DOI: 10.21917/ijct.2023.0428

AN ENSEMBLE NEURAL NETWORK TECHNIQUE FOR IMPROVING SECURITY AMONG VARIOUS DOMAINS OF INFORMATION TECHNOLOGY

R. Thamarai Selvi
PG Department of Computer Applications, Bishop Heber College, Affiliated to Bharathidasan University, India

Abstract
In the era of Internet of Things (IoT), enterprise information Systems (IISs) are becoming increasingly valuable in a range of industries due to the fact that they constitute a network in which connected devices exchange data in an environment that is quite close to real time. In this context, enterprises are provided with the opportunity to make use of virus detection solutions that are either static, dynamic, or hybrid. The research uses ensemble machine learning approaches that have been implemented and are analyzed, and comparisons are drawn between them. The findings of this research have been effective in the identification of malwares in IIS.

Keywords:
Ensemble Neural Network, Security, Malware, Information Technology

1. INTRODUCTION

Internet of Things based enterprise information systems (IoT-EIS) are becoming increasingly valuable in a range of industries due to the fact that they constitute a network in which connected devices exchange data in a manner that is quite close to real time. It is estimated that sales revenues from IoT devices would have topped $14.4 trillion by the year 2022, and it is anticipated that there will be approximately 22 billion IoT devices by the year 2025 [1].

The limited power sources and computational capabilities, IoT devices are vulnerable to a wide variety of different types of attacks. Malware that is put on IoT devices leaves those devices vulnerable to attack. After then, the malware is utilized to access APIs on the business side and steal confidential data. Recent malware attacks on the IoT include one known as Maria, which took place in 2017 and inflicted damage at the rate of approximately $4207.03 per hour [2]. Malware that is put on edge devices has the potential to allow such devices to broadcast skewed or false data regarding the IT infrastructure of an enterprise to a cloud server that is located off-site. Attacks using malware such as this one are common in IIoT networks, and they can cause irreversible harm to the financial position as well as the reputations of those networks [3].

Researchers devote a large amount of their time and attention to improving the security architecture of enterprise information systems that are based on the IoT as a direct consequence of this issue [4]. In the past, the bulk of commercial information technology systems have relied on feature extraction methodologies in order to identify malicious software. This was done in order to protect against computer viruses. Before doing an analysis on the relevant parts of the code, these methodologies require first separating those parts of the code that are of interest [5]. These methods of feature detection, however, do not scale very well in enterprise information systems because of the restricted power and processing capabilities of IoT devices. This is due to the constraints imposed by the gadgets that make up the IoT [6]. Normal methods of malware detection become useless and cannot be employed when adversarial attacks occur, in which the attacker alters the training samples [7].

Within the context of the IoT, enterprise information systems are provided with the opportunity to make use of virus detection solutions that are either static, dynamic, or hybrid. Static analyses include things like signature analysis, n-gram identification, and OpCode analysis, whereas dynamic studies involve actually running the application in a virtual environment [8]. Examples of static analyses include signature analysis, n-gram identification, and OpCode analysis.

2. LITERATURE SURVEY

The research in [9] a variety of methods for the identification of intrusions in an IoT setting. The author focuses the majority of his attention on machine learning algorithms, but he also addresses concerns regarding the security of IoT devices and provides information about datasets that are available to the public. On the other hand, for the purposes of this review study, our primary attention is placed on intrusion detection systems that are capable of recognizing distributed denial of service attacks.

Malicious Android applications [10] are used to break into the security of IoT devices, allowing hackers to steal sensitive information. The author provides a comprehensive look at the many different methods of analysis that were done in order to discover the malicious Android application. In spite of this, the approaches that are utilized to discover malware in Android applications are not the primary focus of the survey articles. The survey papers, on the other hand, offer further information regarding Android applications that can run on IoT devices.

Certain broad information regarding protection measures for IoT devices that make use of machine learning. The author [11] provide any context for the many forms of malware that can be found in a scenario involving the IoT. In their research paper titled Malware Detection in the IoT, which was released researched the many machine learning algorithms that are currently being used for this specific purpose. They found that there are a large number of machine learning algorithms that can be utilized for this particular purpose. The various machine learning approaches for identifying malware in an IoT setting are analyzed, and comparisons and contrasts are drawn between them.

The authors [12] conducted research on several different machine learning algorithms with the intention of identifying malicious software in an IoT setting. The author focuses almost all of their emphasis on machine learning techniques that are derived from OpCode.

