Anomaly-based Network Intrusion Detection using Wavelets and Adversarial Autoencoders

Samir Puuska, Tero Kokkonen, Janne Alatalo, and Eppu Heilimo

Institute of Information Technology, JAMK University of Applied Sciences, Jyväskylä, Finland
{samir.puuska, tero.kokkonen, janne.alatalo, eppu.heilimo}@jamk.fi

Abstract. The number of intrusions and attacks against data networks and networked systems increases constantly, while encryption has made it more difficult to inspect network traffic and classify it as malicious. In this paper, an anomaly-based intrusion detection system using Haar wavelet transforms in combination with an adversarial autoencoder was developed for detecting malicious TLS-encrypted Internet traffic. Data containing legitimate, as well as advanced malicious traffic was collected from a large-scale cyber exercise and used in the analysis. Based on the findings and domain expertise, a set of features for distinguishing modern malware from packet timing analysis were chosen and evaluated. Performance of the adversarial autoencoder was compared with a traditional autoencoder. The results indicate that the adversarial model performs better than the traditional autoencoder. In addition, a machine learning pipeline capable of analyzing traffic in near real time was developed for data analysis.

Keywords: Adversarial Autoencoder · Intrusion Detection · Anomaly Detection · Haar Wavelets.

1 Introduction

The Internet is becoming more secure as encryption becomes more ubiquitous and new standards are adopted. Web pages and other related assets that make up modern web applications are more often transferred using Transport Layer Security (TLS). In addition to providing security to end users, it also allows malicious actors to leverage encryption for evading detection. Therefore, it is extremely important to know the situation of your own valuable assets in the network. For accomplishing that task, one must maintain good visibility into the network, despite of increasing encryption.

Artificial intelligence (AI) and its applications for cyber security are active and growing research fields. Pham et al. compared various machine learning techniques commonly used for intrusion detection [20], while Dhingra et al. outlined several different application areas and challenges for AI in the enterprise information security landscape [6]. Hendler et al. used neural networks for detecting malicious PowerShell commands [9]. Various novel approaches, such as neural immune detectors, Short-Term Memory Recurrent Neural Networks, and Stacked

Auto-Encoders (SAE) have also been studied for attack detection [13, 14, 25]. Intrusion detection systems (IDS) can be divided into anomaly-based detection (anomaly detection) and signature-based detection (misuse detection). Anomaly detection has capability to detect unknown attack patterns; however, anomaly detection usually generates a large amount of false positive indications [19].

Modern intrusions and malware are tasked from the Internet by malicious actors using so-called command and control (C2) channels. Modern malware utilizes various techniques, such as encryption and steganography, for avoiding the detection of communication with a C2 server. TLS is an extremely good way of hiding the command and control traffic because it has become almost ubiquitous, and the recent efforts at hardening TLS infrastructure, such as certificate preloading or easily obtainable free legitimate certificates, have made certificate bumping and other deep inspection methods unreliable. This paper focuses on malware that utilizes TLS for evasion.

This paper presents an anomaly detection -based IDS that leverage Haar wavelet transforms and Adversarial Autoencoders (AA). First, the reasoning about selected features and methods is presented. Next, the implemented solution and validation results are shown. Finally, future research topics are given along with a discussion.

2 Feature Engineering and Selection

Feature selection is the key element in anomaly detection, determining the maximal effectiveness of the detection capability. Chandola et al. [4], and Sommer et al. [22] both listed several challenges in applying machine learning to anomaly detection. One of the challenges they mentioned is how to select features that actually vary between legitimate and malicious traffic, and defining an effective boundary between them. They also noted that when a malicious actor is involved, the adversary is able to adapt.

Due to the increasing ratio of encrypted traffic, the features cannot utilize the payload of the network packets. In the Internet Protocol (IP) packets, the fields cannot be encrypted and are available for feature engineering. Packet timings, and TLS connection parameters, such as handshake parameter negotiation, are also available.

Networks and Internet traffic are not static in volume or content. They experience considerable variance depending on many factors such as workday cycles, scheduled software updates, or changes in workforce structure. More formally, time series based on our features are non-stationary.

