

Real time malware detection in encrypted network traffic using machine learning with time based features

Abhay Pratap Singh *

Mahendra Singh [†]

Department of Computer Science

Gurukula Kangri (Deemed to be University)

Haridwar

Uttarakhand

India

Abstract

With the increasing amount of Internet users, malware attacks are also growing. The purpose of malicious authors creating malware is to attack, damage, or impair electronic devices. In recent times, malware authors are also using HTTPs traffic; therefore, detecting malware in HTTPs traffic is intriguing since network traffic is enciphered. As the network traffic is enciphered, it is an arduous job to identify benign and malicious traffic. It also poses a significant challenge for firewalls and anti-malware software. Hence, it is essential to monitor the network traffic for detecting malware and threats in this way that maintains the encryption integrity. In this paper, a machine learning based model was proposed, which can effectively and efficiently detect malware without deciphering the network traffic. The prime objective of the research work is to apply several of ML techniques to detect malware in real-time utilizing time-based features. The proposed methodology can classify malware attacks in less than one second, achieving an accuracy of 99% on the Central Processing Unit (CPU) and Graphics Processing Unit (GPU) platform, which is sufficient for detecting malware in real-time.

Subject Classification: 68M12, 68M25.

Keywords: Malware, Time based features, Machine learning, Network traffic, Real time detection.

* E-mail: rs.abhaypratapsingh@gkv.ac.in (Corresponding Author)

[†] E-mail: msa@gkv.ac.in

1. Introduction

Network security upholds ongoing concern due to the spread of a wide range of malicious and illegitimate activities through the Internet. The continuous expansions of cyber threats are posing a significant challenge for the network security research community, and the situation is constantly evolving. Currently, cyber-attackers are using various ways such as traffic redirection, click jacking, and encryption to conceal malicious activities and silently propagate malware into the networks [1]. The usage of these advanced techniques is extensively rising. In recent years, the use of encryption on the Internet has been growing very fast because most of the web applications are moving towards HTTPs protocol. The primary objective of HTTPs protocol is to encipher the network traffic between clients and servers; therefore, adversaries or hackers cannot inspect the transmitted packets. As per the report, more than 80% of the existing network traffic is enciphered through HTTPs protocol [2], and additionally, 37% of whole malware is using HTTPs protocol as its own communication channel [3]. Encryption is essential to protect the privacy of benign-users, the confidentiality, and integrity of communications, whereas cyber attackers used it as a weapon to secure their malicious activities and elude detection from antiviruses. The detection of malicious threats in HTTPs traffic is a difficult task as network traffic is enciphered; as a result, malicious and benign traffic looks like similar traffic. Traditional approaches such as DPI (Deep Packet Inspection) are ineffective to detect/identify threats in encrypted traffic. The conventional methods usually requires encrypted traffic to be decrypted and then inspected whether it comprises any malicious threats or not, enciphered again, and sent to the destination address. The key problem of this approach is that it infringes the security and privacy of the communication medium, which is the primary concern of HTTPs protocol. The prime objective of the proposed research is to use ML techniques to identify threats contained in the enciphered network traffic without decrypting it. Even though the network traffic payload is enciphered, there are statistical properties (flow features) in HTTPs based traffic that can provide enough information to identify the threats [4].

The primary contributions of proposed research can be stated as follows:

- a) Proposed a ML based model to detect malware in real time in HTTPs traffic without decrypting the network traffic.

- b) Selected the best machine learning algorithm (out of four classifiers) for detecting/identifying malware.
- c) Developed a lightweight detection method or model which is carefully adapted to achieve time and detection accuracy.
- d) Compared the computational time for malware detection on CPU and GPU platform.

This paper is structured as follows: The proposed model explored in section 2. Experimental results are discussed in section 3, and a comparative analysis of the proposed model on CPU and GPU platform is enumerated in section 4. Section 5 contains the conclusion of the paper.

2. Proposed Model

The goal of our research is to reveal malware in real time enciphered traffic utilizing time based features. The complete process of the proposed work has been explained in Figure 1. To accomplish this goal, we applied several supervised machine learning techniques. The proposed model consists of various phases; feature extraction, preprocessing, feature selection, feature importance, and final training and testing of the model. A detailed description of every phase of the proposed work is illustrated below.

