Malware Classification Using Deep Learning Methods

Bugra Cakir
BearTell, Inc.
Ankara, Turkey
bcakir@beartell.com

Erdogan Dogdu
Cankaya University
Ankara, Turkey
edogdu@cankaya.edu.tr

ABSTRACT

Malware, short for Malicious Software, is growing continuously in numbers and sophistication as our digital presence continuous to grow. It is a very serious problem and many efforts are devoted to malware detection in today's cybersecurity world. Many machine learning algorithms are used in automatic detection of malware problem in recent year. Recently Deep Learning is being used with better performance. Deep Learning models work much better in the analysis of long sequences of system calls. In this paper a shallow deep learning based feature extraction method (word2vec) is used for representing any given malware based on its opcodes. And, Gradient Boosting algorithm for malware classification is chosen for the classification task. K-fold cross-validation is used to validate the model performance without sacrificing a validation split. Evaluation results show up to 96% success with limited sample data.

CCS CONCEPTS

• **Computing methodologies → Supervised learning by classification**; **Neural networks**; • **Security and privacy →** *Malware and its mitigation*;

KEYWORDS

Machine learning, deep learning, supervised learning, classification, malware detection

ACM Reference Format:
Bugra Cakir and Erdogan Dogdu. 2018. Malware Classification Using Deep Learning Methods. In *ACM SE '18: ACM SE '18: Southeast Conference, March 29–31, 2018, Richmond, KY, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3190645.3190692

1 INTRODUCTION

Malware is a malicious software, and it is harmful when executed on a computing device or system. The consequences of running malware could be from minor to catastrophic damages, such as loosing critical data or failure in a nuclear power plant. Therefore, malware affects our daily lives continuously, and our computing systems need protection from malware attacks round the clock. To provide protection, many researchers and cybersecurity firms are working on new and robust methods and tools for the automatic identification and detection of malware software.

Automatic malware detection is a scientific research area and many machine learning algorithms are used and adapted for automatic malware detection problem. Recently deep learning algorithms, that are multilayer neural networks, are being used in machine learning area and they are successful in many learning processes, such as classification. However, Deep Learning requires more computation time to train and retrain the models, which is common in malware detection business since new malware types appear frequently and they need to be added to the training sets. The trade-off is therefore challenging: the classical ML machine learning algorithms are fast but not so much accurate, on the other hand emerging deep learning methods are time consuming but more accurate in malware detection.

Signature method is an old technique used by antivirus software for malware detection. A signature is a short sequence of bytes that uniquely idetifies a malware type. However this method is not so effective due to ever changing malware types [18].

Malware analysis has two phases. In the first phase, so called malware discovery phase, malware is first caught and identifed. In the second phase, which is the malware classification phase, security systems try to identify or classify each threat sample as one of the appropriate malware families. For malware classification, feature vector selection methods are used in the classification process and this could be divided into two types: static and dynamic analysis.

Dynamic analysis method consists of executing the malware and monitoring its behavior, and stating changes in the executed environment. The complexity of environment setup is very high and the time needed to see the outcome of executing all malware is too long. However, this approach gives the most safe, good, and reliable results.

Static analysis method on the other hand implies examining the malware by analyzing the metadata of malware executables, assembly code instructions, and binary data in its content. This method requires a simple environment setup cost and the results can be obtained much faster.

The modern way of doing things for the classification of malware lies in machine learning and its classification sets of computer instructions. Classification sets of computer instructions consists of a process so called "supervised learning" approach. In this approach (more than two, but not a lot of) models are built from malware input data and its labeled class. The created models are used in the classification of malware. Furthermore, feature vectors are used for supervised learning which consists exactly two columns. The first column of a feature vector is a categorical (number or thing that changes) which describes a malware class and the second one has malware opcodes, which consists of code sections describing each malware and play an important role in the classification accuracy

ACM SE '18, March 29–31, 2018, Richmond, KY, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5696-1/18/03…$15.00
https://doi.org/10.1145/3190645.3190692

of a malware. The malware classification accuracy heavily depends on the selection of appropriate feature vectors.