The authors in [12] carried up a survey to explore the many different methods that are now available for locating static malware in an IoT setting. The authors conducted research on a variety of methods that are already in use to recognize static malware. These methods include machine learning, fuzzing, and clustering. Nevertheless, the author does not supply any further information on how to recognize dynamic and hybrid kinds of harmful software.

On the subject of employing machine learning for the detection of malicious software, a significant amount of research has been carried out. Malware detection in the context of the IoT still suffers from inadequate research. IoT refers to the network of interconnected electronic devices [13].

3. PROPOSED MALWARE DETECTION

In this research, we will examine the characteristics of user nodes in the malware propagation network from two distinct vantage points: the structure of the network, and the activity of user nodes in the past. In addition to this, we provide a complete representation method for the feature space related with the spread of malware. We apply the Doc2vec technique to the user node content structure in order to establish the feature vector that is reflective of the user node one-of-a-kind content preferences. In other words, we want to figure out what those preferences are. In order to ascertain the feature vector of the network, the Tensor2vec methodology is put to use once more for the purpose of this particular scenario.

3.1 CHARACTERISTICS OF USER NODES

An examination of the past actions of the nodes that comprise a network for the transmission of a virus enables one to gain insight into the preferences and routines of users of the network. Doc2vec is an unsupervised approach that can learn fixed-length feature representations from texts of varying lengths. It was developed by the Stanford Natural Language Processing Group. It is put to use in the process of constructing vector representations of phrases, paragraphs, and entire documents. Paragraphs are created from the information that a user node persistently relays and actively transmits over the course of a predetermined amount of time. The Doc2vec technique is able to express the user social behavior as a vector that communicates the user node habits and preferences because it makes use of the multiple characteristics that are offered by paragraph feature vectors. This is made possible by the method ability to take advantage of these numerous features. The step in the text classification process that is referred to as preprocessing is an essential part of the overall method.

The effectiveness of feature extraction as well as text classification is directly influenced by the outcomes of Chinese word segmentation and stop words. Both of these processes are influenced in a direct manner. A technique known as Jieba Chinese word segmentation is used to break up the original corpus into its component parts. This approach takes into account the peculiarities that are inherent to Chinese grammatical conventions. Following the segmentation of the data, it is important to apply judgment with reference to the various portions of speech. Additionally, stop words are added to the results of the segmentation in order to filter out unnecessary keywords that may otherwise compromise the accuracy of the classification. This is done in order to identify the nouns and verbs in the sentence. Additionally, stop words are added in order to identify the nouns and verbs in the phrase.

We are now in possession of the candidate keywords that have been derived from the history actions of the user node as a direct consequence of this. The TF-IDF calculation algorithm has been improved so that it more precisely reflects the current state of the network. This improvement was made in order to facilitate the calculation of word frequency. This was made possible by the incorporation of the variable that distinguishes an active user from an ordinary user. This is because active user nodes have a greater impact on the overall rate at which information is disseminated throughout the network.

The candidate words are mined for the user historical behavior data, which is then used to retrieve relevant keywords. This is done in order to ensure the integrity of the core user node data while at the same time filtering out any unnecessary information that may be present in the propagation network. After that, the Doc2vec technique is put to work in order to construct a feature vector that is reflective of the user node prior actions and is constructed on the basis of the keyword sequence that most effectively reflects that behavior. The user receives this feature vector once it has been processed.

$$T = N \times Fu \tag{1}$$

where

N - total users

Fu - representation vector.

Algorithm: Shap value to probability value

3.2 BASE CLASSIFIERS OF ENSEMBLE

The goal of this benchmark is to determine how well deep learning and shallow learning perform on the challenge that is now being faced using the two datasets that we have just finished discussing.

3.2.1 Random Forest:

A random forest is a specific kind of classifier that is produced by bagging together a number of decision trees. This process is described as bagging. A decision tree is a form of graph that represents numerous possibilities and the different effects that could arise from each selection. One way to think about a action plan is as a decision tree, which is one way to think about a action plan. One of the things that differentiates this model from others is the fact that the information is depicted in the form of a tree, which is very straightforward and straightforwardly presented.

Each of the internal nodes in the decision tree serves as a substitute for a different variable. The variable was subjected to either an equality, majority, or minority condition in order to determine where the split should occur, and each fork in the graph depicts the resulting split. As a result of this, each fork is also a representation of the arcs that link a parent node to its progeny. Instead, the leaf nodes in the tree are what indicate the predicted class in the structure.

