Journal of Education and Science (ISSN 1812-125X), Vol: 30, No: 5, 2021 (86-102)

Ransomware Detection System Based on Machine Learning

Omar Shamil Ahmed[1]*, Omar Abdulmunem Ibrahim Al-Dabbagh[2]

[1*] Department of Basic Sciences, College of Agriculture and Forestry, University of Mosul, Mosul, Iraq
[2] Department of Computer Science, College of Computer Sciences and Mathematics, University of Mosul, Mosul, Iraq

E-mail: [1*]omarshamil@uomosul.edu.iq, [2]dr.omar.aldabbagh@uomosul.edu.iq

(Received July 06, 2021; Accepted September 09, 2021; Available online December 01, 2021)

DOI: 10.33899/edusj.2021.130760.1173, © 2021, College of Education for Pure Science, University of Mosul.

Abstract

Every day, there is great growth of the Internet and smart devices connected to the network. Additionally, there is an increasing number of malwares that attack networks, devices, system and applications. One of the biggest threats and newest attacks in cybersecurity is Ransom Software (Ransomware). Although there is a lot of research on detecting malware using machine learning (ML), only a few focus on ML-based ransomware detection, especially attacks targeting smartphone operating systems (e.g., Android) and applications. In this research, a new system was proposed to protect smartphones from malicious applications through monitoring network traffic. Six ML methods (Random Forest (RF), k-Nearest Neighbors (k-NN), Multi-Layer Perceptron (MLP), Decision tree (DT), Logistic Regression (LR), and eXtreme Gradient Boosting (XGB)) are applied to CICAndMal2017 dataset which consists of benign and various kinds of android malware samples. 603288 benign and ransomware samples were extracted from this collection. Ransomware samples were collected from 10 different families. Several types of feature selection techniques have been used on the dataset. Finally, seven performance metrics were used to determine the best feature selection and ML classifiers for ransomware detection. The experiment results imply that DT and XGB outperform other classifiers with best detection accuracy at more than (99.30%) and (99.20%) for (DT) and (XGB) respectively.

Keywords: Malware, Ransomware, Static and Dynamic analysis, Network Traffic, ML algorithms;

نظام الكشف عن برامج الفدية المستند الى التعلم الالي

عمر شامل احمد[1]*، عمر عبد المنعم إبراهيم الدباغ[2]

[1*] قسم العلوم الاساسية، كلية الزراعة والغابات، جامعة الموصل، الموصل، العراق
[2] قسم علوم الحاسوب، كلية علوم الحاسوب والرياضيات، جامعة الموصل، الموصل، العراق

الخلاصة:

في كل يوم، هناك نمو كبير في الإنترنت والأجهزة الذكية المتصلة بالشبكة. من ناحية أخرى، هناك زيادة في عدد البرامج الضارة التي تهاجم الشبكات والأجهزة والأنظمة والتطبيقات. تعد برامج الفدية (Ransomware) أحد أكبر التهديدات وأحدث الهجمات في مجال

الأمن السيبرانى. على الرغم من وجود الكثير من الأبحاث حول اكتشاف البرامج الضارة باستخدام التعلم الآلى ML، إلا أن القليل منها يركز فقط على اكتشاف برامج الفدية المستندة إلى ML. خصوصاً الهجمات التى تستهدف أنظمة تشغيل الهواتف الذكية (مثل Android) والتطبيقات. فى هذا البحث، تم اقتراح نظام جديد لحماية الهواتف الذكية من التطبيقات الضارة من خلال مراقبة حركة مرور الشبكة. يتم تطبيق ستة خوارزميات للتعلم الالى (Random Forest و k-Nearest Neighbors و Decision Tree و Multi-Layer Perceptron و eXtreme Gradient Boosting و Logistic Regression) على مجموعة بيانات CICAndMal2017 التى تتكون من عينات حميدة وأنواع مختلفة من عينات البرامج الضارة لنظام Android. تم استخراج 603288 من العينات الحميدة وبرامج الفدية من هذه المجموعة. تم جمع عينات برامج الفدية من 10 عائلات مختلفة. كما تم استخدام عدة أنواع من تقنيات اختيار الميزات على مجموعة البيانات. أخيرًا، تم استخدام سبعة مقاييس للأداء لتحديد أفضل تقنيات اختيار الميزات وأفضل مصنفات ML لاكتشاف برامج الفدية. تشير نتائج التجارب إلى أن DT و XGB يتفوقان على المصنفات الأخرى بأفضل دقة كشف تتجاوز (99.30٪) و (99.20٪) لـ (DT) و (XGB) على التوالى.

الكلمات المفتاحية: البرامج الضارة، برامج الفدية، التحليل الثابت والديناميكي، حركة مرور الشبكة، خوارزميات التعلم الالي.

1. Introduction

The Android OS covers the world with 85% of smart devices and phones market share and it continues to grow [1]. In the last years, relying on smart devices (especially smartphones) has increased in daily activities ranging from studying, shopping, and entertainment, to financial transactions [2]. The reasons for this are due to recent technological developments, the widespread of smartphones and contemporary conditions (such as the COVID-19 quarantines). These reasons have the Android OS the main target for attackers [3]. Unfortunately, Android differs from other mobile OS, in that it maintains openness and does not impose many restrictions on users for application uploading and downloading. It leaves the safety of the phones and devices in the user's hands through letting them decide whether or not to install an application thus, smartphones have become more liable to cyber-attacks [1].