On the other hand word embedding or vector space modelling is used for the classification of text in machine learning. Word2Vec [7] is a popular tool used in the literature especially for natural languages like English. In this model an association of each word is accomplished with a mathematical vector representation under which some structural or semantic relation holds. Despite being designed for natural language processing, it is also satisfactory for malware assembly code represention.

In section 2 we present the related work in malware detection and classification using machine learning-based techniques. Section 3 explains our deep learning based feature model for malware detection and classification. In section 4 we present and explain the dataset used for development and testing. Our classification architecture is explained in detail in section 5, and the evaluation measures are presented in section 6. We tested our method extensively, present the early results and discussed them in section 7. We conclude in section 8.

2 RELATED WORK

There are a lot of works in the literature on automatic malware detection using data mining and machine learning techniques [16].

In SAVE [13] and SAFE [1], heuristic static malware detection methods have been used and are considered some of the early works in this area, inspiring researchers in malware detection. These works proposed using patterns to detect the malicious content in executable files. Later other methods are developed to detect patterns from the header of portable executables (PE), or the from the body of PE, or both. Pattern detection is further done on the bytecode [14], or by disassembling the code, extracting opcodes, and searching for patterns in the opcodes for malware traces [10]. Packing and obfuscation are frequently used to hide malware code. Recent works addressed this issue by the automatic decyphering of obfuscated malwares with no apriori assumption about the obfuscation techniques[15].

Drew et al [3] used Strand, a gene sequence classifier that provides a robust classification strategy that can easily accommodate unstructured data with any alphabet including source code or compiled machine code, to demonstrate Stand's suitability for classifying malware. They have used the same dataset we used (BIG 2015) and reached 95% and above accuracy levels with less training times.

Santos et al. [10] used opcodes from executables for malware detection using data mining techniques. Popov [9] proposed to use word2vec, a recently developed popular tool to analyze natural language texts, for malware classification. Word2vec generates word embeddings for texts in the form of word vectors and then these vector representations for texts are used to classify texts. Popov used word2vec to generate word embeddings from malware opcodes, after machine instructions are fed into Capstone disassambler and opcodes are generated. Then these word vectors are used to classify executable codes using a convolutional neural network-based classifier. The system is tested with a small dataset of 2400 portable executables (PE) of just two class (malicious vs bening) with up to 97% success.

We also took a similar approach to malware classification using word2vec for PE representation with word embeddings as in [9]. We tested our system on a recently released large dataset with multi-class malware.

3 DEEP LEARNING-BASED MALWARE CLASSIFICATION

A great deal of research use neural networks for malware detection and analysis. For instance, a large scale approach on random projections and neural networks to classify malware has been done by Dahl et al. [2]. However, they have shown that increasing the number of hidden layers in the neural network did not provide significant accuracy gains. Saxe and Berlin [11] used feedforward based deep neural networks for the classification of static malware. But, dynamic analysis results are missing in their research. Disassembling obfuscated binaries may not give satisfactory inputs for classification. Huang et al. [5], on the other hand, use up to four hidden layers of feedforward neural network and evaluated multi-task learning ideas. Pascanu et al. [8] use recurrent networks for system call sequencing modeling, in order to create a "corpus" for malware opcodes. They test Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), and report considerably good classification accuracies.

In this work, classification of malware will be performed on deep neural network architectures. Deep learning based malware classification can be performed by using a Convolutional Neural Network based feature extraction of a malware [9], using an autoencoder based feature learning [17] or recurrent neural network based classifier [8]. In each of aforementioned works, there are more than one feature representing a malware and each of these are combined together for better classification accuracy.

We use a shallow deep learning network based on Word2Vec vector space model for representing malwares and at the end use a Gradient search algorithm based on Gradient Boosting Machine for malware classification task.