The aim of feature selection is to use as much feature engineering as possible to filter out variances in data that are known to be irrelevant. In addition, the features used must not be readily attacker controlled or circumvented.

2.1 TLS Fingerprints

In the TLS handshake the client and server agree what cryptographic suites they will use. The client sends its preferred suites in preferred order in a package dubbed "ClientHello" [21]. The order, number, and types of these suites vary considerably between web browsers, desktop applications and other programs, thus forming a sort of fingerprint.

Common malware and various APT-simulation tools, such as CobaltStrike[1], Empire[2], and Meterpreter[3], were analysed. It was discovered that their TLS handshakes were either unique or different than legitimate programs. Specific versions of Firefox browsers used in Kali Linux[4], a well known penetration testing tool collection, was also detected. This is in line with previous research; for example, Husák et al. obtained similar results when fingerprinting applications [10]. Although preliminary look into this feature gave extremely positive results, it is also something that the adversary may choose to change, as e.g. TOR meek[5] does. The feature is still useful in a limited manner, it groups applications reliably and only the most sophisticated adversary can tailor the malware traffic to look like the one a particular target organization is using. It should be noted that e.g. PowerShell environment, often used for running malicious code, does not allow the scripts to select preferred suites. We did not use TLS fingerprints in the neural network input data.

The malware used in this research was tasked to beacon to the C2 server from several hours to days. After that the malware was used to perform various malicious activities, such as listing processes, transferring files, taking screenshots, and for further lateral movement on the internal networks. First infection was achieved using either phishing e-mails, or custom-made zero-day exploits.

2.2 Network Flows and Time Series

Network flows form a natural time series, especially when considering those made with Transmission Control Protocol (TCP). Depending on the application, these flows will vary with respect to duration, number of packets transmitted and received, and periodicity, among other characterizing statistics. As previously stated, these time series are non-stationary. This is partly due to the inherent nature of TCP flows, as well as the aim to keep once-negotiated tunnels up for subsequent data transfer, rather than renegotiate. This is also true for TLS, which in virtually any application, runs on top of TCP.

There are several well-know legitimate use cases for TLS encrypted connections. The most ubiquitous one is the World Wide Web; virtually every major web application is accessible or mandates the use of TLS. Virtual Private Networks (VPN) may also be deployed on top of TLS. Compared to the web browsing, these connections can be longer-lived, and their activity is more varying. The third major category is desktop applications, which use TLS to connect securely to their back-ends which may reside either on an internal network or the Internet.

[1] https://www.cobaltstrike.com/
[2] https://www.powershellempire.com/
[3] https://www.offensive-security.com/metasploit-unleashed/about-meterpreter/
[4] https://www.kali.org/
[5] https://trac.torproject.org/projects/tor/wiki/doc/meek

As for malware, TLS provides a practical channel for communicating with Command and Control (C2) servers. It blends in easily with legitimate network traffic, and in many deployments is permitted through firewalls. However, these connections are not usually similar to the legitimate use cases. In the analyzed malware and APT tools, the connections were very short-lived when there were no instructions available for them in their C2 server. When they are tasked to e.g. transfer files or take screenshots, the connections looked different. Based on these observations it was concluded that an aggregation of TLS connections using the IP address or Server Name Indication (SNI) record, the result will form a descriptive time series usable for anomaly detection. This series can be constructed from packet timings and sizes made into an impulse signal, where received packets have negative values, sent packets positive values and where the impulse values are the packet sizes.

2.3 Analysis using Haar Wavelets

There are many options for characterizing a time series. It is important to use a representation that retains the essential features for the classification task at hand without overfitting the data. By considering overfitting also at this stage, the input to the classification algorithm can be made less noisy. Due to the non-stationary nature of our data, the methods at our disposal are somewhat limited. The differing lengths of the series also needs to be addressed.

