Assessment of Zero-Day Vulnerability using Machine Learning Approach

SakthiMurugan S [1], Sanjay Kumaar A[2], Vishnu Vignesh[3] and Santhi P[4]

[1, 2, 3, 4]Department of Computer Science and Engineering, Amrita School of Computing, Amrita Vishwa Vidyapeetham, Chennai, India

Abstract

Organisations and people are seriously threatened by zero-day vulnerabilities because they may be utilised by attackers to infiltrate systems and steal private data. Currently, Machine Learning (ML) techniques are crucial for finding zero-day vulnerabilities since they can analyse huge datasets and find patterns that can point to a vulnerability. This research's goal is to provide a reliable technique for detecting intruders and zero-day vulnerabilities in software systems. The suggested method employs a Deep Learning (DL) model and an auto-encoder model to find unusual data patterns. Additionally, a model for outlier detection that contrasts the autoencoder model with the single class-based Support Vector Machine (SVM) technique will be developed. The dataset of known vulnerabilities and intrusion attempts will be used to train and assess the models.

Keywords: zero-day vulnerabilities, machine learning, autoencoder model, neural network, intrusion detection

Received on 07 November 2023, accepted on 21 January 2024, published on 30 January 2024

doi: 10.4108/eetiot.4978

*Corresponding author. Email: vishnuvigneshvicky@gmail.com

1. Introduction

A software susceptibility referred to as a "zero-day vulnerability" is one that neither the Programme vendor nor the general public are aware of [1]. This indicates that the vulnerability is unpatched or unfixed and that attackers can use it to compromise the targeted system or programme. Figure 1. shows the life cycle of a zero-day vulnerability.

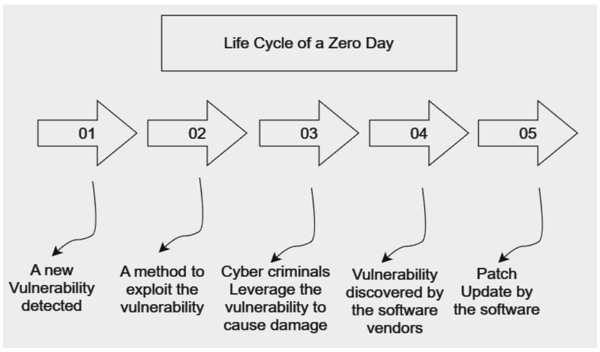
Life Cycle of a Zero Day
01
02
03
04
05
A new Vulnerability detected
A method to exploit the vulnerability
Cyber criminals Leverage the vulnerability to cause damage
Vulnerability discovered by the software vendors
Patch Update by the software

Figure 1. Life Cycle of a Zero Day

Because they allow attackers to attack a system before a patch or fix is released, zero-day vulnerabilities can be extremely harmful. This implies that the assault can take place covertly and cause damage up until a fix is identified. Cybercriminals may come across zero-day vulnerabilities, or they may be identified by researchers who then alert the vendor. Researchers may work with the vendor to build a patch or remedy, while cybercriminals may leverage the vulnerability for their own evil goals [2]. Any sort of software, including operating systems, apps, and web browsers, might have zero-day vulnerabilities. Numerous techniques, such as code analysis, fuzz testing, and reverse engineering, can be used to find them. Software must always be updated with the most recent fixes and additions in order to minimise the risk of zero-day vulnerabilities. This procedure is crucial to guarantee that software like firewalls and IDSs.

ML is becoming a more widely used technology for finding zero-day vulnerabilities because traditional security measures are sometimes unable to detect these kinds of assaults. This research's goal is to create an ML-based system that is capable of quickly identifying zero-day vulnerabilities in software systems. The system will employ different ML, including the use of autoencoders and SVMs [3] methods after being trained on a dataset of existing vulnerabilities to find possible zero-day flaws in fresh software releases [4-6]. The outcomes of this study may be used to enhance cybersecurity generally and assist organizations in better defending their systems against zero-day assaults.

The remainder of this document will discuss: Section 2 discusses the literature review and similar works by other people. Section 3 discusses the proposed method, along with the datasets used. Section 4 discusses the results from and evaluation of the proposed systems, in addition to their limitations. At last, Section 5 provides a conclusion to the paper.