4 DATASET

Microsoft released a large malware dataset in 2015 WWW Conference. It is called Microsoft Malware Classification Challenge Dataset (BIG 2015)[1]. We used this dataset in this work. Training dataset has 10869 sample malware from 9 different classes of malware. These malware classes are (1) Ramnit, (2) Lollipop, (3) Kelihos_ver3, (4) Vundo, (5) Simda, (6) Tracur, (7) Kelihos_ver1, (8) Obfuscator.ACY, (9) Gatak. The data distribution among these malware classes are shown in Figure 1. We sampled 100, 200, 300, and 398 files from all classes (except class 5) and used these samples in order to show the performance of generated classification models trained with these low number of samples. We excluded class 5 (Simda) samples since it has a very low number of samples (42) and we wanted an even number of samples from each class in the experiments to remove the bias.

[1]https://www.kaggle.com/c/malware-classification

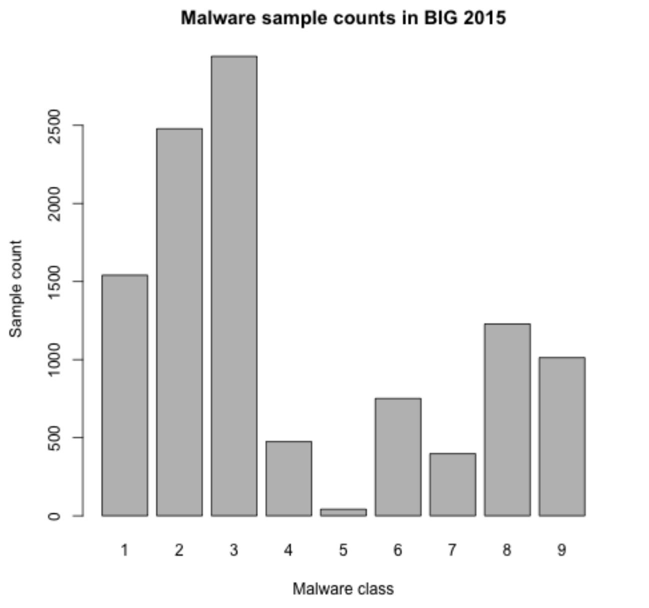

Figure 1: Malware sample distribution in Microsoft Malware Classification Challenge Dataset

5 CLASSIFICATION ARCHITECTURE

The choice of malware features that will be used to represent each malware sample for the classification task directly affects the classification performance in terms of accuracy and execution time. In this section, our malware classification architecture using a malware representation based on Word2Vec vector space representation will be explained.

5.1 Malware Representation

Malware samples can be represented in different ways and in this section some of them will be reviewed. One can use hex codes and the assembly notation when viewing malware code. The sequence of hexadecimal digits is the accumulation of successive 16-bytes words like the one shown in Listing 1.

Listing 1: Assembly code / Opcode

```
0x401180  55  31  C0  89  E5  B9  11  00
          00  00  57  56  8D  55  A4  53
```

The starting address is represented as the first value and the consecutive numbers are the machine codes in the memory, and each value (byte) represents an element for the PE, like instruction codes or data.

The Interactive Disassembler (IDA) [6] is a disassembler, which performs reverse engineering and automatic analysis of binary applications using cross-references between code sections, knowledge of API call stack, and other information. For example, in IDA a byte sequence can be viewed as shown in Listing 2.

Listing 2: Assembly view

```
0x401180:55                  push ebp
0x401181:31 c0               xor eax, eax
0x401183:89 e5               mov ebp, esp
0x401185:b9 11 00 00 00      mov ecx, 0x11
0x40118a:57                  push edi
0x40118b:56                  push esi
0x40118c:8d 55 a4            lea edx, dword
0x40118f:53                  push ebx
```

5.2 Word2Vec

The disassembled codes, namely opcodes, are assembly codes, and are human interpretable representation of binary codes. We strip the assembly codes from their arguments in order to obtain the assembly operation sequences only. The final representation of opcode sentences are similar to the example shown in Listing 3.

Listing 3: Concatenated Opcode Words

```
push xor mov mov push push lea push
```

The proposed feature representation technique used in this paper is Word2vec [7], which recently gained popularity in the analysis of natural language text. Word2Vec is used for generating embeddings for words from text in natural languages. At the end of this learning process, word embeddings (vector representations of words) are generated for each input instance. These vectors show the syntactic and semantic relationships between words; words that share common context are located in close proximity in the vector space.

