Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Review

A review of Machine Learning-based zero-day attack detection: Challenges and future directions

Yang Guo

NIST, Gaithersburg, MD 20899, United States of America

ARTICLE INFO

Keywords:
Zero-day attacks
Attack detection
Machine Learning

ABSTRACT

Zero-day attacks exploit unknown vulnerabilities so as to avoid being detected by cybersecurity detection tools. The studies (Bilge and Dumitraş, 2012, Google, 0000, Ponemon Sullivan Privacy Report, 2020) show that zero-day attacks are wide spread and are one of the major threats to computer security. The traditional signature-based detection method is not effective in detecting zero-day attacks as the signatures of zero-day attacks are typically not available beforehand. Machine Learning (ML)-based detection method is capable of capturing attacks' statistical characteristics and is, hence, promising for zero-day attack detection. In this survey paper, a comprehensive review of ML-based zero-day attack detection approaches is conducted, and their ML models, training and testing data sets used, and evaluation results are compared. While significant efforts have been put forth to develop accurate and robust zero-attack detection tools, the existing methods fall short in accuracy, recall, and uniformity against different types of zero-day attacks. Major challenges toward the ML-based methods are identified and future research directions are recommended at last.

Contents

1. Introduction ... 175
2. Outlier-based zero-day attack detection using normal data ... 176
2.1. One-class SVM-based detection ... 176
2.2. Autoencoder-based detection ... 177
2.3. Zero-day attack detection performance comparison: One-class SVM vs. Autoencoder ... 177
2.4. Ensemble of autoencoders for zero-day attack detection ... 177
3. Supervised and hybrid learning-based zero-day attack detection using labeled data ... 178
3.1. Evaluation of supervised machine learning classifiers for zero-day attack detection ... 178
3.2. Integrating supervised and unsupervised learning for zero-day malware detection ... 178
3.3. Hybrid learning using available unlabeled data for zero-day malware detection ... 179
3.4. Generative adversarial networks (GAN)-based zero-day malware detection ... 180
4. Transfer learning (TL)-based zero-day attack detection ... 180
4.1. Feature-based transfer learning for zero-day network attack detection using spectral transformation ... 181
4.2. Domain adaptation-based zero-day attack detection using manifold alignment ... 182
5. Comparisons, challenges and future directions ... 182
5.1. Future directions ... 183
6. Conclusions ... 184
Declaration of competing interest ... 184
Data availability ... 184
Acknowledgment ... 184
References ... 184

1. Introduction

A zero-day attack is a new cyber-attack that is unknown to the public and cybersecurity community, hence the name "zero-day". It exploits vulnerabilities that have not been disclosed publicly or employs novel attacking tactics to avoid being detected by existing detection tools. Attackers can, thus, attack targets of their choosing while remaining unrecognized. Bilge and Dumitras studied the duration and

E-mail address: yang.guo@nist.gov.

https://doi.org/10.1016/j.comcom.2022.11.001
Received 22 July 2022; Received in revised form 1 November 2022; Accepted 2 November 2022
Available online 25 November 2022
0140-3664/Published by Elsevier B.V.

prevalence of zero-day attacks in [1]. Their results show that a typical zero-day attack lasts 312 days on average before being detected, and the volume of attacks continues to increase by up to five orders of magnitude after vulnerabilities are disclosed publicly. Google's Project Zero [2] publishes their tracking records for publicly known cases of detected zero-day attacks. On average, an ''in the wild" zero-day attack is discovered every 17 days, and it takes 15 days on average to develop a patch for the vulnerability that is being exploited by active attacks. A recent study conducted by Ponemon Institute [3] concludes that 80% of security breaches are derived from zero-day attacks, with the average incurred cost amounting to 1.2 million dollars per attack. All evidences point to the severity of the threats created by the zero-day attacks.

Traditionally, cyberattack detection systems are broadly classified into two categories: signature-based detection systems and anomaly-based detection systems. The signature-based systems have a predefined repository of static signatures, or fingerprints, that represent known attacks. The detection is achieved by matching the incoming signature with an attack signature already in the repository. In contrast, the anomaly detection methods have a notion of normal activity and flag deviations from that profile. Both approaches have been studied extensively. In fact, signature-based detection systems have been successfully deployed in operational environments and have proven to be effective in detecting known attacks with high detection accuracy and recall. However, it is expensive to keep the signature library up to date, and the signature-based detection is prone to miss zero-day attacks with alarmingly low recall [4], which is expected as the zero-day attacks' signatures are typically not available in the repository.

Machine learning (ML) has become a promising technology for cyber-attack detection. Machine learning (ML) algorithms build a model based on sample data, known as training data, and are able to discover complex statistical patterns that help detect similar attacks and respond to attacks' changing behavior. ML-based technology carries the promise to be effective in detecting zero-day attacks. Extensive research has been conducted on using ML model for zero-day attack detection [5–13]. Various ML models, ranging from unsupervised ML, supervised ML, to Transfer Learning, are explored.

While ML is a powerful tool, the adoption of ML to zero-day attack detection faces several challenges. The availability of the training data is vital for ML. By definition, the zero-day attacks are not known until after an attack is discovered. The zero-day attack samples are, thus, not available in the datasets. One solution is to assume that the novel zero-day attacks are same or similar to the existing attacks. Hence, the model trained using the existing data can be used to identify zero-day attacks. This is an important assumption that needs to be validated.

Secondly, how to process the data and derive the feature vector, an n-dimensional vector of numerical features that captures the characteristics of attacks, is challenging. The design of feature vectors often requires the domain knowledge of the cybersecurity practitioners and is especially important in designing a zero-day attack ML model.

Finally, the evaluation of the ML-based zero-day attack detectors is difficult. The testing data for a true zero-day attack are often unavailable, and a comprehensive benchmark that provides fair and thorough comparison is lacking.

In the existing ML-based zero-day attack detection studies, the data used in both the training and the testing are limited. The evaluation results often fail to show that the proposed zero-day attack schemes provide satisfactory detecting precision and are uniformly effective against different types of zero-day attacks. While multiple surveys have been conducted on intrusion detection and malware detection using various approaches [14–18], none has been focused on ML-based zero-day attack detection schemes. A comprehensive review of existing ML-based models so as to compare and contrast their pros and cons and identify the key design and evaluation gaps is highly desired.

In this paper, we conduct such a review, followed by a thorough comparison in terms of types of ML models, training data, zero-day attack testing data, and evaluation results. Challenges and future directions are provided thereafter. The paper is organized as follows. In Section 2, the outlier-based zero-day attack detections are reviewed. In Section 3, the supervised and hybrid learning-based zero-day attack detection is reviewed. The Transfer Learning (TL)-based zero-day attack detection is included in Section 4. The comparisons, challenges, and future directions are included in Section 5, and conclusions in Section 6 wrap up the paper.

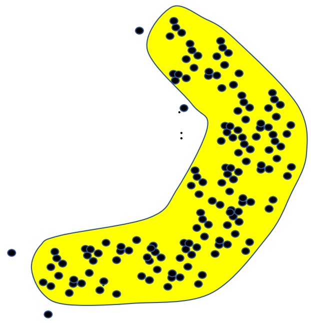

Fig. 1. One-class SVM decision boundary.

2. Outlier-based zero-day attack detection using normal data

Outlier-based zero-day attack detection trains the detection model using the normal data that have been observed in the past. The model detects zero-day attacks by assuming a zero-day attack differs significantly from the normal behavior. One-class Supportive Vector Machine (SVM) [19,20] and auto-encoder [21] are two representative Machine Learning models for outlier-based zero-day attack detection, which are explored in [5,6,10,22,23]. Outlier-based zero-day detection falls into unsupervised learning, a type of ML algorithms that learns pattern from unlabeled data.

2.1. One-class SVM-based detection

One-class Support Vector Machine (SVM) learn a tight and smooth boundary that encloses ''normal" data using non-linear kernels. An outlier is detected if the data fall outside of the decision boundary, as shown in Fig. 1.

Consider the normal data set $\{x_1, x_2, \dots, x_N\}$ where $x_i \in \mathcal{X}$ and $\mathcal{X}$ is the feature space. Let Φ be a feature mapping $\mathcal{X} \rightarrow \mathcal{H}$ that maps the normal data from their original feature space $\mathcal{X}$ to a new feature space $\mathcal{F}$. One-class SVM seeks to define a hyper-plane in feature space $\mathcal{F}$: $w \cdot \Phi(x) - \rho = 0$ such that the hyper-plane has the largest distance to the origin, while all mapped normal data $\Phi(x_i)$ lie at the opposite side of hyper-plane to the origin. For a new point x, its normalcy is determined by evaluating which side of the hyper-plane $\Phi(x)$ falls onto in the feature space $\mathcal{F}$. The new point is deemed to be ''abnormal" if it falls inside the hyper-plane.

