An Unsupervised Learning Based Encrypted Mobile Traffic Cleaning Framework

Kun Qiu, Ying Wang, Baoqian Li
Intel Asia-Pacific Research & Development Ltd, Shanghai, China
{kun.qiu, ying.a.wang, baoqian.li}@intel.com

***Abstract*—Traffic classification, a technique that aims at classifying network traffic into different classes, has been widely deployed in the enterprise and carrier network. With the massive adoption of mobile devices, the encrypted protocol is applied in mobile applications to prevent the raising of privacy issues. However, traditional traffic classification methods such as Deep Packet Inspection (DPI) cannot distinguish encrypted traffic. To overcome this problem, Artificial Intelligence (AI), especially Machine Learning (ML) has emerged as a satisfactory solution for encrypted traffic classification. Traffic data cleaning is the most important step, which removes traffic that does not need to be trained (useless protocol, background, control plane traffic such as long live traffic, etc.). However, the existing solution that is manually checking every packet collected, is too expensive in cleaning encrypted traffic. In this paper, we design a novel framework that can automatically clean the encrypted traffic data. The evaluation results show the effectiveness of our framework with only $2\% \sim 2.5\%$ accuracy loss comparing to the manual solution. Moreover, we also optimize the implementation that can achieve 10-times speedups over the non-optimized implementation.**

***Index Terms*—Traffic Cleaning, Unsupervised Learning**

I. Introduction

Traffic classification, a technique that aims at classifying network traffic into different classes, such as normal or abnormal traffic, or the name of the application (e.g., YouTube, Skype or Netflix), has been widely deployed in the enterprise and carrier network [1], [2]. Usually, traffic can be classified by (1) IANA defined Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) port numbers, or (2) utilizing Deep Packet Inspection (DPI) tools within TCP or UDP payload for seeking specific protocol/application signatures. However, with the massive adoption of mobile devices in recent years, encrypted protocol (e.g., TLS) has been widely utilized in mobile applications to prevent the raising of privacy issues [3]. Thus, the traditional traffic classification approach is inaccurate if applications use encrypted traffic.

To overcome these issues, Artificial Intelligence (AI), especially Machine Learning (ML) has emerged as a satisfactory solution for encrypted traffic classification. The general steps to train an ML model in traffic classification are (1) traffic data collection; (2) traffic data cleaning; (3) classification model training [4]. The quality of a trained classification model depends on the cleanliness of the data collected. Briefly speaking, the traffic data cleaning process should remove traffic that does not need to be trained, such as useless protocol traffic (e.g., TCP traffic with only handshake packet), background service traffic (e.g., Android/iOS service) or even control plane traffic of an application (e.g., keep alive traffic).

One of the factors that affect the complexity of traffic cleaning is the traffic collection method. Appropriate traffic collection methods must introduce as little unnecessary traffic as possible. However, obtaining traffic from mobile applications is much more challenging than obtaining traffic from traditional PC applications (e.g., YouTube via a web browser, Skype or the famous game World of Warcraft). One reason is that collecting PC traffic can directly utilize traffic/packet sniffing tools like Wireshark or Fiddler on the application process [5], but collecting mobile traffic can only obtain traffic from the mobile device by using a modified router or edge device. Comparing to Wireshark or Fiddler, there are numerous traffic that should not be trained (e.g., background Android/iOS service traffic) are collected via router or edge device. Thus, cleaning the traffic data of mobile applications has been a serious problem.

Unfortunately, most of the existing cleaning works focus on unencrypted traffic classification [6]. None of them mentioned how to clean encrypted traffic after traffic collection from the mobile device. A naiv̈e solution is to manually check every packet offline. However, it may cost several days to prepare cleaned traffic for classification model training, which is difficult to be deployed in enterprise and carrier network.

In this paper, we design a novel framework to clean the encrypted mobile traffic data. We propose an online structure that utilizes an unsupervised learning based algorithm to extract clean data from the collected traffic data. We also apply the high-performance instruction such as vector optimization to fully increasing its computing efficiency. The evaluation results show the effectiveness of our framework with only $2\% \sim 2.5\%$ accuracy loss comparing to the manual cleaning solution. Moreover, our optimized implementation can achieve 10-times speedups over the non-optimized implementation.