A number of parameters are used for building Word2Vec representations that are listed in Table 1.

Table 1: Vector Space Parameters

vector length	The length of the word vectors.
window length	Length of the context window word.
sample sent frequency	The frequency of words. Higher frequency words will be down sampled.
learning initial frequency	Initial learning frequency.
training iterations	The number of training epochs.

5.3 Classification

Over the years the scientific community proposed several classification techniques. The choice of the most appropriate classifier for a given task is often guided by previous experience in different domains, as well as by trial and error procedures. However, recently some researchers evaluated the performances of about 180 different classifiers arising from different families, using various datasets, and they concluded that random forests and SVM are the two classification mechanisms that have the highest likelihood to produce good performances [12].

On the other hand, Gradient Boosting Machine [4] (for Regression and Classification) is a forward learning ensemble method. The guiding heuristic is that good predictive results can be obtained through increasingly refined approximations. GBM sequentially builds regression trees on all the features of the dataset in a fully distributed way that is each tree is built in parallel.

A number of parameters are used for building a GBM for the classification task.

6 EVALUATION MEASURES

Classification performance can be assessed using two measures, namely, accuracy and logarithmic loss. Accuracy measures the fraction of correct predictions. Accuracy alone is usually not enough to assess the robustness of the prediction, we also measure the logarithmic loss (logloss), which is a sensitive measurement of accuracy that incorporates the concept of probabilistic confidence. It is the cross entropy between the distribution of the true labels and the predicted probabilities. As shown in Equation 1, it is the negative log likelihood of the model.

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij}) \tag{1}$$

where N is the number of observations, M is the number of class labels, log is the natural logarithm, y_{ij} is 1 if observation i is in class j and 0 otherwise, and p_{ij} is the predicted probability that observation i is in class j.

7 EVALUATION

In this work, we experimented with 4 separate sampled datasets. There are 8 different malware classes in experiment dataset. We conducted 4 different runs for 4 seperate sets. These are listed in Table 2.

Table 2: Experiment Setup Parameters

Experiment No	Sample per Class	Total Samples
1	100	800
2	200	1600
3	300	2400
4	398	3184

k-fold cross validation is used to reduce bias. k-value can be set between values 5 and 10. If a greater value is chosen, then the cross validation time will be greater, too. In order to calculate a baseline model for cross validation we set k-fold cross validation value to 5.

The following Word2Vec parameter values are used. *vector size:200, sent sample rate:0,initial learning rate:1,epochs:10*

Experiments are conducted in parallel within a cluster consisting of 3 servers, each has 32-core Xeon CPU and 32GB of memory.

Models are evaluated with their logloss value. After GBM models are generated, the chosen model is the one that has the smallest logloss value. Confusion matrices for the test dataset results are shown in Table 3, 4, 5, 6 for each of the 4 experiments. Error rates are also listed at the rightmost column in each table. The error rates are at most 6%, which is clearly successful.

Figure 2 shows the error rate drop as the sample size increases. That is a 50% drop (approx.) from 0.0688 to 0.0386. This means that bigger malware sample data will improve the results further.

Table 3: Experiment 1 - Cross Validation (5 fold) Confusion Matrix

Class	1	2	3	4	5	6	7	8	Error	Rate
1	89	5	0	1	0	0	3	2	0.1100	11/100
2	7	90	0	0	2	0	1	0	0.1000	10/100
3	0	0	97	2	0	0	0	1	0.0300	3/100
4	1	1	0	97	1	0	0	0	0.0300	3/100
5	2	0	0	3	93	0	0	2	0.0700	7/100
6	1	0	0	0	0	95	1	3	0.0500	5/100
7	4	1	0	1	2	1	90	1	0.1000	10/100
8	1	0	1	3	1	0	0	94	0.0600	6/100
Total	105	97	98	107	99	96	95	103	**0.0688**	55/800

