IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2347-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 10, Issue 10, October 2021

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 7.569

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 10, Issue 10, October 2021 ||

| DOI:10.15680/IJIRSET.2021.1010113|

BertMal: A Novel Framework for Malware Detection using BERT

Chendev Petrovich[1], Sizova Yakovna[2], Roman Tsarev[3]

Graduate Student, Siberian Federal University, Russia[1]

Graduate Student, Saint Petersburg State University, Russia[2]

Associate Professor, Siberian Federal University, Russia [3]

ABSTRACT: Malware is a worldwide epidemic. Studies suggest that the impact of malware is getting worse. Malware detectors are the primary tools in defence against malware. The quality of such a detector is determined by the techniques it uses. It is therefore imperative that we study malware detection techniques and understand their strengths and limitations. This paper examines several malware detection techniques and offers an opportunity to compare them against one another aiding in the decision-making process involved with developing a secure application/system. Finally, we present a novel framework called BertMal that leverages RoBERTa, a BERT-based transformer-driven model to detect malware at extremely high precision. Our model is able to outperform nearly all traditional methods for malware detection.

KEYWORDS: Malware Detection, Virus Analysis, Deep Learning, IDS

I. INTRODUCTION

A malware detector is the implementation of some malware detection technique(s). The malware detector attempts to help protect the system by detecting malicious behaviour. The malware detector may or may not reside on the same system it is trying to protect. The malware detector performs its protection through the manifested malware detection technique(s) and serves as an empirical means of evaluating malware detection techniques' detection capabilities. Malware detectors take two inputs. One input is its knowledge of the malicious behaviour. In anomaly-based detection, the inverse of this knowledge comes from the learning phase. So theoretically, anomaly-based detection knows what anomalous behaviour is based on its knowledge of what is normal. Since anomalous behaviour subsumes malicious behaviour, some sense of maliciousness is captured by anomaly-based detection.

If the malware detector employs a signature-based method, its knowledge of what is malicious comes from its repository, which is usually updated/maintained manually by people who were able to identify the malicious behaviour and express it in a form amenable for the signature repository, and ultimately for a machine to read. The other input that the malware detector must take as input is the program under inspection. Once the malware detector has the knowledge of what is considered malicious behaviour (normal behaviour) and the program under inspection, it can employ its detection technique to decide if the program is malicious or benign. Although Intrusion Detection Systems (IDS) and malware detectors are sometimes used synonymously, a malware detector is usually only a component of a complete IDS.

Techniques used for detecting malware can be categorized broadly into two categories: anomaly-based detection and signature-based detection. An anomaly-based detection technique uses its knowledge of what constitutes normal behaviour to decide the maliciousness of a program under inspection. A special type of anomaly-based detection is referred to as specification-based detection. Specification-based techniques leverage some specification or rule set of what valid behaviour in order is to decide the maliciousness of a program under inspection. Programs violating the specification are considered anomalous and usually, malicious. Signature-based detection [10] uses its characterization of what is known to be malicious to decide the maliciousness of a program under inspection. As one may imagine this characterization or signature of the malicious behaviour is the key to a signature-based detection method's effectiveness.

Anomaly-based detection usually occurs in two phases–a training (learning) phase and a detection (monitoring) phase. During the training phase the detector attempts to learn normal behaviour. The detector could be learning the behaviour of the host or the PUI or a combination of both during the training phase. A key advantage of anomaly-based detection

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 10, Issue 10, October 2021 ||

| DOI:10.15680/IJIRSET.2021.1010113|

is its ability to detect zero-day attacks. Weaver, et al. [8] describe zero-day exploits. Similar to zero-day exploits, zero-day attacks [9] are attacks that are previously unknown to the malware detector. The two fundamental limitations of this technique is its high false alarm rate and the complexity involved in determining what features should be learned in the training phase.

