applied sciences

MDPI

Article

APT-Attack Detection Based on Multi-Stage Autoencoders

Helmut Neuschmied [1], Martin Winter [1], Branka Stojanović [1], Katharina Hofer-Schmitz [1], Josip Božić [1,*] and Ulrike Kleb [2]

[1] DIGITAL—Institute for Information and Communication Technologies, Joanneum Research GesmbH, A-8010 Graz, Austria; helmut.neuschmied@joanneum.at (H.N.); martin.winter@joanneum.at (M.W.); branka.stojanovic@joanneum.at (B.S.); katharina.hofer-schmitz@joanneum.at (K.H.-S.)

[2] POLICIES—Institute for Economic and Innovation Research, Joanneum Research GesmbH, A-8010 Graz, Austria; ulrike.kleb@joanneum.at

* Correspondence: josip.bozic@joanneum.at

Abstract: In the face of emerging technological achievements, cyber security remains a significant issue. Despite the new possibilities that arise with such development, these do not come without a drawback. Attackers make use of the new possibilities to take advantage of possible security defects in new systems. Advanced-persistent-threat (APT) attacks represent sophisticated attacks that are executed in multiple steps. In particular, network systems represent a common target for APT attacks where known or yet undiscovered vulnerabilities are exploited. For this reason, intrusion detection systems (IDS) are applied to identify malicious behavioural patterns in existing network datasets. In recent times, machine-learning (ML) algorithms are used to distinguish between benign and anomalous activity in such datasets. The application of such methods, especially autoencoders, has received attention for achieving good detection results for APT attacks. This paper builds on this fact and applies several autoencoder-based methods for the detection of such attack patterns in two datasets created by combining two publicly available benchmark datasets. In addition to that, statistical analysis is used to determine features to supplement the anomaly detection process. An anomaly detector is implemented and evaluated on a combination of both datasets, including two experiment instances–APT-attack detection in an independent test dataset and in a zero-day-attack test dataset. The conducted experiments provide promising results on the plausibility of features and the performance of applied algorithms. Finally, a discussion is provided with suggestions of improvements in the anomaly detector.

Keywords: machine learning; autoencoder; anomaly detection; intrusion detection; statistical analysis

Citation: Neuschmied, H.; Winter, M.; Stojanović, B.; Hofer-Schmitz, K.; Božić, J.; Kleb, U. APT-Attack Detection Based on Multi-Stage Autoencoders. *Appl. Sci.* **2022**, *12*, 6816. https://doi.org/10.3390/app12136816

Academic Editors: Howon Kim and Thi-Thu-Huong Le

Received: 7 June 2022
Accepted: 1 July 2022
Published: 5 July 2022

1. Introduction

Advanced technologies enable the interconnection of people, organisations and infrastructure as one system. In this way, they affect the development of social, economic and political life. For this reason, ensuring cybersecurity in network systems represents a critical challenge to ensure the functionality of existing infrastructure. In such complex environments, cyberattacks are becoming more sophisticated as well. Advanced attacks include, for example, evolving malware, and represent a major security challenge due to the emergence of new variants of attacks. Eventually, each variant encompasses new signatures and functionality that is not detected by existing tools or techniques. This is especially the case with attacks that are active over longer periods of time. Usually, such attacks are carried out by different actors, ranging from individuals to state-sponsored groups. Such attacks can lead to unpredictable consequences and are usually very hard to detect in real-time. Due to the above reasons, existing detection techniques are rendered obsolete when dealing with APT attacks. APT attacks often come in the form of multi-stage attacks, which means that they are executed in multiple steps. For example, in the initial step, access is gained to a target system. Subsequently, an open channel is established that

can be used for further actions in the aftermath. For this reason, the identification of the initial step remains the most important challenge in intrusion detection [1].

Inspired by existing ML and deep-learning (DL) methods, neural networks—for example, autoencoders—are applied to anomaly detection. Autoencoders are a special type of unsupervised learning technique that can be used to derive and learn features and packet classification [2]. They are a special type of multi-layer neural network performing non-linear dimensionality reduction in the data. Given a large amount of benign data, they can be trained to reconstruct input data as closely as possible by minimizing the reconstruction error on the network's output. In turn, the reduced representation in the so-called bottleneck layer make the autoencoders useful for outlier or anomaly detection [3]. Anomaly detection is also utilized in application areas such as video-processing [4], network monitoring and intrusion detection [5–7], cyber-physical systems [2] or monitoring industrial control systems [8]. In recent years, however, the scientific community has increasingly focused on anomaly detection methods in cybersecurity [9–12]. This is especially the case for intrusion detection for APT attacks [1,13].

This paper builds upon previous works from [14,15], where a novel two-stage approach for anomaly detection, relying on autoencoders, was introduced. In this work, we additionally investigated several anomaly-detection methods with an emphasis on filtering methods and performance evaluation on two datasets. We analyse neural networks, including standard and variational autoencoders, as well as their combinations and a support vector machine. Each autoencoder is trained with benign or malicious data during an unsupervised learning process. The corresponding algorithms are used to investigate several known malware attacks against networks. Eventually, their authentic imprints in the datasets are identified according to distinctive evaluation metrics.

The reminder of the paper is structured as follows. Section 2 discusses related work on ML-based anomaly detection in multiple domains. Section 3 provides an overview of the applied methodology, including datasets, APT attacks of interest and evaluation metrics. Subsequently, Section 4 discusses the proposed anomaly-detection framework, i.e., the main contribution of this paper, in detail. Finally, Section 7 elaborates on the experimental results and provides final remarks for future research.

2. Related Work

The application of ML-based anomaly-detection methods represents an important research challenge. Thus far, common algorithms are used to detect suspicious deviations from common behavioural patterns. One popular domain of interest for anomaly detection represents traffic data from network systems. In this case, a network is monitored and the extracted data is used in the posterior detection process [5–7]. Other application domains for anomaly-detection algorithms include, for example, cyber-physical systems (CPS) [2] or smart energy environments [16].

In recent times, however, ML has increasingly found its way into the realm of cybersecurity applications [9–12] and, subsequently, the detection of APT attacks [1,13]. Common ML methods in the latter domain include signature-based detection, behaviour-based detection, monitoring [17] and data mining [18]. Regarding APT attacks, however, not many works deal with this aspect of cybersecurity. The existing literature usually focuses on common cyberattacks, such as DDoS, zero-day and web attacks [19]. A general problem for the detection of APT attacks consists of the lack of adequate datasets. The existing datasets usually capture traffic at external endpoints. However, APT attacks occur in internal networks as well. In addition, due to their artificial nature, the generic datasets do not exactly reflect a real-world environment [20]. In general, it is very difficult to distinguish between the behaviour of a benign user and an attacker. As a result, this leads to the generation of a large number of false positives and false negatives. This is especially the case for semi-supervised and unsupervised learning methods [1]. The reason for this difficulty, therefore, lies in the fact that APT attacks try to disguise themselves as common network behaviour. In such way, these attacks are unpredictable and exhibit non-repetitive

behaviour. In addition, the massive amount of sheer data traffic that is generated by many hosts makes the filtration of low and slow activities even more difficult [21]. Due to such reasons, implementing a general defence approach or application for APT attacks proves to be very difficult, even unlikely. Therefore, this work contributes to this challenge by evaluating the multi-stage approach from [14] for anomaly detection. In this way, our proposed approach contributes to the above-mentioned challenges with the goal to make anomaly-detection findings more understandable.

In [19], a survey is provided on DL methods for the detection of APT attacks. The applied algorithms are discussed with regard to their performance, advantages and existing limitations. In addition, proposed improvements for individual methods are addressed for the mentioned datasets. The same authors propose another DL-based approach for APT-attack detection in [22]. Here, a DL stack is presented that relies on a model where attacks are observed in the form of a multi-vector multi-stage attack.