Table 4: Experiment 2 - Cross Validation (5 fold) Confusion Matrix

Class	1	2	3	4	5	6	7	8	Error	Rate
1	177	7	0	2	4	1	5	4	0.1150	23/200
2	6	186	1	1	3	0	0	2	0.0653	13/199
3	0	0	198	2	0	0	0	0	0.0100	2/200
4	0	3	0	196	0	0	0	1	0.0200	4/200
5	6	2	0	0	191	0	0	1	0.0450	9/200
6	0	0	0	2	0	198	0	0	0.0100	2/200
7	8	2	1	6	1	0	181	1	0.0950	19/200
8	1	1	0	2	1	0	0	195	0.0250	5/200
Total	198	201	200	211	200	199	186	204	**0.0482**	77/1599

Table 5: Experiment 3 - Cross Validation (5 fold) Confusion Matrix

Class	1	2	3	4	5	6	7	8	Error	Rate
1	265	12	0	1	2	0	13	6	0.1137	34/299
2	10	284	0	3	1	0	0	0	0.0470	14/298
3	0	0	298	1	0	0	0	1	0.0067	2/300
4	0	2	0	298	0	0	0	0	0.0067	2/300
5	5	0	0	0	287	0	0	4	0.0401	12/299
6	2	0	0	6	0	292	0	0	0.0267	8/300
7	20	2	0	6	2	1	265	3	0.1137	34/299
8	4	2	0	0	1	0	3	290	0.0333	10/300
Total	306	302	298	318	293	293	281	304	**0.0484**	116/2395

Table 6: Experiment 4 - Cross Validation (5 fold) Confusion Matrix

Class	1	2	3	4	5	6	7	8	Error	Rate
1	365	12	0	1	8	0	8	4	0.0829	33/398
2	15	381	0	0	0	0	1	1	0.0427	17/398
3	0	0	397	1	0	0	0	0	0.0025	1/398
4	0	2	0	396	0	0	0	0	0.0050	2/398
5	4	0	0	1	388	1	0	4	0.0251	10/398
6	1	1	0	7	1	387	1	0	0.0276	11/398
7	28	3	0	8	1	1	354	3	0.1106	44/398
8	1	0	0	1	2	0	1	393	0.0126	5/398
Total	414	399	397	415	400	389	365	405	**0.0386**	123/3184

8 CONCLUSION

In this paper, we presented a new malware representation method based on static analysis, we used sequences of opcodes without arguments. We have shown that with a low dimension of Word2Vec

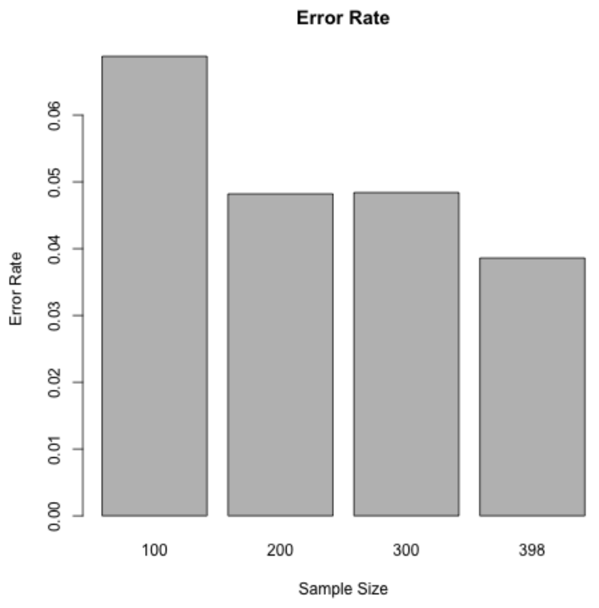

Figure 2: Malware sample distribution in Microsoft Malware Classification Challenge Dataset

feature vectors and a boosting classifier like GBM, malware classification accuracy can easily reach to between 94% and 96%. If the cross-validation k-fold value is greater, the accuracy might be higher. Also noted that GBM model tree depth can be searched with the help of grid search (tree pruning) would facilitate a wide range model lookup in the GBM tree search and this could lead to better accuracy. For future work, hyperparameter optimization stages will be added to the model search, dataset will be extended to include every class of malware in the wild and also finding semantic relationships between malware opcodes will be a research direction.

