Webology (ISSN: 1735-188X)
Volume 18, Number 4, 2021

Victory: A Framework For Fast Detection Of Iot Malware

Henrik Bellante*[1] , Manuel Vivier[2] , Jonathan Kirsch[3]

[1]Department of Computer Science, University of Urbino, Italy.

[2]Department of Computer Science, University of Urbino, Italy.

[3]Department of Computer Science, University of Urbino, Italy.

ABSTRACT

The Internet of Things (IoT) has become an integral part of modern life, with billions of connected devices used in homes, businesses, and industry. Unfortunately, the rapid growth of the IoT has also created a new frontier for malware attacks. To combat these threats, we present VICTORY, a framework for fast detection of IoT malware. Our approach leverages the power of machine learning algorithms to analyze network traffic and detect malicious behavior in real-time. VICTORY is designed to be scalable, efficient, and able to handle the large amounts of data generated by IoT devices. The framework includes several key components, including a feature extractor, a classifier, and a decision engine. The feature extractor is responsible for extracting relevant features from the network traffic data, while the classifier uses these features to identify malware. The decision engine integrates the results of the classifier and provides a unified view of the network. Additionally, we use a transformer-based architecture so as to quickly train the model with new malware samples as they are discovered. Our experiments show that VICTORY outperforms existing methods in terms of accuracy and speed, providing a new and effective approach to IoT malware detection. This framework has the potential to improve the security of IoT systems and help protect the privacy of users.

KEYWORDS Malware, Deep Learning, IoT, Virus, Network Security.

INTRODUCTION

The Internet of Things (IoT) refers to the connection of physical devices, vehicles, home appliances, and other items embedded with electronics, software, sensors, and connectivity which enables these objects to connect and exchange data with each other. The IoT has the potential to revolutionize many aspects of our daily lives by allowing us to monitor, control and automate a wide range of devices and systems through a single interface. The IoT represents a major technological advance, connecting a vast number of devices to the internet and allowing them to share data and work together to achieve common goals.

The growth of the IoT has been driven by the rapid advancement of technology, including the miniaturization of electronics, the development of wireless networking, and the increasing prevalence of high-speed internet connectivity. This growth has led to the creation of a vast network of interconnected devices, which has the potential to transform many industries, including healthcare, energy, transportation, and agriculture. For example, in healthcare, the IoT can help to improve patient outcomes by providing real-time monitoring and alerting, while in agriculture, it can increase efficiency and reduce waste by providing real-time monitoring of crop health and soil conditions.

Despite the many benefits of the IoT, there are also significant security and privacy concerns that must be addressed. With billions of connected devices, the IoT creates a vast network of potential targets for hackers and malicious actors. Additionally, the data generated by IoT devices can be extremely sensitive and private, and it is important to ensure that this data is properly protected and used in an ethical and responsible manner.
To fully realize the potential of the IoT, it is essential that these security and privacy concerns be addressed. This will require a coordinated effort by researchers, industry, and governments to develop new technologies and standards that can protect the privacy and security of IoT devices and the data they generate. This effort must also include the development of new policies and regulations to ensure that the IoT is used in a responsible and ethical manner.
In conclusion, the IoT represents a major technological advance that has the potential to transform many aspects of our lives. However, to fully realize this potential, it is essential that the security and privacy concerns associated with the IoT be addressed. This will require a coordinated effort by researchers, industry, and governments to develop new technologies and standards that can protect the privacy and security of IoT devices and the data they generate.

In this paper, we present VICTORY, our system for efficient and fast detection of IoT Malware. VICTORY consists of a fast feature extraction and decision engine and leverages domain-adapted IoT traffic features. In IoT, the availability of malware data to train on is a worrying issue. In order to solve this issue, we leverage prior work that uses pretrained BERT models for training a classifier with limited malware data. Using this, VICTORY is able to catch up fast as new malware emerges.

The security of IoT devices is a major concern because these devices are often connected to sensitive information and are used to control critical systems. For example, a connected device in a smart home may be used to control the locks, heating, and lighting, while a connected device in a car may be used to control the engine and navigation system. If these devices are not properly secured, they can be vulnerable to hacking, malware, and other security threats.