Cyber criminals are developing malicious applications to target individuals, companies and even governments. Ransomware takes over the victim's device, and blocks or encrypts the data, therefore, preventing the victim from using the device. The victim can get back to using the device or its data only if ransom is paid [4]. Ransomware made history in 2020 as it contributed to the first reported death related to a cyber-attack, when a German hospital was attacked by ransomware, causing a lock out of their systems and preventing treatment of patients. Consequently, a woman in need of urgent help died [5].

According to Cisco Annual Internet Report [6], more than 299 billion mobile applications will be downloaded and used in 2023. With the great growth and increasing use of applications, network interaction and utilization has increased substantially via these applications [7]. Besides, network traffic has increased dramatically due to many permanently linked applications like social networking applications. Now, with the fast evolution of the Internet, the fifth generation will make AI systems, the Internet of Things and self-driving cars the most important tools that humans use in their lives. By 2023, the fifth generation connection will create three times more traffic than the fourth generation connection [6]. To detect Android malware, antivirus software uses standard code analysis and signature detection techniques which are known to hackers. The use of ML methods for network traffic detection is one of the best solutions used as effective ransomware detection [3].

1.1 Malware

The malicious software, which is known as malware, is one of the most dangerous and most common cyber threats. These programs are built to collect sensitive data and information, disrupt, damage, or gain unauthorized access to applications, system, or networks [8]. In addition to losses in money, information, time, and infrastructure for individuals, companies and institutions, these victims could incur other harms like loss of lives. According to the malware function and its proliferation systems, malware can be divided (not exclusive categories) into several types such as (ransomware, adware, virus, worm, trojan, bot, scareware, etc.) [9].

1.1.1 Ransomware

Ransomware is a class of serious safety problems that attacks individuals, companies, organizations and even governments. It locks up or encrypts information, systems, and devices, then asks for ransom payment to access and use them again [7]. Victims (persons, companies, etc.) may be exposed to great damage as a result of paying money as ransom in addition to the costs of downtime and loss of reputation and may even lead to loss of life [10]. In 2020, cybercrime has witnessed a great evolution in ransomware attacks. This threat has included malevolent tactics and targeted some of the most vulnerable industries in the year. A survey of 1,100 IT professionals showed that 90% of them had customers who experienced ransomware attacks in 2020 [5]. By 2021, ransomware is expected to cause $20 billion in loss [11]. The ransomware growth in the recent years is imputed to the rise of Ransomware-as-a-service (RaaS). By (RaaS), a novice attacker can easily pay a service, customize his ransomware, and deploy it on many computers around the world.

There are two common kinds of Ransomware: the Crypto- Ransomware that encrypts the victim's files and data and prevents him/her from access, and provides the decryption files key only if the ransom is paid. The second type is Locker-Ransomware that leaves the victim’s files and data as they are, but it prevents the victim from accessing and using his/her device, and access can only be regained once a ransom is paid. Usually, the Crypto-Ransomware encrypts a specific extension from the memory, like .jpg, .doc, .pdf, that is, files that contain text documents, presentations, and images which usually include important and personal user data.

The techniques of ransomware detection include statistical-based techniques, event-based techniques, data-centric-based techniques, and ML based techniques which are considered as a new research topic. The focus of this research is on using the techniques of ML for android ransomware detection by monitoring network traffic.

1.2 Static & dynamic analysis

To save phones and smart devices from threats that attack android OS, various solutions based on features analysis were proposed. The static analysis is the approved technique by antivirus companies. It is a passive approach based on signature check and source code by educing the features from the source code of the applications or extracting them from the binary strings. This means testing files or applications without running its code. It is a faster and safer approach that generates rich information about malware samples. But it suffers several flaws. Cyber-criminals use different polymorphism and obfuscation techniques to overcome the detection systems via packing and encryption [9].

Another solution is dynamic analysis which involves executing the malicious file in an isolated and safe environment (e.g., sandbox) in order to know the real behavior, that is, the way it interacts with the underlying OS, and analyze their execution logs. But this technique needs more processing capacity and battery power. It is difficult because sometimes the malware that is being analyzed in a safe environment differs from the real one, and it is not available for all researchers [12]. both analysis techniques have their

benefits and flaws. The static analysis of features is safer and faster than dynamic but, malicious programs can avoid detection using techniques of code obfuscation. In contrast, polymorphic malware techniques and code obfuscation hardly evade dynamic analysis because malware is monitored and analyzed at execution time. Therefore, to use dynamic analysis at the lowest cost, one of the best solutions is to use CICAndMal2017 dataset which is available and based on real-time network traffic features.
The contributions of this research are:

1. Using the data preprocessing techniques on the network traffic dataset and explaining the benefits of these techniques on the performance of android ransomware detection.
2. Understanding feature selection techniques and comparing between them to determine the best option.
3. Applying ML methods on the dataset based on best features to determine the best methods for ransomware detection.
4. This research aspires to discover the perfect method to detect ransomware via monitoring network flow, which is done by comparing the research with related works that have used the same dataset.

The remaining parts of this paper are as follows: section 2 reviews the related works to detect malware by ML methods. The details of dataset and network traffic are explained in section 3. The proposed model is discussed in section 4. The data preprocessing techniques are shown in section 5. The experiments in feature selection techniques, ML classifiers, and the results are explained in section 6.