A survey on multi-step attack detection is given in [23], which provides an analysis of multi-step attacks and mechanisms to predict them. The work states that the identification of multi-step attacks is difficult to achieve due to several reasons. One of the problems represents the complexity of available network data. Filtering relevant information, which usually constitutes a small fraction in a vast amount of data, constitutes a challenging task. In addition, with an increasing number of attack steps, detecting similarities and links between attacks becomes increasingly difficult. Since attackers behave in an unpredictable manner, their strategy cannot be determined intuitively. The latter facts represent an issue for the modelling of attack scenarios or defining them in a standard language. In addition, problems may occur in the case of technical limitations of network hardware.

In [24], the authors discuss APT-attack detection techniques and provide a theoretical (holistic) approach to recognize unique APT features in network attacks. For this purpose, existing APTs from multiple case studies are modelled in three distinctive representations, thereby identifying common features between them. The resulting models for each case study include a high-level kill-chain model, a labelled transition-systems (LTS) diagram and a message-sequence (MSQ) diagram. The main objective of the approach is to differentiate complex APTs from more common attacks such as ransomwares or botnets. After validating the models and the identified features, another case study was defined in order to confirm the authors' claim. They conclude that the produced models sufficiently recognize the identified features in the final case study.

The authors of [1] address open challenges for systems to fend off APT attacks. They discuss their claim that defence against such actions must be tackled at different stages of an occurring attack. However, this implies that proper mechanisms must be implemented across multiple points and levels of a system. Thus, they recommend the implementation of an unsupervised clustering approach to identify general information on anomalies. The authors also provide an APT-attack tree and taxonomy and mitigation methods known so far. Subsequently, the existing APT-attack defence methods are classified into three categories, namely, monitoring, detection and mitigation methods.

Another APT-attack detection approach, called HOLMES, is proposed in [25]. The system produces graphs that summarise attacks and assist real-time response operations. The system was evaluated on data from DARPA that contains simulations of attacks in a network system. The authors conclude that HOLMES detects APT activity with a high precision and a low rate of false alarms. On the other hand, the authors of [26] explore a DL-based proactive APT detection approach in the context of Industrial Internet of Things (IIoT). For this purpose, they present a scheme that focuses on long attack sequences. The solution relies on bidirectional encoder representations from transformers (BERT) for detection purposes with lower false-alarm rates. In addition to that, the paper in [27] addresses the shortcomings of existing APT detection methods. For this reason, the paper provides a state-based framework that reconstructs attack scenarios. This framework relies on a three-phase detection model that summarises the critical phases in APT attacks. From

the obtained results, potential attacks can be predicted with greater accuracy, which is demonstrated in an evaluation.

The work in [28] proposes an APT-attack-stage identification method, called APTSID. This method represents a multi-stage approach that relies on the observed network traffic. In addition, the authors of [20] propose a new dataset and benchmark existing anomaly-detection models on that dataset. According to the performance, they claim that the reliable detection of APT attacks proved to be very difficult. Thus, better learning models need to be created to achieve improved detection results.

A general overview of emerging APT attacks and detection techniques can be found in [1,29]. In addition, an APT-attack classification and respective countermeasures for individual attacks can be found in [30]. Furthermore, defence against such attacks in the context of their life cycle is described in [31].

3. Datasets

This section provides information on the datasets that were used for evaluation and performance determination. In general, data that reflects a correct system functionality are considered "benign". On the other hand, data that contains indicators of a cyberattack are referred as "malicious" or "anomalous". One significant research drawback in the domain of anomaly detection consists in the permanent lack of available datasets. This is due to the fact that datasets contain private user data which is restricted for public access. On the other hand, artificial datasets with anonymized data exist but they lack some critical features [32]. In the latter case, datasets must be periodically adapted to evolving attack strategies. These include, among others, the following datasets: DARPA [33], KDD'99 [34], DEFCON [35] and CAIDA [36].

Similar to a previous paper, [15], this work relies on two different datasets, namely, Contagio [37] and CICIDS2017 [38]. Whereas the former contains data on APT, the latter encompasses benign and attack data that resemble real-world data. However, these attacks are not considered part of APT; hence, this work labels them as background data. Both datasets include the results of a network -traffic analysis and contain features that are based on multiple variables. Contagio represents a malware database that encompasses a collection of raw network data in the form of PCAP files. The dataset is made up of a total of 36 files, where each file contains network-traffic data that was subject to a different type of ATP attack. These include the latest malware samples, threats, observation analyses and data that was subjected to attacks from several APTs. On the other hand, CICIDS2017 is a publicly available dataset that was developed by the Canadian Institute for Cybersecurity. In general, this dataset can be considered as a benchmark dataset in the domain of intrusion detection [32]. As such, it is widely used in the research community since its results can be reproduced, and thus, verified and compared. This dataset consists of five files, representing five working days in an enterprise network, namely, Monday to Friday, 09:00–17:00 h. These files are available in two formats. The first format represents CSV files that contain labelled bi-directional network flows including 78 time-based features, i.e., features that occur during daily time intervals, and additional metadata about IP addresses, ports, protocols and attacks. In addition to CSV files, all subsets are available in a raw network format (PCAP files). The authors of the dataset also provided a tool that can be used for features extraction from raw network data files—CICFlowMeter [39]. In this way, this tool is compatible with the Contagio dataset format, which makes it beneficial to work with.

For the purpose of this paper, both datasets are combined so APT attacks from the Contagio dataset are added to the CICIDS2017 network environment. First, in order to ensure compatibility, features were extracted from six preselected Contagio files and CICIDS2017 PCAP files with the help of CICFlowMeter.

The CICIDS2017 dataset itself is captured on a testbed that comprises two separated networks, i.e., a victim network with 13 machines and an attacker network. In this case, Monday represents the only subset that is free of any attacks. Therefore, the network

data for that day is used as a training subset for the conducted experiments. On the other hand, Contagio files contain only filtered communication between attacker(s) and victim computers, identifiable via IP addresses. In order to merge the corresponding data, IP addresses of victim computers from the Contagio dataset were changed to corresponding IP addresses from the CICIDS2017 victim network pool. APT attacks (Contagio files) preselected for merging with the CICIDS2017 dataset include BIN_9002, BIN_Nettravler, TrojanPage, TrojanCookies, Enfal_Lurid and BIN_LURK. All inspected APT attacks were taken from Contagio and encompass several types of malware, including Trojans, spyware and spear phishing attacks. The corresponding IP address mappings used for merging, as well as the attack durations, are shown in Table 1.

Table 1. APT attacks and mapped IP addresses.

Attack	Victim IP	Duration
BIN_9002	192.168.10.5	0:04
BIN_Nettraveler	192.168.10.15	0:08
TrojanPage	192.168.10.15	0:06
TrojanCookies	192.168.10.9	0:40
Enfal_Lurid	192.168.10.14	0:02
BIN_LURK	192.168.10.8	7:00

The resulting merged dataset includes three instances: (i) training dataset, (ii) independent test dataset and (iii) zero-day-attack test dataset (see Table 2). It should be noted that the training data set is used for training purposes only and does not contain any attacks. On the other hand, the test dataset (independent) files encompass benign, i.e., harmless data, background attacks, and APT-attack traces. This dataset is divided into two distinct parts during the training/testing of the system, namely, a validation and a test set. The zero-day-attack records are used for additional testing to demonstrate the performance of the trained system in detecting previously unknown attack types. These types include attacks that were unknown during the training process.

Table 2. Datasets and selected APT attacks.