The hyper-plane can be found by solving the following quadratic optimization program:

$$\min_{w,\xi,\rho} \quad \frac{1}{2}\|w\|^2 + \frac{1}{\upsilon N}\sum_{i=1}^{N} \xi_i - \rho \tag{1}$$

$$\text{subject to} \quad (w \cdot \Phi(x_i)) \geq \rho - \xi_i, \quad \xi \geq 0 \tag{2}$$

where υ is the regularization coefficient that trades off model complexity and training error, and ξ_i is the slack variable that enables One-class SVM to have margin to exclude some noisy training data. The decision function $f(x) = sgn(w \cdot \Phi(x) - \rho)$ will be positive for most samples x_i contained in the training set.

In practice, the mapping Φ is often not explicitly defined. Instead, one usually specifies kernel function (x_i, x_j), where $K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$. For example, Gaussian kernel $K(x_i, x_j) = \exp(-\|x_i - x_j\|^2/\sigma^2)$.

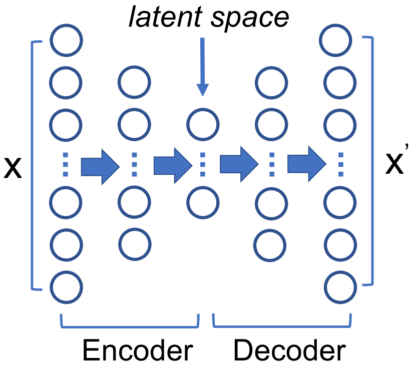

Fig. 2. Autoencoder architecture.

The optimization problem as defined in the Eqn(1) is usually solved in its dual form:

$$\max_{\alpha} \quad -\frac{1}{2}\sum_{i,j=1}^{N} \alpha_i \alpha_j K(x_i, x_j) \tag{3}$$

$$\text{subject to} \quad \sum_{i}^{N} \alpha_i = 1, \quad 0 \le \alpha_i \le \frac{1}{\upsilon N} \tag{4}$$

where $\{\alpha_i\}$ is the dual variable. With selected kernel function and its hyper-parameter υ, the above dual optimization problem can be solved as a quadratic programming problem. Having solved α_i, the value of ρ can be recovered via $\rho = \sum_j \alpha_j K(x_i, x_j)$ with x_i being any training data whose corresponding α_i satisfies $0 < \alpha_i < 1/\nu N$. The decision function is rewritten as $f(x) = \sum_{\alpha_i > 0} \alpha_i K(x, x_i) - \rho$. An incoming new data x is determined as an outlier if $f(x) < 0$.

2.2. Autoencoder-based detection

Autoencoder-based zero-day attack detection belongs to the so-called reconstruction methods with the assumption that the “normal" data can be reconstructed by an autoencoder with low reconstruction error, while the outliers, or abnormal ones, cannot. Autoencoder is a neural network that is trained to attempt to copy its input to its output [24]. It consists of two segments, an encoder and an decoder, as shown in Fig. 2.

The encoder function, denoted by ϕ, maps the original data, $x \in \mathcal{X}$, to a latent space $\mathcal{F}$, which is present at the bottleneck, i.e., $\phi : \mathcal{X} \rightarrow \mathcal{F}$. The decoder function, denoted by ψ, maps the latent space $\mathcal{F}$ at the bottleneck to the output space $\mathcal{X}$, $\psi : \mathcal{F} \rightarrow \mathcal{X}$. The output, in this case, is the same as the input function. Let x be the input to the autoencoder, and x' be its output. The loss function of the autoencoder is defined as $\mathcal{L}(x, x') = \|x - x'\|^2$. In other words, the autoencoder is trained to recreate the input after some generalized non-linear compression. The incoming new data x is determined as an outlier if $\|x - x'\|^2 > \delta$, where δ is pre-set threshold.

2.3. Zero-day attack detection performance comparison: One-class SVM vs. Autoencoder

- **Data sets**. In [5], the performance of One-Class SVM and autoencoder is compared. The experiments use two data sets: CIC-IDS2017 data set [25] and NSL-KDD data set [26]. Both data sets are widely used in network intrusion detection evaluations. The CIC-IDS2017 data set contains five-day pcap files that record the benign, insider and outsider attacks traffic. The dataset contains benign traffic, as well as SSH & FTP brute force attacks, DoS/DDoS, heartbleed, Web attacks, infiltration, portscan, and botnet attacks. The full CIC-IDS2017 description and analysis are available in [27]. The pcap files are pre-processed to generate bidirectional flows features. The features with high correlation are dropped in order to reduce its dimension.

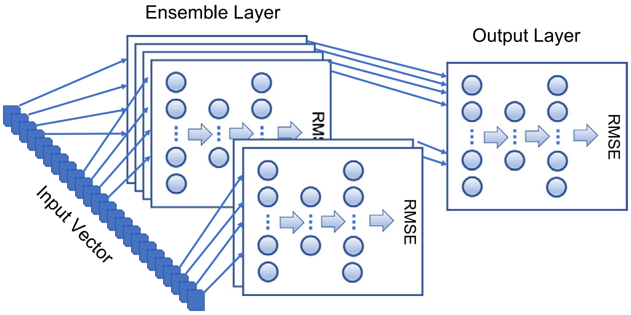

Fig. 3. Kitsune’s anomaly detection framework.

NSL-KDD contains network features extracted from a series of TCP connections captured from a local area network. The data set has 41 network features and 22 different types of attack, which can be grouped into four main categories: Denial of Service (DoS), probing, Remote to Local (R2L), and User to Root (U2R).

- **Model training**. The models are trained solely using benign traffic for both One-Class SVM and autoencoder models. Each of the attack classes mimics a zero-day attack and is used to assess the ability of the model to detect its abnormality

For hyper-parameter optimization in an autoencoder model, random search [28] is used in order to select the architecture of the network, number of epochs, and learning rate. The benign instances are split into 75% and 25% for training and validation, respectively. Once the model training converges, it is evaluated using both benign traffic and attack traffic. An attack instance is flagged as a zero-day attack if the Mean Squared Error (MSE) (reconstruction error) of the decoded (x') and the original instance (x) is larger than a given threshold.

In the One-Class SVM training, a ‘$\upsilon \in [0, 1]$’ value is specified, which is the lower and upper bound on the number of examples that are support vectors and that lie on the wrong side of the hyperplane [29]. The output of the One-Class SVM is a binary value that indicates if an instance is benign or a zero-day attack.

- **Experiment results.** The zero-day attack detection accuracy varies greatly according to the different attack types. For instance, the detection accuracy of the attacks that are very different from benign (for example, Hulk and DDoS), is high, in the rage of 92% to 99%; while the detection accuracy of attacks that are less distinguishable from benign traffic can be low, e.g., DoS-SlowHTTPTest’s detection accuracy is less than 40%. Second, while One-Class SVM is well suited for flagging recognizable zero-day attacks, autoencoders generally out-performs One-Class SVM for complex zero-day attacks as the performance rank is significantly higher. Finally, both of the models demonstrate low miss rate (false-positives).

2.4. Ensemble of autoencoders for zero-day attack detection

The authors in [6] employed an ensemble of Autoencoders for online network intrusion detection. They developed Kitsune, a plug and play network intrusion detection system that can learn to detect attacks on the local network without supervision and in an efficient online manner. Fig. 3 depicts the Kitsune’s core Machine Learning architecture with an ensemble of autoencoders to collectively differentiate between normal and abnormal traffic patterns. Kitsune has a feature extraction component that extracts n features of a network session under monitoring. The collected features are then clustered into k sub-instances, with one sub-instance for each autoencoder. The outputs of the ensemble-layer autoencoders are then fed into a output-layer autoencoder that generates the final score.

Assume that there are k ensemble-layer autoencoders with the input space of m features. Kitsune employs the agglomerative hierarchical clustering method to divide the n features into k clusters. The mapper ensures that each cluster has no more than *m* features and captures the normal behavior well enough to detect anomalous events occurring in the respective sub-instances.