The remainder of the paper is organized as follows. Section II gives the background and related work, Section III gives an overview of our design, Section IV shows the detailed design. Our evaluation results are showed in Section V, and we conclude this paper in Section VI.

II. Background and Related Work

As we have mentioned before, most of the ML-based traffic classification methods need to follow the generic steps: 1) traf-

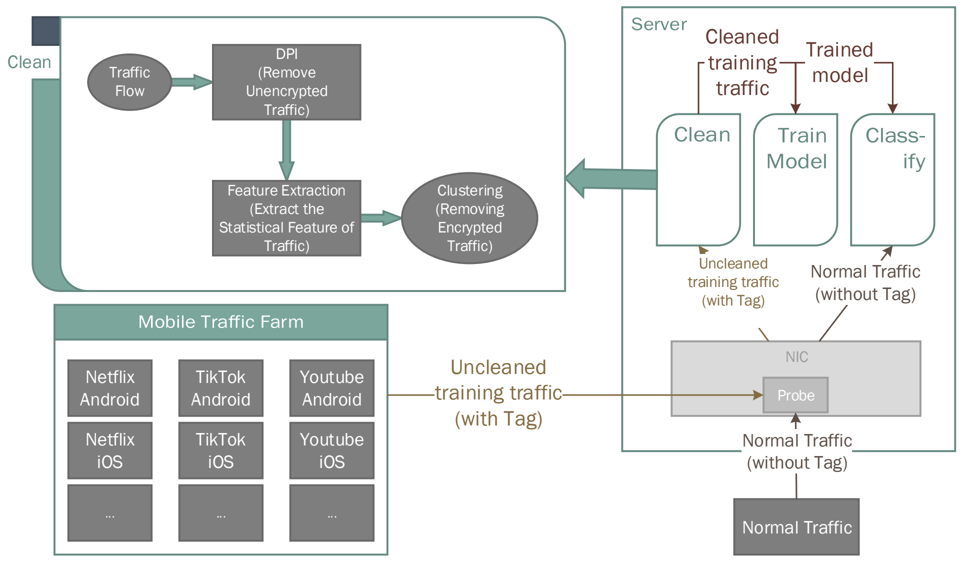
Clean
Traffic Flow
DPI (Remove Unencrypted Traffic)
Feature Extraction (Extract the Statistical Feature of Traffic)
Clustering (Removing Encrypted Traffic)
Server
Cleaned training traffic
Trained model
Clean
Train Model
Class-ify
Uncleaned training traffic (with Tag)
Normal Traffic (without Tag)
NIC
Probe
Mobile Traffic Farm
Netflix Android
TikTok Android
Youtube Android
Netflix iOS
TikTok iOS
Youtube iOS
...
Uncleaned training traffic (with Tag)
Normal Traffic (without Tag)
Normal Traffic

Fig. 1. The overview of our design. The mobile traffic farm is utilized to generate traffic for every application which is needed for training. We deploy the application in the real smartphone or simulator in the traffic farm. We deploy our clean module as a clean Virtual Network Function (VNF) in a server or even in an Edge Box. A tag, such as MAC address or VLAN ID can be used to distinguish different application traffic. Traffic without a tag will not be directed into the clean and train module. The detailed process of the clean module is on the left side of the figure. We can see that there are 3 steps to remove the unnecessary traffic: 1) remove unencrypted traffic; 2) extract traffic features and 3) remove encrypted traffic.

fic data collection; 2) traffic data cleaning and 3) classification model training:

1) **Traffic data collection:** This step aims at obtaining the raw packets and extracting traffic data from raw packets. Raw data can be imported from an offline packet trace [7]–[9], or captured from online packet sniffing software or devices [10], [11]. In order to extract traffic from raw packets, feature extraction (FE), which is one of the most important processes, is proposed to extract statistical features of every traffic [2], [12], [13]. Statistical features can better describe traffic then raw packets. Nearly all existing works perform feature extraction to get statistical features before they train classification model. Most works using Joy [14], an Open Source feature extraction library, to extract statistical features for further training.
2) **Traffic data cleaning:** Several works perform traffic data cleaning in their pre-processing tasks [1], [6]. An Android malware detection framework [4] performs traffic data cleaning to smooth the noisy traffic and resolve inconsistencies in the data. A more detailed work [15] shows that traffic cleaning should detect errors, fix functional dependency and missing/incomplete data. However, none of them mentioned how to perform encrypted traffic cleaning.
3) **Classification model training:** The classification model mainly falls into two classes: supervised and unsupervised model. Naiv̈e Bayes [16], Support Vector Machine (SVM) [17] and Decision Tree [5] have been widely used as traffic classification. Comparing to the supervised model, the unsupervised method does not need label before training. K-means [18] is usually utilized to separate traffic into different QoS classes (e.g., VoIP, browsing, gaming, etc.).

III. Overview

In this section, we will give an overview of our design. Fig. 1 shows the structure of our framework. It is mainly composed by a mobile traffic farm and a server. We will describe them in detail.

A. Mobile traffic farm

We use a mobile traffic farm to generate traffic for every application which is needed for training. Traffic is generated by the mobile application that is deployed in real smartphone and simulator (e.g., Bluestack [19]). Since it is reported that a same application on different mobile platform (e.g, iOS/Android) may lead to significant differences of traffic features [3], we deploy the same application with Android/iOS separately. We use a tag, such as MAC address or VLAN ID to distinguish different application traffic. However, the tagged traffic contains not only the application traffic, but also the background and control plane traffic generated by mobile OS, we call it *uncleaned training traffic*. The *uncleaned training traffic* will be directly sent to the server side for cleaning and training.

B. Server

In order to deploy our framework into existing network, we can pack our clean module as a clean Virtual Network Function (VNF) in a server or even in an Edge Box such as Intel uCPE. When the *uncleaned training traffic* arrived at NIC, the probe module will direct tagged traffic into our clean module. After the traffic is cleaned, the *cleaned training* *traffic* will be redirected into the training service to train a classification model, then the model will be deployed into the classifier.

IV. Design of Clean Module

In this section, we describe the clean module in detail. The left side of Fig. 1 shows there are 3 steps to clean the traffic. First, statistical features are extracted from the packet flow by feature extraction (FE). Next, unnecessary traffic such as DNS query or Android/iOS service can be removed by the DPI tool. At last, encrypted traffic can be cleaned by unsupervised learning. We use the clustering algorithm to cluster traffic into several traffic clusters, and remove traffic in clusters whose traffic features do not fit our requirement (e.g., heartbeat traffic/upload traffic/etc.). Finally, we can get cleaned traffic after these steps.

A. Removing unencrypted traffic and extracting statistical features

As we have mentioned before, FE is the most important step since traffic features are required by the following clustering and classifying algorithm. We can extract statistical features (e.g., *Bytes In/Out*, *Header/Payload* mean size and the duration of traffic) that reflect traffic properties by utilizing the feature extracting library. The result of the FE is a structured table, while each column indicates a feature and each row is a feature observation of traffic.

At the same time, we use the DPI tool to quickly detect unencrypted traffic that we do not need such as Google/Apple service or Cloudflare for training. At last, we get the pre-cleaned traffic flow with the statistical feature by combining the result from the last two steps.