Training Dataset - without Attacks	**CICIDS2017-Monday**
Test dataset - independent	1. CICIDS2017-Wednesday with Contagio BIN_9002: - Background attacks: DoS, HeartBleed - APT attacks: BIN_9002 2. CICIDS2017-Wednesday with Contagio BIN_Nettravler: - Background attacks: DoS, HeartBleed - APT attacks: BIN_Nettravler 3. CICIDS2017-Thursday with Contagio TrojanPage: - Background attacks: Brute Force, Infiltration, SQL Injection, XSS - APT attacks: TrojanPage
Test dataset - zero-day attacks	1. CICIDS2017-Tuesday with Contagio TrojanCookies: - Background attacks: FTP patator, SSH patator - APT attacks: TrojanCookies 2. CICIDS2017-Tuesday with Contagio Enfal_Lurid: - Background attacks: FTP patator, SSH patator - APT attacks: Enfal_Lurid 3. CICIDS2017-Friday with Contagio BIN_LURK: - Background attacks: Bot, DDoS, PortScan - APT attacks: BIN_LURK

4. Proposed Methodology

In general, the proposed two-stage detection of APT attacks belongs to the category of anomaly detection. The stages of this approach encompass (1) data pre-processing, and (2) anomaly detection. The former stage deals with the processing of raw data for

further feature selection. This is realised by conducting statistical analysis on individual data records. On the other hand, the latter stage encompasses testing and validation with several autoencoders. In addition to that, both datasets were evaluated according to pre-specified metrics.

4.1. Data Pre-Processing—Statistical Analysis

In ML, large amounts of data are analysed to identify common patterns between data records. For this purpose, statistical analysis is conducted to derive valuable insights from the raw data records. In this case, the analysis is applied to check the plausibility of features and to assess their suitability for anomaly detection. Subsequently, suitable features are selected and used in conjunction with individual detection methods. The statistical analysis includes the following steps:

- Descriptive analysis: Determination of statistical metrics for the plausibility check of feature values.
- Histograms and boxplots: Visualisation and comparison of statistical distributions of normal flows and attacks in datasets.
- Correlation analysis: The examination of features to identify groups of highly correlated or identical features.
- Principal component analysis (PCA): Estimation of the extent of possible dimensionality reduction for individual features.

In this paper, a statistical analysis was conducted on the combined—i.e., the merged Contagio and CICIDS2017—datasets. This analysis resulted both in the reduction in feature vectors and the reduction in the number of features. In particular, 620 of the feature vectors containing negative values were removed. Furthermore, the feature pool was reduced from 78 to a subset of 54 numerical features as a result of the analysis.

As a matter of fact, the initial analysis of 78 original features revealed 8 features containing only zero values, 3 features containing higher amounts of implausible negative data, and 2 features containing infinity values. Next, a correlation analysis revealed 7 redundant features. Thus, 20 features were excluded from further evaluation, leaving 58 features for further analysis. Subsequently, including boxplots identified suspicious data in 4 features, which were then excluded from further anomaly detection. In concrete terms, the affected features exhibit a connection between the feature values and the day of the week on which the data was recorded. However, this observation stands in strong contrast to the network behaviour described by the remaining features. This inexplicable relationship is most apparent in a boxplot for the "Idle Max" feature, which depicts a linear increase in values from Monday to Friday, as shown in Figure 1.

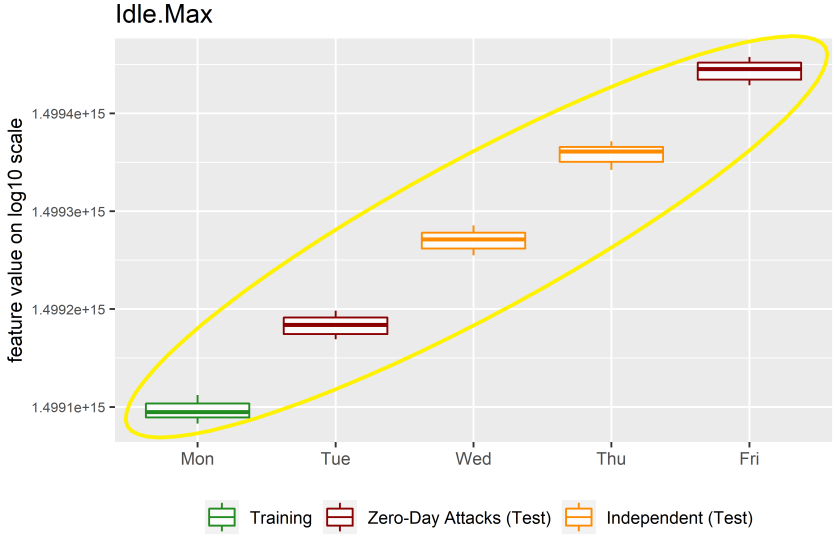

Figure 1. Boxplot of the feature "Idle Max".

Furthermore, the feature distributions of both test data sets—independent ("independent") and completely unknown ("zero-day attack")—were compared with one another. In this case, a distinction was made between benign data ("benign"), background attacks from CICIDS2017 ("attack") and the inserted APT attacks ("APT"). For some features, especially for the data of the background attacks, clear differences in statistical distributions can be recognized (for example, the boxplots framed in yellow in Figure 2). In fact, the results illustrate the difficulty of operating reliable anomaly detection for completely unknown data or application scenarios: changed feature distributions, as they exist for the "zero-day attack" test data set, pose a great challenge for detection algorithms, which usually results in a weaker detection performance for such data.

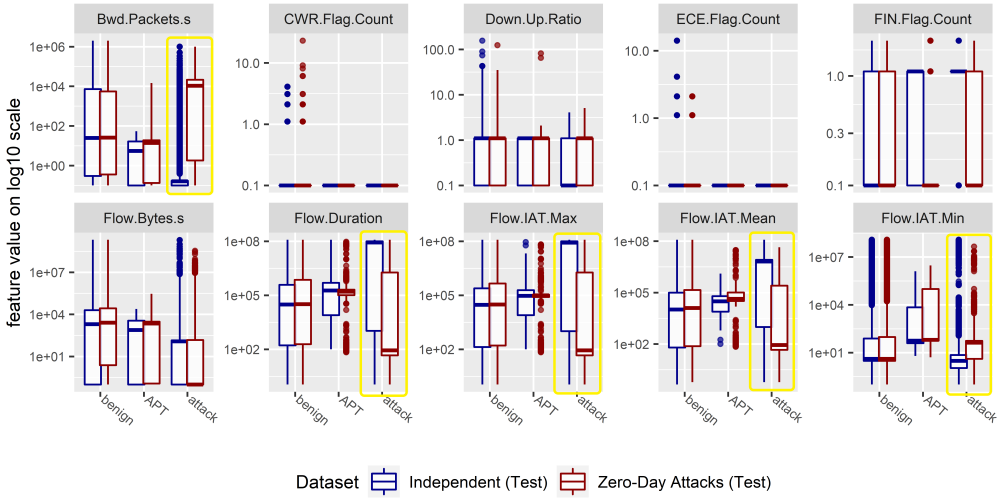

Figure 2. Boxplot comparison of statistical distributions for independent and zero-day attacks.