II. RELATED WORK

According to recent studies, malicious software (malware) is increasing at an alarming rate, and some malware can hide in the system by using different obfuscation techniques. In order to protect computer systems and the Internet from the malware, the malware needs to be detected before it affects a large number of systems. Recently, there have been several studies on malware detection approaches. However, the detection of malware still remains problematic. Signature-based and heuristic-based detection approaches are fast and efficient to detect known malware, but especially signature-based detection approaches have failed to detect unknown malware. On the other hand, behaviour-based, model checking-based, and cloud-based approaches perform well for unknown and complicated malware; and deep learning-based, mobile devices-based, and IoT-based approaches also emerge to detect some portion of known and unknown malware. However, no approach can detect all malware in the wild. This shows that to build an effective method to detect malware is a very challenging task, and there is a huge gap for new studies and methods. This paper presents a detailed review on malware detection approaches and recent detection methods which use these approaches. Paper goal is to help researchers to have a general idea of the malware detection approaches, pros and cons of each detection approach, and methods that are used in these approaches.

Wang and Stolfo [5] present PAYL, a tool which calculates the expected payload for each service (port) on a system. A byte frequency distribution is created which allows for a centroid model to be developed for each of the host's services. This centroid model is calculated during the learning phase. The detector compares incoming payloads with the centroid model, measuring the Mahalanobis distance between the two. The Mahalanobis distance takes into account not only the mean values of a feature vector but also variance and covariance yielding a stronger statistical measure of similarity. If the incoming payload is too far from the centroid model (a large Mahalanobis distance value), then the payload is considered to be malicious. Wang and Stolfo evaluated their technique on the 1999 MIT Lincoln Labs data. This data contains 3 weeks of training data, and 2 weeks of testing data. Of the 201 attacks in the Lincoln Labs data, 97 of them should be detected by Wang and Stolfo's technique. The authors' technique could detect 57 of the 97 attacks it should have been able to catch. Overall, the detection rate for their technique was approximately 60 percent when the false positive rate was 1 percent or lower.

Hofmeyr et al. [2] propose a technique that monitors system call sequences in order to detect maliciousness. First, profiles must be developed that represent the normal behaviour of the system's services. "Normal" in this technique is defined in terms of short sequences of system calls. Although intrusions may be based on other parameters, these other parameters are ignored. Hamming distance is used to determine how closely a system call sequence resembles another. A threshold must be set to determine whether a process is anomalous. Typically, processes showing large Hamming distance values are anomalous. Hofmeyr et al.'s method was able to find intrusions that attempted to exploit various UNIX programs like sendmail, lpr, and ftpd.

Taylor and Alves-Foss [5] propose a computationally low-cost approach to detecting anomalous traffic. Their approach is referred to as the Network Analysis of Anomalous Traffic Events (NATE). This technique focuses on attacks which exploit network protocol vulnerabilities. Their approach relies on the assumption that malicious packets tend to have a large number of syn, fin, and reset packets, while having a low number of ack packets. This approach also relies on abnormalities showing up in the number bytes being transferred for each packet. A session is defined as information flow from a source to some IP address and port number pair. Multivariate cluster analysis is used to group the normal TCP/IP sessions.

Several other works (Mahindru et al. [15], Wu et al. [16], Ni et al. [17]) have attempted to solve the problem of malware detection using deep learning methods. Ravi and Manoharan [18] have also experimented with using API call sequences. However, a major limitation in all of these prior works is that they require significant data for training that includes a high percentage of malware samples. The only work that simulates the real-world low-positive scenario and defines an algorithm for detecting malware when the amount of training data is very low is the one proposed by Oak et al. [4].