B. Utilizing the clustering algorithm to remove encrypted traffic

After the pre-cleaned traffic is obtained, we can clean encrypted traffic by unsupervised learning algorithms. Most of the unsupervised learning algorithms utilize the clustering algorithm. The clustering algorithm is to group a set of objects into several sets or called clusters. Objects in the same cluster are more similar to each other than objects in other clusters. We use a clustering algorithm to cluster pre-cleaned traffic into several clusters. Both research works [5] and our evaluation results show that $3 \sim 4$ clusters are good enough to clean useless traffic. The following traffic features are used to cluster traffic: *BytesIn, BytesOut, PacketsIn, PacketsOut, Duration, Ratio*. *BytesIn* and *BytesOut* indicate the sum of bytes the traffic received and sent. *PacketsIn* and *PacketsOut* indicate the count of packets the traffic received and sent. Duration indicates the time of traffic sustains. *Ratio* is a value ranging from -1 to 1 and is computed by $\frac{BytesIn - BytesOut}{BytesIn + BytesOut}$. If the ratio is closer to 1, it indicates the traffic is similar to download traffic and vice versa. A clustering example is given in Fig. 2 where only two features *BytesIn, BytesOut* are considered. The clustering algorithms used here are k-means and hierarchical clustering. The former has better performance while the latter has higher accuracy.

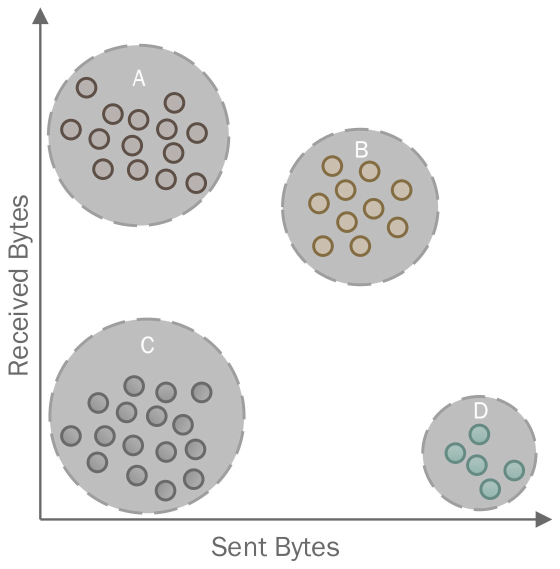

Fig. 2. An example of utilizing the clustering algorithm in cleaning encrypted traffic. In this figure, we only use two features: *Sent Bytes* and *Received Bytes* as an example. After the clustering algorithm is performed, we have 4 clusters (A, B, C, D). Traffic in cluster D is significantly different from traffic in cluster A. Most traffic in cluster D is upload traffic, while traffic in cluster A is download traffic.

C. Using cleaned traffic to train classification model

After the clustering, we can get several traffic clusters. We can choose the clustered class we need as training data to train the classifier for classification. As an example shown in Fig. 3, if we want to choose the data plane traffic of a video-stream application such as Netflix or YouTube, we can find the cluster whose traffic ratio is closer to 1 (larger than 0.9 in most time), and remove other traffic. Another example is extracting some control plane traffic, such as long-live heart-beat traffic. We can remove the traffic in the cluster whose traffic duration is not significantly longer than $1s$ with a minimum $BytesOut$.

V. Evaluation

A. Environment

Before we propose the evaluation results, we firstly give the detail of our evaluation environment. Our server uses a Xeon Gold 6148 @ 2.40Ghz with 196GB DDR4 RAM. The operating system is Linux 3.10.0, with Python 3.6 and GCC 8.4.0. We first implement our framework on *scikit-learn* [20], which is an ML library based on Python. We use *nDPI* [21] as a DPI tool. As mentioned above, we convert our implementation to C++ with the help of *Intel Data Analytics Acceleration Library(DAAL)* [22], a high-performance ML library with vector optimization (e.g., AVX512), to further increase the performance of our framework.