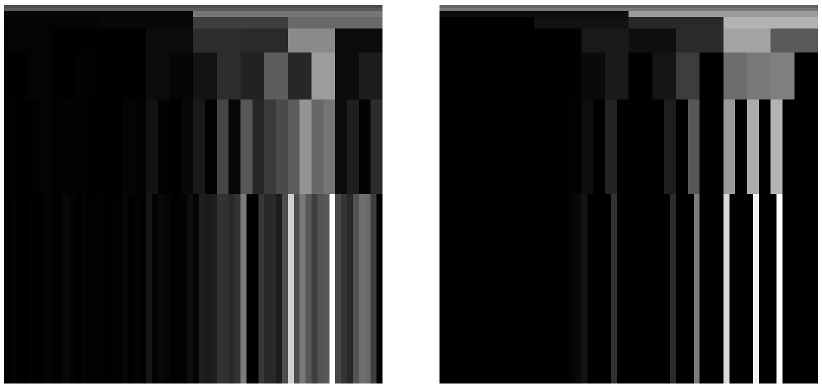
There are two main categories for mathematical time series representations, data-adaptive and non data-adaptive [15]. Haar wavelets [8], a non data-adaptive representation, was chosen. The transform contains both time and frequency elements, and is therefore advantageous for data which is both non-stationary and sparse [5, 3].

Figure 1 illustrates the result of decomposition as it is used in this study. The image represents eight of the lowest frequency coefficient layers from the wavelet transformation result. Brighter areas represent higher coefficient values and the black areas have coefficients values very close to zero. The number of coefficient samples doubles on each layer when more layers are taken. The new layers represent higher and higher frequencies. The solution is designed to examine the long, low frequency traffic patterns so it is safe to discard the high frequency layers. In this study, only the eight coefficient layers that represent the lowest frequencies are kept.

2.4 Adversarial Autoencoders

The wavelet transform provides a starting point for anomaly detection. However, the comparison of the decomposition results is a non-trivial task. There are no obvious ways to assign probabilities to real-valued time series or their transformations in this dataset.

Autoencoders are constructed using artificial neural networks that attempt to reconstruct their input using a relatively small hidden (latent) layer, in a fashion similar to the Principal Component Analysis. Adversarial autoencoders



(a) Legitimate web browser traffic to an authentication portal.

(b) Malicious traffic caused by a CobaltStrike beacon.

Fig. 1: Wavelet decomposition, illustrated here with scalograms, for both legitimate and malicious TLS traffic samples.

have several desired properties for anomaly detection, especially in our use case. Adversarial autoencoders (AA) combine ideas from traditional autoencoders and generative adversarial networks, turning autoencoders into generative models. AAs are suitable for unsupervised learning, or they can also be used in supervised or semi-supervised fashion [17, 7]. Their advantage over traditional AAs is the ability to influence what distribution the hidden layer should approximate. This allows the model to learn additional variance that is not present in the training data, making it less likely overfitted. Adversarial autoencoders use generative adversarial neural networks for regulating the distribution of the autoencoder's latent space (the latent). The encoder of the network is trained to fool the discriminator by generating vectors similar to the chosen distribution, while the discriminator is trained to determine if the sample is generated or from the chosen distribution. Meanwhile, the decoder is trained to reconstruct the input data from the latent space. [17] The reconstruction loss and generation loss are optimized for each batch using separate optimizers, hence the calculated gradients are applied in turns.

Figure 2 shows the general architecture of the neural network design developed during this research. It is based on the architecture proposed by Makhzani et al. [17]. The AA variant in this study (TLS-AAE) is trained unsupervised, and regularized with a continuous distribution. This allows the TLS-AA to better reconstruct input variants not present in the training dataset, resulting in a lower reconstruction error than a traditional autoencoder. Although the reconstruction of the new variants is beneficial for reducing the number of false positives, the network might learn to reconstruct anomalies as well. To counter the unwanted variants, a one-hot categorical distribution was imposed into softmax of the latent, dividing the latent space into clusters of continuous distributions. The amount of clusters can be set using a parameter, that is 20 in this study.