One of the main challenges in securing IoT devices is the diversity of devices and the use cases for these devices. There are many different types of IoT devices, ranging from small wearable devices to large industrial systems. Each device has its own unique security requirements, and the security measures that work for one device may not work for another. Additionally, the way in which these devices are used can greatly impact their security. For example, a device that is used in a hospital environment may have different security requirements than a device that is used in a home environment.

Another challenge in securing IoT devices is the lack of standardization. Many IoT devices are developed by different manufacturers and use different operating systems, communication protocols, and security measures. This lack of standardization makes it difficult to implement consistent security measures across all devices and increases the risk of vulnerabilities.
To secure IoT devices, it is important to follow best practices for device security. This includes ensuring that devices are updated with the latest software and security patches, using strong passwords and authentication measures, and securing communications between devices. It is also important to educate users about the security risks associated with IoT devices and how to use them securely.

In addition, there are a number of security measures that can be implemented at the network level to secure IoT devices. This includes using firewalls, intrusion detection and prevention systems, and virtual private networks (VPNs). These measures can help to prevent unauthorized access to IoT devices and to protect sensitive information that is transmitted between devices.

Finally, it is important to keep in mind that security is an ongoing process. As new threats emerge and new technologies are developed, the security measures that are in place today may become outdated. It is therefore important to regularly review and update security measures to ensure that they remain effective.

In conclusion, the security of IoT devices is a critical issue that must be addressed to ensure the safe and secure use of these devices. By following best practices for device security, implementing network-level security measures, and staying informed about the latest security threats and technologies, organizations can ensure the security of their IoT devices and protect sensitive information.

LITERATURE REVIEW

Traditionally, most techniques to detect IoT malware have been opcode based [2]. Opcode-based detection is a technique used to detect malware in the Internet of Things (IoT) by analyzing the machine code of the malware. This technique is based on the idea that malware has a unique set of instructions, known as opcodes, that can be used to identify it.

Opcode-based detection works by analyzing the machine code of the malware, which is a series of instructions that are executed by the computer's central processing unit (CPU). Each instruction is represented by a unique opcode, and these opcodes can be used to identify the behavior of the malware.

The process of opcode-based detection involves extracting the opcodes from the malware [3] and comparing them to a database of known malware opcodes. If a match is found, the malware is detected and can be neutralized. This technique can be used to detect malware on IoT devices, even if the malware is encrypted or has been modified to evade other forms of detection.

Opcode-based detection is effective because it analyzes the behavior of the malware at the machine code level, which is difficult for malware to modify or hide. Additionally, opcode-based detection can be used to detect zero-day threats, which are threats that are unknown to the security community.

However, opcode-based detection is not a perfect solution, and there are some limitations to this technique. For example, opcode-based detection can be time-consuming and resource-intensive [4], as it requires a detailed analysis of the machine code of the malware. Additionally, opcode-based detection may not be effective against polymorphic malware, which is malware that changes its opcodes in order to evade detection.

Second, some approaches use file headers in order to detect malware. ELF (Executable and Linkable Format) file header contains information about the executable file, including its type, architecture, and entry point. This information can be used for IoT malware detection by analyzing the ELF header and comparing it to known, benign files on the system [6]. If the header of an executable file does not match the expected values for a benign file, it may indicate that the file is malware. Additionally, the presence of certain flags in the ELF header, such as "Dynamic" or "PIE," can also suggest that the file is malicious [5]. However, it is important to note that relying solely on ELF header analysis is not a foolproof method of detecting malware, as attackers can manipulate or obfuscate the header to evade detection.

Finally, more recent approaches have employed machine learning and deep learning to detect malware. Machine learning and deep learning can be used to detect malware by training a model to recognize patterns in malicious and benign software. Here are a few ways this can be achieved:

1. Binary classification: The model is trained to distinguish between malicious and benign software by learning to recognize characteristic patterns in the binary code [13] of each type of file.