2. Related Works

Much research has been conducted in the area of Android malware detection. This research relied on static, dynamic, or hybrid features analysis. In 2017, Chen et al. designed a novel system that utilizes data mining techniques with dynamic analysis to monitor the Application Programming Interface (API), which is the interaction procedures and protocols between applications, for ransomware detection. In order to create API call flow, the authors monitor the behaviors of software, then, they commence mapping the API calls in a feature space. The researchers applied data normalization technique and methods of feature selection to select the best for discriminating between ransomware and benign software. Then, four data mining algorithms were used for building the detection model. The experimental results show that SL algorithm can achieve 98.2% and 97.6% accuracy and detection rate respectively [13].

In 2018, Al-rimy et al. conducted research that provides an important and detailed review of ransomware. This survey provides a comprehensive demonstration and study of the latest technology to detect and prevent ransomware. The authors made a new classification of ransomware from various viewpoints. They explain the factors and circumstances that helped make these attacks successful, and discuss the related research into struggling ransomware, with various solutions for analyzing these attacks, as well as detecting, preventing, and predicting them [10].

In 2018, Cusack et al. used a programmable forwarding engine (PFEs) which collects network monitoring data for per-packet. This data was utilized to monitor the flow of network between the command and control (C&C) server and an infected computer. After feature extraction from the flow, the authors used this data to classify ransomware. The classification model achieved 86% detection rate with 11% false negative rate [12].

In 2018, Zhang et al. utilized static feature analysis for ransomware classification. First, from ransomware samples, the authors transformed opcode sequences into N-gram sequences. Then, they treat N-gram vectors as feature vectors. Next, they introduce these vectors into five ML methods to classify ransomware. The models are validated with six metrics. The proposed approach achieves 91% accuracy, and 99% accuracy of binary classification [14].

In 2018, Alhawi et al. proposed a NetConverse, which is a ML method for ransomware detection. They use TShark to create a dataset from the conversation of network traffic. Features are extracted and fed into ML classifiers that achieved 97% accuracy for DT and 96% for Logistic Model Tree (LMT) [15].

In 2019, another work utilized the features of network to detect ransomware using ML classifiers. Kaiiali et al. focus on crypto ransomware network activities. Network features are extracted and fed for two classifiers that operate in parallel on (packet and flow levels). The detection accuracy of the two proposed levels was 97.92% and 97.08% respectively [16].

In 2019, Noorbehbahani et al. applied two experiments on CICAndMal2017 dataset to analyze six ML techniques (DT, RF, Random Tree (RT), k-NN, NB, and SVM) for ransomware detection. Firstly, a dataset was applied with different forms and classes of ransomware on ML classifiers. Then they were applied to 10 ransomware families separately on classifiers. The results show that RF was the best in both experiments. The highest detection accuracy belongs to RF with accuracy score 83%, and 79% for DT [7].

In 2020, Moussaileb et al. suggested an analysis of different families of ransomware depending on collected logs form a device system and network. In order to packet detection, the authors delved into the malware network traffic that created by these samples. This work shows that using DT to detect zero-day attacks provides high detection rates among other ML algorithms [17].

In 2020, Sangal et al. used ML methods for new android malware detection. They applied many techniques (RF, KNN, SVM, and NB) on a dataset of android applications with permissions and intent features. First, they performed data pre-processing to handle missing values. Next, they used feature selection to minimize the dataset dimensions. They used AndMal2019 dataset which was provided by CIC. The best detection result was 96% using RF classifier [18].

In 2020, during the lockdown due to COVID-19, everyone sat at home and their interactions with others increased mostly through smartphones. Hence, this presented an opportunity for cyber criminals to develop malware-infected applications. For this purpose, Sangal et al. proposed a new system which focuses on machine learning and signature-based methods to detect known Android malware. In this work, 11,000 distinct Android applications belonging to twelve different Android application categories were collected. Ten feature selection methods were used to reduce the dimensions of the dataset. For Android malware detection, Deep Neural Network (DNN) machine learning technology was used, and it achieved 97% of malware detection points from real-world applications [19].

In 2020, 67 research papers for malware detection and classification were reviewed in a deep survey by Mateu et al. This study aimed ats providing a detailed and systematic overview of ML methods (especially Deep learning DL methods) for malware detection. It also offered a description of the features and methods in a traditional ML process, from feature extraction, and selection steps to detection and classification. It explained all feature analysis methods with all the branches [20].

In 2021, a survey for risks of cyber-attacks and Advanced Persistent Threat (APT) attacks was made. Lee et al. touched upon the rapid development of APTs and their use of AI techniques to design the new kind of ransomware, that spreads quickly between users of IoT devices and smartphones to infect the largest number of them at the same time. The authors proposed using detection and response tools which can quickly extract ransomware attack features and respond of this threat. They built an open-source framework that enables ransomware detection at the system and network level [21].

3. The Network Traffic and Dataset used

Network traffic or data traffic refers to the data that moves across the network at any time. It consists of a sequence of packets (packet is the smallest unit of data that is passed over a network). Each packet includes Payloads (raw data) and Headers (metadata) that contain basic flow information [15]. One of the best methods to detect malware is to monitor malicious network traffic which can uniquely offer a clear view of the behavior of malware applications. When a malicious program infects a victim's device, it may establish a connection to an external server to perform a malicious operation like download updates or other malware, to obtain new commands, or to steal sensitive and private information [17]. Therefore, monitoring network traffic that enters the network and leaves it, intra-network traffic and device activity, provides important and useful information to disclose malicious behavior.