B. Dataset

We give the detailed information of our dataset in TABLE I. The traffic data is collected by the mobile traffic farm. We deploy the iOS version in several iPhone 6 smartphones

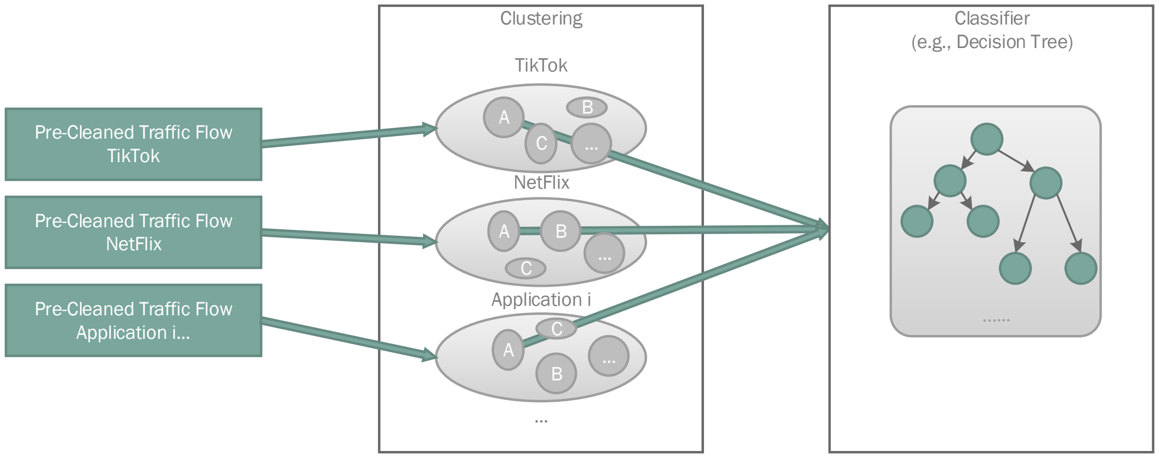
Pre-Cleaned Traffic Flow TikTok
Pre-Cleaned Traffic Flow NetFlix
Pre-Cleaned Traffic Flow Application i...
Clustering
TikTok
NetFlix
Application i
A
B
C
...
Classifier (e.g., Decision Tree)
......

Fig. 3. Utilizing the clustering algorithm to separate unnecessary traffic without any trained data from all traffic flows. The uppercase letters ('A', 'B', 'C') indicate clustered classes after the clustering algorithm is performed. We can choose one of the clustered classes as training data to train the classifier for traffic classification.

and deploy the Android version in a server with several Bluestack [19] Android simulators. The dataset of each application contains mixture iOS/Android traffics. Since all of these applications are video applications, we set a playlist in these applications so they can generate traffic automatically. The duration of all traffic is 2 hours.

TABLE I
DATASET

Dataset	Size	Number of Flows	Date	Duration
Youku	1.94GB	16017	2019-12-6	2 hours
Weishi	1.83GB	12097	2019-11-25	2 hours
Kuaishou	1.52GB	10148	2019-11-24	2 hours
Tiktok	1.47GB	9718	2019-11-25	2 hours
Bilibili	2.02GB	14132	2019-12-6	2 hours

C. Accuracy

In order to obtain the accuracy/precision/recall of our cleaning framework, we choose the 75% traffic for training and 25% traffic for testing. After performing the clustering algorithm, we use the data plane traffic to train a classifier by choosing the cluster whose traffic ratio is closer to 1. We also manually clean the traffic by checking every packet offline. We use all 5 application traffic to train a multi-class random forest classifier based on *scikit-learn*.

From Fig. 4 we can see that utilizing the clustering algorithm only brings 2% $\sim$ 2.5% accuracy loss then the manual traffic cleaning approach. Moreover, the hierarchical clustering algorithm has slightly higher accuracy than the k-means algorithm.

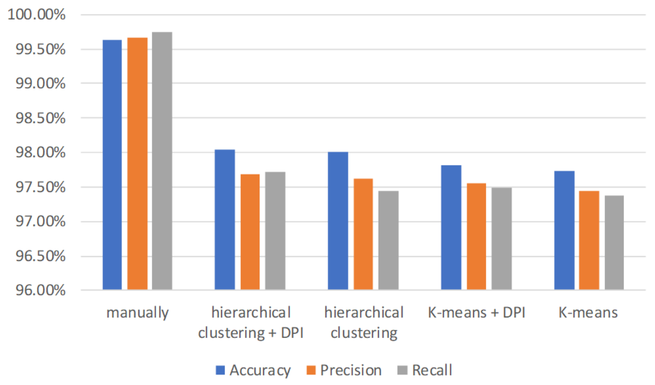

Fig. 4. The accuracy/precision/recall comparison between manually cleaning and our cleaning framework. We do not show the uncleaned result since the accuracy of uncleaned traffic is less than 50%. We use both k-means and hierarchical clustering algorithms to evaluate traffic cleaning. The result shows that our cleaning framework only has 2% $\sim$ 3% accuracy loss than the manually traffic cleaning