REFERENCES

[1] Mihai Christodorescu and Somesh Jha. 2003. Static Analysis of Executables to Detect Malicious Patterns. In *Proceedings of the 12th Conference on USENIX Security Symposium - Volume 12 (SSYM'03)*. USENIX Association, Berkeley, CA, USA, 12–12. http://dl.acm.org/citation.cfm?id=1251353.1251365

[2] George E Dahl, Jack W Stokes, Li Deng, and Dong Yu. 2013. Large-scale malware classification using random projections and neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 3422–3426.

[3] Jake Drew, Tyler Moore, and Michael Hahsler. 2016. Polymorphic malware detection using sequence classification methods. In *Security and Privacy Workshops (SPW), 2016 IEEE*. IEEE, 81–87.

[4] Jerome H. Friedman. 2000. Greedy Function Approximation: A Gradient Boosting Machine, In Annals of Statistics. *Annals of Statistics* 29, 1189–1232. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.29.9093

[5] Wenyi Huang and Jack W Stokes. 2016. MtNet: a multi-task neural network for dynamic malware classification. In *Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 399–418.

[6] "IDA". 2013. "Ida : Disassembler and debugger. https://www.hex-rays.com/products/ida/". (2013).

[7] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 3111–3119. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

[8] Razvan Pascanu, Jack W Stokes, Hermineh Sanossian, Mady Marinescu, and Anil Thomas. 2015. Malware classification with recurrent networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 1916–1920.

[9] Igor Popov. 2017. Malware detection using machine learning based on word2vec embeddings of machine code instructions. In *Data Science and Engineering (SSDSE), 2017 Siberian Symposium on*. IEEE, 1–4.

[10] Igor Santos, Felix Brezo, Xabier Ugarte-Pedrero, and Pablo G. Bringas. 2013. Opcode sequences as representation of executables for data-mining-based unknown malware detection. *Information Sciences* 231 (2013), 64 – 82. https://doi.org/10.1016/j.ins.2011.08.020 Data Mining for Information Security.

[11] Joshua Saxe and Konstantin Berlin. 2015. Deep neural network based malware detection using two dimensional binary program features. In *Malicious and Unwanted Software (MALWARE), 2015 10th International Conference on*. IEEE, 11–20.

[12] Alexander Statnikov and Constantin F Aliferis. 2007. Are random forests better than support vector machines for microarray-based cancer classification?. In *AMIA annual symposium proceedings*, Vol. 2007. American Medical Informatics Association, 686.

[13] A. H. Sung, J. Xu, P. Chavez, and S. Mukkamala. 2004. Static Analyzer of Vicious Executables (SAVE). In *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC '04)*. IEEE Computer Society, Washington, DC, USA, 326–334. https://doi.org/10.1109/CSAC.2004.37

[14] S. Momina Tabish, M. Zubair Shafiq, and Muddassar Farooq. 2009. Malware Detection Using Statistical Analysis of Byte-level File Content. In *Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics (CSI-KDD '09)*. ACM, New York, NY, USA, 23–31. https://doi.org/10.1145/1599272.1599278

[15] B. Yadegari, B. Johannesmeyer, B. Whitely, and S. Debray. 2015. A Generic Approach to Automatic Deobfuscation of Executable Code. In *2015 IEEE Symposium on Security and Privacy*. 674–691. https://doi.org/10.1109/SP.2015.47

[16] Yanfang Ye, Tao Li, Donald Adjeroh, and S Sitharama Iyengar. 2017. A survey on malware detection using data mining techniques. *ACM Computing Surveys (CSUR)* 50, 3 (2017), 41.

[17] Mahmood Yousefi-Azar, Vijay Varadharajan, Len Hamey, and Uday Tupakula. 2017. Autoencoder-based feature learning for cyber security applications. In *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 3854–3861.

[18] Mikhail Zolotukhin and Timo Hamalainen. 2014. Detection of zero-day malware based on the analysis of opcode sequences. (01 2014), 386–391.