2. Literature Review

Joseph et al. proposed a hybrid approach that combined rule-based feature selection with deep feedforward neural networks (DFFNN) for the detection of intrusions in IIoT networks. [7]. *Nikolaos et al.* proposed the use of DL-based detectors for zero-day exploits, which evaluates their performance both with and without transfer learning features [8]. *Fatemeh et al.* proposed a taxonomy that had 4 categories into which deep malware detection and techniques were put: These categories were adversarial resistant, few-shot, semi-supervised, and unsupervised [9]. *Apichit et al.* proposed an Intrusion Detection System (IDS) based on anomaly detection. This model combines K-Means clustering, feature selection, and the XGBoost classification model that enhances the detection of network intrusions [10].

Rania et al. proposed converting pcap files into RGB images and evaluating them using 7 different ML techniques [11]. A hardware Trojan detection technique is proposed by *Priyatharishini et al.*, in which the malicious logic in VLSI circuits is found at the gate level netlist utilising a stacked autoencoder and stacked sparse autoencoder model [12]. *Lirim et al.* proposed a deep neural network utilizing the UNSW-NB15 dataset for detecting intrusions [13].

Shamshair et al. have reviewed different, recent ML algorithms used for the purpose of zero-day vulnerability detection [14]. *Pengzhi et al.* have explored the development of autoencoders in depth, and highlight improved versions of various autoencoders used in various fields [15]. *Rushdan et al.* proposed a system to detect and prevent zero-day attacks for Software-Defined Networks using Cuckoo, a sandbox tool [16]. *Akash et al.* proposed a method to detect the malware family reference number using a Convolutional Neural Network (CNN) model that employed DL based on long short-term memory [17].

3. Methodology

3.1. Datasets

The proposed model is evaluated using two well-known datasets of IDS. The CICIDS2017 dataset produced by CIC from Canada refuges an inclusive range of current outside and inside occurrences of attacks. At last, the original form of data is provided to the researcher to handle the dataset in a flexible way. This dataset is considered for this model to evaluate the performance. The CICIDS2017 dataset contains five days' worth of traffic from benign, insider, and external attacks. It is possible to access the PCAP recordings. The original files in the dataset have undergone pre-processing.

The second dataset that has been used to evaluate the model is NSL-KDD. The problems in the KDD Cup'99 dataset [18] are addressed by using NSL-KDD. It is released by CIC. The dataset of KDD Cup'99 is chosen for more than 50% of IDS evaluations over the preceding 10 years for IDS evaluations and the NSL-KDD dataset for more than 17%. Therefore, a number of difficulties with the KDD Cup'99 are explored in-depth. These problems include unequal class sizes and redundant records. NSL-KDD is thus appropriate for the evaluation objectives of this research and for comparison with related studies. Both normal/harmless traffic were included in the NSL-KDD as well as four types of cyber-attacks: DoS (Denial of Service), U2R (User 2 Root), R2L (Remote to Local), and probing. 'KDDTrain+.csv' and 'KDDTest+.csv' are two files that include the NSL-KDD dataset. The NSL-KDD is offered in CSV file format. The class label and each instance's feature values are displayed together. The feature files have their features encoded categorically for it to be suitable for ML usage. Not only does it study the NSL-KDD dataset, it also analyses it.

3.2. Dataset Pre-Processing

The section explains how the dataset, CICIDS2017, is processed. Map Reduce-based DL model is used for the purpose of detecting the intrusion using spatio and temporal features. To improve the accuracy of feature selection, a black widow-optimized algorithm is used [19]. First, the PCAP files are separated into groups based on

attack types and timestamps. A unique PCAP file is created for each class of attack, and they are able to produce features that could flow bi-directionally.
The characteristics should suit the analysis of traffic, both with and without encryption, due to how much the networks rely on encrypted data and how complex the networks are [20]. Per the analysis of contemporary IDS, bidirectional flow properties are used widely [21]. It is also claimed that features based on flow are more appropriate for the development of IDS.
Thirdly, characteristics with substantial association are removed to decrease model instability. The threshold is set at '0.9'. Features with correlations below the threshold are used for training. The findings are scaled using a Standard Scalar at the end.
As stated, the goal is to assess how well models perform at spotting assaults by utilising benign traffic to train them. Thus, training is only done with safe and safe or typical traffic. 75% of the data is used to train the model, while 25% of the data is used for testing or evaluation purposes. sklearntrain_test_split function is used for this, and the shuffle parameter is True.
The model's capability to identify the anomaly in the case of a zero-day attack is then tested as each attack type simulates one. The evaluation uses attacks from the training and testing files of the NSL-KDD dataset.