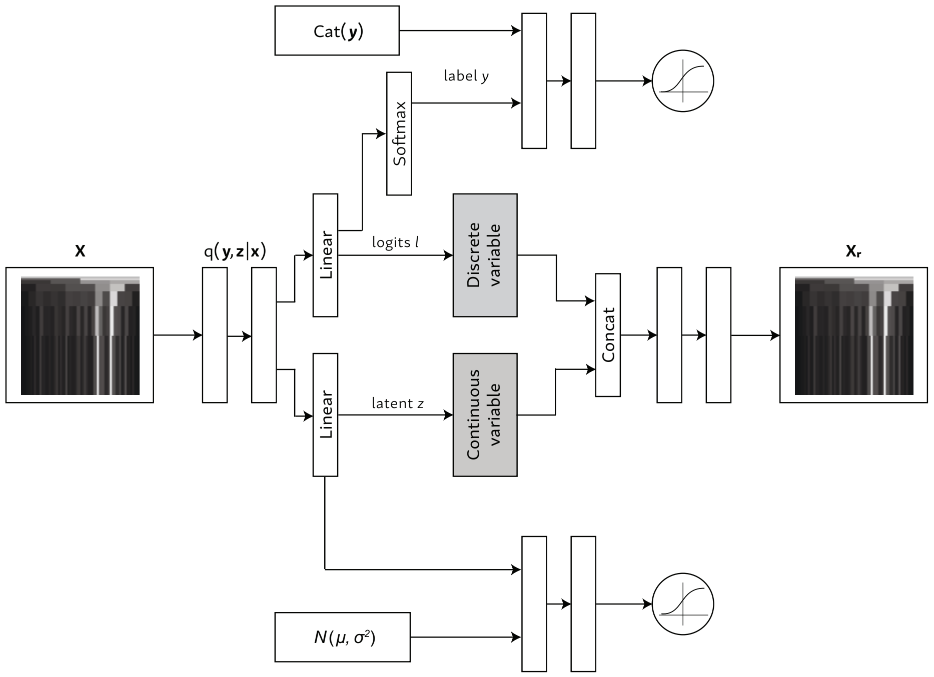


Fig. 2: The general architecture of the neural network design used in this research. The encoder's outputs latent z and label y are enforced to match the selected distributions by establishing a two-player adversarial min-max game between the discriminator and the generator/encoder. [17]. The circles represent adversarial loss, while X_r represents the reconstruction.

The number of clusters does not determine or depend on the labels or classes the network can predict. The TLS-AAE, trained in an unsupervised fashion, uses the classes as a way to utilize the latent space more efficiently.

By utilizing the discrete variable y, the TLS-AAE can output what it considers the best cluster for each input. The clusters are used in a cost function; the cost function is added to the cluster variable (logits l), and it penalizes small Euclidean distances between any two points belonging to different clusters. If the distance between cluster boundaries is over a chosen threshold, it is set to zero. Since the TLS-AAE uses a Gaussian distribution as the continuous distribution, the cluster threshold is set to the length of three standard deviations to ensure that the probability of outlier variants between clusters is minimized. Equation 1 is the cost function, where C is a set containing point sets for each cluster, and $d_{\mathtt{min}}$ is the desired minimum distance between any two clusters.

$$L(C, d_{\mathtt{min}}) = \frac{1}{|C|} \sum_{x \in C_i} \sum_{y \in C_j,\ i \neq j} \max\left\{0, d_{\mathtt{min}} - ||x - y||^2\right\} \tag{1}$$

By utilizing several different optimization targets the latent space can be constructed to approximate what the authors believe are reasonable assumptions about the nature of TLS connections. This reasoning stems from the idea that the TLS traffic can be divided into several categories depending on the application responsible for generating it; web browsing, music streaming services, and malware should form distinct clusters. The anomaly detection is done by calculating the squared Euclidean distance between the input and output image. Squared error magnifies larger errors while disregards small ones.

3 Analysis Pipeline

The TLS-AAE was implemented as a part of an analysis pipeline for evaluating real-world performance and suitability. The implementation was made using open source software frameworks. The pipeline works in any network where the traffic can be mirrored for the analysis system for inspection. Figure 3 illustrates the architecture of the pipeline. The line consists of two main components, the preprocessing pipeline and the machine learning model.