The dataset used in this research was obtained from the Canadian Institute for Cybersecurity [22]. It is a collection of benign samples and several malware types. The AndMal2017 dataset includes network traffic, API/SYS calls, memory dumps, logs, and phone statistics with 42 malware families. The previous works proved that network traffic can be utilized to detect and classify android malware. Therefore, this research focuses on the network traffic feature for detecting ransomware applications. 603288 ransomware and benign data samples were extracted with network flow features that consists of six columns for each flow (FlowID, SourceIP, DestinationIP, SourcePort, DestinationPort, and Protocol) and 79 network traffic features. This dataset was created via CICFlowMeter software [23] which is a network traffic flow generator and analyzer.

3.1. Ransomware Dataset

In this research, 353288 ransomware samples were used with 85 features which were collected from 10 popular ransomware families. Table I lists the behavior and characteristics of ransomware and the number of samples used for each one of the families.

Table I. Description of behavior and characteristics of ransomware dataset

Ransomware family	AV labelled	Num. of samples	Attack							
			Att-1	Att-2	Att-3	Att-4	Att-5	Att-6	Att-7	Att-8
Charger	Sophos	39551	√			√	√	√		
Jisut	ESET	25672					√	√		
Koler	Avast	44555		√	√			√		
LockerPin	ESET	25307					√	√	√	
Simplocker	Symantec	4715	√			√		√	√	
Pletor	Alibaba	46082						√	√	√
PornDroid	Ikarus	39859						√		√
RansomBO	Fsecure	40685						√	√	
Svpeng	Sophos	54161	√					√	√	
WannaLocker	Avast	32701							√	√
Total Ransomware samples		**353288**								

Description:
Att-1: Steal data (credit card credentials, contacts and SMS messages)
Att-2: Harvest data (bookmark history, text messages and mobile accounts)
Att-3: Phishing to the contact list / Send SPAM SMS
Att-4: Download malicious software (malware)
Att-5: Malware spread (uninstall AVs, load dynamic code, source encrypting)
Att-6: Lock up the device
Att-7: Encrypt the user files and data
Att-8: Modify contents / SD card

3.2. Benign Dataset

The benign applications used in this research were published in 2015, 2016 and 2017 in Google play market. These applications are more than six thousand and they have been grouped based on the popularity of the applications (best free new and most free popular) for each class available in the market. These applications were checked in Virustotal Web Service with two Antivirus Products (AV) [24]. 250000 benign samples with 85 features of network traffic were extracted and used in this research. These features can be categorized into classes such as (Flow-ID, Packet-based, Byte-based, Flow-based, Time-based).

4. The Proposed System

The proposed system in this research consists of five steps as shown in Figure I.

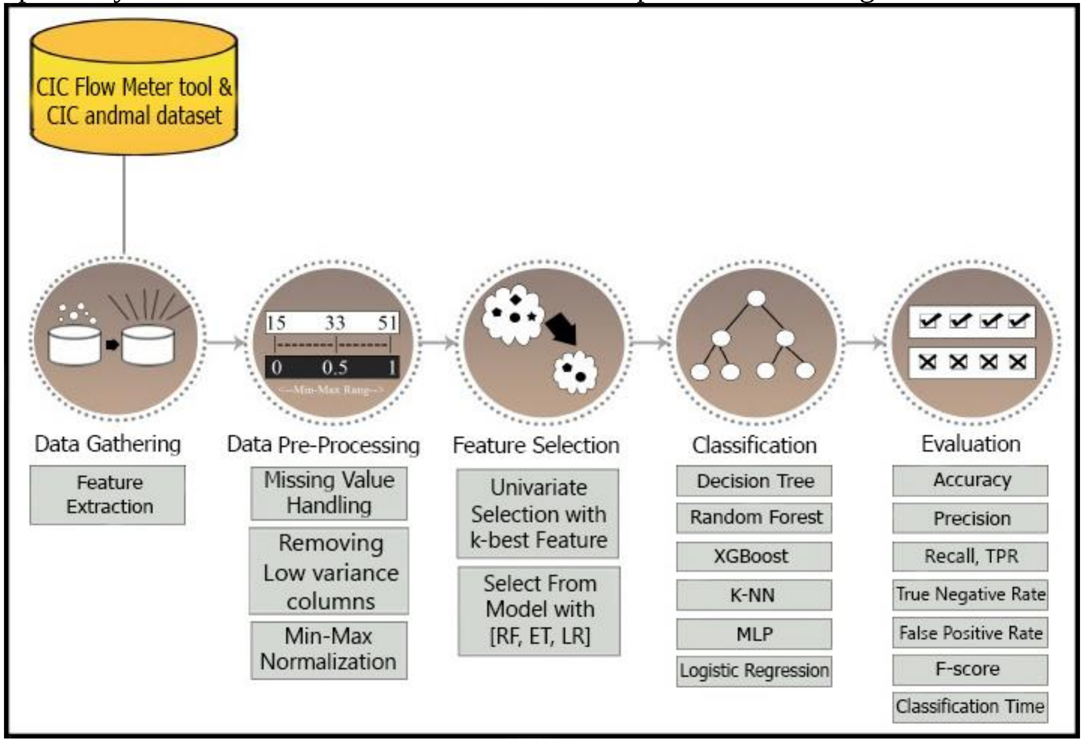

Figure I. The methodology of the proposed system

The five steps can be briefly explained as follows:

A. Gathering a network feature dataset using traffic capture such as CICFlowMeter [23] or from sober website such as [22].
B. Data preprocessing is the second step, through removing the missing value in the columns, removing features with low variance values. Then, the technique of data normalization was used to scale and modify the data in the range [0 and 1] via (Max-Min method). It is notable that in this research, the dataset used does not contain any missing values.
C. The third step is feature selection. Several techniques were used to analyze and select the best features from the dataset.
D. Six algorithms of ML were applied on the selected features from the previous step. The dataset used in this work was divided into 80% for training the algorithms, and the rest for testing.
E. Finally, (Testing and Evaluation step), the 20% remaining dataset that broke up from the total data was used for testing the ML classifiers.