3.3. Model based on an Autoencoder

An Artificial Neural Network (ANN) serves as the core of the proposed autoencoder. The architecture of the network (Figure 2.), epoch number, and the rate of learning are all selected using a random search for the optimization of hyperparameters. It is well known that a grid [22] search takes longer to reach a set of semi-optimal parameters than a random search. Furthermore, it is more effective than grid search because only a few criteria are needed. Not to mention, it lessens the possibility of having over-fitted parameters.

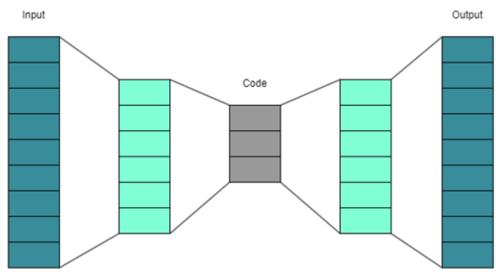

Figure 2. ANN Architecture

As described in Algorithm 2, the hyper-parameters are studied, after which the model is trained. The benign occurrences are broken into 75% training data and 25% testing or validation data. It is then trained for n number of epochs after an ideal ANN design is used to initialize the model. The accuracy and loss curves produced are monitored [23].

It is evaluated using Algorithm 3 when it converges, as depicted in the above image. If the MSE of the decoded and original instances exceeds a threshold that was predefined, that particular instance of attack is flagged as a zero-day attack. For assessment reasons, three different criteria—0.05, 0.1, and 0.15—are considered. These cut-offs are established by hyperparameter optimisation with random search. A zero-day attack is recognised by the value of the MSE values and the threshold.

3.4. Model based on Single-Class SVM

Beneficent cases are used to train a single-class SVM. A "v" value is required in order for the single class SVM to be trained. As shown in Figure 3., the support vectors are positioned on the opposite side of the hyperplane and serve as both the lower and upper bounds for the count of support vectors. When the default value is set to 0.5, the hyper plane contains 50% of the training data. Other numbers (0.2, 0.15, and 0.1) were chosen for the experiment, though. The performance of the autoencoder is evaluated using these results.

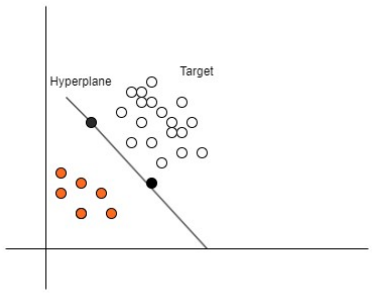

Figure 3. Single Class SVM

The training process for single class SVM is illustrated by the fourth algorithm. The Single Class SVM model is fitted using 75% benign data. When compared to the Autoencoder model, the single class SVM returns the results as 0 or 1, in binary form. The output shows if an instance is part of the class to which the SVM is fitted. As a result, each attack is assessed using the number of cases predicted with a '0' SVM result.

3.5. Algorithms Used

Algorithm 1

```
Correlated features – drop
1. 2D Array of Benign Data and threshold of Correlation are given as inputs and 2D array of Bening Data and Columns that have dropped is extracted as Output.
2: data1.corr().abs() for the correlation_matrix
Correlation matrix[q,w] for the upper matrix
 (I, w ∈ N: I <=w) //Correlation matrix
Drop {I ∈ N: Corr_Mat[q,∗] > threshold}
data1.Drop_Column(drop);
drop returndata
```

Algorithm 2

```
(Autoencoder-Based Model), which takes inputs as "ANN_architecture," "harmless_data," "num_epochs." and "regularization_value"
Autoencoder has been trained as output
Trained 75% with Benign Data
 Training = benign_data and testing
Autoenc_build(ANN_Architect, regularization)
size_batch=1024
Autoenc_train (size_batch, epoch_num, train, test).
 Return autoenc
```