• **Data sets and evaluation** Kitsune is evaluated using a IP camera video surveillance test-bed. The test-bed consists of two sets of HD surveillance cameras, with each set having four cameras. The cameras are connected to the Digital Video Recorder (DVR) via a VPN tunnel. The evaluation deploys the software tools in the network to generate four types, nine specific attacks, including OS Scan, Fuzzing, Video Injection, ARP MitM (ARP Man in the Middle), Active Wiretap, SSDP Flood, SYN DoS, SSL Renegotiation, and Mirai Botnet malware. When first installed, Kitsune assumes that all traffic is benign while in training mode. The online evaluation results are compared with the offline algorithms using Isolation Forests (IF) [30] and Gaussian Mixture Models (GMM) [31]. The offline algorithms have more information and supposes to perform better than the online algorithms. The true positive rate (recall), the false negative rate, the area under the receiver operating characteristic curve (AUC), and the equal error rate (EER) are used as the measurement metrics. The evaluations shows that Kitsune can detect various attacks with a performance comparable to offline anomaly detectors. The true positive rate varies greatly for different types of attacks. For instance, the detection is not effective for attacks such as ARP MitM, OS Scan, SYN DoS, SSL Renegotiation, and Video injection. The true positive rate is smaller than 30% when the False Positive Rate is set to be 0.001.

3. Supervised and hybrid learning-based zero-day attack detection using labeled data

Supervised learning is a type of machine learning technique that maps an input to an output based on example input–output pairs. It infers a mapping function from labeled training data consisting of a set of training examples. A zero-day attack is a novel cyber-attack that is unknown to the public. By definition, the labeled data of zero-day attack are not available. However, the assumption is made that the zero-day attack's feature vector is similar to that of the existing attacks [7,8,32,33], or that zero-day attacks can be modeled by adding noise to existing data [10]. With the help of known attacks' labeled data, supervised learning can be employed to detect zero-day attacks. Below we review some zero-day attack detection techniques that use supervised learning or hybrid learning, which is the combination of supervised and unsupervised learning.

3.1. Evaluation of supervised machine learning classifiers for zero-day attack detection

In [7], Zhou and Pezaros inspect the effectiveness of six popular and well known machine learning classification models for zero-Day attacks detection. They are Random forest classifier, Gaussian naive Bayes classifier, Decision tree classifier, Multi-layer Perceptron (MLP) classifier, K-nearest neighbors classifier, and Quadratic discriminant analysis classifier. Since these classifiers have been studied extensively, we will not introduce them in detail here.

The training data set is CSE-CIC-IDS2018 Data Set [34] (it is called CIC-AWS-2018 in the paper). The data set contains six different intrusion types, Brute-force, Botnet, DoS, DDoS, Web attacks, and infiltration of the network from inside, with a total of 14 different intrusions, namely, Botnet attack, FTP-BruteForce, SSH-BruteForce, BruteForce-Web, BruteForce-XSS, SQL Injection, DDoS-HOIC attack, DDoS-LOIC-UDP attack, DDoS-LOIC-HTTP attacks, Infiltration, DoS-Hulk attack, DoS-SlowHTTPTest attack, DoS-GoldenEye attack, and DoS-Slowloris attack. The data are collected from real networks. The benign data are collected for a week from a typical research network. The traffic include routine daily activities such as emailing, searching, news, video streaming, etc. All attacks and benign traffic are labeled and are used for training the detection models. The data set has 80 bi-directional flow features. To reduce the computation expense, the authors pre-processed the data and removed the features deemed to be "noisy", i.e., the features that does not show significant difference between the benign and the malicious traffic. The leftover 25 features are used in the detection (see Table 1 in [7]).

To simulate the zero-day attack, eight new attacks that are not included in the training CSE-CIC-IDS2018 data set are collected from real-life attacks. These zero-day attacks include Bitcoin miner, Drowor worm, nuclear ransomware, false content injection, ponmocup trojan, DDoS Bot'a Darkness, Google doc macadocs, and ZeroAccess (see Table 2 in [7]).

The evaluation consists of two steps. In the first step, six Machine Learning models are evaluated using the CSE-CIC-IDS2018 Data Set and the most effective model is selected. In the second step, the selected model is evaluated against the zero-day attacks. All attacks are treated as "malicious", and the models are trained to classify the traffic as either "benign" or "malicious".

Extensive cross validation is run on the labeled CSE-CIC-IDS2018 data to validate the fitness of each machine learning model. True positive rate (recall), false positive rate, F1 metric, and running time are performance metrics used in the comparison. While all models are effective against some attack types (e.g., DDoS-LOIC-UDP, SSH-BruteForce, or FTP-BruteForce, with true positive rate close to 100% and near zero false positive rate) some of the other attacks (e.g., Infiltration and SQL Injection) are difficult to detect correctly. For instance, using K-nearest neighbors model, the true positive rate for infiltration attacks runs as low as 28% with the false positive rate of 16%. It turns out the decision tree model is the best performer and is chosen to detect the zero-day attacks.

The authors apply the decision tree model to the zero-day attack data at last. The zero-day attacks are mixed with the benign data. The true-positive rate achieves the best at 96% when the maximum tree depth is 5, and slightly deteriorates to 92% and 90% when the maximum tree depth reduces to 3 and 4. The false-positive rate is lowest at 5% when the maximum depth is 3 and 4, and deteriorates to about 10% when the maximum depth is either lower or higher than 2 or 5, respectively.

3.2. Integrating supervised and unsupervised learning for zero-day malware detection

Malware is a malicious software program that is deployed by attackers to compromise a computer system. Malware, e.g., botnets, spyware, keystroke loggers, and trojans, can disrupt the normal operations of the system, give the attackers unauthorized access to the system resources, or allow them to gather information from users without their consent. Malware is one of the most damaging security threats facing the Internet today.

In addition to newly developed malwares, the existing malwares also evolve, either by automatically reprogramming themselves when being distributed or propagated (so called metamorphic malwares), or by self-mutating or being encrypted to avoid being detected (so called polymorphic malwares). The signature-based approach is, thus, not effective in detecting such zero-day malwares.

In [8], Comar et al. developed a two-level, supervised and unsupervised hybrid learning method for zero-day malware detection. Fig. 4 depicts the detection framework, which consists of the top macro-level classifier and the bottom micro-level multiple classifiers. The macro-level employs a random forest classifier to determine if an incoming flow is malicious or not. If deemed to be malicious, the incoming flow is passed on to the micro-level classifier to check if it belongs to one of the known malwares in the training set, or a zero-day malware.

The macro-level classifier employs a binary random forest model/classifier. The micro-level classifier employs a multi-class Support Vector Machine (SVM) model developed in [35]. The micro-level classifier is made up of multiple single-class SVMs, where each single-class SVM

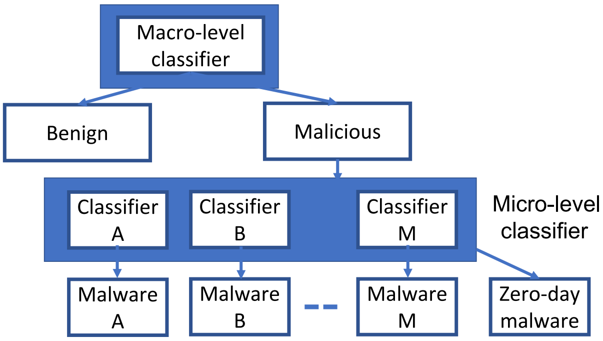

Fig. 4. Two-level hybrid learning-based zero-day malware detection framework.

model specializes in categorizing instances from a specific malware class. Collectively, the micro-level classifiers are able to distinguish zero-day malware from those that are already known.[1]

A single-class classifier consists of two concentric hyperspheres. The hyperspheres for class k are constructed by first labeling the instances belonging to class k as +1 and those that belong to other classes as -1. Two concentric hyperspheres are constructed such that the inner sphere encloses as many instances from class k as possible. The outer sphere is constructed so that instances that do not belong to class k lie outside of it. The radial distance between the two hyperspheres is called the classifier's margin, which defines the objective function to be optimized by the single-class SVM learning algorithm:

$$\min_{R_k,a_k,d_k,\xi_i,\xi_l} \quad R_k^2 - M d_k^2 + \frac{C}{N_k}\sum_{i:y_i=k}\xi_i + \frac{C}{N_{\bar{k}}}\sum_{l:y_l\neq k}\xi_l \tag{5}$$

$$\begin{aligned} \text{subject to} \quad & \|x_i - a_k\|^2 \le R_k^2 + \xi_i, \quad \forall i : y_i = k \\ & \|x_l - a_k\|^2 \ge R_k^2 + d_k^2 - \xi_l, \quad \forall l : y_l \neq k \\ & \xi_i \ge 0, \quad \forall i : y_i = k \\ & \xi_l \ge 0, \quad \forall l : y_l \neq k \end{aligned} \tag{6}$$

where a_k and R_k represent the center and radius of the inner hypersphere for the k-th class. The value of d_k represents the margin between inner and outer hyperspheres. Specifically, the radius of outer hypersphere is $\sqrt{R_k^2 + d_k^2}$. C and M are parameters that penalizes misclassification errors and control the trade-offs between R_k^2 and d_k^2. The values of N_k and $N_{\bar{k}}$ are the number of instances that belong to the k-th class and its complement. ξ_i and ξ_l are the slack variables. The above optimization problem is solved using the dual approach [8].