3.1 Data processing

In the start of the pipeline, a slightly modified version of Suricata IDS software [23] was used for constructing network flows from the individual mirrored packets. The modification to Suricata software was made for collecting individual packet timings for each flow.

The main pipeline functionality was implemented using Apache Kafka [1] as a message queue. Suricata was configured to send the flows to one of the Kafka topics. From there, the flows are consumed using Apache Spark [2] platform running a custom preprocessing and feature extraction script. The extracted features are then sent back to Kafka, and delivered to the machine learning algorithm. In the final step of the pipeline the TLS-AAE returns the anomaly scores back to Kafka, after predicting the anomaly score.

The Apache Spark platform was used for extracting the needed features from flows. It was used for flow filtering, flow aggregation, and wavelet calculation by writing a custom script. The script filters all the flows that are not TLS traffic and aggregates the flows using a time window. The aggregation joins flows in a specified time window using source IP, destination IP, destination port and JA3 hash[6]. The JA3 hash is used in aggregation to separate different applications on a host.

The flows include the timing and size information for every packet. Packet timings are reduced from microseconds to seconds in accuracy to reduce unnecessary computational complexity. The packet timings and sizes are made into an impulse signal, where the received packets have negative values, sent packets positive values, the impulse values being the packet sizes. The signal is zero-padded from both ends so that the length of the signal is power of two and the

[6] https://github.com/salesforce/ja3

minimum length is reached. The minimum length of the signal depends on how many layers from the wavelet transformation are needed.

The wavelet transform is calculated for the signal using the Haar mother wavelet. Only the last N layers of the transform result are maintained so that all the results are of the same size regardless of the original signal length. The wavelet result is then transformed to an X x Y matrix. The transformation is made so that the results are easy to visualize. Absolute values are taken from the coefficients and the matrix values are normalized per matrix. The matrix is made in a way that each detail coefficient value populates equal amount of elements in the matrix. Figure 1 shows the matrix transformation visualized as an image. The color in the image represents the element value. The lower squares in the image represent the higher frequencies and the position from the left represents in which time the frequency has happened in the signal.

The results are sent back to Kafka where the machine learning algorithm can consume it and produce the anomaly score as the result.

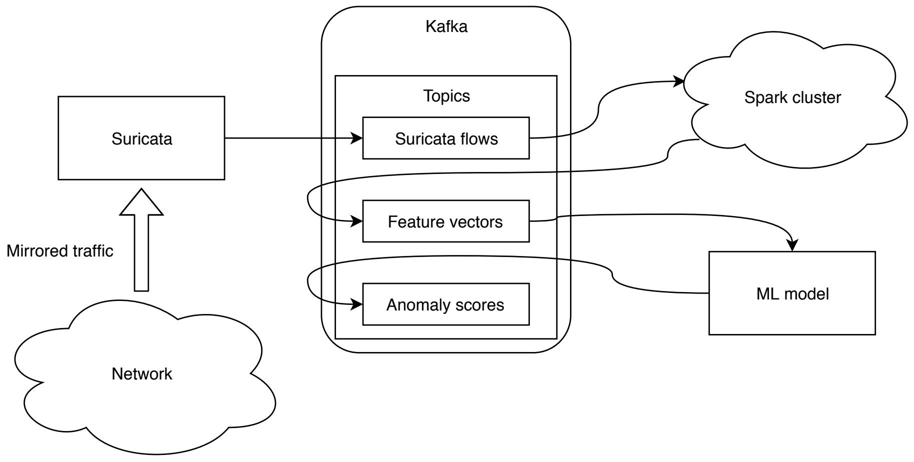


Fig. 3: Architecture of the implemented IDS solution

3.2 Dataset

A relevant and modern dataset is required for training, testing, and evaluating a IDS solution. Although some public datasets, such as the DARPA intrusion detection evaluation dataset [16] exists, they tend to be dated and not suitable for modern research. They lack both up-to-date cyber attacks, and modern traffic profile.