Algorithm 3

```
The training of autoencoder has been done and thresholds and attacks are taken as the inputs for evaluation.
Output: Accuracy of detection
accuracy_detect
predict(a) is s //s-prediction, a-attack
 thresholds, do
 precision (s, attack) > th) / len (a)
accuracy_detect. add(threshold, acc) acc-accuracy
for loop end
returndetection_accuracy
```

Algorithm 4

```
2D Array of Benign Data, Correlation, and N as inputs for One-Class SVM Model.
Threshold
2D Array of Benign Data and Columns which have been dropped as  the Output
 Train with 75% of Benign Data
Training = benign_data and testing
oneclasssvmOneClassSVM(no_value, rbf')
oneclasssvm.fit(train)
returnoneclasssvm
```

4. Results

4.1. Evaluation of Autoencoder

For the training of the autoencoder, 75% of the harmless events are used. The input and output layers of the ANN network for CICIDS2017 have a total of 18 neurons. 1024 is recommended as the batch size. 50 epochs, MSE loss, and a 0.001 regularisation of L2 make up the additional parameters that are ideal for this purpose [24].
It is important to establish that the definition of accuracy for benign is different. In contrast to attacks, the accuracy of the benign class represents the proportion of cases correctly identified as non-zero-day, which reflects specificity. For the attack classes, it indicates recall.
It is observed that the benign accuracy for the threshold of 0.05 is 81.10%, 90.45% for a threshold of 0.10, and 95.21% for a threshold of 0.15. It is also emphasized that there are three different categories for the accuracy of detection of different attacks. The detection accuracy for harmful attacks is high [over 90%], independent of the threshold. Second, lower limits for classes that only slightly deviate from harmless show improved accuracy. This emphasises the function of the threshold. Thirdly, albeit less precisely, classes that cannot be differentiated from safe traffic are also recognised. An example is SQL injection attack.

4.2. Evaluation of Single Class SVM Results for CICIDS2017

The single class SVM results are shown in Table 1. below. By examining this, it is possible to make the following two claims: (1) changing the v value has little impact on the accuracy of detection; and (2) the classes that had a high rate of detection in the results of the Autoencoder are additionally identified by the single class SVM, but it does not identify the remaining two groups. This arises as a result of limitations in the single class SVM approach, which tries to fit a hyper plane that is shaped like a sphere to differentiate the harmless class from others while always designating classes within it as benign/normal.

Table 1. Results of Single Class SVM

Class	Table column heading		
Threshold	Subheading	Subheading	Subheading
Benign	89.76%	84.79%	79.66%
Bruteforce (FTP)	10.14%	15.11%	20.24%
Bruteforce (SSH)	79.46%	80.21%	80.90%
Slowloris (DoS)	7.61%	8.33%	10.31%
GoldenEye (DoS)	71.82%	72.34%	72.80%
Hulk (DoS)	90.64%	91.30%	91.50%
Slowhttps (DoS)	98.54%	98.61%	98.66%
DDoS Attacks	39.30%	39.89%	40.91%
Heartbleed	99.44%	99.49%	99.53%
BF (Web)	20.6%	23.46%	35.79%
XSS (Web)	9.53%	9.71%	10.08%
SQL (Web)	5.72%	6.26%	6.80%
Dropbox 1 (Infiltration)	38.84%	38.84%	38.84%
Dropbox 2 (Infiltration)	29.36%	35.24%	35.24%
Dropbox 3 (Infiltration)	57.09%	57.09%	57.09%
Cooldisk (Infiltration)	92.10%	93.30%	94.86%
PortScan	59.22%	46.10%	49.95%
Botnet	44.18%	46.10%	49.95%

To further visualize this, refer to Figure 4. below. With One-Class SVM, identifiable zero-day threats can be found. However, autoencoders have a significantly higher performance rating, which makes them more appropriate for sophisticated zero-day vulnerabilities. The efficiency of both the Autoencoder and the single class SVM are also contrasted in Figure 4. on a class-wide basis. The results are plotted using a threshold of 0.15 in the autoencoder and v=0.2 in the One-Class SVM.

