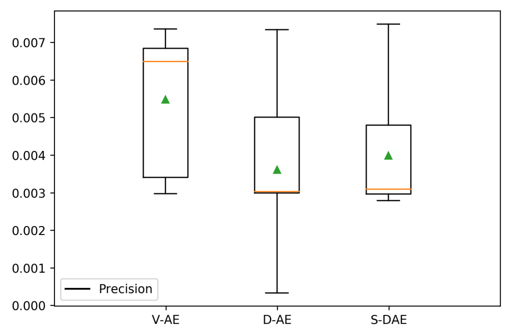
(1) *Resources*: Our challenges to manage, manipulate, and train our model on the original LANL dataset was limited by our computational resources. We did utilize GPUs for the bulk of this work, however, testing preliminary ideas and experimentation on-the-fly was only feasible with smaller datasets. Having more dedicated computing power would have greatly helped our experimentation.
(2) *Data Selection*: Our dataset was formulated to encompass data that would mimic a theoretical anomaly detection schema. We used all red team events as our base data and then selected all source and destination communications from within this subset from the overall dataset. To allow for noise that would be seen in a real-world system, we added additional selection of non-red team source hosts which may cause some underlying bias in our model; however, we believe that fundamentally our technique is validated. Re-evaluating the results on the full data set as well as exploring real-time solutions with this model would be an improvement.
(3) *Algorithm Design*: Choosing only a few models to evaluate has limited us in our protocol. In future work, we want to explore deeper and more complex architecture as a solution.



(a) False positive rates.



(b) True positive rates.



(c) Calculated precision.

Figure 5: Performance results for our three models.

We also believe that ensemble methods would be valuable to evaluate against our models.

7 DISCUSSION

Anomaly detection for lateral movement is not new to the industry. Multiple methods, traditional and non-traditional, have been available with varying degrees of application. Some of the traditional signature-based tools, such as Snort, yield too many false positives, and so would flood analysts with alert messages such as '`ET POLICY PSexec service created`', when system administrators perform these actions regularly [10]. An analyst would exempt these system administrators' hosts from this specific detection leading to a potential blind spot for attackers to utilize. Outside of traditional signature-based approaches addressing this issue, there have been other research in cybersecurity for detecting lateral movement using graphs. For instance, Sqrrl suggests detecting this traffic by learning common login patterns for users and ranking every login according to its anomalous nature[26]. In addition, there have been successful approaches using game theory and solving for saddle point strategies enabling the researchers to successfully detect the lateral movement and respond immediately by blocking the attacker [4].

Autonomous cyber breach detection holds the promise of utility in a real-time system. Network administrators would find it advantageous to be alerted of a cyber attack while it is underway as opposed to hours, days or months later. Moreover, the paucity of verified network data containing known cyber attacks suitable for training supervised algorithms leaves autonomous systems as a favorable option. Such an approach applies to the more general problem of detecting unanticipated cyber attacker techniqes than traditional tools which seek to detect a particular kind of attack.

Autoencoders presently enjoy widespread application in the general area of anomaly detection. They provide a robust method for learning generalized associations inside a dataset, in an unsupervised format[12, 13]. We built and evaluated multiple autoencoders and tested them on the LANL dataset to determine their effectiveness in detecting anomalous network activity. We found that suitably structured autoencoders generally performed well on network authentication data. We also discovered that increasing the dimension of the decoder improved several measures of the autoencoder's efficacy. Additional research is needed.

8 CONCLUSION

Utilizing autoencoders for anomaly detection in authentication data has the potential to improve and automate parts of cybersecurity. We have demonstrated an autoencoder application, using a real-world use case for detecting anomalies, enabling increased productivity, preventing loss, and serving as a mechanism to detect lateral movement and other malicious events prematurely.

9 ACKNOWLEDGEMENTS

We would like to express our gratitude to Professor Todd Gary of Lipscomb University for suggesting this direction for our research. Thanks also to prof. Qingguo Wang for his advice to us for this research.

REFERENCES

[1] *Cyber Crime to Reach $2 Trillion By 2019: What Can We Do?*, Jana Rooheart, https://www.business.com/articles/cyber-crime-to-reach-2-trillion-by-2019-what-can-we-do/, Feb. 22, 2017.

[2] Steve Morgan, editor-in-chief, *Cybercrimes Magazine* https://cybersecurityventures.com/hackerpocalypse-cybercrime-report-2016/, Oct. 16, 2017.

[3] Lewie Dunsworth, L., and Kapadia, K., The Herjavec Group, *The 2019 Healthcare Cybersecurity Report* https://www.herjavecgroup.com/wp-content/uploads/2018/11/Herjavec-Group-2019-Healthcare-Cybersecurity-Report.pdf 2018.

[4] Fawaz, A., Bohara, A., Cheh, C., & Sanders, W. H. (n.d.), *Lateral Movement Detection Using Distributed Data Fusion*, Retrieved from, https://www.perform.illinois.edu/Papers/USAN_papers/16FAW02.pdf.