D. Performance

As we have claimed that manually checking every packet is expensive, which may consume over hours or days, we evaluate the time consuming of our traffic cleaning framework to show the computing performance.

Fig. 5 shows that the time consuming of our framework is less than $33s$ with DPI, $12s$ without DPI, which is significantly less than manually traffic cleaning. Moreover, the hierarchical cluster algorithm uses more time than the k-means algorithm.

As mentioned above, we also utilize high-performance vector optimization to further increase the performance of the traffic cleaning framework. We use other traffic data sets generated by Youku with different numbers of traffic to evaluate the performance of our optimized implementation. We use k-means as our clustering algorithm in this evaluation.

From Fig. 6 we can see that with the C++ and vector optimized implementation, the traffic cleaning efficiency has

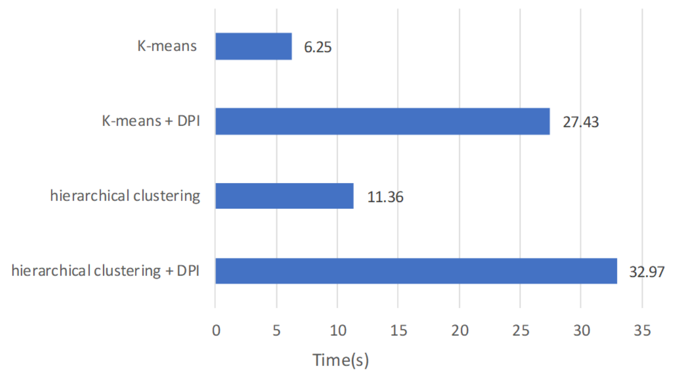

Fig. 5. The performance comparison between different clustering algorithm. We do not show the time of manually traffic cleaning since it costs more than several hours. The result shows that the hierarchical clustering algorithm uses more time than the k-means algorithm.

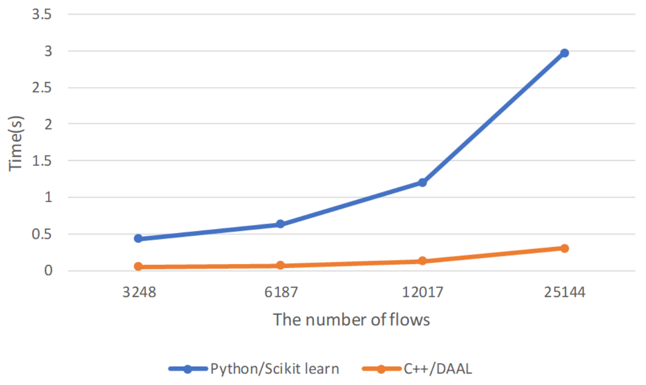

Fig. 6. The performance comparison between *python/scikit-learn* and *C++/DAAL*. With the increasing number of flows, the time of our optimized traffic cleaning framework does not raise much. The time cost of non-optimized solution on cleaning 25144 traffic is about 10 times than our optimized solution.

a significant increase.

E. Analysis

First of all, the evaluation results show that our traffic cleaning framework can effectively clean the traffic without manual intervention. The DPI seems to be less effective since most of unencrypted traffic is not data plane traffic. However, if we choose control plane traffic to train a classifier, the DPI is indispensable in cleaning unencrypted traffic. Also, there is a tradeoff in choosing the clustering algorithm. The hierarchical clustering has a better accuracy but the k-means has a better computing performance. But most of the time, the cleaning accuracy of k-means is enough for traffic classification.

