Journal of Sensor and Actuator Networks

MDPI

Article

Effective One-Class Classifier Model for Memory Dump Malware Detection

Mahmoud Al-Qudah [1], Zein Ashi [2], Mohammad Alnabhan [1] and Qasem Abu Al-Haija [1,*]

[1] Department of Cybersecurity/Computer Science, Princess Sumaya University for Technology, Amman 11941, Jordan
[2] Princess Sarvath Community College, Amman 11941, Jordan
* Correspondence: q.abualhaija@psut.edu.jo

Abstract: Malware complexity is rapidly increasing, causing catastrophic impacts on computer systems. Memory dump malware is gaining increased attention due to its ability to expose plaintext passwords or key encryption files. This paper presents an enhanced classification model based on One class SVM (OCSVM) classifier that can identify any deviation from the normal memory dump file patterns and detect it as malware. The proposed model integrates OCSVM and Principal Component Analysis (PCA) for increased model sensitivity and efficiency. An up-to-date dataset known as "MALMEMANALYSIS-2022" was utilized during the evaluation phase of this study. The accuracy achieved by the traditional one-class classification (TOCC) model was 55%, compared to 99.4% in the one-class classification with the PCA (OCC-PCA) model. Such results have confirmed the improved performance achieved by the proposed model.

Keywords: Novelty-class; One-class SVM (OCSVM); Memory dump; Malware; Principal Component Analysis (PCA); Dimensionality Reduction

Citation: Al-Qudah, M.; Ashi, Z.; Alnabhan, M.; Abu Al-Haija, Q. Effective One-Class Classifier Model for Memory Dump Malware Detection. *J. Sens. Actuator Netw.* **2023**, *12*, 5. https://doi.org/10.3390/jsan12010005

Academic Editor: Lei Shu

Received: 26 October 2022
Revised: 6 January 2023
Accepted: 11 January 2023
Published: 17 January 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The progress and development of computer information systems and their indivisible connection to our lives have dramatically increased the motivation of attackers to target these systems [1,2]. Malicious software (malware) widely affects computer information systems in various ways, interrupting their normal operations, harming and removing files, programs, or services, and allowing illegitimate access to sensitive and private information.

In particular, malware disrupts regular user activity in computer systems by conducting undesired or harmful actions [3]. According to the AV-Test Institute, the number of malware attacks on operating systems has increased by 722.505 million since 2022, compared to 13.365 million in 2008 [4]. Many malware devices, such as ransomware, spyware, rootkits, worm, viruses, bots, botnets, Trojan horses, and other malware types, exist and target many parts of information systems, especially memory dump files. Memory dump aims to discover faults within working applications or programs.

Memory dump files regularly contain information on the final state of programs and applications. Memory areas and program status are considered points of interest for attackers to steal vital information such as passwords and encryption keys, causing a breach and a major threat to confidentiality, integrity, and authenticity [5]. Manual detection methods of memory dump attacks are linked with limited capability due to low-accuracy rate and time-consuming issues, This can be developed by different machine learning systems, where training data is used to generate the most rapid and accurate evaluation possible [6]. In contrast, a few machine learning techniques focus on speed, and others focus on precision and accuracy. Subsequently, it is crucial to select the most capable machine

learning model for the memory dump context that can achieve the optimal model results [7,8].

Obfuscated malware [9] is malware that hides to avoid detection and extermination. In this research, we employed a dataset generated from an obfuscated malware dataset simulating real-world situations. "MalMemAnalysis2022" is a balanced dataset designed to test obfuscated malware detection methods through memory and is made up of spyware, ransomware, and Trojan malware. The structure of the "MalMemAnalysis2022" dataset is shown in Figure 1.

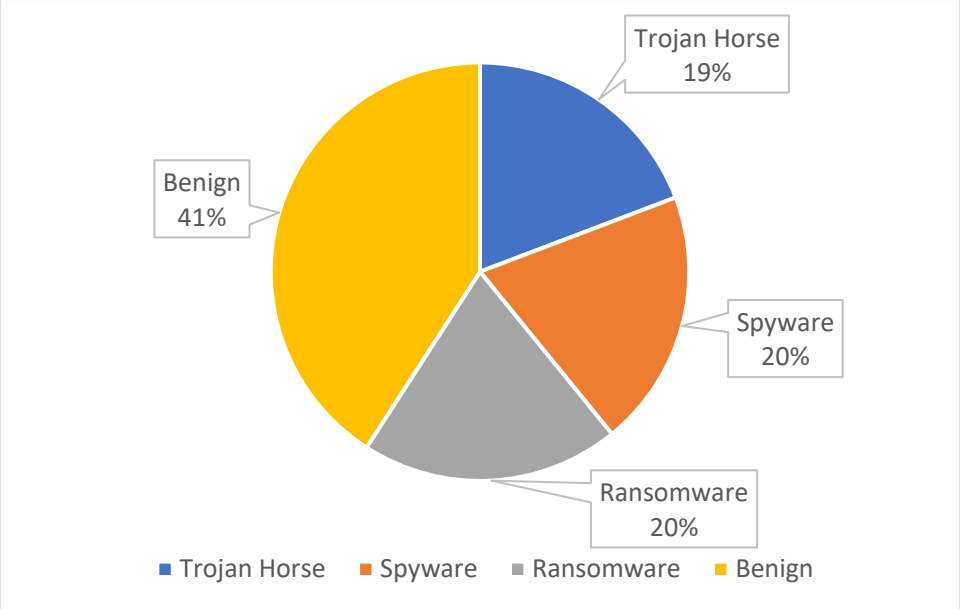


Figure 1. The Memory Dump Dataset Structure.

This paper aims to provide an enhanced memory dump detection model by increasing the model's sensitivity, improving its ability to be generalizable, and improving its efficiency in detecting different types of memory malware, especially zero-day malware. The OCSVM classifier is utilized to identify any deviation from the normal memory dump file patterns. In addition, a technique of dimensionality reduction, PCA, is combined with the OCSVM training phase to achieve the desired goals.

The "MalMemAnalysis2022" dataset [7] is utilized in two scenarios, as illustrated in Figure 2. The first scenario, known as TOCC, is to train an OCSVM classifier and evaluate its performance. The second scenario, OCC-PCA, is to reduce the dataset dimensionality using the PCA technique before training the OCSVM classifier. Several accuracy matrices have been used to evaluate the performance of both scenarios and to determine whether the PCA improved the standard OCSVM performance. The proposed OCSVM model achieved results and was compared with results reported in related studies for benchmarking and validation.