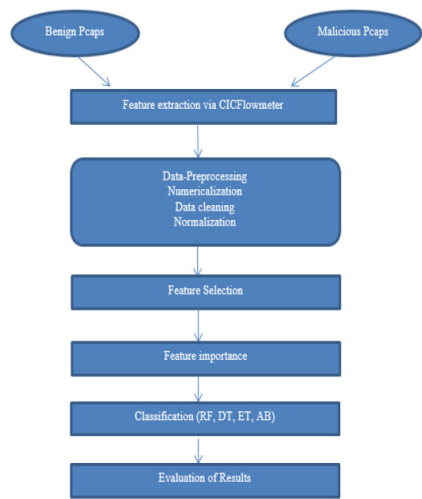


Figure 1
Process of the Proposed Model

2.1 Dataset Description

The Malware Capture Facility Project (MCFP) is a scientific project launched by the Czech Technical University ATG group to collect, analyze and produce real malicious network traffic [5]. The MCFP dataset is an amalgamation of several malicious and benign captures. It contains information about such as Pcap files, log files, and DNS information. Pcap files are used in our work because it is easy to extract relevant information (flow based features) without using the payload data.

2.2 Feature Extraction

The use of raw data in the machine learning method is problematic because it is unstructured and comprises redundant information. Therefore, it is necessary to construct a structured representation of the original data that is informative to the particular problem domain. Constructing this representation is known as feature extraction. In this step, we extract several features from benign and malicious Pcaps via the CICFlowmeter tool [6]. This tool can retrieve more than 80 statistical network flow features.

2.3 Preprocessing Phase

The preprocessing phase is an essential step to improve the efficiency and efficacy of the machine learning model. The primary objective of preprocessing is to reduce the irrelevant features and duplicate values of the dataset. Some instances may affect the detection accuracy negatively, such as packet retransmissions, duplicity acknowledgments, and in-order packets. Data cleaning and normalization were executed to enhance the performance of the model. The detailed description is as follows:

2.3.1 Numericalization

The dataset has various statistical features and one non-statistical feature. The input value in the machine learning based model should be in numeric form because, in machine learning algorithms, back-end calculations are performed using numeric values and not nominal values. Therefore, it is essential to convert the label value into a numeric vector. For example, in the label feature for binary-class classification, benign and malware are converted into 1 and 0, respectively.

2.3.2 Data Cleaning

Data cleaning is the method of eliminating or modifying incorrect or improperly formatted data. It also enhances the integrity of data. In proposed model, Pandas library was used for data cleaning.

2.3.3 Data Normalization

Data normalization is a mechanism which is usually practiced to statistical data, while the variety of raw data differs broadly. Standard score is a normalization technique which amends the data in this way that the novel feature vector contains a zero mean and unit SD [7]. We have used Z-score to normalize the dataset in the following way:

$$Z = \frac{(X - \mu)}{\sigma} \quad \dots(1)$$

Where Z indicates standard score, X is data, μ defines mean and σ defines SD.

2.4 Feature Selection

It is a significant procedure of determining the subclass of the important features from the dataset. Choosing relevant and appropriate

Table 1
Time related features [8, 9]

Feature	Description
Duration	The duration of the flow.
D.port	Destination port
Fiat	Forward Inter Arrival Time (mean, min, max, std).
Biat	Backward Inter Arrival Time (mean, min, max, std).
Flowiat	Flow Inter Arrival Time, (mean, min, max, std).
Active	The amount of time, the time a flow was active before going idle (mean, min, max, std).
Idle	The amount of time, the time a flow was idle before becoming active (mean, min, max, std).
fb_psec	Flow bytes per second.
fp_psec	Flow packets per second.

features for the machine learning model is a challenging task. To provide an independent, lightweight, and real time detection model, we need to select the time based features to speed up the efficiency and to pursue an encryption independent network traffic classifier[8][9]. Time based features contain amount of time flow, packet length, inter arrival time, and the direction of consecutive packets [10]. With time based features, the first few packets are sufficient enough for reasonable accuracy for enciphered traffic classification [11]. The time based features are selected out of more than 80 statistical network flow features illustrated in Table 1.