- **Zero-day malware detection.** If a new malware's flow-based features are significantly different from those of the trained malware, the test instance lies outside the hyperspheres for all the single-class SVM models. This instance is predicted to be a zero-day malware.

On the other hand, a zero-day malware may be a variant of existing malware, and, thus, be identified as the malware by one or multiple single-class SVM models. In such cases, a probabilistic class-based profiling is developed for zero-day malware detection. Let the inner hypersphere of the k-th single-class SVM model be defined by (a_k, R_k), where a_k is the center and R_k is the radius of the hypersphere. Further assume that the radial distances for all training instances follow a Gaussian distribution with mean μ_k and standard deviation σ_k. The values of μ_k and σ_k are estimated using the training examples. The probability p_k of a test instance belonging to the class k is estimated using its radical distance and the corresponding Gaussian model. A final score is computed based on the probability vector $[p_1, p_2, \dots, p_M]$. If the final score is greater than a preset threshold, the test instance is deemed to be a zero-day malware.

- **Data sets and evaluations.** The scheme is evaluated using a private data set from an Internet service provider. The 108 network flow-level features, such as bytes per second, packets per flow, packet inter-arrival times, etc., are used as the features. The class labels are obtained by analyzing network traffic's corresponding payload using a commercial IDS/IPS system. 38 different types of malicious flows are identified. The flows that were unlabeled by the IDS system are labeled as benign traffic.

The data are partitioned into training set and test set. To simulate zero-day malware, some of the malware classes were withheld from the training set and appear only in the test set. Specifically, twelve most prevalent malware classes, e.g., Sality, Conficker (Downadup), Tidserv, and Trojans, are selected as known malware and are included in the training set (together with an equal number of benign flows). The remainder of the network flows were assigned to the test set that includes all 38 malware flows.

The random forest model based macro-level binary classifier is able to detect all known malware and 88.54% of the zero-day malware, with the F-1 score of 90.96% on the malicious classes. For micro-level classification, the AUC score (the total area underneath the ROC curve) reaches 91% to detect the zero-day malware. However, the F1 score of zero-day attack detection is only 0.50.

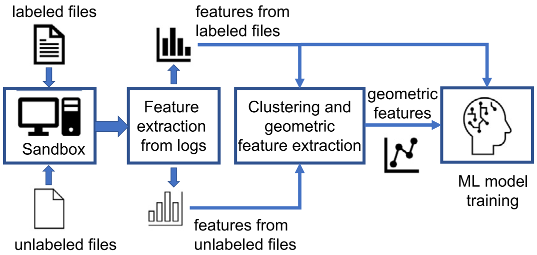
labeled files
features from labeled files
Sandbox
Feature extraction from logs
Clustering and geometric feature extraction
geometric features
ML model training
unlabeled files
features from unlabeled files

Fig. 5. Zero-day malware detection using labeled and unlabeled data.

3.3. Hybrid learning using available unlabeled data for zero-day malware detection

In [9], a hybrid learning approach is proposed that automatically integrates the knowledge of the unknown malware from available unlabeled data into the detection system. Fig. 5 depicts the proposed scheme. First, the files, both labeled and unlabeled, are executed in a sandbox where dynamic run-time logs are collected. The collected logs are processed to extract features—the frequencies of API calls. Then the feature vectors of both labeled and unlabeled data sets are merged and fed into a k-mean cluster, where multiple clusters are formed (since the clustering is unsupervised learning, the labels are not used in this step). The centroids of individual clusters are computed, and the geometric distance from a labeled sample to the clustering centroids are calculated as the augmented features. Finally, the labeled samples with the run-time features and the augmented geometric features are used as the training data to train the supervised classifier, such as Random Forest, SVM, etc. The main intuition of the design is that the unlabeled data set may already contain the zero-day malware. Thus, the augmented geometric distance shall reflect the patterns of zero-day malware.

- **Data sets and evaluations**. The experiments use the data from CA Technologies VET Zoo and two other publicly available data sources. Unfortunately all of these data sets are no longer available currently. Besides the clean files, the data set contains three malware types, Trojan, worms, and viruses, and twenty malware families. The training data set contains all malware families except *emerleox*, a virus family that is left out as the zero-day attacker. A total of 907 examples are in the training set while 506 examples are in the testing set.

[1] Note that the single-class SVM is different from one-class SVM as described in Section 2.1. The single-class SVM is a special case of multi-class SVM and is trained using labeled data. The one-class SVM is trained using unlabeled data.

All the data from both the training and test sets are merged into one set for unsupervised clustering. A global k-means algorithm is used to cluster the data into three groups. The cluster centroids are calculated and the distances of the examples to the cluster centroids are computed. The geometric distances are considered as the knowledge from unlabeled data. For the purpose of comparison, SVM, decision trees(J48), Naive Bayes, instance-based (IB) classifiers, and Random forest are used as the binary classifiers. True positive rate, false positive rate, AUC, and accuracy are used as the accuracy metrics. The experiments show that using run-time analysis and malware API frequency as the feature vector (but without augmented geometric distances), Random Forest achieves the highest accuracy of 98.5348%, and true positive rate of 0.985, false positive rate of 0.001, and AUC of 0.998. With the augmented geometric distances, both SVM and Random forest achieve perfect detection results—true positive rate (1), false positive rate (0), AUC (1), and accuracy (100%).

3.4. Generative adversarial networks (GAN)-based zero-day malware detection

In [10], Kim et al. proposed a Generative Adversarial Networks (GAN)-based zero-day malware detector. Generative Adversarial Networks (GAN) consists of a discriminator and a generator that interacts adversely and are trained together. The generator takes uniform random variable z as the input and generates fake data that are statistically similar to the training data. The discriminator is a classifier that is trained to differentiate the generated data from the real data. The GAN is trained using the malware data. After a successful training, the GAN's discriminator shall be able to classify a testing sample either as a malware or a benign software.

Below we describe the working mechanism of a GAN neural network. Eqn (7) is the objective function of a GAN, where $p_{data}(x)$ is the probability distribution of the real/training data. $G(z)$ is the generated data by the generator G driven a random variable z, where z follows a probability distribution of $p(z)$. The discriminator D is trained to maximize $V(D,G)$, i.e., it strives to classify the real data x to be one ($D(x) = 1$), and classifies the generated data to be zero ($D(G(z)) = 0$). In contrast, G is trained to minimize $V(D,G)$ such that $D(G(z))$ is one, i.e., the discriminator D cannot distinguish $G(z)$ from real training data x.

$$\begin{aligned}\min_G \max_D V(D,G) =& \mathbb{E}_{x\sim p_{data}(x)}\left[\log D(x)\right] \\ &+ \mathbb{E}_{z\sim p_z(z)}\left[\log(1-D(G(z)))\right]\end{aligned} \tag{7}$$

When the GAN training converges, the distribution of the data generated by the generator G becomes the same as the distribution of real data. The discriminator D is able to distinguish the data different from the real data distribution accurately. In other words, the discriminator can serve as a zero-day malware detector to classify the malware from the normal data.

One issue with the GAN training is its instability in training convergence. The authors proposed to train a autoencoder first using the entire training data set. As described in 2.2, the encoder compresses the input into latent vector, while the decoder reconstruct the output. The trained decoder is transferred to the GAN as the preliminary generator. The GAN is trained using the malware samples in the training data set. Finally, the trained GAN discriminator is used as the detector. In addition, since the malware code is represented as an image, and the deep-convolutional neural network is effective in image generation and detection, the deep-convolution technique is used in both autoencoder and GAN. For example, the Deep Convolutional GAN (DCGAN), an extension of the GAN architecture using deep convolutional neural networks for both the generator and discriminator models, is employed in [10].

• **Data sets and evaluations**. The malware data set used in training and testing is from the Kaggle Microsoft Malware Classification Challenge [36]. The data in the form of binary codes are used and are converted into the image. Because sizes of images are too large, the images were reduced to 0.1 times of their original sizes. The zero-day attacks are generated by introducing noise into existing malware. Specifically, the zero-day attack is produced by combining two malware images using Structural Similarity Index (SSIM) method.

The proposed method is compared with other ML-based detection methods such as SVM, random forest, decision tree, naive Bayes, etc. The results show that the proposed GAN discriminator-based detector consistently out-performs other models. For zero-day attacks, the proposed scheme achieves greater than 98% detection accuracy and is robust against the zero-day attacks generated with different noise levels.