Consequently, what comprises a decision tree is a set of rules for making decisions that change depending on the values of the variables. Using the information included in the dataset, a decision tree is constructed. During the training phase, the stopping criteria, which are also known as halting, need to be set since a tree that is extremely complex and has numerous branches contributes very little to the accuracy of the classification.

The method known as bagging entails selecting a number of models from the same dataset that are comparable to one another and then sampling from those models without making any replacements. During the training process for the individual decision trees that go into the construction of a random forest, only a subset of the total number of variables that are contained within the dataset are used. When doing a task that involves classification, the end classification is established by taking the median of the individual decision tree classifiers. When performing a job that involves regression, on the other hand, the final classification is determined by taking the mean of the findings.

3.2.2 XGBoost:

The problem-solving community that works on machine learning developed a technique called gradient boosting to address issues with classification and regression. One way to construct models that can be used for making predictions is by combining a number of more fundamental forecasting tools, such as decision trees. This is only one of many possible approaches. This method is useful because of its generalizability, which makes it possible to optimize any arbitrary differentiable loss function while the model is still being generated. This is a benefit to the method. One of the many reasons why this strategy is so helpful is because of something like this. This method, in contrast to others, offers remarkable portability as a result of its support for a cross-platform between a number of programming languages and operating systems. Specifically, this portability is made possible by the fact that the method can translate between Windows and Linux. Because of its support, the approach can be implemented on a number of different systems.

Because it incorporates both structural and algorithmic enhancements, XGBoost is able to make the Gradient Boosting Machines (GBM) architecture, which is the underlying algorithm, more effective. Because it allots internal buffers to each thread in order to keep computed statistics, XGBoost is in a position to make use of algorithms that are aware of the cache. This is how it manages to achieve its goals. The software makes use of parallelization since the process of creating trees using XGBoost is completed in parallel by exchanging nested loops with one another. This switching of loops is the key to parallelization. Rather than relying on the standard stop criterion, you can make adjustments to the depth at which each tree is separated by making use of the max_depth option. This is done instead of the default stop criterion that would normally be used. The ability to design more complex algorithms is made feasible by the following characteristics:

When compared to other algorithms, XGBoost models perform exceptionally well in classification and regression tasks. This is true in terms of the precision of their forecasts as well as the speed with which they are computed. This is the case with regard to both the precision of their forecasts and the efficacy with which they are computed.

3.2.3 CatBoost:

When it comes to utilizing the gradient boosting process with decision trees, the open-source software package known as CatBoost claims that it is superior to any other choice that is currently available. During training, a series of trees are constructed one after the other in sequential order, with the amount of loss used to construct each tree becoming increasingly smaller than what was used to construct the previous tree in the sequence as the training progresses.

For the underlying tree structure is one that is analogous to that of greed. Importantly, it is also able to automatically do quantization on feature values, which means that it is able to find the thresholds to employ in order to break feature values and labels into discrete ranges. This is an extremely useful capability. This is a capability that is of tremendous value. (bins). It is easy to use, accurate, dependable, versatile, and extendable, and there is even a version of it that can be computed by a GPU and provides support for categorical data formats.

CatBoost is easily compatible with well-known deep learning libraries such as Apple Core ML and Google TensorFlow, the latter of which was developed by Apple, and the former was built by Google. It is compatible with a large number of data formats and provides explainable artificial intelligence by making use of ranking features to decide the order in which data is retrieved. In addition, it is compatible with a wide variety of data formats. In this scenario, the model for basic supervised classification, which consists of a predetermined feature set, is put to use.

3.2.4 Extra Trees:

The output of the prediction is created by making use of a forest of trees, each of which is reliant on the results of the several base models that were employed in the calculation. The Extra Trees method generates a significant number of decision trees without carrying out any type of pruning on them and by automatically producing them from the supplied data. for making predictions for classification, a majority vote is used, but the outcomes of each individual tree are averaged for making predictions for regression. It is based on the idea that overfitting may be avoided when trees are built by randomly selecting the feature to divide at each level for each tree. This is the foundation of the argument.

The degree to which the trees in the ensemble are similar to one another is reduced, while the variability of those trees is increased. One method that can be used to deal with the greater variability is to increase the number of trees that are present in the forest.