2.5 Feature Importance

Feature Importance is a method to select features in the dataset that provides the most important contribution in predicting the target variable. Using the selected features instead of all features, can mitigate the risk of over-fitting, enhances accuracy, and decreases training time. We used the Pycaret library to retrieve the most relevant features among the time-based features. It utilizes an amalgamation of various supervised feature selection techniques to select the appropriate features most useful for modeling. Figure 2 depicts the feature importance plot.

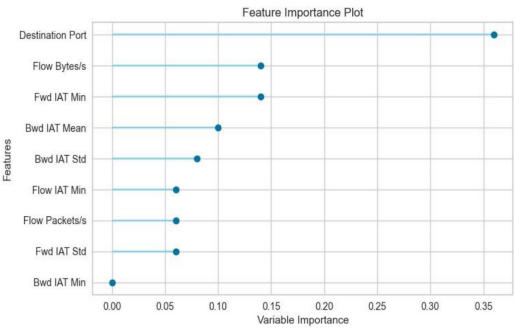


Figure 2
Feature Importance Plot

3. Experimental Result and Analysis

To implement our experiment, we used the Pycaret machine learning library of Python that helps build up a machine learning model. In the total network flow samples, 70% of the data is utilized as training data, and 30% of the data is utilized as testing data. In addition, to evaluate the performance of the ML model to mitigate the risk of over-fitting and

Table 2
Comparison of Different ML Algorithms on CPU Platform

Model	Accuracy	Precision	Recall	F1-score	Model building time (sec)	Detection time(sec)
RF Classifier	0.9963	0.9970	0.9951	0.9960	8.1650	1.35
DT Classifier	0.9955	0.9961	0.9942	0.9951	5.2139	1.19
ET Classifier	0.9965	0.9970	0.9956	0.9962	52.3192	1.56
Ada Classifier	0.9733	0.9864	0.9562	0.9710	48.4804	2.96

to acquire a generalized result, we utilized a 10-fold cross validation approach. We have applied four machine learning classifiers (RF, DT, ET, and Ada) using the CPU platform, and the results have been explored using the confusion matrix as shown in Figure 3, Figure 4, Figure 5, and Figure 6. The four models were analyzed with regard to accuracy, precision, recall, F1-score, model-building time (sec), and detection time (sec), as depicted in Table 2.

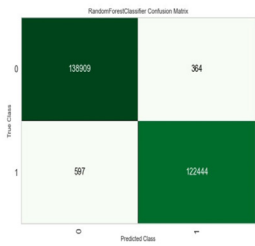


Figure 3
RF Confusion Matrix

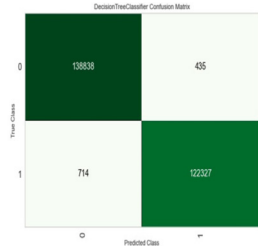


Figure 4
DT Confusion Matrix

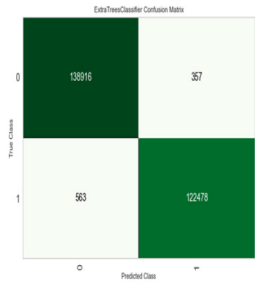


Figure 5
ET Confusion Matrix

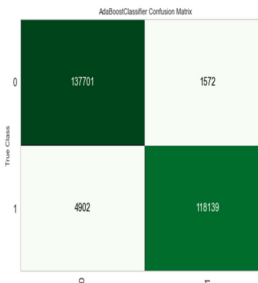


Figure 6
Ada Confusion Matrix

4. Comparative Analysis of Models on CPU and GPU Platform

We have measured the performance metrics for the four classifiers on the CPU platform, as shown in Table 2. Similar experiments were carried out on the GPU platform using Google Colab, which is a cloud based service provided by Goggle. These machine learning models were also evaluated on the accuracy, precision, recall, F1-score, and model building time (sec), as illustrated in Table 3.