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 10, Issue 10, October 2021 ||

| DOI:10.15680/IJIRSET.2021.1010113|

Oak et al [4] propose a new approach to malware detection that uses a pre-trained BERT model to detect malware using extremely minimal samples of malware. This technique shows that pretraining on a resource-heavy task and dataset (such as any English language task) tunes a model to obtain the optimal set of parameters, which can easily be fine-tuned on a downstream task like malware detection. The presented algorithm is able to achieve an F-1 score of more than 90% using less than 1% of malware in the training phase, solving for the first time the important label unavailability problem that is prevalent in malware and virus detection. This remains the state-of-the-art technique in low-positive malware detection till date. We use this framework and improve upon it to build BertMal, as it accurately reflects real-world scenarios and allows detecting malware with very few samples.

III. METHODOLOGY

We propose BertMal, a new framework for malware detection. We leverage API call sequences as well as processor-level instruction sequences and construct a high-dimensional feature vector. Following Oak et al [4], we use BERT as the model of choice for classification. We also adopt the low-positive algorithm from Oak et al [4], that uses a pretrained model on high-resource tasks as the base model.

However, our framework differs from them in the choice of the pretraining task. We replace the model trained on Wikipedia by a model trained over the Drebin Malware Dataset as a base. Our hope is that by choosing a base model close to the actual task, we will be able to obtain a more optimal set of weights and thus better performance.

Our complete architecture is shown in Figure 1, and it consists of two modules: online and offline modules.