The fundamental difference between ExtraTrees and Random Forest is in the way in which the decision trees that are included within the forest are generated. ExtraTrees uses a top-down approach, whereas Random Forest uses a bottom-up approach. ExtraTrees selects a branching point for each feature in a fully arbitrary manner as opposed to Random Forest, which strives to achieve a flawless split in the process of selecting a point at which to split the data.

3.2.5 TabNet:

The goal of the TabNet project is to create a neural network that will be able to deal with tabular data in an effective manner. This is the project primary objective. It has been established that the application of deep learning strategies, as opposed to machine learning methods, can significantly improve performance as the dataset continues to develop. This has been noticed in a variety of application areas, including the field of photography. It was created to train models that are analogous to decision trees, and it shares the benefits of decision trees, such as being interpretable and including selection, just like those models. It accomplishes this goal by employing the strategy of multi-headed attention in a sequential manner to choose which features to utilize at each stage of the decision-making process. Every single input asks for its own unique set of features, which are selected on a case-by-case basis depending on the specifics of the scenario.

3.2.6 Neural Oblivious Decision Ensemble (NODE):

The model is constructed in stages, with input examples being passed from one stage to the next; it is made up of a transformer with numerous multi-headed attentions acting in parallel to produce a sparse feature selection matrix. Input examples are transmitted from stage to stage as the model is constructed. During the process of the model construction, the input instances are passed along from one stage to the next. Because the weights of the attributes, which are often referred to as their importance, are retrieved on a per-instance basis, the interpretability of the data has significantly improved.

The dataset is initially handled in its original, unprocessed condition before any feature engineering work is carried out on it. After being subjected to batch normalization, examples are then sent to the feature transformer, where they are processed by numerous layers of fully connected neurons that are activated by a variety of gated linear units. This occurs after the instances have been subjected to batch normalization. Feature transformer (GLUs).

This procedure is carried out as many times as necessary until the appropriate degree of precision is reached. It is essential to keep in mind that normalization with 0.5 makes learning more consistent by reducing the frequency of sudden shifts in network variance. This is something that needs to be borne in mind at all times. In reference to this specific subject, we make use of an automatic version of the classification model feature engineering and feature selection processes. The requirement of an extremely large amount of information in order to learn is one of the most significant problems of the multi-headed attention model. This is also one of the fundamental limitations of this strategy.

An architecture for deep learning known as the NODE algorithm was designed specifically for use with tabular data. This design is constructed from the oblivious decision tree, which has the peculiar characteristic of requiring the same feature for the split and the same split threshold for all nodes of the same depth. This is an unusual requirement for any design. The network is trained using backpropagation from the very beginning to the very end.

In the deep form, in which numerous NODE layers are stacked atop one another, the residual connection from the ResNet work is used to connect the many NODE layers. The outputs from each layer are mixed with the features of the previous layer to produce the input for the next NODE layer. When it comes down to it, just like with Extra Trees, we take the results of all of the layers and average them, and when it comes to categorization, we use majority vote rather than majority vote. The deep version of the model is utilized by NODE when it comes to the classification process.

3.3 ENSEMBLE NEURAL NETWORK

For the purpose of developing a model for churn prediction, artificial neural networks, one of the most powerful and well-known methods for machine learning, were applied. The strategy that has been developed combines the most useful aspects of two neural networks in order to generate very accurate forecasts regarding the number of customers who will leave an organization. Because of this, at this point in time, two multilayer feed-forward neural networks were built. One of these networks used a solution obtained from the alpha wolf, and the other network used a solution developed by the beta wolf. In the end, the final response of the prediction model was derived using, which is a weighted average, and the dataset was taught to the neural networks utilizing the newly acquired features from the feature selection stage.

$$\text{Output} = 0.5 \times net1 + 0.5 \times net2. \qquad (2)$$

where,

$net1$ and $net2$ - neural networks outputs.

The ultimate solution of the neural network is the combined solution of two neural networks that have the same weighted average. This ultimate solution is also represented by the output of the neural network.

4. EVALUATION

This research was validated by testing it on a dataset of malware that is freely available to the public and was collected from the VirusHare database. In the year 2020, a total of 2,000 examples of malicious software, originating from a wide variety of families, were gathered and put into the virus database. The 1744 secure programs were selected from a large number of different software packages. (games, browsers, word processors, business, etc.). Following the completion of the collection, VirusTotal was used to do a scan of the clean files using thirty distinct anti-malware engines.