Since 2013 the Finnish national cyber security exercise has been conducted using the RGCE Cyber Range. In 2018 the exercise was organized by The Ministry of Defence, The Security Committee, and JAMK University of Applied

Sciences. The exercise is a large-scale live cyber security exercise, with more than 100 individuals from different national security authorities exercising co-operation during the cyber incidents. [18]

The data set used in this study was created from network traffic captured during the exercise. The whole traffic capture, at full packet level, consists of over 100,000,000 network flows from which a subset of 56,408,665 flows were captured from a place where anomaly traffic was present. This subset was used to create the training and testing data sets. The data set contains 729,998 flows that are TLS traffic, of which 665 flows are malicious.

The flows contain both human and auto-generated web browsing traffic, authentication portal logins, automatic updates of software, e-mail protocols that use TLS, and other common benign activity. Malicious flows were generated by Meterpreter, Empire, or CobaltStrike.

3.3 RGCE Cyber Range

The Realistic Global Cyber Environment (RGCE) is a holistic cyber range suitable for various tasks such as training, exercises, research and development. RGCE mimics the structure and traffic of real Internet. For example, ISP tiers are emulated using real hardware and structure. Network distances and latencies reflect those in real world, up to including packet losses. The network traffic of RGCE Cyber Range is generated according to a realistic end user traffic model, which augments the traffic generated by humans. RGCE includes industry specific organization environments, with complex deployments. For example, financial organization, electricity company and Internet service providers all have realistic AD infrastructure, SCADA systems, and other specialized production assets. [12, 11]

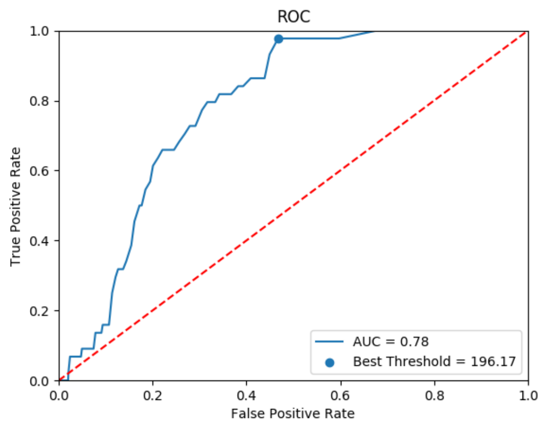
4 Results

The evaluation of the performance was made by using the receiver operating characteristic (ROC) curve by plotting the true positive rate (TPR) to y-axis and false positive rate (FPR) to x-axis [24]. Overall, the following characteristics were considered: TPR, FPR, and accuracy [24, 19].

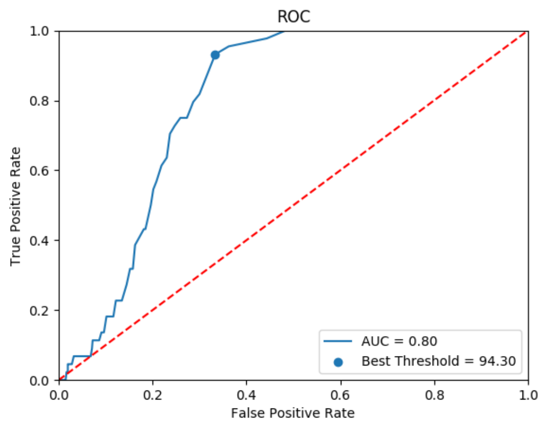
For evaluation of TLS-AAE, the prediction results were compared against a traditional autoencoder. The performance characteristics are listed in Table 1 and ROC curves are plotted in Figure 4.

Method	**TPR**	**FPR**	**Accuracy**
TLS-AAE	95%	36%	65%
Plain autoencoder	97%	47%	55%

Table 1: Performance characteristics



(a) Traditional autoencoder



(b) TLS-AAE

Fig. 4: Receiver operating characteristic curves for both plain autoencoder and the TLS-AAE. Best thresholds are based on the best ratio.

The results indicate that the TLS-AAE achieves similar TPR as the plain counterpart. However, the FPR is considerably lower, resulting in better accuracy. The input image of the autoencoder is 128x128 pixels and the grayscale values vary between 0 and 255 integers. Both autoencoders have two dense layers of 2048 units in both encoder and decoder. The bottleneck of the adversarial autoencoder consists of 10 dimensional latent and 20 dimensional cluster linearly activated variables. For consistency, the traditional autoencoders latent is 30 dimensional. The output of the decoder uses sigmoid activation to map the decoded values between 0 and 1.