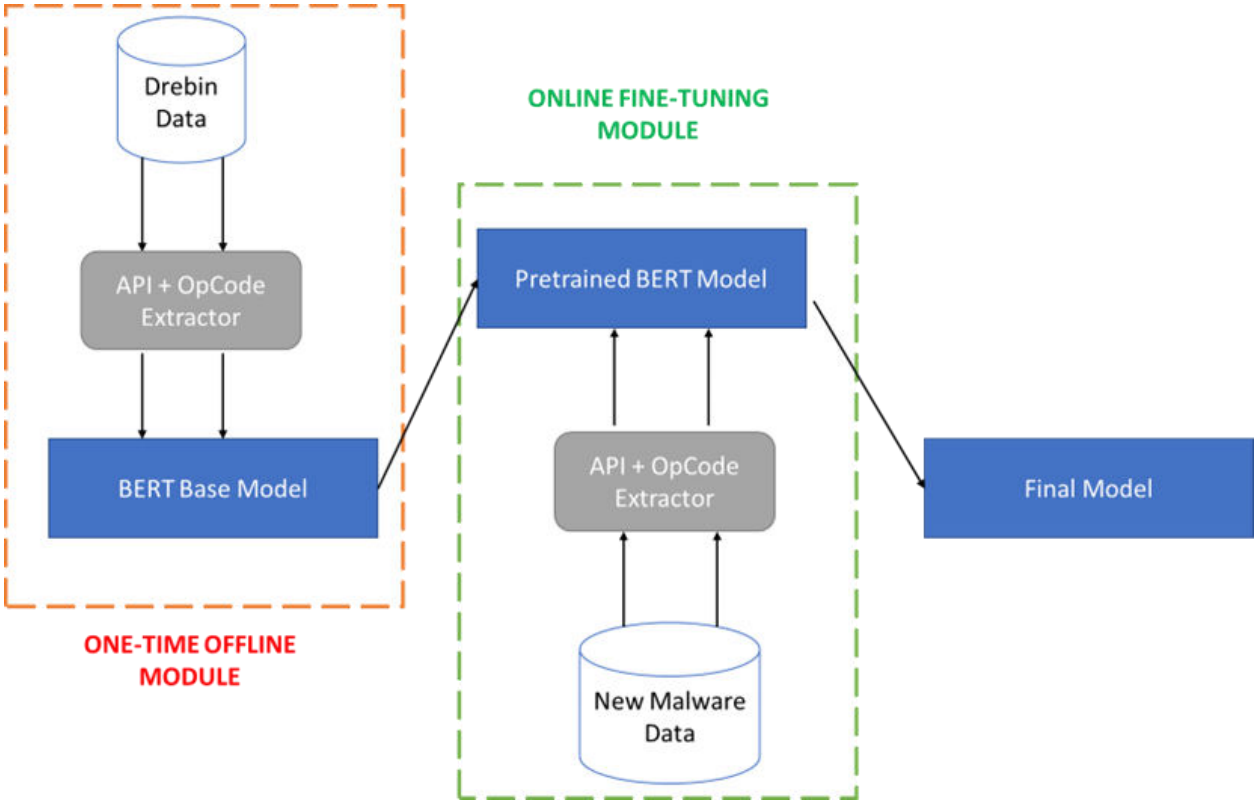

Fig.1.Model Architecture with Online and Offline Modules

Offline Module. It is responsible for the pretraining part. This is directly adapted from the algorithm in Oak et al [4], and we reuse the same configurations they did, except with the Drebin dataset. This is a one-time effort, meaning that pretrained models can be stored for future use. As it is one-time, this can be implemented on a GPU for quicker training. Here, we also extract opcode along with API call sequence. Opcode analysis can help detect malware by examining the underlying machine instructions executed by a program to determine its behaviour. Malware often has a

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 10, Issue 10, October 2021 ||

| DOI:10.15680/IJIRSET.2021.1010113|

distinctive pattern of opcodes that can be identified through analysis. If the opcode sequence of a binary matches the known pattern of malware, the system can flag it as potentially malicious. Additionally, opcode analysis can also reveal obfuscation techniques used by malware authors to hide their code, allowing for more effective detection.

Pretraining in machine learning refers to the process of training a model on a large dataset before fine-tuning it for a specific task. This can be done in several ways, but one common method is to use a pre-trained model as the starting point for a new model. Pretraining can help in machine learning in several ways:

1. It can reduce the amount of data required to train a model for a specific task, as the model has already learned some general features from the large dataset.
2. It can improve the performance of a model by providing it with a better set of initial weights, as the pre-trained model has already learned useful features from the large dataset.
3. It can help in situations where the data for a specific task is limited, by leveraging knowledge learned from a larger dataset.
4. It can save time and computational resources by using pre-trained models, as training a model from scratch can be computationally expensive.
5. It can boost the performance of models on a variety of tasks, such as natural language understanding, computer vision, and speech recognition.

Online Module. This module uses the pretrained model and continuously tunes it on newly available data. Any time new malware comes into discovery, it can be used for fine-tuning to update the model. As the algorithm we are using [4] is specifically for low-positive settings, we do not need to wait for a large number of malware to be discovered. This will produce a final model that can be used for real-time inference.

Pretrained models can be used for inference by loading their trained parameters and using them to make predictions on new data without additional training. Here are the steps for using a pretrained model for inference:

1. Load the pretrained model's architecture and trained parameters into the memory.
2. Pre-process the input data to match the format expected by the model.
3. Pass the input data through the model to make predictions.
4. Postprocess the predictions to obtain the final results.

In inference mode, the model's parameters are not updated and only the forward pass is executed to make predictions. Pretrained models can save a significant amount of time and resources compared to training a model from scratch, as they have already learned relevant features from a large amount of data.

IV. EXPERIMENTAL RESULTS

In order to examine the effectiveness of our approach, we compared it against two baselines: a standard random forest (which is traditionally used in computer security problems for classification and fraud detection) and the algorithm from Oak et al [4] which is state-of-the-art and hence a reliable baseline. We evaluate the models under two settings: a low positive setting (0.5% malware) and a well-balanced setting (50% malware).

Accuracy is a commonly used metric in machine learning to evaluate the performance of a classifier. It is defined as the ratio of the number of correct predictions made by the classifier to the total number of predictions made.Accuracy is a simple and intuitive metric, but it may not always provide a complete picture of a classifier's performance, especially when the dataset is imbalanced (i.e., has unequal number of instances of different classes). In such cases, precision, recall, F1 score, and other metrics may be more informative.