In addition, the effectiveness of ENN was determined by testing it with 8,750 examples of malware hailing from 25 distinct families contained inside the MaleVis dataset and 9339 Malimg malware images serving as a benchmark.

The experiment was carried out inside a Google Collaboratory with a graphics processing unit (GPU), random access memory (RAM), and a hard drive provided by the business.

4.1 EVALUATION

The evaluation criteria that were applied in order to evaluate the efficacy of ENN. Accuracy, precision, recall, and F1-score were some of the measures that were used. The ENN was tested to determine whether or not it could differentiate between malicious and benign labels using only a small amount of training data. During the course of the experiment, a total of 2,000 examples of malicious software and 1,744 examples of benign software were tested. The outcomes of an experiment are summarized in Table.1, which shows how ENN performed in comparison to other classifiers.

The findings indicate that the recommended technique resulted in improvements in terms of accuracy, precision, and recall. The proposed ENN method was successful in attaining a higher accuracy (96.5%) as compared to the other machine learning approaches shown in Figure 8, which also includes a bar chart displaying the outcomes of the binary classification. The ensemble classifiers Gradient Boosting and Extra Tree Classifier came in second and third place, respectively, with an accuracy of 95.6% and 95.3%.

4.1.1 Mallmg Dataset:

Only 10 of 25 families of malicious software were selected to be used in the training of the ENN on Mallmg dataset. According to Table.2, the accuracy of detection achieved by ENN was 99.8%, which was significantly higher than that of competing technologies.

4.1.2 MaleVis Dataset:

Using the MaleVis dataset, which contained 25 families with comparable sample sizes, the same process was carried out. According to Table 3, ENN has a success percentage of 97%, whereas NODE has a success rate of 92%, SVM has a success rate of 88%, KNN has a success rate of 78%, and Bagging has a success rate of 93%.

Table.1. Training Performance

Models	Class	Accuracy	Precision	Recall	F1-score
ENN	Benign	0.9960	0.9366	0.9463	0.9366
	Malware		0.9463	0.9366	0.9463
Resnet50	Benign	0.9640	0.9366	0.8780	0.9073
	Malware		0.8976	0.9366	0.9171
Extra Trees Classifier	Benign	0.9836	0.9073	0.9366	0.9171
	Malware		0.9366	0.9073	0.8293
Gradient Boosting Classifier	Benign	0.9867	0.9073	0.9366	0.9171
	Malware		0.9366	0.9073	0.9268
Bagging	Benign	0.9671	0.8585	0.9561	0.9073
	Malware		0.9561	0.8683	0.9073
SVM	Benign	0.9753	0.9171	0.9171	0.9171
	Malware		0.9268	0.9171	0.9171
KNN	Benign	0.8886	0.9463	0.7122	0.8098
	Malware		0.7805	0.9561	0.8585
Naïve Bayes	Benign	0.9557	0.9073	0.8585	0.8878
	Malware		0.8780	0.9171	0.8976
Random Forest	Benign	0.9712	0.8683	0.9659	0.9171
	Malware		0.9659	0.8683	0.9171
Decision Tree Classifier	Benign	0.9310	0.8780	0.8585	0.8683
	Malware		0.8780	0.8878	0.8780

Table.2. Accuracy comparison on Mallmg Dataset

Malware Family	Proposed				NODE				SVM				KNN				Bagging			
	A	P	R	F1	A	P	R	F1	A	P	R	F1	A	P	R	F1	A	P	R	F1
Adialer.C	0.997	0.975	0.975	0.975	0.998	0.975	0.975	0.975	0.995	0.975	0.975	0.975	0.954	0.643	0.975	0.770	0.988	0.975	0.975	0.975
Agent.FYI		0.974	0.974	0.974		0.974	0.974	0.974		0.974	0.974	0.974		0.974	0.974	0.974		0.974	0.974	0.974
Alueron.gen!J		0.972	0.972	0.972		0.972	0.972	0.972		0.972	0.972	0.972		0.875	0.972	0.923		0.972	0.972	0.972
Lolyda.AA1		0.969	0.969	0.969		0.950	0.969	0.959		0.950	0.969	0.959		0.940	0.862	0.901		0.950	0.969	0.969
Lolyda.AA2		0.972	0.972	0.972		0.952	0.952	0.952		0.972	0.952	0.962		0.952	0.952	0.962		0.972	0.972	0.972
Malex.gen!J		0.937	0.966	0.957		0.966	0.937	0.957		0.966	0.947	0.957		0.966	0.947	0.957		0.928	0.947	0.937
Obfuscator.AD		0.971	0.971	0.971		0.971	0.971	0.971		0.971	0.971	0.971		0.971	0.971	0.971		0.971	0.971	0.971
Rbot!gen		0.970	0.970	0.970		0.941	0.941	0.941		0.951	0.970	0.960		0.931	0.970	0.951		0.951	0.931	0.941
Swizzor.gen!I		0.969	0.969	0.969		0.940	0.969	0.959		0.969	0.969	0.969		0.940	0.911	0.921		0.940	0.940	0.940
VB.AT		0.968	0.958	0.968		0.968	0.958	0.968		0.968	0.968	0.968		0.968	0.813	0.881		0.958	0.949	0.949