[5] Soria-Machado, M., Abolins, D., Boldea, C., & Socha, K., *Detecting Lateral Movements in Windows Infrastructure*, Retrieved from http://cert.europa.eu/static/WhitePapers/ `CERT-EU_SWP_17-002_Lateral_Movements.pdf` 2017, February 27.

[6] L. M. (n.d.), *The Cyber Kill Chain*, Retrieved from https://www.lockheedmartin.com/us/what-we-do/aerospace-defense/cyber/cyber-kill-chain.html.

[7] M. (n.d.), *Lateral Movement*, Retrieved from https://attack.mitre.org/wiki/Lateral_Movement.

[8] Shu, X., Tian, K., Ciambrone, A., & Yao, D., *Breaking the Target: An Analysis of Target Data Breach and Lessons Learned*, Retrieved from https://arxiv.org/pdf/1701.04940.pdf, 2017.

[9] Abe, S., *Detecting Lateral Movement in APTs–Analysis Approach on Windows Event Logs*, https://www.first.org/resources/papers/conf2016/FIRST-2016-105.pdf, June 17, 2016.

[10] Cyrus, R., *Detecting Malicious SMB Activity Using Bro. Retrieved from* https://www.sans.org/reading-room/whitepapers/detection/detecting-malicious-smb-activity-bro-37472, December 13, 2016.

[11] Kent, A.D., *Comprehensive, Multi-Source Cybersecurity Events*, Los Alamos National Laboratory, Retrieved from http://dx.doi.org/10.17021/1179829, 2015.

[12] M. Sakurada, T. Yairi, *Anomaly Detection Using Autoencoders with Non-linear Dimensionality Reduction*, ACM Press, 2014, pp. 4–11, http://dx.doi.org/10.1145/2689746.2689747.

[13] Jinwon An and Sungzoon Cho., *Variational Autoencoder based Anomaly Detection using Reconstruction Probability (Technical Report)*, pp.1–18, SNU Data Mining Center, 2015.

[14] Marc Aurelio Ranzato, Y-Lan Boureau, Sumit Chopra, and Yann LeCun, *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics, vol.2*, Proceedings of Machine Learning Research (Meila, M. and Shen, X., ed.), pp. 371–379, http://proceedings.mlr.press/v2/ranzato07a/ranzato07a.pdf, 21–24 March, 2007.

[15] David Charte, Francisco Charte, Salvador García, María José del Jesús, and Francisco Herrera, *A Practical Tutorial on Autoencoders for Non-linear Feature Fusion: Taxonomy, Models, Software and Guidelines*, http://arxiv.org/abs/1801.01586, Mon, 13 Aug 2018.

[16] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. *Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research, Vol. 12, pp. 2825–2830* 2011.

[17] Diederik P. Kingma and Jimmy Ba, *Adam: A Method for Stochastic Optimization, CoRR, Vol. abs/1412.6980, Mon, 13 Aug 2018*, http://arxiv.org/abs/1412.6980, 2014.

[18] Chollet, François and others, *Keras*, https://keras.io, 2015.

[19] Yoshua Bengio, Aaron Courville, Ian Goodfellow, *Deep Learning*, MIT Press, https://www.deeplearningbook.org, 2016.

[20] N. Heard, N. Adams, P. Rubin-Delanchy, and M. Turcotte, *Data Science for Cyber-Security*, WORLD SCIENTIFIC (EUROPE), https://www.worldscientific.com/doi/abs/10.1142/q0167 2018.

[21] A. Bohara, M. A. Noureddine, A. Fawaz and W. H. Sanders, *An Unsupervised Multi-Detector Approach for Identifying Malicious Lateral Movement*, 2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS), pp. 224-233, Hong Kong, 2017.

[22] Aaron Tuor, Samuel Kaplan, Brian Hutchinson, Nicole Nichols, and Sean Robinson, *Deep Learning for Unsupervised Insider Threat Detection in Structured Cybersecurity Data Streams* arXiv preprint arXiv:1710.00811 2017.

[23] Michal Choras and Rafal Kozik, *Machine learning techniques applied to detect cyber attacks on web applications* Logic Journal of the IGPL, Volume 23, Issue 1, 1 February 2015, Pages 45?56, https://doi.org/10.1093/jigpal/jzu038 2014.

[24] H. Siadati, B. Saket and N. Memon, *Detecting malicious logins in enterprise networks using visualization*, 2016 IEEE Symposium on Visualization for Cyber Security (VizSec), pp. 1–8., Baltimore, MD, 2016.

[25] N. Heard and P. Rubin-Delanchy, *Network-wide anomaly detection via the Dirichlet process*, 2016 IEEE Conference on Intelligence and Security Informatics (ISI), pp. 220-224, Tucson, AZ, 2016.

[26] Adam Fuchs (Sqrrl), Ryan Nolette (Sqrrl) *Threat Hunting for Lateral Movement*, Retrieved from https://resources.sei.cmu.edu/asset_files/Presentation/2018_017_001_512070.pdf.