The remaining part of this paper is structured as follows: Section 2 provides a background of memory dump attacks, the concept of novelty detection, the OCSVM, and PCA classifiers. Section 3 demonstrates recent related work, whereas Section 4 comprehensively describes the methodology. Section 5 describes and analysis the evaluation results. Section 6 is the conclusion.

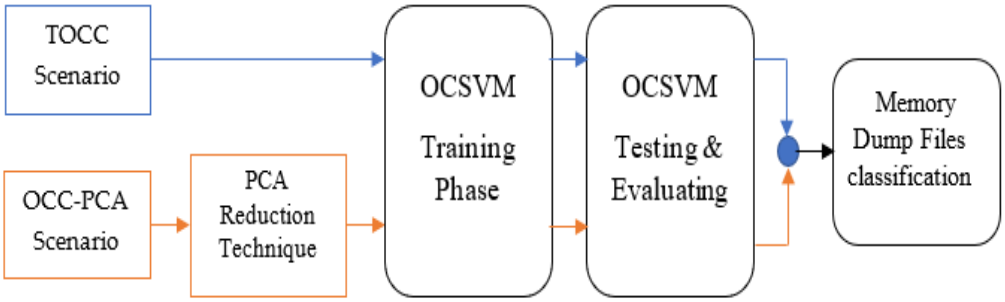


Figure 2. The Proposed Scenarios for OCC-PCA and TOCC Models.

2. Background

2.1. Memory Dump and Their Attacks

A memory dump occurs when all of the information in RAM is written onto a hard drive. Memory dumps are widely used to collect diagnostic information during a crash to aid in debugging and learning more about the event and to help solve problems with operating systems and other programs. Many computer problems are unfixable because they need a reboot, yet the code that caused the problem is still stored in RAM at the time of failure. Due to the volatile nature of RAM, memory dumps save data that could otherwise be lost or overwritten. Some customers are concerned about privacy because these dumps might contain anything in the computer's dynamic RAM. Dumps may pose a security concern since they are saved on the hard drive. Hackers may be able to extract cleartext passwords or decryption keys from a memory dump that would be hard to obtain otherwise [1].

Some malware encrypts the user's data and demands payment to access the key needed to recover this information, known as ransomware. In the same respect, spyware attempts to compromise machines allowing for the surveillance of various system activities that could significantly compromise the user's personal information. Trojan horse is a program that appears to be helpful and legitimate; however, it contains a secret code that, when executed, performs an undesirable or dangerous function. Trojan horses perform tasks that the attacker cannot work on directly.

2.2. Novelty Detection

The detection of unique or uncommon data within a dataset is known as novelty detection. A machine learning system might be trained entirely on correct data to classify this data appropriately in novelty detection. However, one-class classification achieves novelty detection, requiring distinguishing one class (the specified normal, positive class) from all other alternatives. The positive type is commonly considered well-sampled, whereas the opposite category is drastically under-sampled [10,11]. The novelty detection approach provides the ML model with more flexibility. It can be generalized to unknown memory dump patterns as the approach focuses on the deviation from the target class to detect any outlier as an anomaly [12].

2.3. OCSVM and PCA Classifier

OCSVM classifier uses techniques for identifying outlier data and creating a boundary to separate the numeric values from the rest of the input space. In addition, the domain of the minor class is measured. Data points outside parameters are considered outliers [13]. OCSVM applies SVM concepts to one-class settings. Kernels that perform dots between points from the input data determine the distance in a high-dimensional space [14]. The PCA is a classifier that utilizes a statistical manner to reduce the dataset dimensions. This involves creating uncorrelated parameters and selecting from a linear collection of the input features from the original dataset. Hence, the variance will increase, and relevant

features will be produced from the entire dataset. PCA relies on eliminating all features that are not necessary and focuses on the crucial features that produce more robust results. In this paper, the PCA classifier handles dataset features to minimize the feature number, which can contribute to enhancing the OCSVM classifier in the training and testing phases. The PCA is a classifier that utilizes a statistical manner to reduce the dimensionality of the feature. Uncorrelated features are extracted from a linear collection of the input features, maximizing their variance and improving the ML model's performance. In addition, low dimensional feature space reduces computational costs and memory usage [15].

In this paper, the PCA classifier handles dataset features to minimize the feature dimensionality that can enhance the OCSVM classifier in the training and testing phase. PCA relies on eliminating all features that are not necessary and concentrating effort on the crucial features that produce more robust results, which helped us achieve the best results.

3. Literature Reviews

Many researchers have employed different one-class classification techniques to detect novel attacks efficiently as soon as they occur. Consequently, most experimental attempts are combined with various enhancement techniques to improve the one-class classifier performance.

This section provides a comprehensive review of studies that adopted different class classification techniques. The focus was on how classifiers are employed, what attempts were used to enhance the performance, the utilized datasets, and what results were achieved.

Ref. [16] proposed an anomaly-based and signature-based NIDS that overcame other systems in terms of reducing false alarm rates and detecting zero-day attacks. To build the NIDS, they presented two parallel subsystems, each utilizing OCSVM. The target class of the first OCSVM subsystem was the normal network packets to detect any outliers, while in the second OCSVM subsystem the target class was the attack packets to identify the known attacks. This research used the "KDDCUP-99" dataset for the training phase. Then the optimal features were selected using the Pigeon Inspired Optimizer (PIO) from the training sets. In the proposed methodology, the two subsystems were integrated in a parallel manner to judge each packet; in this way, false alarms were reduced, and detection rates were increased. For evaluation, "KDD CUP-99", "NSL-KDD," and "UNSW-NB15" datasets were used in the proposed NIDS testing phase. This research was carried out with the "KDD CUP-99" dataset: (99.7%) accuracy rate, (99.8%) DR, and (0.02%) FPR.

The authors of [17] proposed using SIMCA and OCSVM models to identify impurities in cassava starch. Both models used a one-class classification technique. The SIMCA model used PCA instead of the OCSVM model, which used the OCSVM classifier. The evaluation results showed the two models' accuracy rates, 78.8%, and 86.9%, respectively, in forecasting benign data. In the same respect, OC-SVM, WOC-SVM-DD, WOC-SVM (ND), and AWOC-SVM classifiers were implemented utilizing eight different datasets.

Preliminary results in ref. [18] confirmed the viability and effectiveness of the WOC-SVM-DD classifier, which improved the weight calculation procedure, and addressed limited sample and high-dimension classification. Further experiments indicated the outstanding performance of OCSVM, 99.3% accuracy, utilizing the banknote authentication dataset.