4.2. Anomaly Detection with Autoencoders

As described in previous sections, the conducted experiments were tested on a combination of the two datasets. The combined result represents a more complex test dataset, which is used for validation and testing. In addition, the test dataset is further used for the additional testing of algorithms on previously unknown attack types. For the conducted tests, the following algorithms were applied:

- Standard autoencoder (AE): This neural network comes with multiple, dense layers that are trained with benign data. It does not consider attack data during this process.
- Autoencoder based on convolutional networks (AE-CNN): This comes in a combination of one-dimensional convolutional and dense connected layers. In addition, this system is trained with only benign data.
- Variational autoencoder (VAE): This sampling-based autoencoder proposed by An and Cho [40] consists of dense connected layers that are trained with only benign data.
- Variational autoencoder using reconstruction probability (VAE-Prob): This probability-based autoencoder consists of dense connected layers. In addition, it is trained with only benign data.
- One-class support vector machine (OCSVM): This ML model represents the "gold standard" in classic ML. Similar to the networks above, it is trained only with benign data.
- Combination of the standard autoencoder and the variational autoencoder (AE+VAE): This two-stage approach, as presented in the previous work in [14], combines both models, which underlying idea is depicted in Figure 3. In a first step—referred to as the pre-processing or filtering step—a fast anomaly detector filters out data which, with a very high probability, does not belong to any anomaly. The remaining data is then evaluated by a second, more specific, anomaly detector that provides a more accurate decision.

- Autoencoder with labelled data (AEC): this autoencoder is used together with the custom loss function to achieve the best DL-based results. The "custom" loss function (used by AE-Custom) takes into account the anomaly data for the calculation of the reconstruction error err_{rec} as follows:

$$err_{rec} = \frac{n_{benign}}{n} \tanh(mse(y_{benign}, y_{pred})) + \frac{n_{anomaly}}{n} (1 - \tanh(mse(y_{anomaly}, y_{pred}))) \quad (1)$$

where:

err_{rec} = reconstruction error;
n = number feature vectors;
n_{benign} = number of benign vectors;
$n_{anomaly}$ = number of anomaly vectors;
y_{benign} = benign feature vectors;
$y_{anomaly}$ = anomaly feature vectors;
y_{pred} = predicted feature vectors;
mse = mean square error function.

However, it should be mentioned that this autoencoder serves only for comparison purposes, since it represents a supervised method. In this way, it provides insights regarding the complexity of datasets.

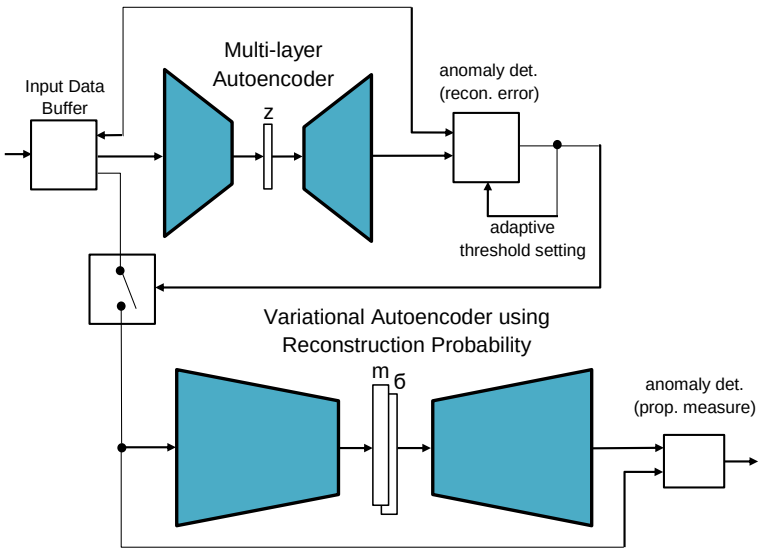

Figure 3. Two-stage anomaly detection with autoencoders.

Hyper-Parameter Optimisation

The implementation of neural-network models for different autoencoders is realised with Talos [41]. This framework is used for the optimisation of hyperparameters. This means that Talos enables a random parameter search by applying a correlation method. The corresponding performance metric for model optimisation represents the area under the ROC curve (AUC). The parameter set that was selected for optimisation can be observed in Table 3, as shown in the previous work in [14].

In this work, most of the parameters of interest are defined with the Keras framework [42]. In this DL framework, the "layer reduction" parameter defines how the number of neurons changes after each layer. This means that the number of encoder neurons is reduced or decoder neurons is increased (value: True). In addition, the number of neurons can remain unchanged (value: False). Some of the parameters are applied only for

some autoencoder variants. For example, the "number of convolutional layers" is only used in AE-CNN. Other parameters were predefined to meet specific performance conditions. Thus, they are obtained from personal experience after conducting some initial tests during the so-called exploration phase. For example, for one-dimensional convolution layers, the filter size is set to 16. However, if "layer reduction" is set to False, then the neuron number equals the input feature size s. Otherwise, the number of neurons for the i layer equals the following equation:

$$n_i = \begin{cases} s & \text{for } i = 0 \\ s - (s - s_{lat}) \frac{i}{(n_{layer}-1)} & \text{for } i < n_{layer} \end{cases} \quad (2)$$

where:

n_i = number of neurons at the layer i ;
s = input feature vector size;
s_{lat} = latent (bottleneck) vector size;
n_{layer} = number of layers.

Table 3. Parameter used for model optimisation.

Hyperparameter	Values
Learning rate	0.01, 0.001
Batch size	64, 128
Epochs	100, 600
Layer red.	True, False
Dropout rate	0.0, 0.3
Optimizer	Adam, Adadelta, Adamax
Activation	elu, selu, relu, tanh, sigmoid
Loss function	mse, custom
Initializer	he_normal (he_n), lecun_normal (lecun_n), glorot_normal (glorot_n), lecun_uniform (lecun_n)
Regularizer	L2, L1, None
Hidden neurons	10, 12, 14, 16, 18
Dense layers	1, 2, 3, 4, 5
Conv. layers	0, 1, 2, 3, 4, 5
CNN kernel size	3, 5

The resulting optimised parameters for each autoencoder algorithm are shown in Table 4.

Table 4. Optimised parameters with hyperparameter optimisation.

Type/Value	AE	AE-CNN	VAE	VAE-Prob	AEC
Activation	selu	relu	relu	relu	relu
Batch size	64	128	128	524	64
Dropout rate	0.3	0.3	0	0	0
Epochs	100	100	100	100	100
Initializer	lecun_n	lecun_n	lecun_n	lecun_n	he_n
Regularizer	l1	l2	l1	l1	None
CNN kernel size	-	3	-	-	-
Hidden neurons	10	14	14	16	10
Layer red.	True	True	False	True	True
Learning rate	0.01	0.001	0.01	0.01	0.001
Loss function	mse	mse	mse	mse	custom
Conv. layers	-	2	-	-	-
Dense layers	5	2	4	3	4
Optimizer	Adam	Adamax	Adamax	Adadelta	Adamax

5. Evaluation

5.1. Evaluation Metrics

In order to evaluate the applied ML methods in our experiments, we rely on existing performance metrics from this domain. Therefore, we rely on the ROC (receiver operating characteristic) curve for the subsequent analysis. This graphical plot depicts the performance of anomaly detection based on multiple threshold values. However, the respective overall performance measurement across all of ROC's threshold settings represents AUC-ROC (area under the ROC) curve. This metric is applied in ML for checking the performance of the classification model, as required in this paper. In general, the ROC curve depicts probabilities, whereas AUC measures how the ML model distinguishes between different classes. The higher the AUC, the better the model performs this task.

In addition to ROC and AUC, the following metrics derived from the individual threshold values and based on the confusion matrix are commonly considered for evaluation: balanced accuracy, precision, recall and f_1 score [43,44].

In practice, the basic evaluation measures (confusion matrix) represent the absolute number of correctly recognized anomalies (TP—true positives), the number of correctly recognized benign data (TN—true negatives), benign data wrongly recognized by the algorithm as anomalies (FP—false positives) and the wrongly overlooked by the algorithm anomalies (FN—false negatives). In order to compare data sets of different sizes, they are normalized according to the dataset size, i.e., the amount of benign data or anomalies (TP rate or TPR, TN rate or TNR). Subsequently, derived parameter values in the evaluation are accuracy (acc), balanced accuracy (balAcc), precision (prec or PPV), recall (rec, equal to TPR) and f_1 score, as common in the literature.