5. The Data preprocessing

The analysis shows that the CICAndMal2017 dataset includes 85 features of network traffic. The names of the features are shown in the Table II, along with their values that are an instance in the dataset.

Table II. Example of an instance (the features and their values) in CICAndMal2017 dataset

No.	Feature name	Feature value	No.	Feature name	Feature value
1	Flow ID	198.11.132.53-10.42.0.211-443-57835-6	44	Bwd Packets/s	0.261789451
2	Source IP	10.42.0.211	45	Min Packet Length	0
3	Source Port	57835	46	Max Packet Length	1452
4	Destination IP	198.11.132.53	47	Packet Length Mean	268.2142857
5	Destination Port	443	48	Packet Length Std	468.915325
6	Protocol	6	49	Packet Length Variance	219881.582
7	Timestamp	28/08/2017 02:20:24	50	FIN Flag Count	0
8	Flow Duration	61117818	51	SYN Flag Count	0
9	Total Fwd Packets	11	52	RST Flag Count	0
10	Total Backward Packets	16	53	PSH Flag Count	1
11	Total Length of Fwd Packets	1048	54	ACK Flag Count	0
12	Total Length of Bwd Packet	6462	55	URG Flag Count	0
13	Fwd Packet Length Max	419	56	CWE Flag Count	0
14	Fwd Packet Length Min	0	57	ECE Flag Count	0
15	Fwd Packet Length Mean	95.27272727	58	Down/Up Ratio	1
16	Fwd Packet Length Std	139.3908827	59	Average Packet Size	278.1481481
17	Bwd Packet Length Max	1452	60	Avg Fwd Segment Size	95.27272727
18	Bwd Packet Length Min	0	61	Avg Bwd Segment Size	403.875
19	Bwd Packet Length Mean	403.875	62	Fwd Header Length	240
20	Bwd Packet Length Std	580.0636603	63	Fwd Avg Bytes/Bulk	0
21	Flow Bytes/s	122.8774234	64	Fwd Avg Packets/Bulk	0
22	Flow Packets/s	0.441769698	65	Fwd Avg Bulk Rate	0
23	Flow IAT Mean	2350685.308	66	Bwd Avg Bytes/Bulk	0
24	Flow IAT Std	11753837.31	67	Bwd Avg Packets/Bulk	0
25	Flow IAT Max	59977363	68	Bwd Avg Bulk Rate	0
26	Flow IAT Min	14	69	Subflow Fwd Packets	11
27	Fwd IAT Total	1140235	70	Subflow Fwd Bytes	1048
28	Fwd IAT Mean	114023.5	71	Subflow Bwd Packets	16
29	Fwd IAT Std	117387.5319	72	Subflow Bwd Bytes	6462
30	Fwd IAT Max	296745	73	Init_Win bytes_forward	65535
31	Fwd IAT Min	148	74	Init_Win bytes_backward	11680
32	Bwd IAT Total	61029770	75	act_data pkt_fwd	5
33	Bwd IAT Mean	4068651.333	76	min_seg size_forward	20
34	Bwd IAT Std	15473328.797	77	Active Mean	1140235
35	Bwd IAT Max	59999671	78	Active Std	0
36	Bwd IAT Min	14	79	Active Max	1140235
37	Fwd PSH Flags	0	80	Active Min	1140235
38	Bwd PSH Flags	0	81	Idle Mean	59977363
39	Fwd URG Flags	0	82	Idle Std	0
40	Bwd URG Flags	0	83	Idle Max	59977363
41	Fwd Header Length	240	84	Idle Min	59977363
42	Bwd Header Length	352	85	Label	RANSOMWARE
43	Fwd Packets/s	0.179980247			

After analyzing the dataset, it was noted that its features have different value ranges, in addition to features (columns) that have complete zero values. Therefore, the dataset needs data preprocessing operations before using ML algorithms on it, as follows.

5.1 Deleting the low variance columns (features)

When a dataset includes features that has values with a very slight variance or has same value for all rows in the column, then these features will not add any informative power to the model [25]. Hence, using these features also adds an unnecessary computational burden and should be removed from the dataset. In order to improve the model performance, the technique of deletion features with low variance is used. The Variance Threshold technique which was provided by sklearn [26] was used in this research. VarianceThreshold is a simple basic feature selector that deletes the low-variance columns. This technique only handles the input columns (X), not to the target column (y), and it is most useful when used for unsupervised learning. After applying this technique on the dataset, twelve low-variance features were removed from the whole feature set as shown in Table III. The remaining features are 69 out of a total of 85 features. Four columns were removed from the data that were entered to ML algorithms because their values are string and cannot be used for training the algorithms. Up to this step, the data entered is 69 features.