4. Transfer learning (TL)-based zero-day attack detection

Traditional Machine Learning assumes that the collected training data and future data share the same features and have the same distribution. The machine learning models are trained on the training data and make predictions on the future data. Transfer learning, in contrast, allows the feature space, data distribution, and label prediction tasks to be different. In a real world, human beings can intelligently apply knowledge learned in previous tasks to solve a new problem faster and better, which motives the development of transfer learning techniques [37,38].

Formally, a *domain* $\mathcal{D}$ is defined as the tuple of $\{\mathcal{X}, P(X)\}$, where $\mathcal{X}$ is the feature space and $P(X)$ is the probability distribution of X, where $X = \{x_1, \dots, x_n\} \in \mathcal{X}$. A *task* $\mathcal{T}$ consists of a label space $\mathcal{Y}$ and a predictive function $f(\cdot)$, i.e., $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$. The predictive function $f(\cdot)$ is expected to be learned from the *domain data* of $\{(x_i, y_i)\}$, where $x_i \in X$ and y_i either belongs to $\mathcal{Y}$ or not available. In a later case (x_i, y_i) is a non-labeled data. The function f can be used to predict the corresponding label $f(x)$ of a new instance x.

Consider a source domain $\mathcal{D}_S$ and learning task $\mathcal{T}_S$, and a target domain $\mathcal{D}_T$ and learning task $\mathcal{T}_T$, transfer learning aims to help improve the learning of the target predictive function f_T using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$. If the source and target domain are the same, i.e., $\mathcal{D}_S = \mathcal{D}_T$, and their learning tasks are the same, i.e., $\mathcal{T}_S = \mathcal{T}_T$, the learning problem becomes a traditional machine learning problem.

The transfer learning (TL) techniques can be categorized into different categories based on either the TL settings or the TL approaches. Based on TL settings, there are three types of TL: *Inductive TL*, *Transductive TL*, and *Unsupervised TL*:

- *Inductive TL*: the source and target tasks are different but related. The source and target domains may or may not be the same. Some labeled data in the target domain are required for the purpose of induction.
- *Transductive TL*: the source and target tasks are the same, but the source and target domains are different. A lot of labeled data in the source domain are available while no labeled data in the target domain are available.
- Unsupervised TL: similar to the inductive TL, the target task is different from the source task. The labels are not available in both source domain and target domain. Unsupervised TL focuses on solving unsupervised learning tasks in the target domain such as clustering or dimensionality reduction.

In the zero-day attack detection, the source and target tasks are the same, i.e., to detect the attacks. The labeled data in the target domain, i.e., the labels for zero-day attacks, are typically not available. Hence, the transductive TL is most suitable for the zero-day attack detection. In Section 4.2, Transductive TL-based domain adaption method is described for zero-day attack detection.

Based on different TL approaches, TL techniques can be categorized into four categories: *Instance-based TL*, *Feature-based TL*, *Parameter-based TL*, and *Relational knowledge-based TL*:

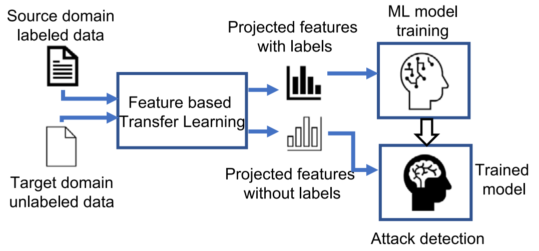
Source domain labeled data
Target domain unlabeled data
Feature based Transfer Learning
Projected features with labels
Projected features without labels
ML model training
Trained model
Attack detection

Fig. 6. Feature-based TL for zero-day attack detection using labeled source domain data and unlabeled target domain data.

- *Instance-based TL*: The Instance-based TL re-weights some labeled data in the source domain for the use in the target domain.
- *Feature-based TL*: In the Feature-based TL, a good feature representation is found so as to reduce the difference between the source and target domains and, thus, improve the prediction accuracy in the target domain.
- *Parameter-based TL*: The parameter-based TL discovers shared parameters or priors between the source and target domain to improve the transfer learning.
- *Relational knowledge-based TL*: Relational knowledge-based TL builds the mapping of relational knowledge between the source domain and the target domain.

The feature-based TL has been explored in the zero-day attack detection as described in Section 4.1. Feather-based TL can adapt features in a target domain with deficient/no labeled data by transferring learned knowledge from a related source domain. The intuition behind this is that a human's transitive inference ability can extend what has been learned in one domain to a new similar domain. Zero-day attacks often belong to variants of known attack families and share common traits in feature spaces, which suggested a good fit for applying transfer learning.

4.1. Feature-based transfer learning for zero-day network attack detection using spectral transformation

In feature-based transfer learning, a new feature representation is learned from the source and the target domain and is used to transfer knowledge across domains. Let the source domain training examples be $S = \{x_i\}$, $x_i \in R^m$, and labels be $L_S = y_i$. The target domain data are $T = \{u_j\}$, $u_j \in R^n$, and the target domain does not have labels. The source and target domain samples have different feature dimension, m and n respectively. The goal is to accurately predict the labels on the target domain T.

In [11,12], the authors proposed the spectral transformation-based approach that transforms the source and target data onto a common latent feature space (see Fig. 6). After the transformation, a Machine Learning-based classifier is trained using the transformed source domain data and their labels. The same classifier can then be used to classify the transformed target samples for the zero-day network attack detection. We next described how the feature transformation is done.

Given source data S and target data T, an optimal projection of S and T onto an optimal subspace V_S and V_T with the feature size of k is conducted according to the following optimization objective:

$$\min_{V_S,V_T} l(V_S,S) + l(V_T,T) + \beta D(V_S,V_T), \tag{8}$$

where $l(\cdot,\cdot)$ is a distortion function that evaluates the difference between the original data and the projected data. $D(V_S,V_T)$ denotes the difference between the projected source data and the projected target data. The parameter β controls the trade-off between $l(V_S,S)+l(V_T,T)$ and $D(V_S,V_T)$. The function $l(\cdot,\cdot)$ and $D(\cdot)$ are defined as:

$$D(V_S,V_T) = ||V_T - V_S||^2, \tag{9}$$

$$l(V_S,S) = ||S - V_S P_S||^2,\ l(V_T,T) = ||T - V_T P_T||^2, \tag{10}$$

where V_S and V_T are linearly transformed back to original domain and target space by matrix $P_S \in R^{k\times m}$ and $P_T \in R^{k\times n}$, respectively. $||\cdot||^2$ is the Frobenius norm that can also be expressed as a matrix trace norm.

Substituting (9) and (10) into (8), we obtain the following optimization objective to minimize with regard to V_S, V_T, P_S and P_T as follows:

$$\begin{aligned}\min G(V_S,V_T,P_S,P_T) = min\big(&||S - V_S P_S||^2 \\ &+ ||T - V_T P_T||^2 + \beta \cdot ||V_T - V_S||^2\big)\end{aligned} \tag{11}$$

In [11], a gradient method, called HeTL, is employed to obtain the global minimums. However, the performance of the algorithm depends on the choice of β value as well as the row order of the data samples in S and T, which affects the results of $D(V_S,V_T)$. Practically, we know little about the new attack in T, so the transformation process in (11) could be misleading. In [12], a clustering enhanced hierarchical transfer learning is proposed to mitigate the issue.

• **Data sets and evaluation**. NSL-KDD data set [26] is used in the evaluation. NSL-KDD contains network features extracted from a series of TCP connections captured from a local area network. The data set has 41 network features and 22 different types of attack, which can be grouped into four main categories: DoS, R2L, Probe, and User to root (U2R). The portion of U2R is small and, thus, is not used in the evaluation.

The zero-day attacks are simulated by assuming attacks in the target data have no labels and differ from attacks in the source domain. One main attack category, e.g., DoS, R2l, or Probe, and normal examples form the source domain; a different attack type combined with normal samples from the target domain. Three source–target domain pairs are constructed: DoS → Probe, DoS → R2L and Probe → R2L. The experiments are repeated ten times, and the average results are reported.

The authors argue that different feature sets may be used to detect different attacks. For example, traffic feature is more distinguishable for DoS attack, while the content feature is more distinguishable for the R2L attack. In the experiments, the most relative features are selected for the source and target domains using information gain, resulting in unequal feature dimensions. The final selected features are included in the Appendix in [12]. For the purpose of comparison, the baseline approach always uses the features selected for the source domain and apply the traditional classifiers.

The C4.5 decision tree (CART), linear SVM, and KNN are chosen as the classifiers for baselines and TL-based approaches. The accuracy, F1 score, and ROC curve are used as the performance metrics. The results show that the baseline models performed poorly, with accuracy of 0.47–0.74 and F1 score of 0.1–0.65. The proposed TL-based approaches, HeTL and CeHTL, significantly outperformed the baselines, obtained over 0.70 accuracy and 0.75 F1 score. CeHTL also outperformed HeTL in all experiments.