The f_1 score, the most commonly used evaluation metric in similar applications, represents an evaluation measure between "recall" and "precision" in the form of a single number. The respective value can be further used depending on the area of application and desired statement. This equation is defined in the following manner:

$$f_1 = \frac{2 \times \mathrm{TP}}{2 \times \mathrm{TP} + \mathrm{FP} + \mathrm{FN}} \tag{3}$$

Balanced accuracy (balAcc), in contrast to "accuracy", is particularly suitable for the evaluation of unbalanced datasets—datasets that exhibit large differences in the amount

of positive and negative data, i.e., anomalies and benign data. It is defined as the average value of TPR and TNR. The equation for "balanced accuracy" is given as follows:

$$balAcc = \frac{\text{TPR} + \text{TNR}}{2} \tag{4}$$

5.2. Evaluation–Independent Test Dataset

This section discusses the obtained evaluation results for the detection of anomalies in the independent test dataset, as shown in Table 5 and 6. In total, this dataset contains 1,637,230 entries, out of which 1,334,003 entries represent benign data and 303,227 are attack traces. The corresponding contamination rate equals 18.52%.

Table 5. Resulting confusion matrix for the detection of anomalies in the independent test dataset.

Method	Cont. Rate	TP	TN	FP	FN	TPR	TNR
AE		262,865	1,165,452	168,551	40,362	86.7%	87.4%
AEC		278,736	1,305,643	28,360	24,491	91.9%	97.9%
AE-CNN		189,636	1,291,406	42,597	113,591	62.5%	96.8%
VAE	18.52%	215,020	1,210,259	123,744	88,207	70.9%	90.7%
VAE-Prob		268,052	1,081,114	252,889	35,175	88.4%	81.0%
AE+VAE-Prob		266,852	1,155,967	178,036	36,375	88.0%	86.7%
OCSVM		203,878	1,240,254	93,749	99,349	67.2%	93.0%

Table 6. Results for the detection of anomalies in the independent test dataset.

Method	Balanced Accuracy	Accuracy	Precision	Recall	f_1 Value	Running Time (s)
AE	0.870	0.872	0.609	0.867	0.716	9.56
AEC	0.949	0.968	0.908	0.919	0.913	4.56
AE-CNN	0.797	0.905	0.817	0.625	0.708	14.17
VAE	0.808	0.871	0.635	0.709	0.670	243.04
VAE-Prob	0.847	0.824	0.515	0.884	0.650	286.31
AE+VAE-Prob	0.873	0.869	0.600	0.880	0.713	93.25
OCSVM	0.801	0.882	0.685	0.672	0.679	44,305.81

With respect to the TP and TN rates, all algorithms exhibit good and practically useful results. However, AEC stands out as the theoretically best possible result. Unfortunately, this is only the case for unrealistic and practically irrelevant scenarios. In addition, AE-CNN and VAE achieve a slight improvement compared to the implemented standard ML method (OCSVM). As expected, the results of the purely autoencoder-based methods are below those of the variation autoencoder in terms of runtime. In any case, the OCSVM can be attested to have a "negative" runtime performance.

It is important to mention that all values relate to a very special, selected working point during the process. All such points were determined during training in the course of the validation. A better assessment of the performance behaviour can be obtained from ROC curves, which are shown in Figure 4. All relevant TN and TP rates are plotted on the x and y axis, respectively. They include associated rates for different algorithms, shown in color. The red dots mark the optimal threshold values determined in the course of the training, whereas the black triangles show the position of the optimal threshold values with regard to the independent test data. The dotted line represents the performance of a "random" anomaly detector. The depiction demonstrates the superb efficiency of the normal autoencoder AE (blue), sampling-based autoencoder—VAE (red)—and the probabilistic autoencoder variant VAE-Prob (violet), compared to the standard OCSVM (brown). The same behaviour, however, is not pronounced for the convolution-based autoencoder AE-CNN (green) and the clearly superior, practically unusable—but theoretically best possible—autoencoder with custom loss-AEC (orange).

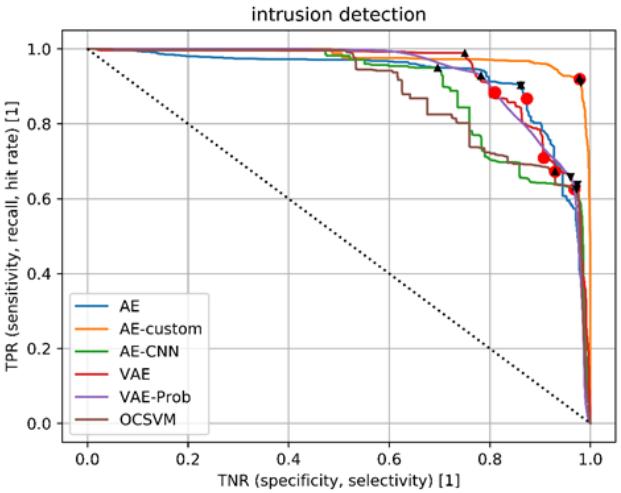

Figure 4. ROC curves for the tested algorithms on the independent test dataset.

Curiously, another interesting assumption can also be derived from the representation. The combined, two-stage approach—although very attractive in terms of the runtime performance (see Table 6)—cannot fully exploit the qualitative performance advantages in comparison to the normal autoencoder in the scenario examined. The reason, therefore, lies in the (coincidentally) very good agreement between the threshold value, as determined during training (red dot), and the "optimal" threshold value (triangle pointing upwards) for the AE (blue line), as shown in Figure 4. However, this behaviour cannot be observed for most other algorithms. Thus, it can be assumed that, in practice—especially under changing boundary conditions in the observed network—an adaptation of the threshold values at runtime is necessary and extremely useful. This should be performed from time to time, e.g., by experts, and based on actually observed anomalies.

In any case, the latter is also evident from the analyses of the determination of threshold values. These values are optimised for certain quality measures in relation to the threshold values (red dots), which are optimised during the course of the training. An exemplary analysis for the sampling-based variation autoencoder (VAE) is shown in Figure 5. As can be seen, an adjustment of the threshold value (determined during training) with regard to a slightly lower TNR can lead to significantly better performance (balanced accuracy). This means that the TNR implies only a slight increase in incorrectly reported anomalies.

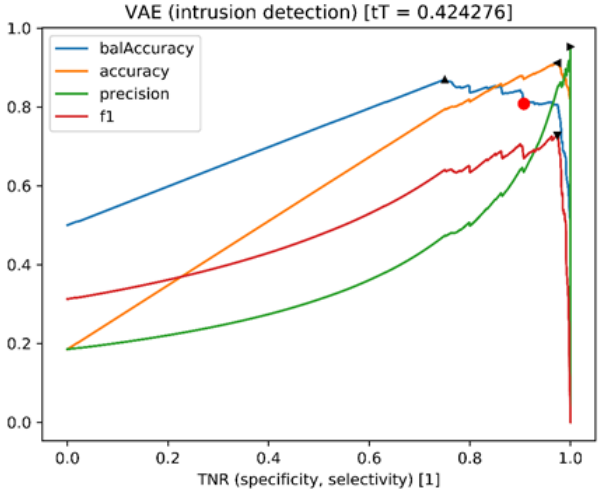

Figure 5. Analysis of the sampling-based variation autoencoder (VAE).

It should be noted that the above observation should not apply to the probabilistic autoencoder variant (VAE-Prob). In fact, it is shown that the threshold value (red dot) optimised during the course of the training is close to the actual optimum for the balanced accuracy (black, upward-pointing triangle), as shown in Figure 6. The user also benefits from the behaviour of the two-stage approach (AE+VAE-Prob) (Table 5 and 6). In fact, the precise analysis of "suspicious" data filtered out during the first stage remains independent of the choice of the threshold value for different scenarios.