Table III. The features with low variance (zero values in all columns)

No.	Feature's Name	No.	Feature's Name
1	Bwd PSH Flags	7	Fwd Avg Bytes/Bulk
2	Fwd URG Flags	8	Fwd Avg Packets/Bulk
3	Bwd URG Flags	9	Fwd Avg Bulk Rate
4	RST Flag Count	10	Bwd Avg Bytes/Bulk
5	CWE Flag Count	11	Bwd Avg Packets/Bulk
6	ECE Flag Count	12	Bwd Avg Bulk Rate

5.2 Normalizing the dataset values

Researchers always aspire to get the best performance of the designed system. Data normalization is a very important technique that is used to improve the performance of the ML system [13]. The reason for this is that some datasets (e.g., CICAndMal2017 dataset) include features with very different values, ranges, and scales. For example, one of the CIC dataset features is "FIN Flag Count" which is (the number of packets with FIN), the values of this feature are 0 or 1. Whereas the values of "Flow Duration" feature (duration of the flow in microsecond) is in the range of tens to millions of microseconds. When a feature selection technique is used on these features, then this technique tends to bias toward larger values over smaller values. To solve this problem, a data normalization technique is used. In this research, the Max-Min technique was used to normalize data attribute in the dataset within the range [0-1] using equation 1 [25], where Z ϵ [min, max] and z' ϵ [0, 1].

$$z' = \frac{Z - min}{max - min} \quad \dots (1)$$

6. Experiments

Several experiments were performed on the proposed system in order to obtain a perfect result. These experiments included:

6.1 The Techniques of Feature Selection

The efficiency of the detection and classification system usually depends on the nature and quality of the dataset. Also, data with higher dimensions increases noise and may lead to a complex detection model thus, the need to use feature selection appears. The feature selection or (feature reduction) technique is defined as the operation of identifying and selecting the feature that more related to the desired output variable, and thus, reducing the mathematical and statistical operations [27, 28]. The results are (reducing classification time, reducing overfitting, and increasing accuracy). It is especially important to use feature selection technique particularly when the dataset is very huge. To find out the benefit of using this technique, first all feature sets (69 features) were used to train and test the six ML classifiers. After this, the next two feature selection techniques were used to train and test the ML classifiers.

6.1.1 Univariate Feature Selection Technique

The statistical test Univariate feature selection was used to determine which of the features are the best and choose the one that has a strong relationship with the target via univariate statistical tests. When it analyzes the relationship between one feature and its goal, it ignores the other features. That is the reason to called it 'univariate' because each feature has its result. In the end, all the results are compared, and then f-test or (f- statistic) method was used to select features with top scores. f-test is a method that is used when the input data is in numerical form and the output is categorical. The sklearn Python provides f_classif() function which is implementation of f-test method [29]. Finally, SelectKBest( ) function [30] was used to choose the best (10, 15, 20, 25) features out of the 69 total features.

6.1.2 SelectFromModel technique

SelectFromModel is a technique that is used with a model (estimator) which has feature_importantance attribute. The best features, which are the most important features, are chosen according to feature weights [31]. SelectFromModel handles all features at the same time, thus it can capture interactions compared to univariate feature selection. Three models (Random_Forest, Extra_Trees, and Logistic_Regression) were used to fit data and select features. The results of applying SelectFromModel technique on the dataset was selection of 21 features with RF model, 20 features with ET model, and 17 features with LR model out of the whole number of features. The effect of using these features to train and test the ML classifiers is shown in the results in Table V.

6.2 Machine Learning Classifiers

Six ML classifiers which are commonly used in the field of cybersecurity [3, 9, 15] have been used to detect Android ransomware in this research. The focus on these classifiers was to measure their efficacy in detecting ransomware when used with data normalization and feature selection techniques that were not used in previous works. Another focus is also to design a simple and perfect security system based on machine learning techniques and methods for early detection of ransomware in network traffic before it hits the target. The ML classifiers used are:

6.2.1 Decision Tree (DT): DT is a simple regression and classification method. It is a Supervised ML sequential model where the data is constantly split according to a certain parameter with a series of tests, similar to a flow chart structure where the inner node denotes a test on a feature. The leaf node holds a class label, and each branch of tree represents a result of the test. The DT flow starts when the features which were extracted from a new sample are introduced to the tree. Then it creates a group of questions to ask of this sample's features sequentially. The max_depth parameter (maximum depth of the tree) is very important to control the efficacy of the tree. The tree with most depth produces the best result, but it will require a lot of time and calculations to process [3, 32]. In decision tree algorithm, the input is the maximum depth of the tree, which is 40 in this research, and the strategy (best or random) used to select the split at each node, was the 'best' in this research. The most important feature will be the root of the DT, then other features will be distributed from the top of the DT to the bottom depending on several sequential questions (decisions) and the results of information gain. The output in DT is the final nodes which represent the results of the questions (the target feature values), in this research the output is the class of the sample (benign or ransomware).

6.2.2 Random Forest (RF): RF is an ensemble method that is based on and consists of many DTs and bagging techniques. Bagging demands train each DT on a part of the whole dataset. Each tree gets its classification, and finally the classification is done using majority voting on the DTs results. The most important parameters are max_depth that define the maximum depth of the tree, and n_estimators, which define the number of trees in the forest [9, 32]. In random forest algorithm, the input is the number of decision trees and the maximum depth of the tree used to train the dataset. In this research, the number of the trees in the forest was 50, and the maximum depth of each tree was 25. Several decision trees will produce several classification results. Thus, the output of random forest for classification problems is the majority vote of the most trees, which represent the classification result (ransomware or benign).

6.2.3 Logistic Regression (LR): LR is an algorithm that is based on 'Statistical Learning' method. It is used for regression and classification tasks. LR is a probability-based prediction algorithm that uses sigmoid function to transform the output and returns value of probability. It separates between the samples by making a boundary (hyperplane or line). For the new samples, LR examines these samples to learn on which side of the hyperplane they are located, then it makes the decision. The most important parameters are max_iter parameter which define the maximum number of iterations [3, 9]. In logistic regression algorithm, the input data with N samples and M features was used to train the algorithm with maximum number of iterations, which is 150 iterations. The multi_class parameter was set to 'ovr' to handle binary classification issues. The solver parameter (the strategy of the optimization problem) was set to 'lbfgs' that process complex dataset in faster way. These parameters were set to produce a predictive model for the output variable (benign or ransomware) using sigmoid function, and this is the output of LR.