In [19], an HIDS model that combines the C5 and OC-SVM classifiers was developed and evaluated. The model was tested on the NSL-KDD and ADFA datasets. Three evaluation stages were conducted to reach high-accuracy results; in the last stage, OCSVM with an RBF kernel was applied using LIBSVM to achieve a detection accuracy of 76.4% for the ADFA dataset and 72.17% for the NSL-KDD dataset.

The authors of [20] described a semi-supervised novelty identification technique based on OC-SVM for SMS spam detection. The researchers used a chi-squared feature selection algorithm, and only normal data were trained and had a 98% accuracy rate.

The authors of [21] introduced an unsupervised deep learning strategy for IDS. NSL-KDD and UNSW-NB15 datasets were implemented, and the proposed CAE + OCSVM classifier was combined with a 1D CAE approach to a joint optimization framework. Convolutional auto-encoder and CNN methods were implemented to accomplish significant feature illustrations for both datasets. This method boosts OCSVM's prediction accuracy to 91.58% with the NSL-KDD dataset and 94.28% with the UNSW-NB15 dataset.

In addition, in [22], an anomaly-based NIDS that utilizes unsupervised methods to detect zero-day attacks was presented. Furthermore, unsupervised NIDS demonstrated their capacity to identify unidentified zero-day attacks provided that the malicious traffic diverges from legitimate traffic. The OCSVM produced the highest AUROC scores of 97% on the CIC-IDS-2017 dataset and 94% on the CSE-CIC-IDS-2018 dataset. At the same time, PCA achieved a good classification performance with the lowest recorded AUROC of 84% on the CSE-CIC-IDS-2018 dataset.

Moreover, in [23], a cutting-edge NIDS model that used the OCSVM technique was developed and tested. The suggested approach relies on identifying regular traffic. An accuracy rate of 97.61% was achieved in an experiment using a recent honey network. Consequently, according to the experimental findings, OCSVM has 97.6% accuracy for predicted benign behavior.

In [24], the authors proposed an unsupervised learning model memory augmented auto-encoder (MemAE); performance results of the model proposed were compared with OCSVM and AE models. All three models were trained on benign records and implemented using UNSW-NB15, NSL-KDD, and CICIDS 2017 datasets. The OCSVM model accuracy rate values were 94% on NSL-KDD, 81% on UNSW-NB15, and 76% on CICIDS 2017 datasets. Accordingly, MemAE was proposed as a solution to the over-generalization problem of auto-encoders.

Furthermore, in [25], a unique method for exploiting PMU data to detect cyber-attacks on smart grids was proposed and built. It uses publicly accessible datasets on power system hacks and is based on semi-supervised anomaly identification. A performance comparison was carried out between four semi-supervised algorithms and four supervised algorithms. The four semi-supervised techniques were set up with PCA and a deep auto-encoder feature extraction approach. The OCSVM classifier had 84%, 85%, and 86% accuracy rates for all features, PCA, and DAE, respectively.

The authors of [26] built an anomaly-based NIDS and then proposed an improvement to achieve a higher intrusion detection rate utilizing only the normal class of the KDD99 dataset. Basically, unsupervised training was conducted with an OCSVM algorithm. To obtain the best performance with minimal false alarms, they applied a nested approach to the OCSVM algorithm to find the optimum hyperparameter for it. The latter is called the "data-driven approach for intrusion detection using nested OCSVM." The two approaches were evaluated, and the nested OCSVM approach achieved the best results: (12%) and (98.25%) for FP and accuracy rates, respectively.

A hybrid SAE-1SVM model was proposed by [27], where the stack autoencoder and the OCSVM algorithm were merged. The authors aimed to construct an NIDS that detects DDoS attacks in software-defined networks (SDNs). The unsupervised anomaly detection approach was based on only the legitimate traffic flows represented in the CICIDS2017 dataset. First, feature dimensionality reduction was performed using the stack autoencoding algorithm. Second, the OCSVM algorithm was trained with the resulting low-dimensional feature set. The proposed model has proven its efficiency with real-time detection along with its effectiveness in detecting DDoS attacks with an accuracy rate of (99.35%), (99.97%) for the precession score, (98.28%) for the recall score, and (99.11%) for the F1 score.

Looking at the literature reviews mentioned, obviously, the researchers who combined the OCSVM algorithm with a dimensionality reduction technique achieved higher accuracy rates. The core point is eliminating all features that are not necessary and focusing on the crucial features to produce more robust results. Therefore, this work uses a PCA classifier to extract vital features from every characteristic of the dataset and improve OCSVM performance. The achieved results reached a 99.4% accuracy rate. Table 1 shows a comparison of the proposed approach with other related works.

Table 1. A Comparison of the Proposed Approach with other Existing OCSVM-Based Models.

References	Dimensionality Techniques	Dataset	Accuracy Rate	Number of Instances	Number of Features
[16]	Cosine PIO	KDD CUP-99	99.70%	145,584	40
[17]	PCA	Cassava starch samples	86.90%	244	17
[18]	Manual	banknote authentication samples	99.30%	282,910	5
[19]	Manual	ADFA	76.40%	37,000	7
[20]	chi-squared	SMS Spam collection	98.00%	5574	200
[21]	AE, CNN	UNSW-NB15	94.28%	257,673	42
[22]	Manual	CIC-IDS-2018	97.00%	1,396,787	67
[23]	Manual	honey network	97.61%	41,770	25
[24]	Manual	NSL-KDD	94.00%	125,973	121
[25]	DAE, PCA	power system samples	86.00%	150	29
[26]	OCSVM nested	KDD CUP-99	98.25%	4601	41
[27]	Stack autoencoding	CIC-IDS-2017	99.35%	691,406	80
Proposed	OCC-PCA	MalMemAnalysis-2022	99.40%	58,027	10

4. Methodology

This section describes the study's methodology and practice. Many phases were implemented to accomplish the desired goals and objectives. The pre-processing stage was applied to the dataset in five steps: partitioning the dataset, handling missing values, removing duplicate entries, and encoding. The OCC-PCA model utilized the PCA classifier on the dataset to minimize the number of features from 53 to 10 after the complete preprocessing phase. However, 53 features from a dataset were retained by the TOCC method. The OCSVM classifier was then implemented for two models using dataset training samples to train and learn. Two models will then be tested with the OCSVM classifier.