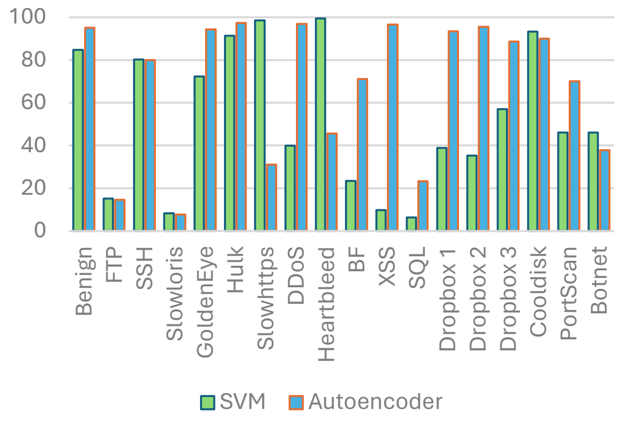

Figure 4. Comparison – SVM vs Autoencoder

The input and output layers of the ANN has a total of 122 neurons. There are also 3 hidden layers which contain 100, 60, and 100 neurons in each, which make up the NSL-KDD dataset's autoencoder-optimized design. A batch size of 1024 is advised. The average absolute error value of loss, fifty epochs, and 0.001 regularisation of L2 are additional ideal parameters.

4.3. Limitations and Future Work

The proposed model evaluates the system using CICIDS2017 and KSL-KDD datasets. While these datasets are commonly used for IDS evaluations, they are small datasets and might not include all the real-world scenarios. This also makes the system unable to detect unknown vulnerabilities, if there are any. Even though the system was evaluated using small datasets, it should be possible to scale it to using fairly large datasets as well. In future work, we plan on evaluating the system using larger datasets and datasets that cover a specific kind of vulnerability. This will also test the scalability of the proposed system to larger datasets more thoroughly. False positives and false negatives exist. However, the rates are minimal. In future work, we also plan on minimizing this further to increase the overall accuracy and reliability of the proposed system.

5. Conclusion

An innovative outlier-based strategy for identifying zero-day hacks is suggested by the research presented in this paper. The primary aim was to build an intelligent and effective IDS model that could detect zero-day cyber-attacks. This was to be achieved with higher accuracy while also overcoming the obstacles presented by present IDS models. The autoencoder model for locating zero-day attacks is examined in this research. The findings from the datasets used showcase the autoencoder model's high detection accuracy.

A Single Class SVM is an unsupervised outlier-based ML technique that is commonly used to compare the autoencoder model. Both the models have low false-positive and false-negative rates. This means that both models are achieving high accuracy and reliability. However, while the Single Class SVM is effective in identifying zero-day attacks in the data sets used in this research, the autoencoder-based model proves to be more effective in detecting said attacks.

References

[1] Kaloudi, Nektaria, Li, Jingyue. The AI-Based Cyber Threat Landscape: A Survey. ACM Computing Surveys (CSUR). 2020; 53(1):1-34.

[2] Rajesh KP, Santhi P. Unified DL approach for Efficient IDS using Integrated Spatial–Temporal Features. Knowledge-Based Systems. 2021; 226:107-132.

[3] Chen P, Lin C, Schölkopf B. A Tutorial on v-support vector machines. Applied Stochastic Models in Business and Industry. 2005; 21:111-136.

[4] Jayachitra S, Prasanth A, Rafi, Shaik Mohammad, Zulaikha Beevi S. Hierarchical-Based Binary Moth Flame Optimization for Feature Extraction in Biomedical Application. In: Khare, Nilay, Tomar, Deepak S, Ahirwal, Mitul K, Semwal, Vijay B, Soni, Vaibhav, editors. Machine Learning, Image Processing, Network Security and Data Sciences. Proceedings of the 4th International Conference on ML, Image Processing, Network Security and Data Sciences; 2022. Springer Nature Switzerland; 2022. p. 27-38.

[5] Jayachitra S, Prasanth A, Hariprasath S, Benazir Begam R, Madiajagan M. In AI Models for Blockchain-Based Intelligent Networks in IoT Systems: Concepts, Methodologies, Tools, and Applications. Springer International Publishing; 2023. Chapter 7, AI Enabled Internet of Medical Things in Smart Healthcare; pp. [141-161].