Table.3. Accuracy comparison on MaleVis Dataset

Malware Family	Proposed				NODE				SVM				KNN				Bagging			
	A	P	R	F1	A	P	R	F1	A	P	R	F1	A	P	R	F1	A	P	R	F1
Adposhel		0.955	0.975	0.965		0.975	0.975	0.975		0.955	0.975	0.965		0.897	0.965	0.926		0.945	0.975	0.955
Agent		0.954	0.828	0.857		0.701	0.886	0.857		0.682	0.857	0.759		0.662	0.613	0.633		0.915	0.828	0.867
Allaple		0.884	0.904	0.894		0.884	0.904	0.894		0.855	0.777	0.816		0.943	0.641	0.758		0.962	0.865	0.904
Amonetize		0.940	0.950	0.940		0.882	0.940	0.911		0.882	0.959	0.921		0.921	0.911	0.911		0.969	0.911	0.940
Androm		0.758	0.797	0.787		0.777	0.671	0.719		0.671	0.690	0.680		0.496	0.709	0.583		0.787	0.836	0.807
Autorun		0.860	0.860	0.860		0.744	0.802	0.773		0.792	0.734	0.763		0.783	0.580	0.667		0.841	0.676	0.754
BrowseFox		0.951	0.951	0.951		0.942	0.913	0.922		0.883	0.932	0.913		0.816	0.883	0.845		0.913	0.951	0.932
Dinwod		0.951	0.951	0.951		0.960	0.960	0.960		0.970	0.912	0.941		0.689	0.854	0.766		0.970	0.960	0.960
Elex		0.950	0.901	0.921		0.940	0.950	0.951		0.901	0.969	0.930		0.921	0.940	0.930		0.921	0.862	0.891
Expiro		0.949	0.900	0.920		0.687	0.774	0.726		0.591	0.726	0.649		0.174	0.048	0.077		0.571	0.862	0.687
Fasong		0.965	0.975	0.965		0.975	0.975	0.975		0.975	0.975	0.975		0.955	0.975	0.965		0.965	0.975	0.965
HackKMS		0.867	0.974	0.896		0.964	0.974	0.964		0.974	0.974	0.974		0.974	0.964	0.964		0.974	0.974	0.974
Hlux	0.997	0.972	0.972	0.972	0.925	0.972	0.972	0.972	0.898	0.962	0.962	0.962	0.789	0.952	0.962	0.962	0.925	0.972	0.962	0.962
Injector		0.930	0.795	0.853		0.853	0.727	0.824		0.630	0.736	0.678		0.562	0.727	0.630		0.882	0.862	0.872
InstallCore		0.962	0.972	0.962		0.972	0.952	0.962		0.962	0.962	0.962		0.972	0.739	0.845		0.972	0.962	0.962
Multiplug		0.918	0.899	0.908		0.850	0.899	0.879		0.850	0.860	0.850		0.850	0.812	0.831		0.966	0.870	0.918
Noreklami		0.951	0.961	0.951		0.942	0.951	0.942		0.913	0.922	0.922		0.951	0.922	0.932		0.971	0.922	0.951
Neshta		0.786	0.776	0.776		0.737	0.582	0.650		0.630	0.630	0.630		0.165	0.388	0.233		0.689	0.863	0.766
Regrun		0.843	0.959	0.891		0.959	0.969	0.959		0.969	0.969	0.969		0.969	0.969	0.969		0.969	0.959	0.959
Sality		0.736	0.610	0.668		0.687	0.581	0.629		0.649	0.368	0.465		0.755	0.068	0.116		0.678	0.639	0.658
Snarasite		0.975	0.975	0.975		0.975	0.975	0.975		0.975	0.975	0.975		0.945	0.975	0.955		0.975	0.975	0.975
Stantinko		0.964	0.964	0.964		0.964	0.954	0.954		0.974	0.935	0.954		0.974	0.925	0.954		0.974	0.974	0.974
VBA		0.972	0.972	0.972		0.972	0.972	0.972		0.972	0.972	0.972		0.972	0.972	0.972		0.972	0.972	0.972
VBKrupt		0.911	0.901	0.911		0.843	0.921	0.882		0.969	0.882	0.921		0.572	0.804	0.669		0.950	0.930	0.940
Vilsel		0.972	0.952	0.962		0.972	0.952	0.962		0.972	0.952	0.962		0.972	0.952	0.962		0.972	0.962	0.962