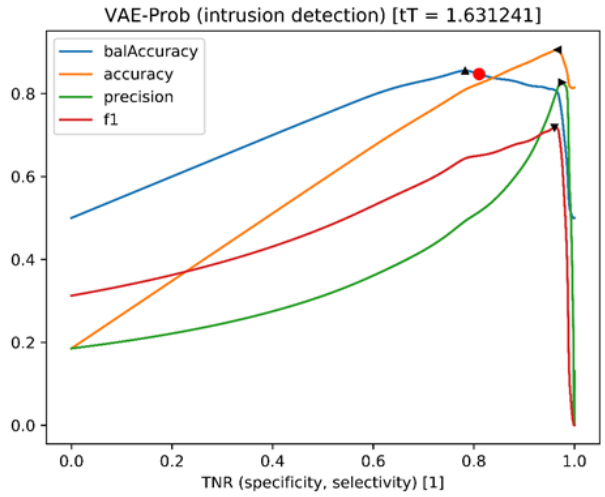

Figure 6. Results for the probability-based variational autoencoder.

5.3. Evaluation—Zero-Day-Attacks Test Dataset

An important challenge for ML algorithms constitutes the evaluation of methods for scenarios with unknown attacks. These "zero-day attacks" do not correspond to trained application scenarios in boundary conditions. Analogous to the evaluations for the independent test dataset, Table 7 and 8 show numerical results analogous to the measures and metrics, as defined above. It should be noted that this dataset contains 1,593,831 entries, out of which 1,276,109 entries constitute benign data with 317,722 attack traces. In addition, the contamination rate equals 19.93% in this case.

Table 7. Resulting confusion matrix for the detection of anomalies in the zero-day-attacks test dataset.

Method	Cont. Rate	TP	TN	FP	FN	TPR	TNR
AE		94,301	1,143,117	132,992	223,421	29.7%	89.6%
AEC		81,252	1,236,751	39,358	236,470	25.6%	96.9%
AE-CNN		80,846	1,221,599	54,510	236,876	25.4%	95.7%
VAE	19.93%	89,072	1,143,910	132,199	228,650	28.0%	89.6%
VAE-Prob		133,394	1,049,943	226,166	184,328	42.0%	82.3%
AE+VAE-Prob		133,182	1,137,382	138,727	184,540	41.9%	89.1%
OCSVM		82,771	1,169,204	106,905	234,951	26.1%	91.6%

Table 8. Results for the detection of anomalies in the zero-day-attacks test dataset.

Method	Balanced Accuracy	Accuracy	Precision	Recall	f_1 Value	Running Time (s)
AE	0.596	0.776	0.415	0.297	0.346	10.28
AEC	0.612	0.827	0.674	0.256	0.371	4.56
AE-CNN	0.606	0.817	0.597	0.254	0.357	14.13
VAE	0.588	0.774	0.403	0.280	0.331	288.29
VAE-Prob	0.621	0.742	0.371	0.420	0.394	250.70
AE+VAE-Prob	0.655	0.797	0.490	0.419	0.452	79.10
OCSVM	0.588	0.786	0.436	0.261	0.326	43,237.95

In the case of the confusion matrix, it is noticeable that, with threshold values determined in the training, consistently similar TNRs are achieved as in the evaluation of the independent test data set. However, the TPRs are significantly lower, in some cases in the range of 20–30%. This can also be observed in the reference approach of classic ML. The same conclusion applies for the best possible autoencoder in unrealistic scenarios (AEC). Thus, it can be claimed that this dataset is characterized by significantly increased complexity and difficulty. The most remarkable result is the excellent performance of the probabilistic variation autoencoder (VAE-Prob) with a TPR of over 40%. The two-stage approach (AE+VAE) exhibits a comparable performance as well. Both observations can be regarded as remarkable results, especially when considering the significantly reduced runtimes (see final column in Table 8). Similar outcomes are also achieved for f_1 values in the context of AE-VAE. In addition, the runtime of OCSVM is extremely high in this case. This means that it is analogous to the results for the independent test data sets. Another overview of the results provides the receiver operating curve (ROC) in Figure 7. This representation depicts the obtained threshold values for each algorithm. The red dots mark the optimal threshold values that are determined during the training. On the other hand, the black triangles indicate the position of optimal threshold values with regard to zero-day attacks.

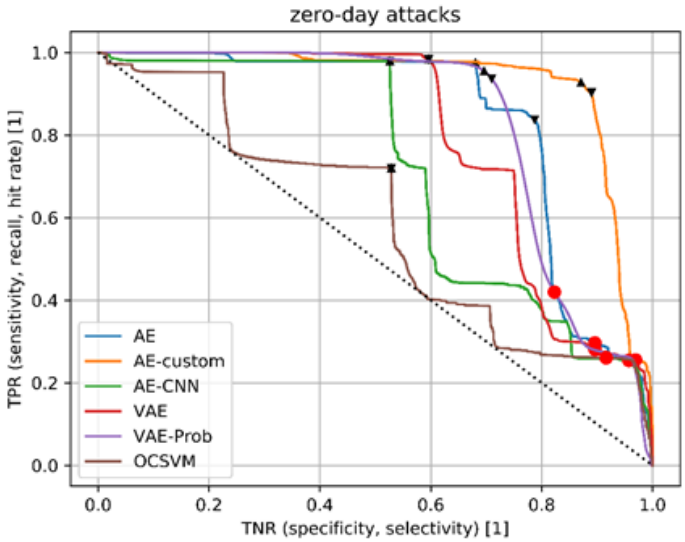

Figure 7. ROC curves for the detection of unknown zero-day attacks.

It should be noted that OCSVM does not achieve any meaningful results in this case. In addition, the results are even totally random with some threshold settings (brown curve). The autoencoder with custom loss AEC (orange), which cannot be used in practical operation despite its theoretical efficiency, also performs significantly worse than in the previous evaluation for the independent test data set.

A qualitatively similar conclusion emerges for the other algorithms as well. Whereas AE-CNN performs in the weakest manner, AE and VAE achieve better results. However, it is noticeable that all optimised threshold values from training (red dots) can be found in a rather low TPR range. Here, too, corresponding adaptations and adjustments to threshold values can lead to significantly better detection rates in this scenario. This includes operators based on previous knowledge that are limited by the number of possible false detections—FPs. This claim is also strengthened by the analysis of the values (red dots) that are optimised in the course of the training. This is achieved with regard to the theoretically "optimal" threshold values for zero-day attacks (black triangles), which applies different metrics from the "normal autoencoder", as shown in Figure 8. A significantly higher balanced accuracy is also be possible, but the operating point must be shifted towards a slightly lower TNR.

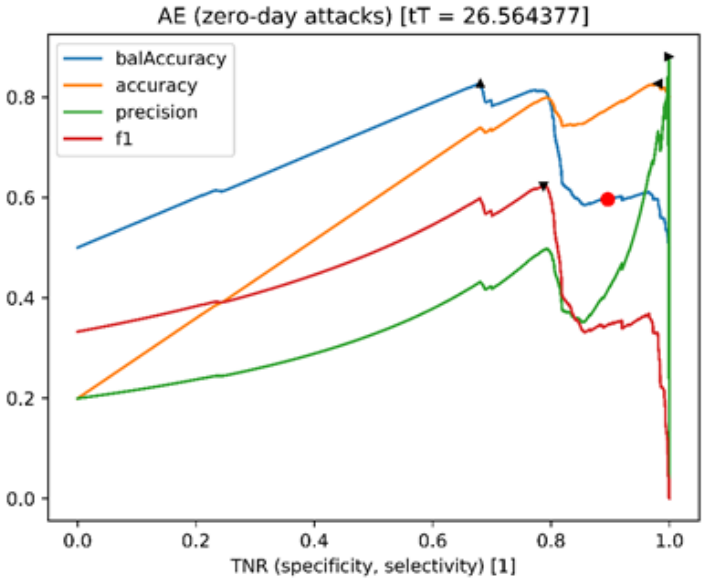

Figure 8. Optimised threshold values for zero-day attacks.