5 Discussion

Based on the findings, the Haar wavelet transform seems to provide adequate representation on the nature of the TLS connections in the dataset, allowing categorization. Time window aggregation of distinct but related TLS flows captures malicious programs that infrequently poll the C2 server, opening a new TLS connection each time.

Considering that the connections were encrypted and only the size and timing information was available for analysis, the unsupervised TLS-AAE was able to construct a relatively representative latent space. Even though the dataset is relatively extensive in size, the variance of the flows is constricted. The relatively high false positive rate is partly explained by non-malicious outliers. The number of false positives can be drastically lowered by augmenting the results with other means of traffic analysis, such as IP / domain reputation.

The modified Suricata IDS, Spark, Kafka, and TensorFlow – in combination – proved to be a working base for an IDS solution. As Suricata can either process live mirrored traffic or replay an existing packet capture, developing models using the platform is relatively straightforward process.

6 Conclusion and Future Work

In this study Haar wavelet transforms and adversarial autoencoders were applied for constructing an anomaly detection based network intrusion detection system. For evaluation, a data pipeline based on open source software, including Suricata IDS, TensorFlow framework, Kafka message bus, and Spark framework, was constructed.

Network data from Finnish national cyber security exercise was used for the evaluation of the proposed model. The data was also used for finding and engineering suitable features for encrypted TLS connections. The test data included various attack vectors made by malware and exploitation frameworks.

Future work includes a more thorough statistical analysis on the TLS-AAE's latent space and its structure. Possible avenues of expansion are combining the current model with a more sophisticated predictor network. The wavelet transform and its applicability for TLS traffic analysis should be also further studied.

Acknowledgment

This research project is funded by MATINE - The Scientific Advisory Board for Defence.

References

1. Apache Kafka: Apache kafka A distributed streaming platform. `https://kafka.apache.org/`, accessed: 31 August 2018

2. Apache Spark: Apache Spark Lightning-fast unified analytics engine. https://spark.apache.org/, accessed: 31 August 2018
3. Chan, K., Fu, W.: Efficient time series matching by wavelets. In: Proceedings 15th International Conference on Data Engineering (Cat. No.99CB36337). pp. 126–133 (March 1999). https://doi.org/10.1109/ICDE.1999.754915
4. Chandola, V., Banerjee, A., Kumar, V.: Anomaly Detection: A Survey. ACM Comput. Surv. **41**(3), 15:1–15:58 (Jul 2009). https://doi.org/10.1145/1541880.1541882
5. Daubechies, I.: The wavelet transform, time-frequency localization and signal analysis. IEEE Transactions on Information Theory **36**(5), 961–1005 (Sept 1990). https://doi.org/10.1109/18.57199
6. Dhingra, M., Jain, M., Jadon, R.S.: Role of artificial intelligence in enterprise information security: A review. In: 2016 Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC). pp. 188–191 (Dec 2016). https://doi.org/10.1109/PDGC.2016.7913142
7. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 27, pp. 2672–2680. Curran Associates, Inc. (2014)
8. Haar, A.: Zur theorie der orthogonalen funktionensysteme. Mathematische Annalen **69**(3), 331–371 (Sep 1910). https://doi.org/10.1007/BF01456326
9. Hendler, D., Kels, S., Rubin, A.: Detecting Malicious PowerShell Commands Using Deep Neural Networks. In: Proceedings of the 2018 on Asia Conference on Computer and Communications Security. pp. 187–197. ASIACCS '18, ACM, New York, NY, USA (2018). https://doi.org/10.1145/3196494.3196511
10. Husák, M., Čermák, M., Jirsík, T., Čeleda, P.: Https traffic analysis and client identification using passive ssl/tls fingerprinting. EURASIP Journal on Information Security **2016**(1), 6 (Feb 2016). https://doi.org/10.1186/s13635-016-0030-7
11. JAMK University of Applied Sciences, Institute of Information Technology, JYVSECTEC: Rgce cyber range. http://www.jyvsectec.fi/en/rgce/, accessed: 23 August 2018
12. Kokkonen, T., Puuska, S.: Blue Team Communication and Reporting for Enhancing Situational Awareness from White Team Perspective in Cyber Security Exercises. In: Galinina, O., Andreev, S., Balandin, S., Koucheryavy, Y. (eds.) Internet of Things, Smart Spaces, and Next Generation Networks and Systems. pp. 277–288. Springer International Publishing, Cham (2018)
13. Komar, M., Kochan, V., Dubchak, L., Sachenko, A., Golovko, V., Bezobrazov, S., Romanets, I.: High performance adaptive system for cyber attacks detection. In: 2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS). vol. 2, pp. 853–858 (Sept 2017). https://doi.org/10.1109/IDAACS.2017.8095208
14. Le, T., Kim, J., Kim, H.: An Effective Intrusion Detection Classifier Using Long Short-Term Memory with Gradient Descent Optimization. In: 2017 International Conference on Platform Technology and Service (PlatCon). pp. 1–6 (Feb 2017). https://doi.org/10.1109/PlatCon.2017.7883684
15. Lin, J., Keogh, E., Lonardi, S., Chiu, B.: A symbolic representation of time series, with implications for streaming algorithms. In: Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery. pp. 2–11. DMKD '03, ACM, New York, NY, USA (2003). https://doi.org/10.1145/882082.882086