The MalMemAnalysis2022 dataset was created in 2022 to imitate real-world settings similar to malware seen in the real. Collecting malicious and benign dumps, the malMemAnalysis2022 dataset consists of 58,596 records, with 29,298 benign and 29,298 attack records, including 56 features and three main categories of memory dump malware (ransomware, spyware, and Trojan horse)[7]. Many reasons for implementing this recent dataset are considered since it is a balanced binary classification dataset with a few missing values. In addition, we are among the earlier researchers to use this dataset. This methodology employs the Python programming language, and the OCSVM classifier is utilized to determine the experimental environment. After the testing step, the method offers information on the performance matrices used to assess the findings. Figure 3 depicts the methodology of the two proposed approaches.

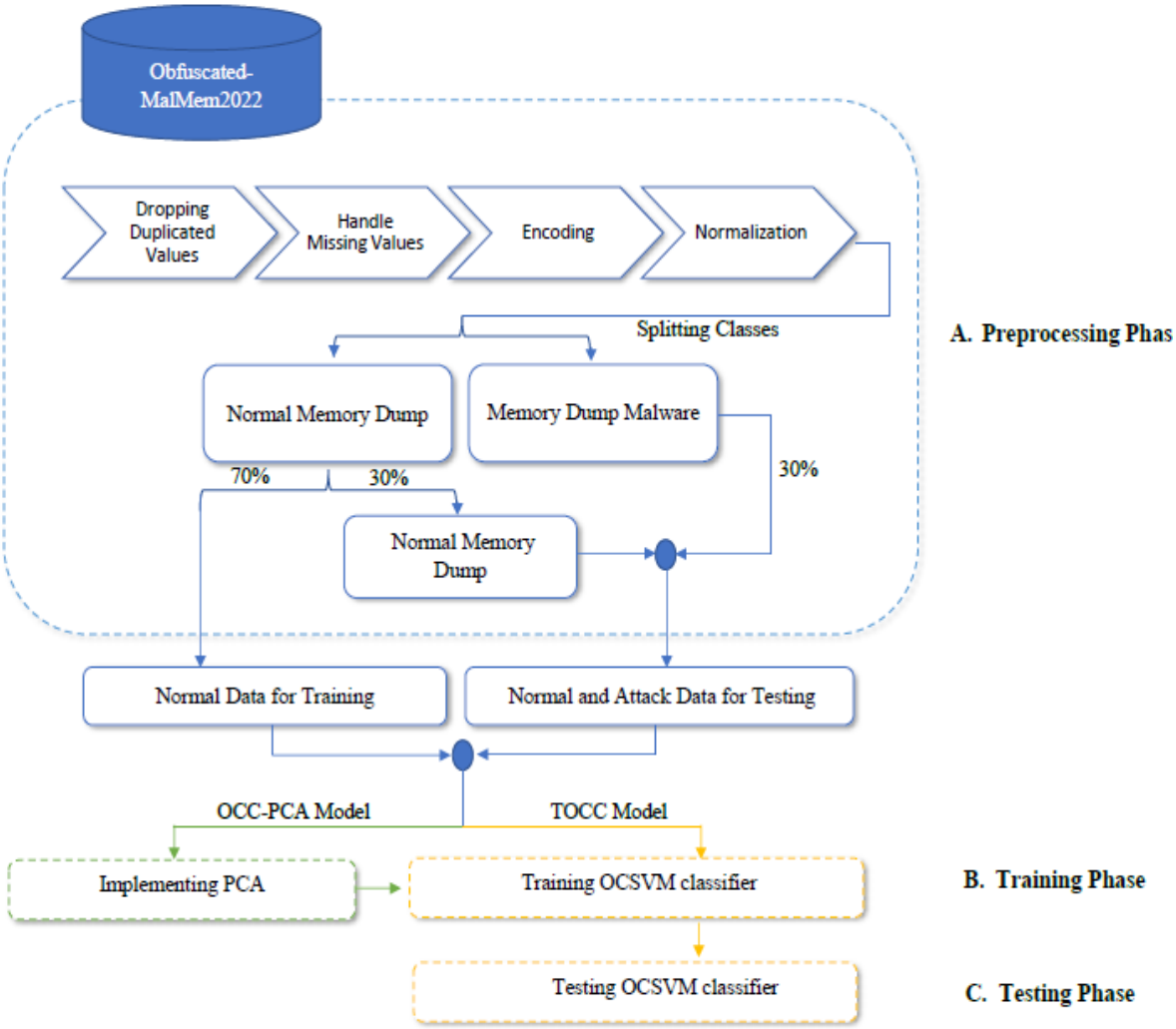


Figure 3. The Methodology of the Proposed Approaches.

4.1. Dataset Pre-Processing Phase

The dataset goes through various processes in this phase to remove noise and then adapt it to the chosen ML methods, as mentioned in Figure 3.

4.1.1. Dropping Duplicate Values

All duplicate rows with missing values were dropped. This stage eliminates duplicate features in the dataset for two approaches by removing the redundant feature column. The first column feature, known as "category," with the type "object," was dropped. Another two columns named "handles. nport" and "svcscan.interactive_process_services" were dropped since they have zero numbers; the dataset, after removing three columns, has 53 features; the last column, formerly known as a "class" with the type "object" was renamed "label".

4.1.2. Handle Missing Values

The dataset's missing values and duplicated rows are dealt with in two approaches at this level. Depending on their implications, missing values were adjusted by eliminating entire rows or replacing them with appropriate values as median [28].

4.1.3. Encoding Dataset

To be acceptable for the two approaches, 0 and 1 numbers define all benign attacks.

4.1.4. Value Normalization

The dataset's independent values are not fairly distributed; all values are set closely together to normalize with the classifier for the best accuracy and measurement performance matrices.

4.1.5. Dataset Partitioned

After completing all the previous steps in the pre-processing phase, all duplicate rows, duplicate features, and missing values are dealt with by eliminating, dropping, or replacing. The dataset after that was divided into 28,796 records for memory dump malware and 29,231 for benign records. To prepare the OCSVM classifier to be trained and tested, the dataset depends on more samples for the training phase to make the model more generalizable. Malware records are divided into 30%, with 8770 records utilized for the testing phase, while the remaining 70%, with 20,026 records, remain inactive. Benign records were split into 70%, with 20,461 records for training the classifier, and 30%, with 8770 records, for the testing phase. Table 2 depicts the number of records in the training and testing phase.

Table 2. The Number of Records in the Training and Testing Phase.

Type of Records	Training Phase	Testing Phase
Benign	20,461	8770
Attack	Null	8770

4.2. Training Phase

After completing dataset pre-processing, the OCSVM classifier was set to depend on benign records for each model to train the classifier. Twenty-nine thousand two hundred thirty-one benign records of the dataset were split into two halves; 70% of benign records were to be utilized for the training classifier on both models, while 30% were to be utilized for the testing phase, as mentioned in Figure 3. On another side, the experimental setup of two models was implemented in a 64-bit Windows 11 pro computer with 12 GB RAM and 1.80 GHz CPU; the ML models were implemented using Python 3.8, and SPYDER 4.2.5 provide libraries, Panda, Scikit-learn, and Numpy. Table 3 depicts the experimental setup for other related works.