6.2.4 k-Nearest Neighbor (k-NN): it is one of supervised ML algorithm that are used in the tasks of regression and classification. It assumes that similar things exist in close proximity (near) to each other. It stores the part of training data and does not make the prediction until it receives the part of test data. The process of prediction is done when getting the instance of test data, then it scans the training data for the k most similar neighbors. For this, it is computationally expensive. The most important parameter is n_neighbors, which is defined as the number of neighbors to use [3, 9]. In k-NN algorithm, the input is the k- nearest training samples in the dataset, which equal to 5 in this research. All points in neighborhood have same weights through using 'uniform' weight function. The prediction for each sample in the test data is calculated using Euclidean metric. The output in k-NN for classification problems is a class membership, which represents a prediction of whether the sample is ransomware or benign by a majority vote of the sample's neighbors.

6.2.5 XGBoost (XGB): It is an open-source library that has recently been used in many ML applications. It provides a high-performance implementation of gradient boosted DT to solve many data science issues in an accurate and fast way. Boosting involves training multiple poor DTs at successive steps to enhance the prediction. A poor DT model can only perform well on part of the training data, where multiple poor learners are combined selectively to produce a much powerful learning model. The learning_rate parameter is a hyperparameter that controls the changing in the weights updating of a model in response to the estimated error [3]. In XGBoost algorithm, the input is the number of gradient boosted trees (which set to 50 with n_estimators parameter) and the maximum depth of the tree (which set to 25 with max_depth parameter). Repeatedly, each tree makes its prediction and computes the error for the output variable, and these errors are then used to build the next tree. The output of XGBoost is computing through adding the new tree prediction to the predictions of the previous trees, then the final prediction result (ransomware or benign) is found.

6.2.6 Multi-Layer Perceptron (MLP): The Deep Learning MLP is a class of feedforward ANN. It utilizes backpropagation supervised learning technique for training. It consists at least of 3 layers of nodes (input, hidden, and output) with linear or nonlinear activation function. Each node in any layer is connected to all the nodes in the next layer (Fully-Connected Layer). The most important parameters are the hidden_layer_sizes which are define as the number of hidden layers with the number of its elements, and the activation parameter which is defined as the activation function for the hidden layer [3, 32]. In MLP algorithm, the input is defined by the nodes in input layer which equal to the features count, and the hidden layers (3 in this research) with different numbers of nodes via the parameter 'hidden_layer_sizes = (64, 128, 64)'. The activation function is 'relu' and the maximum number of iterations = 20. After several times of calculating the weighted sum, applying the activation function, and weights adjusting, the MLP output is found with one node in output layer which its result is (Ransomware or Benign).

6.3 Performance Metrics

To evaluate ML classifiers, seven metrics based on confusion matrix were used in this experiment, Confusion Matrix (CM) is an error matrix which is used in ML fields specifically in statistical classification issues. It is a table layout that shows an algorithm performance. Each column of CM refers to the instances in a predicted class, and each row in CM refers to instances in an actual class. CM is explained in Table IV:

Table IV. Confusion Matrix

Parameter		**Prediction**	
		Ransomware	Benign
Actual	Ransomware	True Positive (TP)	False Positive (FP)
	Benign	False Negative (FN)	True Negative (TN)

Where:

- TP: malware samples count that are correctly classified.
- TN: benign samples count which are correctly classified.
- FP: benign samples count which are incorrectly classified.
- FN: malware samples count which are incorrectly classified.

The Machine Learning metrics are:

- Accuracy (Auc): the total percentage of cases classified correctly, it is the correctly classified samples divided by all of the classifications

$$Auc = \frac{TP + TN}{TP + TN + FP + FN} \quad \dots \quad (2)$$

- Precision (Pr): It returns the rate of relevant results

$$Pr = \frac{TP}{TP + FP} \quad \dots \quad (3)$$

- Recall (Re), or True Positive Rate (TPR): the detection rate of malware instances that detected by the system

$$Re, TPR = \frac{TP}{P} = \frac{TP}{TP + FN} \quad \dots \quad (4)$$

- True Negative Rate (TNR): it's the ratio between correctly predicted negative and all negative samples, it is proportions of benign instances that correctly detect by the system.

$$TNR = \frac{TN}{N} = \frac{TN}{TN + FP} \quad \dots \quad (5)$$

- False Positive Rate (FPR): it is the percentage of misidentified malware

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN} \quad \dots \quad (6)$$

- F-score (F): it is a measure of a test's accuracy that evaluates the system performance by uniting both precision and recall in on value

$$F = 2 \times \frac{Pr \times Re}{Pr + Re} \quad \dots \quad (7)$$

- Classification Time (T): the required time to train the model

6.4 Results

The Table V presents the results of using ML methods for android ransomware detection.