6. Discussion

The majority of existing research works on anomaly detection in networks focus on the application of intrusion detection. However, this represents only the initial step in the bigger picture of attack detection. Thus, the multi-stage process of detecting APT attacks has been mostly ignored so far. In general, the latter represents a critical matter for further practical applications of anomaly detection. As already mentioned in this paper, one technical obstacle represents the absence of benchmark data sets. For this reason, project-specific training and test data are proposed in this work. Basically, both test datasets (independent and zero-day-attacks datasets) are derived with the help of available statistical and feature extraction methods from two existing cybersecurity datasets—CICIDS2017 and Contagio. Whereas CICIDS2017 encompasses benchmark data for a general (one-stage) detection of cyberattacks, Contagio considers APT attacks as well. In addition, the latter includes a set of concrete attack examples that are obtained through so-called honey pots. Both test datasets were subjected to a descriptive and explorative statistical analysis prior to the actual anomaly detection. The subsequent investigation provided information on the plausibility of features, including the ones that contain deliberate data errors and constant values. In addition to that, histograms and boxplots were used to visualise the statistical distribution of features. In addition, these representations reveal an explorative comparison of benign data and data that contains APT attacks. In such way, an initial impression is given of the relevance of features for anomaly detection.

Additionally, within the framework of correlation analysis and visualisation, groups of correlated and identical features were identified. In this way, entire feature groups in datasets can be partially ignored without the expected loss of information. Thus, feature complexity was reduced and the training speed was increased. In fact, the latter observation denotes a faster convergence of the corresponding ML model. Finally, a principal component analysis (PCA) made it possible to estimate the extent of possible dimensional reductions in the features in question.

As already mentioned, the implementation of the two-stage anomaly detector for intrusion detection is presented in our previous work in [14]. The proposed approach combines the benefits of fast methods and their moderate detection quality with slow methods, i.e., fast detection quality, respectively. In this way, large amounts of data were analysed and dynamically adapted to practical challenges. In particular, the AE-CNN method is well-suited to the initial filtering phase. For example, it allows for a reduction by 82% of analysed network data in cases where more undetected cyberattacks (up to

10%) are accepted. The combination with VAE or VAE-Prob during the next phase can reduce the calculation time, when compared to the one-stage approach. Another effect of the two-stage approach is an increase in precision value. This is due to the additional processing by using two methods.

7. Conclusions and Future Work

In this paper, the analysed datasets relate to multiple independent application scenarios, and also include APT-attack traces (derived from Contagio and merged with CICIDS2017). For example, the evaluation of the combined CICIDS2017/Contagio dataset shows that variational autoencoder approaches (VAE, VAE-Prob) achieve better results than normal autoencoders. However, all the inspected algorithms exhibit satisfying results, especially with regard to the practically relevant TP and TN rates. On the other hand, new methods in the scenario with "completely unknown", i.e., zero-day, attacks exhibit similar TN rates, but with lower TP rates. Similar results are also observed when applying the reference approach of unrealistic scenarios from classical ML. The reason for this, therefore, is the increased dataset complexity, which does not represent a problem of the procedure itself. For both datasets, it can be concluded that the running time of the pure autoencoder methods is slower than that of the variational counterpart. The best performance is achieved with VAE-Prob, which is independent of the selected threshold value. In addition, the two-stage approach exploits these advantages without losses in the runtime. Nevertheless, it can be assumed that a further adaptation of threshold values at runtime would improve the results. This is especially the case with changing boundary conditions, which should be checked on a regular basis.

A major conclusion of the work proposed in this paper is that unsupervised machine-learning methods can be successfully used to detect advanced cyberattacks targeting network infrastructures. Even in a case of a new attack type with a pattern unknown during the training and validation phases, trained models could be used successfully without a need for re-training. Beneficial improvements in this case would include developing methods for a threshold adaptation.

Given all the results and observations, our future work will include the automation of threshold selection and consider practical applications. It will also include the testing of simulation environments and deriving new and comprehensive datasets for APT-attack detection evaluation.