Precision and recall are important evaluation metrics in classification because they provide different perspectives on the performance of a classifier.Precision measures the proportion of true positive predictions (correct positive predictions) out of all positive predictions made by the classifier. A high precision indicates that the classifier is confident in its positive predictions and has a low rate of false positive predictions. Precision is important in situations where false positive predictions have high costs or consequences.

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 10, Issue 10, October 2021 ||

| DOI:10.15680/IJIRSET.2021.1010113|

Recall (also known as sensitivity or true positive rate) measures the proportion of true positive predictions out of all actual positive instances. Recall indicates the ability of the classifier to identify all positive instances in the data. A high recall indicates that the classifier has a low rate of false negative predictions and is good at detecting positive instances. Recall is important in situations where false negative predictions have high costs or consequences.

It is important to note that precision and recall are often in trade-off with each other. Increasing precision may decrease recall, and vice versa. When a classifier makes a prediction, it can be one of the following:

1. True Positive (TP): The classifier correctly predicts a positive instance.
2. False Positive (FP): The classifier incorrectly predicts a positive instance.
3. True Negative (TN): The classifier correctly predicts a negative instance.
4. False Negative (FN): The classifier incorrectly predicts a negative instance.

When a classifier is designed to minimize false positives, it becomes more selective in making positive predictions, which leads to a decrease in the number of false positive predictions but also a decrease in the number of true positive predictions, resulting in a decrease in recall.On the other hand, when a classifier is designed to maximize recall, it becomes less selective in making positive predictions, which leads to an increase in the number of true positive predictions but also an increase in the number of false positive predictions, resulting in a decrease in precision.

Thus, precision and recall are in trade-off with each other. The best classifier is the one that balances precision and recall in a way that is appropriate for the specific problem being solved.The F1 score is a balanced metric that takes into account both precision and recall to provide a single number for evaluating the performance of a classifier.

Under a low positive setting, we find that our model achieves performance comparable to Oak et al [4], but greatly outperforms the random forest. The F-1 score that BertMal achieves is comparable to the one from [4].

Model	Accuracy	Precision	Recall	F-1 Score
Random Forest	0.985	0.611	0.709	0.656
Oak et al [4]	0.991	0.962	0.846	0.900
BertMal	0.895	0.911	0.868	0.890

Under the balanced data setting, we find that our model outperforms both the baselines. The F-1 score of 0.942 is higher than either of the two models. For a balanced dataset (i.e., when enough malware examples are available for training a classifier), our approach can be used. This can be used to detect traditional malware, or malware that has enough propagation in devices.

Model	Accuracy	Precision	Recall	F-1 Score
Random Forest	0.985	0.611	0.709	0.656
Oak et al [4]	0.992	0.950	0.896	0.922
BertMal	0.993	0.989	0.899	0.942

.

V. CONCLUSION

Malware is a menace that affects computers, smartphones and multiple other devices around the world. Malware can cause catastrophic consequences such as theft of sensitive user data, misuse of resources for nefarious activities like crypto mining, or corrupt critical infrastructure leading to your power outages or even loss of life. Detecting malware, therefore, is the need of the hour. In this paper, we present BERTMal, a novel framework for detecting malware. Extending prior work, we develop a system based on the BERT model. Our methodology uses a BERT base model and

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 10, Issue 10, October 2021 ||

| DOI:10.15680/IJIRSET.2021.1010113|

pretrains it on malware data. In addition to API calls that prior work uses, we also extract opcode level features that provide fine-grained information on the operational aspects of the malware file. This pretrained model is then fine-tuned on specific instances of malware. Our results on the standard Drebin dataset show that BERTMal is able to achieve a performance comparable with prior work in both low and high positive settings.