2. Anomaly detection: The model is trained on a large dataset of benign software to learn what is normal behavior. Then, when it encounters a new file, it uses this knowledge to identify any deviations [11] that may indicate the presence of malware.
3. Behaviour analysis: The model is trained to analyze the behavior of a file [10] during its execution, including system calls and network traffic, to identify any unusual or malicious behavior.
4. File signature analysis: The model is trained to identify characteristic patterns in the file header or other file metadata that can indicate the presence of malware.

In all of these cases, the model is trained on a large dataset of both malicious and benign software [12], and its accuracy can be improved over time as new data is added to the training set. However, it is important to note that machine learning and deep learning models are not foolproof, and attackers may be able to evade detection by developing new forms of malware that do not match the patterns the model has been trained on.

SYSTEM DESIGN

Feature Extractor

This module analyzes the source malware file and extracts a series of features from it. Features can be extracted from IoT malware for the purpose of detection and classification. We extract features belonging to a wide variety of domains and categories, as listed below

1. **File metadata**: Information about the file itself, such as its size, creation date, and file format.
2. **Binary code analysis**: The binary code of the file can be analyzed to extract features such as instruction sequences, function calls, and system calls.
3. **Network behavior**: The behavior of the malware when it communicates with other systems on a network, such as the addresses it contacts, the data it transfers, and the protocols it uses.
4. **System behavior**: The behavior of the malware when it is executing on a system , such as the processes it creates, the files it modifies, and the registry keys it accesses.
5. **API calls**: The API calls made by the malware, such as system calls, library calls, and function calls.

Feature Processor

Data preprocessing is an essential step in the process of building machine learning and deep learning models. It plays a crucial role in improving the quality and reliability of the data used for training and helps the model better fit the data. The feature processing module performs the following key tasks:

- Data Cleaning: Raw data often contains missing values, incorrect values, and irrelevant information that needs to be removed before it can be used to train a model. This step is crucial because having incorrect or missing data can negatively impact the performance of the model. By removing these issues, the model can be trained on cleaner and more accurate data, which can lead to better results.

- Feature Scaling: Machine learning and deep learning algorithms work best when the features have similar scales. This means that preprocessing includes transforming the features so they have similar ranges, which helps the model converge more quickly and improves its accuracy. By scaling the features, the model can more effectively learn the relationships between the features and the target variable, leading to better predictions.
- Data Reduction: This can include filtering or smoothing, which helps the model better handle irrelevant information and improves its performance. By reducing the noise in the data, the model can focus on the important patterns in the data and make more accurate predictions.
- Dimensionality Reduction: High-dimensional data can be difficult for machine learning models to handle, so preprocessing often includes techniques to reduce the dimensionality of the data, such as principal component analysis or feature selection. By reducing the number of dimensions in the data, the model can more effectively learn the relationships between the features and the target variable and make more accurate predictions.

BERT Block

A key challenge in detecting IoT malware is the lack of available training data. Malware samples are hard to come by, and it takes a significant amount of time before enough data is accumulated for training. Training a classifier with a small number of malware is hard. Oak et. al [9] addresses this issue by using a transformer pre-trained on language data. This work has shown significant advantages in prior works and is now the de-facto standard for low-positive malware detection. Directly following the approach in [8], we use the BERT model trained on Wikipedia and fine-tune it on API call sequences. As specified in the algorithm, we use the embeddings generated by the first API call and pass it to a linear layer for classification. Adapting the algorithm from Oak et al. [9] allows us to develop a fast-evolving classifier, as retraining can be done with extremely few samples.

Classifier Block

The classifier block is responsible for using the cooked features from the feature extraction block along with the embeddings from the BERT block to classify the file as malware or benign. We use a multi layer perceptron (MLP) for classification. A Multi Layer Perceptron (MLP) is a type of artificial neural network that is commonly used in supervised learning tasks. It is called a multi-layer perceptron because it consists of multiple layers of artificial neurons, with each layer connected to the next. The first layer is the input layer, which receives the input data, and the last layer is the output layer, which produces the predictions. The intermediate layers are called hidden layers. Each artificial neuron in an MLP receives input from the previous layer, processes the input using an activation function, and then passes the result to the next layer. The activation function determines the output of a neuron based on its inputs and a set of weights that are learned during training. The weights are adjusted during training so that the model can learn to make accurate predictions based on the input data.