• **Discussion**. The proposed feature-based TL methods require that the data sets from both the source and target domains are available when computing the optimal feature vectors in the latent space. Thus, the methods will not work in the real-time zero-day attack detection. In addition, the feature vector of the target domain is set to be the same as that of the source domain in the baseline experiments, i.e., not all features are used in the baseline experiments, which may contribute to the low accuracy and F1 scores. A fairer approach is to use the entire set of available features in the baseline experiments. As shown in Section 2.3, the results of accuracy and F1 score are much better using the entire feature set. It will be interesting to see if TL-based approaches can still outperform the baseline methods using the entire feature set.

4.2. Domain adaptation-based zero-day attack detection using manifold alignment

Manifold alignment-based domain adaptation assumes there are K data sets, where the data instances belong to c different classes (labels). Let $X_k = (x_k^1, \dots, x_k^{m_k})$ represent the k-th input data set with p_k features. X_k can be viewed as a matrix of size $p_k \times m_k$. Some data points in a data set are labeled, while others are not. When X_k corresponds to a source domain, the number of labeled data is usually large; when X_k corresponds to a target domain, the number of labeled data is small.

Manifold alignment-based domain adaption is introduced in [39]. It treats each input domain as a manifold. The goal is to construct K mapping functions $f_1, \dots, f_K$ to project the input domains to a new lower-dimension latent space while preserving the topology of each domain, i.e., the instances with the same labels become neighbors, the instances with different labels are separated, and the topology of each set is preserved. The resulting feature space is a common underlying space shared by all the input domains, and can be directly used for knowledge transfer across domains.

Define similarity matrix W_s and dissimilarity matrix W_d as follows:

$$W_s = \begin{pmatrix} W_s^{1,1} & \cdots & W_s^{1,K} \\ \cdots & \cdots & \cdots \\ W_s^{K,1} & \cdots & W_s^{K,K} \end{pmatrix} \tag{12}$$

$$W_d = \begin{pmatrix} W_d^{1,1} & \cdots & W_d^{1,K} \\ \cdots & \cdots & \cdots \\ W_d^{K,1} & \cdots & W_d^{K,K} \end{pmatrix} \tag{13}$$

Both W_s and W_d are $(m_1 + \cdots + m_K) \times (m_1 + \cdots + m_K)$ matrix, where $W_s^{a,b}$ and $W_d^{a,b}$ are $m_a \times m_b$ matrix. $W_s^{a,b}(i,j) = 1$ if x_a^i and x_b^j have the same label; $W_s^{a,b}(i,j) = 0$ otherwise (including the case when the label is not available). $W_d^{a,b}(i,j) = 1$ if x_a^i and x_b^j have different labels; $W_d^{a,b}(i,j) = 0$ otherwise (including the case when the label is not available). Finally define $W_k(i,j) = e^{-||x_k^i - x_k^j||^2}$. The mapping function $f_1, \dots, f_K$ are obtained by minimizing the cost function:

$$C(f_1, \dots, f_K) = (A + C)/B, \tag{14}$$

where

$$A = 0.5 \sum_{a=1}^{K} \sum_{b=1}^{K} \sum_{i=1}^{m_a} \sum_{j=1}^{m_b} ||f_a^T x_a^i - f_b^T x_b^j||^2 W_s^{a,b}(i,j), \tag{15}$$

$$B = 0.5 \sum_{a=1}^{K} \sum_{b=1}^{K} \sum_{i=1}^{m_a} \sum_{j=1}^{m_b} ||f_a^T x_a^i - f_b^T x_b^j||^2 W_d^{a,b}(i,j), \tag{16}$$

$$C = 0.5\mu \sum_{k=1}^{K} \sum_{i=1}^{m_k} \sum_{j=1}^{m_k} ||f_k^T x_k^i - f_k^T x_k^j||^2 W_k(i,j). \tag{17}$$

Minimizing A encourages the instances from the same class to be projected to similar locations in the new space. Maximizing B encourages the instances from different classes to be separated in the new space. Finally, minimizing C preserves topology of each given domain, where μ is a weight parameter.

• **Domain Adaptation-based zero-day attack detection using manifold alignment.** The authors in [13,40] proposed modified domain adaptation-based zero-day attack detection method using manifold alignment. The method assumes there are one source domain and one target domain, where only the data points in the source domain are labeled as either normal or malicious. In order to overcome the lack of labels in the target domain, the authors employed clustering method to construct similarity and dissimilarity matrices as required by the manifold alignment method. The methods developed in [13,40] differ in how the similarity and dissimilarity matrices are constructed. Since [13] is a newer work, we focus on its description below.

The source domain is first clustered into k clusters using k-medoid clustering method. The value of k is selected so as to minimize the cluster purity value. The target domain is also clustered into k clusters. In each domain, clusters are sorted in the ascending order of their respective distance from the cluster mean to the domain global mean. For instance, the highest rank (rank one) is assigned to the cluster that is closest to the global mean.

The cluster rankings are used to decide the similarity and dissimilarity values. The rationale is that the neighboring clusters are more similar to each other than the clusters that are far apart; and the i-th cluster of one domain may have correspondence with the i-th cluster as well as its nearby clusters (i-1-th and i+1-th)of the other domain. Hence, the similarity value among the i-th cluster of one domain and the corresponding i-th cluster of the other domain is set to be one. The similarity value among the i-th cluster and the i-1-th and i+1-th clusters is set to be 0.5, others are set to be zero. Similarly, in case of constructing the dissimilarity matrix, the dissimilarity value among the i-th cluster and the corresponding i-th cluster of the domain is set to be zero. The dissimilarity value among the i-th cluster and the i-1-th and i+1-th clusters is set to be 0.5, and others are set to be one. The manifold alignment algorithm is then applied to obtain the mapping functions for all clusters.

To increase the number of available labels, the authors [13] also proposed the soft label generation method for target domain, using the rational similar to assign the similarity and dissimilarity values. See [13] for details.

Once the data are mapped to the latent space, a Machine Learning classifier, such as Deep Neural Network, SVM, etc., is trained using the labeled data. The trained classifier is used as the zero-day attack detector.

• **Data sets and evaluation**. The NSL-KDD data set [26] is used for the evaluation. The zero-day attack is simulated by hiding an attack's labels. The Transfer Learning tasks of DoS → R2L and DoS → Probe are evaluated. Each data set contains the specific attack instances and some normal instances. In addition, another cloud intrusion detection data set, namely CIDD [41], is used as the zero-day attacks and the NSL_DoS → CIDD is also evaluated. Multiple Machine Learning classifiers, such as Deep Neural Network, SVM, K-Nearest Neighbors, Random Forrest, and Decision Tree, are used in the experiment, and the Deep Neural Network offers the best results.

In the DoS → R2L transfer learning task, the accuracy reaches 0.9183 and the false positive rate is 0.0423. The similar accuracy (0.9175) and the false positive rate (0.0822) for the DoS → Probe task. However, the accuracy is only 0.7885 in the NSL_DoS → CIDD task, with the false positive rate of zero. The authors also compare the performance of manifold alignment-based scheme with that of CeHTL [12], HeTL [11], PCA-based TL [42], and the transfer learning-based approach as described in [40], the manifold alignment-based scheme out-performs the aforementioned approaches.

5. Comparisons, challenges and future directions

Table 1 compares different ML-based zero-attack detection techniques in terms of ML model employed, training data set, zero-day testing data set, evaluation results, and main challenges and issues. Among nine zero-day detection schemes reviewed by this paper, seven schemes use a total of five publicly available data sets for the model training. The rest two schemes make use of a test-bed and private data sets, respectively. In addition, these five public data sets are from several different domains, e.g., malware, network attacks, etc., and differ in the features space.

The reviewed schemes in the Table 1 show promise in accurately detecting zero-day attack using Machine Learning models. However, the proposed detection schemes often exhibit large variation in detection accuracy against different types of attacks. In addition, quantitative head-to-head comparison of different schemes is difficult as the evaluation data sets, evaluation metrics, and comparison schemes are often different. The training/testing data sets are limited and siloed and a

Table 1
Summary of Machine Learning (ML)-based zero-day attack detection methods.