Table 3. The Experimental Setup for Related Works.

References	O.S	RAM	Programming Language	Tool
[16]	-			
[17]	For Windows 10 Enterprise, an Intel Core i7-6700K processor was used to train OC-SVM models	64 GB RAM	A script supplied by Cardillo was used to conduct McNemar's test	MATLAB R2016b. PLS Toolbox 8.1. The Data Description Toolbox version 2.1.2 and the toolbox LIBSVM 3.23 for MATLAB
[18]	Personal computer running on Windows 10 and equipped with an Intel(R) Core(TM) i5-7400 CPU clocked at 3.00 GHz	-	-	-
[19]	-	-	-	-
[20]	Ubuntu Linux 16.04 64-bit machine	4 GB RAM	Scikit Learn	
[21]	A personal computer running on Windows 10 and Intel Core i7-8565H processor clocked at 1.8GHz	128 GB RAM	Keras and TensorFlow libraries with Python 3	Jupyter

[22]	Distributed job-based platform. Each project obtained 4 Intel(R) Xeon(R) Silver 4108 CPUs running at 1.80GHz A GeForce GTX 1080 Ti GPU	16 GB RAM	NA	NA
[23]	Ubuntu 16.05 LTS servers	NA	NA	Google Cloud
[24]	NA	NA		
[25]	NA	NA	Python, PyOD	a Python
[26]	NA	NA	NA	NA
[27]	NA	NA	NA	NA
OCC-PCA	In a 64-bit Windows 11 pro computer and 1.80 GHz CPU	12 GB RAM	Provide libraries, Panda, Scikit-learn, and Numpy	Python 3.8 and SPYDER 4.2.5

4.2.1. The OCC-PCA Approach

Automated dimensional reduction is implemented by utilizing a PCA classifier to select ten features (n = 10) out of 53 feature selections from the dataset where "n" is the number of feature extractions; then, the OCSVM classifier is trained on 20,461 benign records. To select the best n value that suits the number of available features, an experiment with n = 30, as in [25,29], is conducted. Furthermore, in an attempt to decrease the time consumed, another experiment is conducted with n = 10. The python code for implementing the PCA is shown in Figure 4.

```
n=10
pca = TruncatedSVD(n_components=n)
Xp=pca.fit_transform(X_train)
pca.explained_variance_ratio_
scaler = StandardScaler()
Xp = scaler.fit_transform(Xp)
Xt = pca.transform(X_test)
Xt = scaler.transform(Xt)
```

Figure 4. Python Code for Implementing the PCA Technique.

The results of the two experiments are shown in Table 4. The results illustrate that setting n to 10 is the best for achieving a higher accuracy rate and lower consumption time.

Table 4. The Results of Choosing the Best Number of Features for the PCA Classifier.

Experiment	Precision	Recall	F1 Score	Accuracy	Consumed Time (s)
n = 30	99%	99%	99%	98.8%	1
n = 10	99%	99%	99%	99.4%	0.23

4.2.2. The TOCC Approach

This approach was trained using the identical 20,461 benign records without utilizing any dimensional reduction techniques and dealt with 53 features from the dataset after the pre-processing phase to train the OCSVM classifier.

4.3. Testing Phase

Testing the OCSVM classifier for both novelty-class techniques follows the completion of OCSVM classifier training for two approaches. The OCC-PCA and TOCC models use 8770 benign and 8770 malware records, totaling 30% benign and 30% malware

records, respectively. A total of 17,540 data were employed. The result demonstrates the prediction error for the final model and the suggested model's generalizability, as mentioned in Figure 4.

5. Analysis and Evaluation of the Results

In the evaluation phase, confusion matrix variables consist of four variables, detecting benign records correctly as true negative (TN), correctly detecting attack records as true positive (TP), incorrectly recognizing benign records as false positive (FP), and incorrectly recognizing attack records as false negative (FN) [30]. This was developed by comparing actual label values to predicted label values in the testing phase and measuring performance matrices for both models. Recall value is a way to measure how many predicted positives were TPs. There is an inverse relationship between sensitivity/recall and FN alarm rates [31]. The authors mentioned that the model that produces no FN alarms has a recall score of 1. The proposed model seeks to reduce the false negative rates (FNRs) by obtaining the highest possible sensitivity. The recall score shown in Equation (1) indicates the sensitivity of the classifying model [32].

$$Recall\ value/Sensitivity = \frac{TP}{TP + FN} \quad (1)$$

Furthermore, it aims to increase the true negative rates (TNRs) by obtaining the highest possible specificity measured by Equation (2).

$$Specificty/TNR = \frac{TN}{TN + FP} \quad (2)$$

The precision value can measure the actual positive results; in other words, it measures the true positive rates (TPRs) among all projected positive results. Equation (3) represents the precision value [33].

$$Precision/TPR = \frac{TP}{TP + FP} \quad (3)$$

Precision and recall are both used to measure the F1 value. Equation (4) represents the F1 measure.

$$F1 = 2\left(\frac{Precession\ X\ Recall}{Precession\ +\ Recall}\right) \quad (4)$$

The percentage of accurately predicted events from all predicted events, whether positive or negative, is measured by accuracy value. Equation (5) represents the accuracy value [33].