MLPs are versatile and can be used to solve a wide variety of problems, such as image classification, speech recognition, and natural language processing. They are also relatively simple to implement and can be trained using standard optimization algorithms, such as gradient descent. Overall, MLPs are an important building block in the field of deep learning and have been used in many successful applications. While more advanced models have been developed in recent years, MLPs continue to be widely used due to their simplicity, flexibility, and effectiveness.

EVALUATION

Dataset

We evaluate VICTORY on the Android Malware Genome dataset. The Android Malware Genome Project dataset is a collection of representative Android malware samples and associated metadata. The goal of the project is to support research in the field of mobile malware by providing a comprehensive and structured dataset of Android malware. The dataset contains various attributes such as: file names, file sizes, package names, requested permissions, and API calls made by each sample. This information can be used by researchers to study the behavior and evolution of Android malware, as well as to develop new methods for detecting and mitigating such threats.

Train-Test Split

The train-test split is a common technique in machine learning used to evaluate the performance of a model. It involves dividing a dataset into two parts: the training set and the test set. The training set is used to train the model and tune its parameters. The model is fit on the training data, and the algorithm learns to make predictions based on the patterns in the training data. The test set is used to evaluate the performance of the model. Once the model has been trained, it is applied to the test set, and its predictions are compared to the actual outcomes. This provides an estimate of how well the model is likely to perform on unseen data. The train-test split is important because it helps to avoid overfitting, which is a problem that occurs when a model is too closely fitted to the training data and does not generalize well to new, unseen data. The split helps ensure that the model has seen a diverse range of examples and has learned general patterns that can be applied to new data.

Baselines

To evaluate our method, we use three popular models for IoT malware detection: Ren et al. [7], Xiao et al. [8] and Karbab et al. [10]. These techniques have shown good performance in IoT malware detection in the past. Compared to these, VICTORY has an enhanced feature set and the adaptation of the BERT-block from [9]. We compare the performance of these models using accuracy and F-1 score under varying malware sample rates.

Results

The comparative results are shown below in Table 1.

Malware Rate	10%		1%		0.1%		0.01%	
Model	Acc	F-1	Acc	F-1	Acc	F-1	Acc	F-1
Ren et al. [7]	98.82	94.03	94.61	92.00	84.12	80.09	72.11	66.65
Xiao et al. [8]	97.55	89.13	89.00	80.43	72.80	64.31	58.71	52.07
Karbab et al. [10]	99.19	98.27	84.87	81.1	69.06	60.22	42.06	8.21
VICTORY (ours)	99.28	98.47	98.96	98.27	98.74	98.03	98.11	97.82

Table 1: Accuracy and F-1 score of classification models with varying malware rates. Values shown are percentages.

The results clearly indicate two important findings:

- Overall, VICTORY has better performance (in terms of accuracy and F-1 score than all of the baseline models.
- While all the four models have comparable performance at the 10% malware rate, VICTORY is the only one which is able to sustain the performance at lower malware rates.

Discussion

Better Performance of Victory. The feature set we use contains a variety of features, which complement and improve upon prior work. The effect of our superior feature extraction module is clearly seen in the fact that VICTORY has a higher accuracy and F-1 score in detecting malware samples. In order to validate this hypothesis, we repeat the analysis but using features from each of the prior works individually. In each case, we observe that the accuracy and F1 score are consistently lower than what we obtain with our enhanced feature set.