ENN delivers better results in malware classification by utilizing adversarial malware development in conjunction with a deep convolutional neural network that has been finely trained.

5. CONCLUSION AND FUTURE WORK

As a result of this research, a ENN is developed for the purpose of analyzing and classifying malware. This was done in view of the continually developing strategies that malware authors employ to evade detection. Using ENN, malware and benign features are taught, and then the results are sorted into various families of malware. In the end, ENN performance was assessed and compared to that of other industry-leading malware visualization techniques. It was discovered that using ENN improved both precision and productivity across the board. The findings of this research have applications not only in the visualization but also in the identification of malicious software. In the future, we intend to evaluate ENN using larger datasets and incorporate the design of the proposed framework into a system for the sake of testing its accuracy and performance.

REFERENCES

[1] Chris McNab, "*Network Security Assessment*", 2nd Edition, O'Reilly Media, 2007.

[2] Haifeng Wu, "Research of Network security Assessment System Based on Vulnerability Scan", *Proceedings of International Conference on Advanced Computer Control*, pp. 566-569, 2011.

[3] P. Jayashree, "Security Issues in Protecting Computers and Maintenance", *Journal of Global Research in Computer Science*, Vol. 4, No. 1, pp. 55-58, 2013.

[4] K. Praghash and T. Karthikeyan, "Data Privacy Preservation and Trade-off Balance Between Privacy and Utility using Deep Adaptive Clustering and Elliptic Curve Digital Signature Algorithm", *Wireless Personal Communications*, Vol. 78, 1-16, 2021.

[5] J. Kim and J.H. Yi, "MAPAS: A Practical Deep Learning-based Android Malware Detection System", *International Journal of Information Security*, Vol. 21, No. 4, pp. 725-738, 2022.

[6] M.E.Z.N. Kamba and K. Taghva, "A Survey on Mobile Malware Detection Methods using Machine Learning",

Proceedings of *Annual Workshop and Conference on Computing and Communication*, pp. 215-221, 2022.

[7] X. Song and Y. Wang, "Homomorphic Cloud Computing Scheme based on Hybrid Homomorphic Encryption", *Proceedings of International Conference on Computer and Communications*, pp. 13-16, 2017.

[8] D.R. Kumar Raja and S. Pushpa, "Diversifying Personalized Mobile Multimedia Application Recommendations through the Latent Dirichlet Allocation and Clustering Optimization", *Multimedia Tools and Applications*, pp. 1-20, 2019.

[9] C. Paar and J. Pelzl, "*Understanding Cryptography*", Springer, 2010.

[10] Madiha Khalid, Umar Mujahid and Najam-Ul-Islam Muhammad, "Ultralightweight RFID Authentication Protocols for Low-Cost Passive RFID Tags", *Security and Communication Networks*, Vol. 2019, pp. 1-25, 2019.

[11] C. Li and Y. Qiao, "A Novel Deep Framework for Dynamic Malware Detection based on API Sequence Intrinsic Features", *Computers and Security*, Vol. 116, pp. 102686-102698, 2022.

[12] J.Y. Kim and S.B. Cho, "Obfuscated Malware Detection using Deep Generative Model based on Global/Local Features", *Computers and Security*, Vol. 112, pp. 102501-102513, 2022.

[13] Jong Sik Moon and Im-Yeong Lee, "An AAA Scheme using ID-Based Ticket with Anonymity in Future Mobile Communication", *Computer Communications*, Vol. 34, No. 3, pp. 295-304, 2011.