$$Accuracy\ rate\ (AR) = \frac{TP+TN}{TP+FN+FP+TN} \quad (5)$$

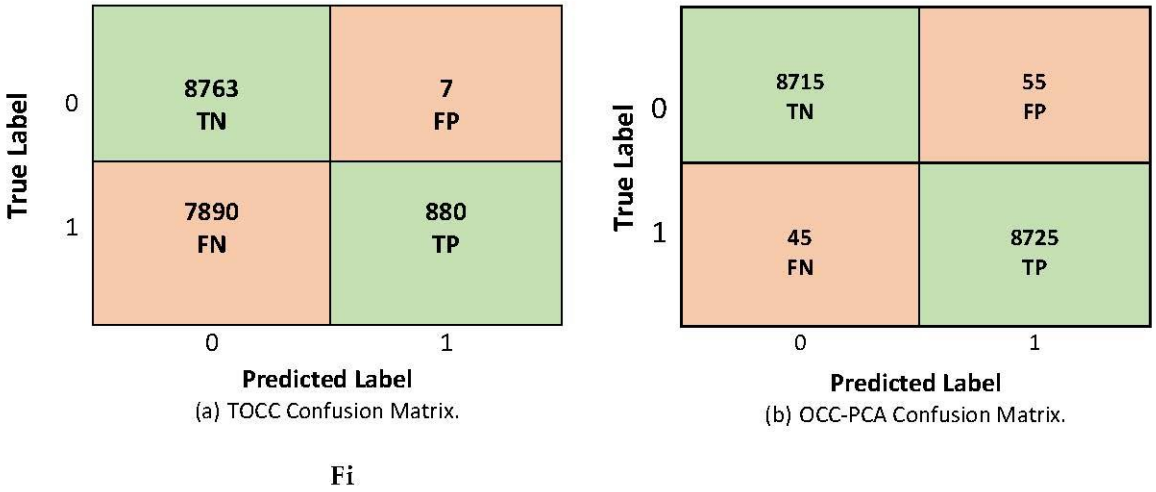
The findings of the evaluation phase, shown in Figure 5a, reveal that the TOCC approach achieved 880 TP samples and 8763 TN samples, while there were seven FP and 7890 FN samples. The TOCC approach achieved high FN rates; consequently, it suffers low sensitivity (recall score of 55%). Furthermore, it achieves low specificity (TNR rate of 53%), as shown in Table 5.

Table 5. OCC-PCA and TOCC Models Result in Comparison.

Model	Precision	Recall	F1 Score	TNR	Accuracy	Consumed Time (s)
TOCC	76%	55%	44%	53%	55%	0.64
OCC-PCA	99%	99%	99%	99.4%	99.4%	0.23

At the same time, the OCC-PCA approach achieved 8725 TP samples and 8715 TN samples, while there were 55 FP samples and 45 FN samples, as shown in Figure 5b. With

this model, the OCSVM could accurately predict normal and abnormal behavior. Utilizing the PCA enhanced the model performance results, as shown in Table 5, and increased the sensitivity and specificity to 99% and 99.4%, respectively. Increasing the model sensitivity indicates its ability to detect any deviation of the target class, consequently increasing its ability to be generalized and to detect zero-day attacks.



Fi

Figure 5. The Confusion Matrix of Both Models: (**a**) Description TOCC Confusion Matrix; (**b**) Description OCC-PCA Confusion Matrix.

The performance matrices of the evaluation phase reveal that the OCC-PCA model achieved 99%, 99%, 99%, 99.4%, and 99.4% in precision, recall/sensitivity, F1, TNR/specificity, and accuracy, respectively.

In this model, the results illustrate that concentrating on significant features enhances the algorithm's accuracy and predictability with respect to identifying benign behavior from malicious behavior. Apart from its accuracy, the algorithm's sensitivity in identifying and detecting normal behaviors can be enhanced and developed using the PCA algorithm. On another side, the TOCC model achieved 76%, 55%, 44%, 53%, and 55% precision, recall/sensitivity, F1, TNR/specificity, and accuracy, respectively. These properties contain information that is not necessarily useful in the training and testing of the algorithm. As a result, the TOCC model demonstrates that the OCSVM algorithm interacts negatively when the number of features is significant. In contrast to the OCC-PCA model, which uses the PCA algorithm to remove unnecessary features, the OCSVM algorithm appears to affect the results in anticipating normal behaviors when improving those predictions and concentrating on the necessary traits.

In conclusion, the PCA excluded the redundant and irrelevant features and selected the most relevant ones that increased the model's sensitivity and specificity. Moreover, memory malware usually suffers from high FP due to the inaccuracies caused by benign software bugs. The concept of merging the one-class classification with the PCA reduction techniques improved the model's detection results and reduced the FPs. Table 5 depicts the comparison between OCC-PCA and TOCC models. Noticeably, the TOCC approach consumed more time than the OCC-PCA in the testing phase.

The authors of [17,23] employed the PCA classifier; their findings were less than those in this study. That may be attributed to the chosen dataset's unsuitability. Refs. [26,27] utilized different dimensionality reduction techniques and achieved lower results than the proposed model's results. The dataset's suitability to the chosen algorithm and the suitable number of extracted features are factors for a successful performance. Although [16] achieved a better accuracy rate than the currently achieved rate, the proposed

model is characterized by utilizing a modern dataset that keeps pace with modern attacks. Table 6 compares the OCC-PCA model with other existing and published related work.

Table 6. Comparison Between the OCC-PCA Model and Literature Reviews.

References	Dimensionality Techniques	Dataset	Classifier	Accuracy Rate
[16]	PIO	KDDCUP-99	OCSVM	99.80%
[17]	PCA	Cassava starch samples	OCSVM	86.90%
[23]	DAE, PCA	Power System Samples	OCSVM	86.00%
[26]	OCSVM nested	KDDCUP-99	OCSVM	98.25%
[27]	Stack autoencoding	CIC-IDS-2017	OCSVM	99.35%
OCC-PCA	PCA	MalMemAnalysis-2022	OCSVM	99.40%

The hierarchical approach adopted in this study had a crucial role in the OCC-PCA model's ability to achieve a greater accuracy rate. The OCC-PCA and TOCC models were chosen following training, testing, and comparing the accuracy results by implementing the OCSVM classifier to assess both models. The authors then modified the variables in the OCC-PCA model from n = 10 to n = 30 and reevaluated the model to see if the variable modification improved the model results. The suggested model was then compared to the models of earlier research. One further factor that enhanced the performance of the proposed model was that OCSVM could have a better accuracy rate when dataset features extracted automation via the PCA classifier and utilized current, balanced datasets with a high augmentation of pre-proccing datasets. The variation in the PCA algorithms used on various datasets in prior research and the diversity in the OCSVM algorithms used on various models, as described in Tables 1 and 6, shows that the model we have presented may provide the best outcomes based on the findings.

As a result, the OCC-PCA approach and the nature of the OCSVM classifier in dealing with a low number of feature selections can be considered enhancements for detecting any attack. That can be seen as a mainly unknown attack or, using its other name, a zero-day attack. The accuracy rate of the OCC-PCA approach was achieved at 99.4%, and all performance matrices such as recall, precision, TPR, and F1 were achieved at 99%. Thus, PCA relies on eliminating all features that are not necessary and concentrating effort on the crucial features that produce more robust results, which helped us achieve better results. At the same time, the TOCC approach performed poorly, with an accuracy rate of 55%, and caused exceedingly low sensitivity to recognize normal behavior during the test phase. In the future, after detecting normal flow, we suggest stepping over to the next layer by determining what kind each attack is by utilizing multi-class classification in different datasets. Figure 6 depicts the comparison between TOCC and OCC-PCA results.