Detector Name	ML model	Training data set	Zero-day Testing data set	Evaluation results	Challenges & Issues
Outlier detector [5]	One-class SVM	CIC-IDS2017, NSL-KDD (benign traffic only)	Attacks withheld from training and used in testing	Recall varies from 27% to 99% based on attack types	Varying accuracy, inconsistency against different attacks
Outlier detector [5]	Autoencoder	CIC-IDS2017, NSL-KDD (benign traffic only)	Attacks withheld from training and used in testing	Out-perform One-class SVM for complex attacks	Varying accuracy, inconsistency against different attacks
Kitsune [6]	An ensemble of autoencoders	IP camera video surveillance test-bed (benign traffic only)	Emulated attacks on test-bed	Out-perform offline algorithms; vary greatly based on attack types	Varying accuracy, limited test cases
Comparison of six Supervised ML detectors [7]	Random forest, Gaussian naive Bayes, Decision tree, MLP, K-nearest neighbors, Quadratic discriminant analysis	CSE-CIC-IDS2018	Eight new attacks collected in real networks, including Bitcoin miner, Drowor worm, nuclear ransomware, false content injection, etc.	Vary greatly based on attack types; decision tree model is the best performer; true-positive rate bests at 96% and false-positive rate gets lowest at 5%	Varying accuracy, inconsistency against different attacks
Two-level supervised and unsupervised learning method [8]	Binary random forrest model and SVM	Private data set from an Internet service provider	Attacks withheld from training and used in testing	At top-level, 88.54% of zero-day attacks detected; at the second-level, AUC score reaches 91% with F1 score of 0.50	Evaluated using private data, difficult to compare, limited test cases
Hybrid learning method [9]	k-mean cluster, Random Forrest, SVM, etc.	Data from CA Technologies VET Zoo and two other data sets; unlabeled testing data used in training as well	Attack labels withheld in training and used in testing	Perfect detection is achieved—true positive rate (1), false positive rate (0), AUC (1), and accuracy (100%)	Limited test cases, data sets not public available anymore, difficult to compare
Generative Adversarial Networks (GAN)-based detector [10]	GAN, autoencoder	Kaggle Microsoft Malware Classification Challenge; binary codes are converted into the images	Generated by introducing noise to existing malware	> 98% detection accuracy against zero-day attacks; robust against zero-day attacks generated with different noise levels	Limited test cases
Feature-based Transfer Learning (TL) detector [11,12]	Spectral transformation-based approach, decision tree, SVM, KNN	NSL-KDD; three source–target domain pairs used in training; unlabeled testing data as target domain	Attack labels withheld in training and used in testing	Achieve 70% accuracy and 0.75 F1 score	Low detection accuracy, limited test cases
Domain adaptation-based Transfer Learning (TL) detector [13]	Manifold alignment-based domain adaptation	Two source–target domain pairs from NSL-KDD data set; the third source–target domain pair uses NSL-KDD as source domain and CIDD data set as target domain	Attack labels from NSL-KDD withheld in training and used in testing; CIDD data set is also used as zero-day attacks	Vary according to different source domain and target domain pairs; Out-perform feature-based Transfer Learning detector	Varying accuracy, limited test cases

large scale, representative data set is lacking. Different schemes are often evaluated using different set of performance metrics. Due to the significant effort needed to implement comparison schemes, the proposed schemes are often compared to a small number of competing schemes and draw the conclusions. The combination of non-representative data sets and limited model comparisons make the outcome inconclusive and sometimes questionable.

The zero-day detection schemes in the Table 1 fall into three categories: unsupervised learning-based zero-day attack detection, supervised and hybrid learning-based zero-day attack detection, and transfer learning-based zero-day attack detection. Outlier detectors [5,6] employ the unsupervised learning method. Unsupervised learning is a type of ML algorithms that learns patterns from unlabeled data. The zero-day attack detection model seeks to learn a compressed representation of normal data to detect zero-day attacks. The outlier detection methods do not require zero-day attack data for the training purpose. This is a great advantage since the zero-day attack data is typically not available at the model training phase.

Supervised and hybrid learning-based zero-day attack detection methods [7,9,9,10] use the supervised learning or the combination of supervised and unsupervised learning method. With representative training data sets, supervised and hybrid learning-based zero-day attack detectors can perform accurate detection. Unfortunately, the data set for zero-day detection typically lacks the representation of zero-day attacks. The detection schemes need to assume that the zero-day attacks behave similarly to the known attacks, an assumption that has not been validated so far.

Transfer learning-based zero-day attack detection methods [11–13] transfer knowledge across domains. Transfer learning seeks to leverage unlabeled and limited data in the target task or domain to the most effect. Given limited or no zero-data attack data, transfer learning is a promising method for zero-day attack detection.

Training and detection speed is another important factor for ML-based zero-day attack detection. While training often takes longer time than detection, and different methods vary in training/detection speed, all the reviewed methods are able to complete training/detection in reasonable time.

5.1. Future directions

Multi-front efforts are required to address the challenges in designing effective ML-based zero-day attack detection schemes. For instance, to address the issue of lacking zero-day attack information in the training data set, honeypot [43] can be used to collect the zero-day attack data before the attacks become known.

Conducting effective feature engineering with domain expertise is another way to improve detection accuracy. Attackers may circumvent the detection if the attack features are similar to legitimate ones. It is important to integrate domain experts' knowledge into the process of feature engineering to ensure that the new attacks will reveal themselves in the designated feature space.

ML technique evolves quickly. Taking advantage of the latest advancement in ML and adopting the new ML models is another way to defend against zero-day attacks. For example, Reinforcement Learning (RL) is another type of machine learning techniques that enables an agent, or decision maker, to learn in an interactive environment by trial and error using feedback from its own actions and experiences. The agent only needs to access induced feedbacks but is not required to know all the factors that determine feedbacks. So RL is particularly well suited to zero-day attack problems where the vulnerabilities and attack targets are not known in advance.

In [44], RL is explored to defend the system against zero-day attacks without actually detecting them. The reinforcement learning based schemes are able to defeat three classes of zero-day attacks: strategic zero-day attacks where the interactions between an attacker and a defender are modeled as a non-cooperative game; non-strategic random zero-day attacks where the attacker chooses its actions by following a predetermined probability distribution; and zero-day attacks that follow the Bayesian attack graphs. We look forward to new zero-day attack defense methods that take advantage of novel ML models.

Finally, the development of a ML-based zero-day attack detection benchmark shall address many hurdles that hinder the progress of research and development. A benchmark suite with rich and standardized data sets, a plethora of representative models and automated testing-and-evaluation capability will greatly expedite the development of ML-based zero-day attack detection tools.

6. Conclusions

Zero-day attacks happen often (an ‘‘in the wild’’ zero-day attack is discovered every 17 days [2]), last for a long time (a typical zero-day attack lasts 312 days on average before being detected [1]) and cause grave consequences (the average incurred cost amounting to 1.2 million dollars per attack [3]). Machine Learning (ML)-based detection represents the most promising and effective method to detect zero-day attacks.

In this paper, a comprehensive review is conducted on the ML-based zero-day attack detection methods based on the types of employed ML models, from unsupervised learning, supervised learning, to hybrid learning and transfer learning. By comparing and contrasting different approaches, key challenges are identified. The ML-based zero-day detection faces the fundamental challenge in lacking the zero-day attack representation in the data sets. The limited and siloed data sets and incomprehensive feature space also makes the proposed models not as accurate, robust, and reliable as desired.

Moving forward, we recommend continuing leveraging the latest advancement in the ML research, and better incorporating domain experts' knowledge into the ML model development. In addition, developing a data-rich, standardized benchmark shall help tremendously to continuously improve ML-based zero-day attack detection models.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgment

Certain commercial equipment, instruments, or materials are identified in this paper in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

References

[1] L. Bilge, T. Dumitraş, Before we knew it: An empirical study of zero-day attacks in the real world, in: Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS '12), ACM, 2012, pp. 833–844.

[2] Google, Project Zero, URL https://googleprojectzero.blogspot.com/p/0day.html.

[3] Ponemon Sullivan Privacy Report, The economic value of prevention in the cybersecurity lifecycle, 2020.

[4] R.A. Bridges, S. Oesch, M.E. Verma, M.D. Iannacone, K.M.T. Huffer, B. Jewell, J.A. Nichols, B. Weber, J.M. Beaver, J.M. Smith, D. Scofield, C. Miles, T. Plummer, M. Daniell, A.M. Tall, Beyond the hype: A real-world evaluation of the impact and cost of machine learning-based malware detection, 2021, arXiv:2012.09214.

[5] H. Hindy, R. Atkinson, C. Tachtatzis, J.-N. Colin, E. Bayne, X. Bellekens, Utilising deep learning techniques for effective zero-day attack detection, Electronics 9 (10) (2020) http://dx.doi.org/10.3390/electronics9101684, URL https://www.mdpi.com/2079-9292/9/10/1684.

[6] Y. Mirsky, T. Doitshman, Y. Elovici, A. Shabtai, Kitsune: An ensemble of autoencoders for online network intrusion detection, NDSS (2018).

[7] Q. Zhou, D. Pezaros, Evaluation of machine learning classifiers for zero-day intrusion detection – an analysis on CIC-aws-2018 dataset, 2021, arXiv:1905.03685.