Author Contributions: Conceptualization, B.S., K.H.-S., H.N. and M.W.; methodology, B.S., K.H.-S., H.N., M.W. and U.K.; software, H.N., M.W., K.H.-S., J.B. and U.K.; validation, H.N., M.W. and B.S.; investigation, H.N., M.W., B.S., K.H.-S., J.B. and U.K.; data curation, K.H.-S., J.B. and U.K.; writing—original draft preparation, H.N., M.W., B.S., J.B. and U.K.; writing—review and editing, B.S., H.N. and M.W.; visualisation, J.B., H.N., M.W. and U.K.; project administration, K.H.-S. and B.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the Austrian Federal Ministry of Climate Action, Environment, Energy, Mobility, Innovation and Technology (BMK) under the project SecFIT (Design and Runtime Security for Internet of Things).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: We would like to express our gratitude to Konstantin Böttinger (Fraunhofer AISEC) for his constructive project review and valuable insights.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analysis, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Alshamrani, A.; Myneni, S.; Chowdhary, A.; Huang, D. A Survey on Advanced Persistent Threats: Techniques, Solutions, Challenges, and Research Opportunities. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 1851–1877. [CrossRef]
2. Schneider, P.; Böttinger, K. High-Performance Unsupervised Anomaly Detection for Cyber-Physical System Networks. In Proceedings of the CPS-SPC@CCS, Toronto, ON, Canada, 19 October 2018.
3. Chen, J.; Sathe, S.; Aggarwal, C.; Turaga, D. Outlier detection with autoencoder ensembles. In Proceedings of the 2017 SIAM International Conference on Data Mining, Houston, TX, USA, 27–29 April 2017; pp. 90–98.
4. Ravi Kiran, M.T.; Parakkal, R. An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos. *arXiv* **2018**, arXiv:1801.03149.
5. Kwon, D.; Kim, H.; Kim, J.; Suh, S.C.; Kim, I.; Kim, K. A survey of deep learning-based network anomaly detection. *Clust. Comput.* **2017**, *22*, 949–961. [CrossRef]
6. Hodo, E.; Bellekens, X.; Hamilton, A.; Tachtatzis, C.; Atkinson, R. Shallow and deep networks intrusion detection system: A taxonomy and survey. *arXiv* **2017**, arXiv:1701.02145.
7. Javaid, A.; Niyaz, Q.; Sun, W.; Alam, M. A deep learning approach for network intrusion detection system. In Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS), New York, NY, USA, 3–5 December 2016; pp. 21–26.
8. Yüksel, Ö.; den Hartog, J.; Etalle, S. Reading between the fields: Practical, effective intrusion detection for industrial control systems. In Proceedings of the 31st Annual ACM Symposium on Applied Computing, Pisa, Italy, 4–8 April 2016; pp. 2063–2070.
9. Duessel, P.; Gehl, C.; Flegel, U.; Dietrich, S.; Meier, M. Detecting zero-day attacks using context-aware anomaly detection at the application-layer. *Int. J. Inf. Secur.* **2017**, *16*, 475–490.
10. Fraley, J.B.; Cannady, J. The promise of machine learning in cybersecurity. In Proceedings of the SoutheastCon, Charlotte, NC, USA, 30 March–2 April 2017; pp. 1–6.
11. Tuor, A.; Kaplan, S.; Hutchinson, B.; Nichols, N.; Robinson, S. Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. *arXiv* **2017**, arXiv:1710.00811.
12. Xin, Y.; Kong, L.; Liu, Z.; Chen, Y.; Li, Y.; Zhu, H.; Gao, M.; Hou, H.; Wang, C. Machine Learning and Deep Learning Methods for Cybersecurity. *IEEE Access* **2018**, *6*, 35365–35381. [CrossRef]
13. Ghafir, I.; Hammoudeh, M.; Prenosil, V.; Han, L.; Hegarty, R.; Rabie, K.; Aparicio-Navarro, F.J. Detection of advanced persistent threat using machine-learning correlation analysis. *Future Gener. Comput. Syst.* **2018**, *89*, 349–359. [CrossRef]
14. Neuschmied, H.; Winter, M.; Hofer-Schmitz, K.; Stojanović, B.; Kleb, U. Two Stage Anomaly Detection for Network Intrusion Detection. In Proceedings of the 7th International Conference on Information Systems Security and Privacy (ICISSP), Online, 11–13 February 2021.
15. Hofer-Schmitz, K.; Kleb, U.; Stojanović, B. The Influences of Feature Sets on the Detection of Advanced Persistent Threats. *Electronics* **2021**, *10*, 704. [CrossRef]
16. Siniosoglou, I.; Radoglou-Grammatikis, P.; Efstathopoulos, G.; Fouliras, P.; Sarigiannidis, P. A Unified Deep Learning Anomaly Detection and Classification Approach for Smart Grid Environments. *IEEE Trans. Netw. Serv. Manag.* **2021**, *18*, 1137–1151. [CrossRef]
17. Cho, D.X.; Nam, H.H. A Method of Monitoring and Detecting APT Attacks Based on Unknown Domains. *Procedia Comput. Sci.* **2019**, *150*, 316–323. [CrossRef]
18. Sai Charan, P.V.; Mohan Anand, P.; Shukla, S.K. DMAPT: Study of Data Mining and Machine Learning Techniques in Advanced Persistent Threat Attribution and Detection. In *Data Mining Concepts and Applications*; IntechOpen: London, UK, 2021.
19. Bodström, T.; Hämäläinen, T. State of the Art Literature Review on Network Anomaly Detection with Deep Learning. In *Internet of Things, Smart Spaces, and Next Generation Networks and Systems, NEW2AN ruSMART 2018*; Galinina, O., Andreev, S., Balandin, S., Koucheryavy, Y., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2018.
20. Myneni, S.; Chowdhary, A.; Sabur, A.; Sengupta, S.; Agrawal, G.; Huang, D.; Kang, M. DAPT 2020 - Constructing a Benchmark Dataset for Advanced Persistent Threats. In Proceedings of the International Workshop on Deployable Machine Learning for Security Defense (MLHat), San Diego, CA, USA, 24 August 2020.
21. Alrehaili, M.; Alshamrani, A.; Eshmawi, A. A Hybrid Deep Learning Approach for Advanced Persistent Threat Attack Detection. In Proceedings of the 5th International Conference on Future Networks & Distributed Systems (ICFNDS), Dubai, United Arab Emirates, 15–16 December 2021.
22. Bodström, T.; Hämäläinen, T. A Novel Deep Learning Stack for APT Detection. *Appl. Sci.* **2019**, *9*, 1055. [CrossRef]
23. Navarro, J.; Deruyver, A.; Parrend, P. A systematic survey on multi-step attack detection. *Comput. Secur.* **2018**, *76*, 214–249. [CrossRef]
24. Atapour, C.; Agrafiotis, I.; Creese, S. Modeling Advanced Persistent Threats to enhance anomaly detection techniques. *J. Wirel. Mob. Netw. Ubiquitous Comput. Dependable Appl.* **2019**, *9*, 71–102.
25. Milajerdi, S.; Gjomemo, R.; Eshete, B.; Sekar, R.; Venkatakrishnan, V. HOLMES: Real-time APT Detection through Correlation of Suspicious Information Flows. In Proceedings of the 2019 IEEE Symposium on Security and Privacy, San Francisco, CA, USA, 20–22 May 2019.
26. Yu, K.; Tan, L.; Mumtaz, S.; Al-Rubaye, S.; Al-Dulaimi, A.; Bashir, A.K.; Khan, F.A. Securing Critical Infrastructures: Deep-Learning-Based Threat Detection in IIoT. *IEEE Commun. Mag.* **2021**, *59*, 76–82. [CrossRef]

27. Xiong, C.; Zhu, T.; Dong, W.; Ruan, L.; Yang, R.; Cheng, Y.; Chen, Y.; Cheng, S.; Chen, X. Conan: A Practical Real-Time APT Detection System With High Accuracy and Efficiency. *IEEE Trans. Dependable Secur. Comput.* **2022**, *19*, 551–565. [CrossRef]
28. Wang, F.; Li, R.; Zhang, Z. APTSID: An Ensemble Learning Method for APT Attack Stage Identification. In Proceedings of the 5th Asian Conference on Artificial Intelligence Technology (ACAIT), Haikou, China, 29–31 October 2021.
29. Xuan, C.D. Detecting APT Attacks Based on Network Traffic Using Machine Learning. *J. Web Eng.* **2021**, *20*, 71–190. [CrossRef]
30. Singh, S.; Sharma, P.K.; Moon, S.Y.; Moon, D.; Park, J.H. A comprehensive study on APT attacks and countermeasures for future networks and communications: Challenges and solutions. *J. Supercomput.* **2019**, *75*, 4543–4574. [CrossRef]
31. Quintero-Bonilla, S.; del Rey, A.M. A New Proposal on the Advanced Persistent Threat: A Survey. *Appl. Sci.* **2020**, *10*, 3874. [CrossRef]
32. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP), Funchal, Madeira, Portugal, 22–24 January 2018.
33. 1998 DARPA Intrusion Detection Evaluation Dataset. Available online: https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset (accessed on 11 April 2022).
34. KDD Cup 1999 Data. Available online: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (accessed on 11 April 2022).
35. HackerEarth DEFCON. Available online: https://www.kaggle.com/datasets/seraphwedd18/hackerearth-defcon (accessed on 11 April 2022).
36. CAIDA Data-Completed Datasets. Available online: https://www.caida.org/catalog/datasets/completed-datasets/ (accessed on 11 April 2022).
37. Contagio. Available online: http://contagiodump.blogspot.com/ (accessed on 19 April 2022).
38. Intrusion Detection Evaluation Dataset (CIC-IDS2017). Available online: https://www.unb.ca/cic/datasets/ids-2017.html (accessed on 19 April 2022).
39. CICFlowMeter. Available online: https://github.com/ahlashkari/CICFlowMeter (accessed on 25 April 2022).
40. An, J.; Cho, S. Variational autoencoder based anomaly detection using reconstruction probability. *Spec. Lect. IE* **2015**, *2*, 1–18.
41. Talos. Available online: https://github.com/autonomio/talos (accessed on 28 June 2022).
42. Keras: The Python deep learning API. Available online: https://keras.io/ (accessed on 29 June 2022).
43. Baddar, S.W.A.H.; Merlo, A.; Migliardi, M. Anomaly Detection in Computer Networks: A State-of-the-Art Review. *JoWUA* **2014**, *5*, 29–64.
44. Hindy, H.; Brosset, D.; Bayne, E.; Seeam, A.; Tachtatzis, C.; Atkinson, R.; Bellekens, X. A Taxonomy and Survey of Intrusion Detection SystemDesign Techniques, Network Threats and Datasets. *arXiv* **2018**, arXiv:1806.03517.