VI. Conclusion

In this paper, we have proposed a traffic cleaning framework to address the encrypted mobile traffic cleaning problem. We have designed an online structure that utilizes the unsupervised learning algorithm to automatically clean encrypted traffic data. Briefly speaking, we propose a framework that utilizes an unsupervised learning based algorithm to extract clean data from the collected traffic data. Moreover, we apply high-performance instruction such as vector optimization to fully increasing computing efficiency. Our evaluation results demonstrate the effectiveness of our framework with only $2\% \sim 2.5\%$ accuracy loss comparing to manual traffic cleaning. Moreover, our optimized implementation can achieve 10-times speedups over the non-optimized implementation.

References

[1] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, and J. Aguilar, “Towards the deployment of machine learning solutions in network traffic classification: a systematic survey,” *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1988–2014, 2018.
[2] A. K. Marnerides, A. Schaeffer-Filho, and A. Mauthe, “Traffic anomaly diagnosis in internet backbone networks: A survey,” *Computer Networks*, vol. 73, pp. 224–243, 2014.
[3] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, “Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges,” *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 445–458, 2019.
[4] A. Zulkifli, I. R. A. Hamid, W. M. Shah, and Z. Abdullah, “Android malware detection based on network traffic using decision tree algorithm,” in *International Conference on Soft Computing and Data Mining*. Springer, 2018, pp. 485–494.
[5] T. Bakhshi and B. Ghita, “On internet traffic classification: A two-phased machine learning approach,” *Journal of Computer Networks and Communications*, vol. 2016, 2016.
[6] R.-y. WANG, L. Zhen, and L. ZHANG, “Method of data cleaning for network traffic classification,” *The Journal of China Universities of Posts and Telecommunications*, vol. 21, no. 3, pp. 35–45, 2014.
[7] (2020) Caida. [Online]. Available: http://www.caida.org/data/
[8] (2020) Kdd cup 1999 data. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[9] (2020) Darpa intrusion detection data sets. [Online]. Available: https://ll.mit.edu/ideval/data
[10] Y. Elovici, A. Shabtai, R. Moskovitch, G. Tahan, and C. Glezer, “Applying machine learning techniques for detection of malicious code in network traffic,” in *Annual Conference on Artificial Intelligence*. Springer, 2007, pp. 44–50.
[11] D. M. Divakaran, L. Su, Y. S. Liau, and V. L. Thing, “Slic: Self-learning intelligent classifier for network traffic,” *Computer networks*, vol. 91, pp. 283–297, 2015.
[12] J. J. Davis and A. J. Clark, “Data preprocessing for anomaly based network intrusion detection: A review,” *computers & security*, vol. 30, no. 6-7, pp. 353–375, 2011.
[13] J. J. Davis and E. Foo, “Automated feature engineering for http tunnel detection,” *Computers & Security*, vol. 59, pp. 166–185, 2016.
[14] (2020) Cisco joy. [Online]. Available: https://github.com/cisco/joy
[15] C. Zhong, H. Liu, and A. Alnusair, “Leveraging decision making in cyber security analysis through data cleaning,” *Southwestern Business Administration Journal*, vol. 16, no. 1, p. 1, 2017.
[16] A. W. Moore and D. Zuev, “Internet traffic classification using bayesian analysis techniques,” in *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, 2005, pp. 50–60.
[17] R. Yuan, Z. Li, X. Guan, and L. Xu, “An svm-based machine learning method for accurate internet traffic classification,” *Information Systems Frontiers*, vol. 12, no. 2, pp. 149–156, 2010.
[18] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, “Traffic classification on the fly,” *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 2, pp. 23–26, 2006.
[19] (2020) Bluestack, an android simulator. [Online]. Available: https://www.bluestacks.com/
[20] (2020) Scikit-learn, ml in python. [Online]. Available: https://scikit-learn.org/stable/
[21] (2020) ndpi, the open source dpi tool. [Online]. Available: https://https://github.com/ntop/nDPI
[22] (2020) Intel data analytics acceleration library. [Online]. Available: https://software.intel.com/dpd/daal-library