16. Lippmann, R.P., Fried, D.J., Graf, I., Haines, J.W., Kendall, K.R., McClung, D., Weber, D., Webster, S.E., Wyschogrod, D., Cunningham, R.K., Zissman, M.A.: Evaluating intrusion detection systems: the 1998 darpa off-line intrusion detection evaluation. In: Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00. vol. 2, pp. 12–26 vol.2 (Jan 2000). https://doi.org/10.1109/DISCEX.2000.821506
17. Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I.: Adversarial Autoencoders. In: International Conference on Learning Representations (2016), http://arxiv.org/abs/1511.05644
18. Ministry of Defence Finland: The national cyber security exercises is organised in Jyväskylä - Kansallinen kyberturvallisuusharjoitus kyha18 järjestetään Jyväskylässä, official bulletin 11th of may 2018. https://valtioneuvosto.fi/artikkeli/-/asset_publisher/kansallinen-kyberturvallisuusharjoitus-kyha18-jarjestetaan-jyvaskylassa (May 2018), accessed: 23 August 2018
19. Mokarian, A., Faraahi, A., Delavar, A.G.: False positives reduction techniques in intrusion detection systems-a review. IJCSNS International Journal of Computer Science and Network Security **13**(10), 128–134 (2013)
20. Pham, T.S., Hoang, T.H., Vu, V.C.: Machine learning techniques for web intrusion detection — A comparison. In: 2016 Eighth International Conference on Knowledge and Systems Engineering (KSE). pp. 291–297 (Oct 2016). https://doi.org/10.1109/KSE.2016.7758069
21. Rescorla, E., Dierks, T.: The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (August 2008). https://doi.org/10.17487/RFC5246
22. Sommer, R., Paxson, V.: Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. In: 2010 IEEE Symposium on Security and Privacy. pp. 305–316 (May 2010). https://doi.org/10.1109/SP.2010.25
23. Suricata: Suricata Open Source IDS / IPS / NSM engine. https://suricata-ids.org/, accessed: 31 August 2018
24. Suyal, P., Pant, J., Dwivedi, A., Lohani, M.C.: Performance evaluation of rough set based classification models to intrusion detection system. In: 2016 2nd International Conference on Advances in Computing, Communication, Automation (ICACCA) (Fall). pp. 1–6 (Sept 2016). https://doi.org/10.1109/ICACCAF.2016.7748991
25. Vartouni, A.M., Kashi, S.S., Teshnehlab, M.: An anomaly detection method to detect web attacks using stacked auto-encoder. In: 2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS). pp. 131–134 (Feb 2018). https://doi.org/10.1109/CFIS.2018.8336654