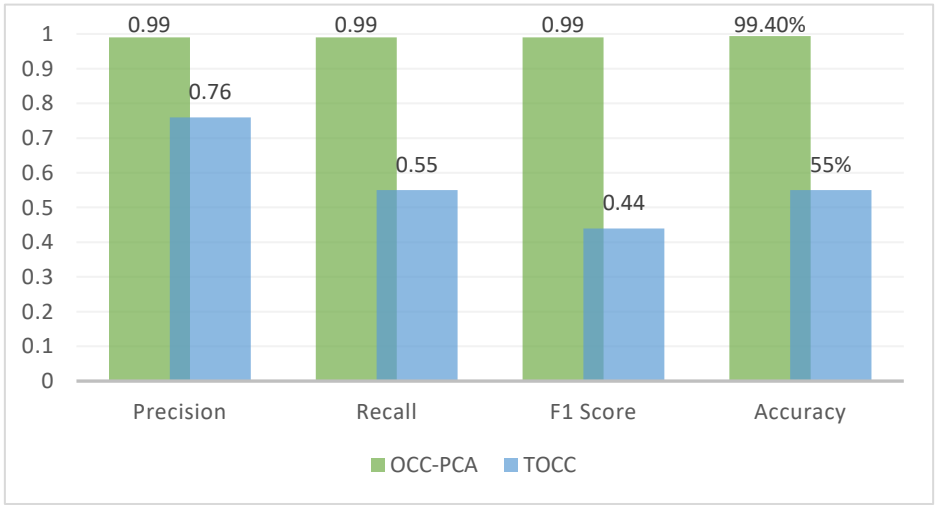


Figure 6. Comparison between TOCC and OCC-PCA Results.

6. Conclusions

This paper has developed a one-class classification (OCSVM) by integrating it with Principal Component Analysis (PCA), a dimensionality reduction technique. This integrated solution focuses on accurately detecting memory dump malware; even novel ones were considered. PCA was used to improve the sensitivity, specificity, and ability to generalize. To efficiently detect any deviation from the normal memory dump file patterns, the OCSVM classifier was utilized.

An intensive evaluation methodology was implemented based on a recently published dataset known as "MalMemAnalysis2022" to compare the performance of the standard OCSVM classifier with the proposed OSCVM with dimensionality reduction technique, PCA (OCC-PCA). The OCC-PCA model achieves a 99.4% accuracy rate, 99.3% TNR, and 99% for F1, recall, and precision scores, compared to the limited low performance of the standard model. Hence, an OCSVM classifier with a PCA classifier is recommended to identify benign behaviors.

Author Contributions: Conceptualization, M.A.-Q. and Z.A.; Methodology, M.A.-Q., Z.A., M.A. and Q.A.A.-H.; Software, M.A.-Q.; Validation, M.A.-Q., Z.A. and M.A.; Formal analysis, Q.A.A.-H.; Investigation, M.A.-Q., Z.A., M.A. and Q.A.A.-H.; Resources, M.A.-Q., Z.A., M.A. and Q.A.A.-H.; Writing—original draft, M.A.-Q., and Z.A., M.A. and Q.A.A.-H.; Writing—review & editing, M.A.-Q., Z.A., M.A. and Q.A.A.-H.; Visualization, M.A.-Q. and Z.A.; Funding acquisition, M.A.-Q., Z.A., M.A. and Q.A.A.-H.; Supervision, M.A. and Q.A.A.-H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The dataset associated with this research can be retrieved from: https://www.unb.ca/cic/datasets/malmem-2022.html (accessed on 17 July 2022).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gibert, D.; Mateu, C.; Planes, J. The rise of machine learning for detecting and classifying malware: Research payments, trends, and challenges. *J. Netw. Comput. Appl.* **2020**, *153*, 102526. https://doi.org/10.1016/j.jnca.2019.102526.
2. Abu Al-Haija, Q.; Al-Dala'ien, M. ELBA-IoT: An Ensemble Learning Model for Botnet Attack Detection in IoT Networks. *J. Sens. Actuator Netw.* **2022**, *11*, 18. https://doi.org/10.3390/jsan11010018.
3. McGraw, G.; Morrisett, G. Attacking malicious code: A report to the Infosec Research Council. *IEEE Software* **2000**, *17*, 33–41. https://doi.org/10.1109/52.877857.
4. The Independent IT-Security Institute. Available online: https://portal.av-atlas.org/ (accessed on 22 August 2022).
5. Abu-Zaideh, S.; Snober, M.A.; Al-Haija, Q.A. Smart Boosted Model for Behavior-Based Malware Analysis and Detection. In *IoT Based Control Networks and Intelligent Systems*; Lecture Notes in Networks and Systems; Joby, P.P., Balas, V.E., Palanisamy, R., Eds.; Springer: Singapore, 2023; Volume 528. https://doi.org/10.1007/978-981-19-5845-8_58.
6. Qalaja, E.K.; Al-Haija, Q.A.; Tareef, A.; Al-Nabhan, M.M. Inclusive study of fake news detection for COVID-19 with new dataset using supervised learning algorithms. *Int. J. Adv. Comput. Sci. Appl.* **2022**, *13*, 1-12. https://doi.org/10.14569/IJACSA.2022.0130867.
7. Carrier, T.; Victor, P.; Tekeoglu, A.; Lashkari, A. Detecting obfuscated malware using memory feature engineering. In Proceedings of the 8th International Conference on Information Systems Security and Privacy, Online, 9–11 February 2022. https://doi.org/10.5220/0010908200003120.
8. Al-Haija, Q.A.; Saleh, E.; Alnabhan, M. Detecting port scan attacks using logistic regression. In Proceedings of the 4th International Symposium on Advanced Electrical and Communication Technologies (ISAECT), Alkhobar, Saudi Arabia, 6–8 December 2021. https://doi.org/10.1109/isaect53699.2021.9668562.
9. Venable, M.; Chouchane, M.R.; Karim, M.E.; Lakhotia, A. Analyzing memory accesses in obfuscated x86 executables. In *Lecture Notes in Computer Science, International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Vienna, Austria, 7–8 July 2005*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 1–18. https://doi.org/10.1007/11506881_1.
10. Novelty and Outlier Detection. Scikit. Available online: https://scikit-learn.org/stable/modules/outlier_detection.html (accessed on 25 October 2022).
11. Abu Al-Haija, Q.; Odeh, A.; Qattous, H. PDF Malware Detection Based on Optimizable Decision Trees. *Electronics* **2022**, *11*, 3142. https://doi.org/10.3390/electronics11193142.
12. Farnia, F. Low-rate false alarm anomaly-based intrusion detection system with one-class SVM. Ph.D. Thesis, Ecole Polytechnique, Montreal, QC, Canada, 2017.