[6] Kavitha M, Roobini S, Prasanth A, Sujaritha M. Machine Learning and Artificial Intelligence in Healthcare Systems. 1st Edition. Boca Raton: CRC Press; 2023. Systematic View and Impact of Artificial Intelligence in Smart Healthcare Systems; pp. [25-56].

[7] Bamidele, Awotunde, Chakraborty, Chinmay, Adeniyi, Emmanuel. Intrusion Detection in Industrial Internet of Things Network-Based on Deep Learning Model with Rule-Based Feature Selection. Wireless Communications and Mobile Computing. 2021; 2021:1-17.

[8] Peppes N, Alexakis T, Adamopoulou E, Demestichas K. The Effectiveness of Zero-Day Attacks Data Samples Generated via GANs on Deep Learning Classifiers. Sensors. 2023; 23:900.

[9] Deldar F, Abadi M Deep Learning for Zero-day Malware Detection and Classification: A Survey. ACM Comput. Surv. 2023; 56(2):36.

[10] Pattawaro, Apichit, Polprasert, Chantri. Anomaly-Based Network Intrusion Detection System through Feature

Selection and Hybrid Machine Learning Technique. In: Proceedings of the ICT Knowledge Engineering (ICTKE) Conference; November 2018. p. 1-6.
[11] Musleh D, Alotaibi M, Alhaidari F, Rahman A, Mohammad RM. Intrusion Detection System Using Feature Extraction with Machine Learning Algorithms in IoT. Journal of Sensor and Actuator Networks. 2023; 12(2):29.
[12] Priyatharishini M, Nirmala. A DL-based malicious module identification using stacked sparse autoencoder network for VLSI circuit reliability. In: Measurement (Ed.). Measurement: Proceedings of the Elsevier Conference, 15 May 2022. International Measurement Confederation (IMEKO); 2022. p. 18.
[13] Lirim A., Cihan D. Network IDS using DL. Procedia Computer Science. 2021; 185:239-247.
[14] Ali S, Rehman SU, Imran A, Adeem G, Iqbal Z, Kim KI. Comparative Evaluation of AI-Based Techniques for Zero-Day Attacks Detection. Electronics. 2022; 11:1-17.
[15] Li P, Pei Y, Li J. A comprehensive survey on design and application of autoencoder in deep learning. Appl. Soft Comput. 138(C):21.
[16] Rushdan, Huthifh, Shurman, Mohammad, Alnabelsi, Sharhabeel, Qutaibah, Althebyan. Zero-Day Attack Detection and Prevention in Software-Defined Networks. In: Proceedings of the Advanced Computer and Information Technology (ACIT) Conference, December 2019.
[17] Akash S, Prabahara P, Vijay K, Soman KP. A Detailed Investigation and Analysis of DL Architectures and Visualization Techniques for Malware Family Identification. Cybersecurity and Secure Information Systems. 2019; 17:241-286.
[18] Tavallaee M, Bagheri E, Lu W, Ghorbani A. A detailed analysis of the KDD CUP 99 data set. In: Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA). 2009. p. 30-36.
[19] Kanna P, Rajesh, Santhi P. Hybrid Intrusion Detection using MapReduce based Black Widow Optimized Convolutional Long Short-Term Memory Neural Networks. Expert Systems with Applications. 2022; 194:27-43.
[20] Rezaei S, Liu X. Deep Learning for Encrypted Traffic Classification: An Overview. IEEE Communications Magazine. 2019; 57(1):76-81.
[21] Aceto G, Ciuonzo D, Montieri A, Pescapè A. Toward Effective Mobile Encrypted Traffic Classification through Deep Learning. Neurocomputing. 2020; 409.
[22] Liashchynskyi P, Liashchynskyi P. Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS. arXiv. 2019:1-8.
[23] Abri F, Siami-Namini S, Khanghah MA, Soltani FM, Namin AS. Can Machine/Deep Learning Classifiers Detect Zero-Day Malware with High Accuracy In: Proceedings of the 2019 IEEE International Conference on Big Data (Big Data); December 2019; Los Angeles, CA, USA. p. 3252-3259.
[24] Hindy H, Atkinson R, Tachtatzis C, Colin J-N, Bayne E, Bellekens X. Utilising Deep Learning Techniques for Effective Zero-Day Attack Detection. Electronics. 2020; 9(10):1684.