[8] P.M. Comar, L. Liu, S. Saha, P.-N. Tan, A. Nucci, Combining supervised and unsupervised learning for zero-day malware detection, in: 2013 Proceedings IEEE INFOCOM, 2013, pp. 2022–2030, http://dx.doi.org/10.1109/INFCOM.2013.6567003.

[9] S. Huda, S. Miah, M. Mehedi Hassan, R. Islam, J. Yearwood, M. Alrubaian, A. Almogren, Defending unknown attacks on cyber-physical systems by semi-supervised approach and available unlabeled data, Inform. Sci. 379 (2017) 211–228, http://dx.doi.org/10.1016/j.ins.2016.09.041, URL https://www.sciencedirect.com/science/article/pii/S0020025516309380.

[10] J.-Y. Kim, S.-J. Bu, S.-B. Cho, Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders, Inform. Sci. 460–461 (2018) 83–102, http://dx.doi.org/10.1016/j.ins.2018.04.092, URL https://www.sciencedirect.com/science/article/pii/S0020025518303475.

[11] J. Zhao, S. Shetty, J.W. Pan, Feature-based transfer learning for network security, in: MILCOM 2017 - 2017 IEEE Military Communications Conference, MILCOM, 2017, pp. 17–22, http://dx.doi.org/10.1109/MILCOM.2017.8170749.

[12] J. Zhao, S. Shetty, J.W. Pan, C. Kamhoua, K. Kwiat, Transfer learning for detecting unknown network attacks, EURASIP J. Inf. Secur. 2019 (2019) http://dx.doi.org/10.1186/s13635-019-0084-4.

[13] N. Sameera, M. Shashi, Deep transductive transfer learning framework for zero-day attack detection, ICT Express 6 (4) (2020) 361–367, http://dx.doi.org/10.1016/j.icte.2020.03.003, URL https://www.sciencedirect.com/science/article/pii/S2405959519303625.

[14] A.L. Buczak, E. Guven, A survey of data mining and machine learning methods for cyber security intrusion detection, IEEE Commun. Surv. Tutor. 18 (2) (2016) 1153–1176, http://dx.doi.org/10.1109/COMST.2015.2494502.

[15] A. Khraisat, I. Gondal, P. Vamplew, J. Kamruzzaman, Survey of intrusion detection systems: techniques, datasets and challenges, in: Cybersecur, 2, (20) 2019.

[16] H. Liu, B. Lang, Machine learning and deep learning methods for intrusion detection systems: A survey, Appl. Sci. 9 (20) (2019) http://dx.doi.org/10.3390/app9204396, URL https://www.mdpi.com/2076-3417/9/20/4396.

[17] Y. Ye, T. Li, D. Adjeroh, S.S. Iyengar, A survey on malware detection using data mining techniques, ACM Comput. Surv. 50 (3) (2017) http://dx.doi.org/10.1145/3073559.

[18] K. Liu, S. Xu, G. Xu, M. Zhang, D. Sun, H. Liu, A review of android malware detection approaches based on machine learning, IEEE Access 8 (2020) 124579–124607, http://dx.doi.org/10.1109/ACCESS.2020.3006143.

[19] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, J. Platt, Support vector method for novelty detection, in: Proceedings of the 12th International Conference on Neural Information Processing Systems, NIPS '99, MIT Press, Cambridge, MA, USA, 1999, pp. 582–588.

[20] S. Wang, Q. Liu, E. Zhu, F. Porikli, J. Yin, Hyperparameter selection of one-class support vector machine by self-adaptive data shifting, Pattern Recognit. 74 (2018) 198–211, http://dx.doi.org/10.1016/j.patcog.2017.09.012, URL https://www.sciencedirect.com/science/article/pii/S0031320317303564.

[21] N. Japkowicz, C. Myers, M. Gluck, A novelty detection approach to classification, in: IJCAI, 1995.

[22] H. Hindy, R.C. Atkinson, C. Tachtatzis, J.-N. Colin, E. Bayne, X. Bellekens, Towards an effective zero-day attack detection using outlier-based deep learning techniques, 2020, ArXiv arXiv:2006.15344.

[23] M. Gharib, B. Mohammadi, S. Hejareh Dastgerdi, M. Sabokrou, AutoIDS: Auto-encoder Based Method for Intrusion Detection System, 2019, arXiv e-prints arXiv:1911.03306.

[24] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016, http://www.deeplearningbook.org.

[25] Intrusion detection evaluation dataset (CIC-IDS2017), URL https://www.unb.ca/cic/datasets/ids-2017.html.

[26] NSL-KDD Dataset, URL https://www.unb.ca/cic/datasets/nsl.html/.

[27] R. Panigrahi, S. Borah, A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems, Int. J. Eng. Technol. 7 (3.24) (2018) 479–482.

[28] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, J. Mach. Learn. Res. 13 (2) (2012).

[29] P.-H. Chen, C.-J. Lin, B. Schölkopf, A tutorial on ν-support vector machines, Appl. Stoch. Models Bus. Ind. 21 (2) (2005) 111–136.

[30] F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation forest, in: 2008 Eighth IEEE International Conference on Data Mining, 2008, pp. 413–422, http://dx.doi.org/10.1109/ICDM.2008.17.

[31] D. Reynolds, Gaussian mixture models, in: S.Z. Li, A. Jain (Eds.), Encyclopedia of Biometrics, Springer US, Boston, MA, 2009, pp. 659–663, http://dx.doi.org/10.1007/978-0-387-73003-5_196.

[32] F. Abri, S. Siami-Namini, M.A. Khanghah, F.M. Soltani, A.S. Namin, Can machine/deep learning classifiers detect zero-day malware with high accuracy? in: 2019 IEEE International Conference on Big Data (Big Data), 2019, pp. 3252–3259, http://dx.doi.org/10.1109/BigData47090.2019.9006514.

[33] P. Parrend, J. Navarro, F. Guigou, A. Deruyver, P. Collet, Foundations and applications of artificial intelligence for zero-day and multi-step attack detection, EURASIP J. Inf. Secur. 2018, Number 1 (2018).

[34] A realistic cyber defense dataset (CSE-CIC-IDS2018), URL https://registry.opendata.aws/cse-cic-ids2018/.

[35] P.-Y. Hao, J.-H. Chiang, Y.-H. Lin, A new maximal-margin spherical-structured multi-class support vector machine, Appl. Intell. 30 (2009) 98–111.

[36] Kaggle: Microsoft malware classification challenge (BIG 2015), URL https://www.kaggle.com/c/malware-classification.

[37] S.J. Pan, Q. Yang, A survey on transfer learning, IEEE Trans. Knowl. Data Eng. 22 (10) (2010) 1345–1359, http://dx.doi.org/10.1109/TKDE.2009.191.

[38] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, Q. He, A comprehensive survey on transfer learning, Proc. IEEE 109 (1) (2021) 43–76, http://dx.doi.org/10.1109/JPROC.2020.3004555.

[39] C. Wang, S. Mahadevan, Heterogeneous domain adaptation using manifold alignment, in: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two, IJCAI '11, AAAI Press, 2011, pp. 1541–1546.

[40] Z. Taghiyarrenani, A. Fanian, E. Mahdavi, A. Mirzaei, H. Farsi, Transfer learning based intrusion detection, in: 2018 8th International Conference on Computer and Knowledge Engineering, ICCKE, 2018, pp. 92–97, http://dx.doi.org/10.1109/ICCKE.2018.8566601.

[41] R. Kumar, S.P. Lal, A. Sharma, Detecting denial of service attacks in the cloud, in: 2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech), 2016, pp. 309–316, http://dx.doi.org/10.1109/DASC-PICom-DataCom-CyberSciTec.2016.70.

[42] N. Sameera, M. Shashi, Transfer learning based prototype for zero-day attack detection, in: IJCAI, 2019.

[43] C. Musca, E. Mirica, R. Deaconescu, Detecting and analyzing zero-day attacks using honeypots, in: Proceedings of the 2013 19th International Conference on Control Systems and Computer Science, CSCS '13, IEEE Computer Society, USA, 2013, pp. 543–548, http://dx.doi.org/10.1109/CSCS.2013.94.

[44] Z. Hu, P. Chen, M. Zhu, P. Liu, Reinforcement learning for adaptive cyber defense against zero-day attacks, in: S. Jajodia, G. Cybenko, P. Liu, C. Wang, M. Wellman (Eds.), Adversarial and Uncertain Reasoning for Adaptive Cyber Defense: Control- and Game-Theoretic Approaches To Cyber Security, Springer International Publishing, Cham, 2019, pp. 54–93, http://dx.doi.org/10.1007/978-3-030-30719-6_4.