13. Al-Haija, Q.A. Exploration of Tools for Data Science. In *Data Science with Semantic Technologies*; Patel, A., Debnath, N.C., Bhusan, B., Eds.; CRC Press: Boca Raton, FL, USA, 2022. https://doi.org/10.1002/9781119865339.ch2.
14. Domingues, R.; Filippone, M.; Michiardi, P.; Zouaoui, J. A comparative evaluation of Outlier Detection Algorithms: Experiments and analyses. *Pattern Recognit.* **2018**, *74*, 406–421. https://doi.org/10.1016/j.patcog.2017.09.037.
15. Carter, J.; Mancoridis, S.; Galinkin, E. Fast, lightweight IOT anomaly detection using feature pruning and PCA. In Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing, Online, 25–29 April 2022. https://doi.org/10.1145/3477314.3508377.
16. Alazzam, H.; Sharieh, A.; Sabri, K.E. A lightweight intelligent network intrusion detection system using OCSVM and Pigeon Inspired Optimizer. *Appl. Intell.* **2021**, *52*, 3527–3544. https://doi.org/10.1007/s10489-021-02621-x.
17. Kelis Cardoso, V.G.; Poppi, R.J. Cleaner and faster method to detect adulteration in cassava starch using Raman spectroscopy and one-class support vector machine. *Food Control.* **2021**, *125*, 107917. https://doi.org/10.1016/j.foodcont.2021.107917.
18. Zhao, Y.-P.; Huang, G.; Hu, Q.-K.; Li, B. An improved weighted one-class support vector machine for Turboshaft Engine Fault Detection. *Eng. Appl. Artif. Intell.* **2020**, *94*, 103796. https://doi.org/10.1016/j.engappai.2020.103796.
19. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J.; Alazab, A. Hybrid intrusion detection system based on the stacking ensemble of C5 decision tree classifier and one class support vector machine. *Electronics* **2020**, *9*, 173. https://doi.org/10.3390/electronics9010173.
20. Yerima, S.Y.; Bashar, A. Semi-supervised novelty detection with one class SVM for SMS spam detection. In Proceedings of the 29th International Conference on Systems, Signals and Image Processing (IWSSIP), Sofia, Bulgaria, 1–3 June 2022. https://doi.org/10.1109/iwssip55020.2022.9854496.
21. Binbusayyis, A.; Vaiyapuri, T. Unsupervised deep learning approach for network intrusion detection combining convolutional autoencoder and one-class SVM. *Appl. Intell.* **2021**, *51*, 7094–7108. https://doi.org/10.1007/s10489-021-02205-9.
22. Verkerken, M.; D'hooge, L.; Wauters, T.; Volckaert, B.; De Turck, F. Towards model generalization for intrusion detection: Unsupervised Machine Learning Techniques. *J. Netw. Syst. Manag.* **2021**, *30*, 12. https://doi.org/10.1007/s10922-021-09615-7.
23. Mahfouz, A.; Abuhussein, A.; Venugopal, D.; Shiva, S. Network intrusion detection model using one-class support vector machine. In *Advances in Machine Learning and Computational Intelligence*; Springer: Singapore, 2020; pp. 79–86. https://doi.org/10.1007/978-981-15-5243-4_7.
24. Min, B.; Yoo, J.; Kim, S.; Shin, D.; Shin, D. Network anomaly detection using memory-augmented deep autoencoder. *IEEE Access* **2021**, *9*, 104695–104706. https://doi.org/10.1109/access.2021.3100087.
25. Qi, R.; Rasband, C.; Zheng, J.; Longoria, R. Detecting cyber-attacks in smart grids using semi-supervised anomaly detection and Deep Representation Learning. *Information* **2021**, *12*, 328. https://doi.org/10.3390/info12080328.
26. Nguyen, Q.T.; Tran, K.P.; Castagliola, P.; Huong, T.T.; Nguyen, M.K.; Lardjane, S. Nested one-class support vector machines for network intrusion detection. In Proceedings of the IEEE Seventh International Conference on Communications and Electronics (ICCE), Hue, Vietnam, 18–20 July 2018; pp. 7–12.
27. Mhamdi, L.; McLernon, D.; El-Moussa, F.; Zaidi, S.R.; Ghogho, M.; Tang, T. A Deep Learning Approach Combining Autoencoder with One-Class SVM for DDoS Attack Detection in SDNs. In Proceedings of the IEEE Eighth International Conference on Communications and Networking (ComNet), Hammamet, Tunisia, 27–30 October 2020. https://doi.org/10.1109/ComNet47917.2020.9306073.
28. Brink, H.; Richards, J.; Fetherolf, M.; Cronin, B. *Real-World Machine Learning*; Manning Publications: Shelter Island, NY, USA, 2017.
29. Ashi, Z.; Aburashed, L.; Al-Fawa, M.; Qasaimeh, M. Fast and Reliable DDoS Detection using Dimensionality Reduction and Machine Learning. In Proceedings of the15th International Conference for Internet Technology and Secured Transactions (ICITST), London, UK, 8–10 December 2020; pp. 1–10. https://doi.org/10.23919/ICITST51030.2020.9351347.
30. Al-Haija, Q.A.; Alsulami, A.A. High Performance Classification Model to Identify Ransomware Payments for Heterogeneous Bitcoin Networks. *Electronics* **2021**, *10*, 2113. https://doi.org/10.3390/electronics10172113.
31. Basnet, R.B.; Shash, R.; Johnson, C.; Walgren, L.; Doleck, T. Towards detecting and classifying network intrusion traffic using deep learning frameworks. *J. Internet Serv. Inf. Secur.* **2019**, *9*, 1–17.
32. Ashi, Z.; Aburashed, L.; Qudah, M.; Qusef, A. Network intrusion detection systems using supervised machine learning classification and Dimensionality Reduction Techniques: A systematic review. *Jordanian J. Comput. Inf. Technol.* **2021**, *7*, 373–390. https://doi.org/10.5455/jjcit.71-1629527707.
33. Abu Al-Haija, Q.; Krichen, M. A Lightweight In-Vehicle Alcohol Detection Using Smart Sensing and Supervised Learning. *Computers* **2022**, *11*, 121. https://doi.org/10.3390/computers11080121.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.