Sustained Performance of VICTORY. We also note above that VICTORY is able to sustain the same performance even when the malware rate is reduced drastically. According to us, the sustained performance is due to implementation of the BERT block and fine-tuning algorithm from Oak et al. [9]. To verify this, we evaluated the system with our enhanced feature set but omitted the BERT block this time. While at the 10% rate we still saw a high accuracy and F-1 score, the performance rapidly degraded with reduced malware rate. Thus, the BERT block and algorithm from [9] is able to provide us sustained performance with a drop of less than 1.5% in accuracy even when the malware rate is reduced by a thousand times (1% vs 0.01%).

CONCLUSION

In this paper, we presented VICTORY, a novel framework for detecting malware in IoT devices. The VICTORY framework provides a highly efficient and effective solution for detecting IoT malware. By leveraging machine learning algorithms and big data analysis techniques, the framework is capable of rapidly identifying malware threats in real-time. The results of our experiments demonstrate the superior performance of the VICTORY framework compared to traditional detection methods, in terms of accuracy and speed. Furthermore, the framework is flexible and can be easily integrated into existing security systems. Our enhanced feature set shows superior performance over prior models and adaptation of a transformer-based block allows fast updates and training at lower positive rates. Overall, the VICTORY framework is a valuable contribution to the field of IoT security and has the potential to greatly enhance the protection of IoT devices against malicious attacks.

REFERENCES

1. Allix K, Jérome Q, Bissyandé TF, Klein J, State R, Le Traon Y. A Forensic Analysis of Android Malware--How is Malware Written and How it Could Be Detected?. In2014 IEEE 38th Annual Computer Software and Applications Conference 2014 Jul 21 (pp. 384-393). IEEE.
2. HaddadPajouh H, Dehghantanha A, Khayami R, Choo KK. A deep recurrent neural network based approach for internet of things malware threat hunting. Future Generation Computer Systems. 2018 Aug 1;85:88-96.
3. Dovom EM, Azmoodeh A, Dehghantanha A, Newton DE, Parizi RM, Karimipour H. Fuzzy pattern tree for edge malware detection and categorization in IoT. Journal of Systems Architecture. 2019 Aug 1;97:1-7.
4. Darabian H, Dehghantanha A, Hashemi S, Homayoun S, Choo KK. An opcode-based technique for polymorphic Internet of Things malware detection. Concurrency and Computation: Practice and Experience. 2020 Mar 25;32(6):e5173.
5. Shahzad F, Farooq M. Elf-miner: Using structural knowledge and data mining methods to detect new (linux) malicious executables. Knowledge and information systems. 2012 Mar;30:589-612.
6. Bai J, Yang Y, Mu S, Ma Y. Malware detection through mining symbol table of Linux executables. Information Technology Journal. 2013 Jan 15;12(2):380.
7. Ren Z, Wu H, Ning Q, Hussain I, Chen B. End-to-end malware detection for android IoT devices using deep learning. Ad Hoc Networks. 2020 Apr 15;101:102098.

8. Xiao F, Lin Z, Sun Y, Ma Y. Malware detection based on deep learning of behavior graphs. Mathematical Problems in Engineering. 2019 Feb 11;2019:1-0.
9. Oak R, Du M, Yan D, Takawale H, Amit I. Malware detection on highly imbalanced data through sequence modeling. InProceedings of the 12th ACM Workshop on artificial intelligence and security 2019 Nov 11 (pp. 37-48).
10. Karbab EB, Debbabi M, Derhab A, Mouheb D. MalDozer: Automatic framework for android malware detection using deep learning. Digital Investigation. 2018 Mar 1;24:S48-59.
11. Amin M, Shehwar D, Ullah A, Guarda T, Tanveer TA, Anwar S. A deep learning system for health care IoT and smartphone malware detection. Neural Computing and Applications. 2020 Nov 6:1-2.
12. Kumar R, Zhang X, Wang W, Khan RU, Kumar J, Sharif A. A multimodal malware detection technique for Android IoT devices using various features. IEEE access. 2019 May 23;7:64411-30.
13. Kumar A, Lim TJ. EDIMA: Early detection of IoT malware network activity using machine learning techniques. In2019 IEEE 5th World Forum on Internet of Things (WF-IoT) 2019 Apr 15 (pp. 289-294). IEEE.