Table 3
Comparison of Different ML Algorithms on CPU Platform

Model	Accuracy	Precision	Recall	F1-score	Model building time (sec)	Detection time(sec)
RF Classifier	0.9964	0.9970	0.9953	0.9961	7.1173	0.642
DT Classifier	0.9956	0.9963	0.9944	0.9953	3.6180	0.416
ET Classifier	0.9966	0.9972	0.9956	0.9963	36.2292	1.423
Ada Classifier	0.9729	0.9854	0.9563	0.9706	28.4158	0.981

Table 2 and Table 3 indicate that with ET classifier, we achieve better accuracy, precision, recall, and F1-score compared to other classifiers but has a high detection time. However, RF and DT classifiers also produced nearly equally good results with reduced detection time. The least accurate is the Ada classifier. The key goal of the research work is to achieve high accuracy for malware detection without decrypting the network traffic with an optimal detection time. It can be observed that performance metrics are improved when models are evaluated on the GPU platform. The detection time of the models are also reduced comparatively on the GPU, as illustrated in Figure 7.

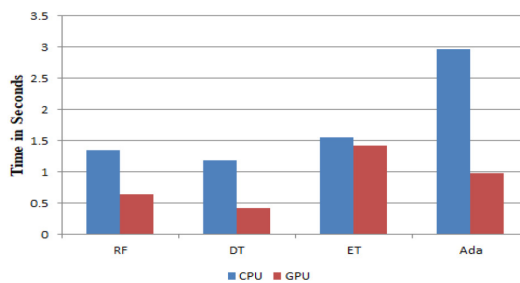


Figure 7

Computational Time Comparison between CPU and GPU

5. Conclusion

Detection of malware in real time has assumed high significance considering the new threats to the cyberspace security. The empirical results showed that our proposed methodology is very robust to enciphered traffic. It utilized a selection of time-related features and four machine learning algorithms. Experimental results prove that the DT classifier is most effective among all classifiers with regard to accuracy and detection time. DT based model produced the 99.55 % accuracy with detection time of 1.19 sec on the CPU platform and 99.56% accuracy with detection time of 0.416 sec on the GPU platform.

References

- [1] Yask and B. S. Kumar, "Construct the Filter for Malware Detection in Distributed Control System," *Journal of Information and Optimization Sciences*, Vol. 40, No. 5, pp. 1155-1166 (2019).
- [2] Let's encrypt stats. URL <https://letsencrypt.org/stats/#percent-pageloads>, Last Accessed, 2021.11.21.
- [3] Arna Magnúsardóttir. Malware is Moving Heavily to Https. <https://www.cyren.com/blog/articles/over-one-third-of-malware-uses-https>, Last Accessed, 2018.10.19.
- [4] Kohut, Jan, Tomáš Komárek, Přemysl Čech, Jan Bodnár, and Jakub Lokoč. "Learning Communication Patterns for Malware Discovery in HTTPs Data", *Expert Systems with Applications*, Vol.101, pp. 129-142 (2018).
- [5] M. J. Erquiaga and S. Garcia, "Malware Capture Facility Project," <https://mcfp.weebly.com>, CVUT University, 2013
- [6] CICFlowmeter, <https://www.unb.ca/cic/research/applications.html> Last Accessed 28.07.2021.
- [7] D.B. Bisandu, R. Prasad, and M.M. Liman, "Data Clustering using Efficient Similarity Measures," *Journal of Statistics and Management Systems*, Vol. 22, No. 5, pp. 901-922 (2019).
- [8] Draper-Gil, Gerard, Arash Habibi Lashkari, Mohammad Saiful Islam Mamun, and Ali A. Ghorbani. "Characterization of Encrypted and VPN Traffic Using Time-Related", In *Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP)*, pp. 407-414 (2016).

- [9] Bagui, Sikha, Xingang Fang, Ezhil Kalaimannan, Subhash C. Bagui, and Joseph Sheehan. "Comparison of Machine-Learning Algorithms for Classification of VPN Network Traffic Flow Using Time-Related Features", *Journal of Cyber Security Technology*, Vol. 1, No. 2, pp. 108-126 (2017).
- [10] Rezaei, Shahbaz, and Xin Liu. "Deep Learning for Encrypted Traffic Classification: An overview", *IEEE Communications Magazine*, Vol. 57, No. 5, pp. 76-81 (2019).
- [11] Lopez-Martin, Manuel, Belen Carro, Antonio Sanchez-Esguevillas, and Jaime Lloret. "Network Traffic Classifier with Convolutional and Recurrent Neural Networks for Internet of Things", *IEEE Access*, Vol. 5, pp. 18042-18050 (2017).