REFERENCES

[1] D. Ellis, J. Aiken, K. Attwood, and S. Tenaglia. A behavioral approach to worm detection. In Proceedings of the 2004 ACM Workshop on Rapid Malcode, pages 43–53, 2004

[2] S. Hofmeyr, S. Forrest, and A. Somayaji. Intrusion detection using sequences of system calls. Journal of Computer Security, pages 151 – 180, 1998.

[3] W. Masri and A. Podgurski. Using dynamic information flow analysis to detect attacks against applications. In Proceedings of the 2005 Workshop on Software Engineering for secure systems –Building Trustworthy Applications, 30, May 2005.

[4] R. Oak, M. Du, D. Yan, H. Takawale, and I. Amit, "Malware detection on highly imbalanced data through sequence modeling," in Proceedings of the 12th ACM Workshop on artificial intelligence and security, 2019, pp. 37-48.

[5] C. Taylor and J. Alves-Foss. Nate – network analysis of anomalous traffic events, a low-cost approach. New Security Paradigms Workshop, 2001.

[6] A. Vasudevan and R. Yerraballi. Spike: Engineering malware analysis tools using unobtrusive binary-instrumentation. In Proceedings of the 29th Australasian Computer Science Conference, pages 311–320, 2006.

[7] G. E. Suh, J. Lee, and S. Devadas. Secure program execution via dynamic information flow tracking. International Conference Architectural Support for Programming Languages and Operating Systems, 2004

[8] N. Weaver, V. Paxon, S. Staniford, and R. Cunningham. A taxonomy of computer worms. In Proceedings of the 2003 ACM Workshop on Rapid Malcode, pages 11–18, 2003.

[9] Vinod, P., Jaipur, R., Laxmi, V. and Gaur, M., 2009, March. Survey on malware detection methods. In Proceedings of the 3rd Hackers' Workshop on computer and internet security (IITKHACK'09) (pp. 74-79).

[10] Moser, A., Kruegel, C. and Kirda, E., 2007, December. Limits of static analysis for malware detection. In Twenty-third annual computer security applications conference (ACSAC 2007) (pp. 421-430). IEEE.

[11] Ye, Y., Wang, D., Li, T. and Ye, D., 2007, August. IMDS: Intelligent malware detection system. In Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 1043-1047).

[12] Gavriluț, D., Cimpoeșu, M., Anton, D. and Ciortuz, L., 2009, October. Malware detection using machine learning. In *2009 International multiconference on computer science and information technology* (pp. 735-741). IEEE.

[13] Liu, L., Wang, B.S., Yu, B. and Zhong, Q.X., 2017. Automatic malware classification and new malware detection using machine learning. *Frontiers of Information Technology & Electronic Engineering*, *18*(9), pp.1336-1347.

[14] Ham, H.S. and Choi, M.J., 2013, October. Analysis of android malware detection performance using machine learning classifiers. In *2013 international conference on ICT Convergence (ICTC)* (pp. 490-495). Ieee.

[15] Mahindru, A. and Singh, P., 2017, February. Dynamic permissions based android malware detection using machine learning techniques. In *Proceedings of the 10th innovations in software engineering conference* (pp. 202-210).

[16] Wu, W.C. and Hung, S.H., 2014, October. DroidDolphin: a dynamic Android malware detection framework using big data and machine learning. In *Proceedings of the 2014 conference on research in adaptive and convergent systems* (pp. 247-252).

[17] Ni, S., Qian, Q. and Zhang, R., 2018. Malware identification using visualization images and deep learning. *Computers & Security*, *77*, pp.871-885.

[18] Ravi, C. and Manoharan, R., 2012. Malware detection using windows api sequence and machine learning. *International Journal of Computer Applications*, *43*(17), pp.12-16.

[19] Fatima, A., Maurya, R., Dutta, M.K., Burget, R. and Masek, J., 2019, July. Android malware detection using genetic algorithm based optimized feature selection and machine learning. In *2019 42nd International conference on telecommunications and signal processing (TSP)* (pp. 220-223). IEEE.

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor:
7.569

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details