Table V. Results of Performance Metrics for ML Classifiers

Num of	Decision Tree						
features	Auc	Pr	Re	TNR	FPR	F	Time
ALL	98.35	98.24	98.44	97.89	0.023	98.34	40.28
10_best	98.38	98.16	98.55	97.62	0.023	98.34	4.42
15_best	98.39	98.17	98.55	97.62	0.023	98.34	7.18
20_best	98.93	98.76	99.08	98.24	0.017	98.91	10.48
25_best	99.02	98.86	99.15	98.39	0.016	98.99	12.89
RF = 21	99.32	99.20	99.42	98.85	0.011	99.31	7.60
ET = 20	99.31	99.18	99.41	98.82	0.011	99.29	9.51
LR = 17	99.37	99.25	99.46	98.92	0.010	99.35	5.71
Num of	Random Forest						
features	Auc	Pr	Re	TNR	FPR	F	Time
ALL	95.83	95.48	96.16	92.21	0.057	94.76	164.95
10_best	93.69	93.26	94.16	91.46	0.078	93.58	70.61
15_best	93.99	93.58	94.41	91.95	0.075	93.88	71.07
20_best	94.67	94.27	95.08	92.36	0.073	94.33	77.74
25_best	94.74	94.30	95.89	92.63	0.071	94.57	87.22
RF = 21	96.97	96.22	97.35	95.03	0.048	96.91	75.07
ET = 20	97.02	96.63	97.45	95.11	0.047	97.04	71.98
LR = 17	95.99	95.63	96.36	94.20	0.056	95.91	68.22
Num of	Gradient Boosting XGB						
features	Auc	Pr	Re	TNR	FPR	F	Time
ALL	96.46	96.37	96.54	96.08	0.039	97.15	385.52
10_best	96.73	96.39	96.98	95.52	0.044	96.65	77.69
15_best	96.73	96.39	96.98	95.52	0.044	96.65	84.92
20_best	97.37	97.07	97.62	96.18	0.038	97.31	111.15
25_best	97.55	97.26	97.79	96.41	0.035	97.49	124.50
RF = 21	99.28	99.15	99.39	98.78	0.011	99.26	106.86
ET = 20	99.22	99.08	99.33	98.68	0.013	99.20	91.90
LR = 17	99.26	99.14	99.37	98.75	0.012	99.24	95.52
Num of	K-Nearest Neighbor						
features	Auc	Pr	Re	TNR	FPR	F	Time
ALL	97.13	96.92	97.55	95.41	0.0448	97.18	491.91
10_best	97.09	96.73	97.45	95.38	0.0461	97.02	384.23
15_best	97.09	96.73	97.45	95.38	0.0461	97.02	573.81
20_best	97.70	97.37	98.01	96.16	0.0383	97.64	745.41
25_best	97.70	97.38	98.02	96.16	0.0383	97.65	900.18
RF = 21	97.92	97.61	98.23	96.46	0.0353	97.87	177.95

ET = 20	97.89	97.59	98.19	96.40	0.0359	97.85	145.34
LR = 17	98.16	97.87	98.43	96.88	0.0311	98.12	115.78
Num of	**Multi-Layer Perceptron**						
features	**Auc**	**Pr**	**Re**	**TNR**	**FPR**	**F**	**Time**
ALL	58.30	79.14	50.03	65.45	0.0025	61.30	196.53
10_best	89.27	88.78	89.55	87.97	0.1202	89.06	99.13
15_best	89.27	88.78	89.55	87.97	0.1202	89.06	103.10
20_best	89.88	89.42	90.24	88.11	0.1188	89.70	145.75
25_best	89.95	89.49	90.28	88.63	0.1136	89.75	162.25
RF = 21	91.64	91.19	91.99	89.93	0.1006	91.47	128.48
ET = 20	91.30	90.93	91.93	88.18	0.1081	91.18	115.69
LR = 17	91.24	90.78	91.48	90.07	0.1092	91.05	106.63
Num of	**Logistic Regression**						
features	**Auc**	**Pr**	**Re**	**TNR**	**FPR**	**F**	**Time**
ALL	69.88	77.87	64.45	77.26	2.027	62.86	37.35
10_best	84.52	84.22	83.67	88.51	0.114	83.91	11.12
15_best	84.52	84.22	83.67	88.51	0.114	83.91	11.12
20_best	85.05	84.80	84.23	89.00	0.109	84.48	15.99
25_best	85.16	84.92	84.34	89.10	0.108	84.59	21.90
RF = 21	85.66	85.30	85.07	88.51	0.114	85.18	14.37
ET = 20	85.16	84.80	84.59	87.99	0.120	84.69	12.83
LR = 17	85.41	84.99	84.90	87.85	0.121	84.94	10.91

From Table V, the benefits of using feature selection techniques can be noted, especially where it selects the best number of the most important features, which contributed to improving the performance of the system in terms of high detection accuracy, precision, recall and f-score, and reducing the FPR rate. In addition, feature selection technique has reduced training time to a quarter of the required time to train all features, and the reason for this was the decrease of computations and processing operations.

Also, from Table V, it can be concluded that both DT and XGB classifiers are the best method used for ransomware detection with detection accuracy exceeding 99%. DT is faster than XGB when compared in terms of "classification time" metric because DT is a single tree that can process a large dataset in a short time. The XGB classifier provides exceedingly high accuracy and it prevents overfitting, but it is more difficult than DT and it requires more arithmetic operations (as with RF) and a lot of training time as it consists of several trees. k-NN classifier provided superior performance with a detection accuracy of more than 98%. Although the efficiency of k-NN depends on n_neighbors parameter, it requires more training time and more storage space because it is a lazy algorithm, k-NN does not learn data generalization at the training phase, it delays that to the testing phase. The detection accuracy for MLP classifier exceeded 91%, and the experiments proved that increasing the number of iterations in MLP leads to better results, but it takes more training time. The detection accuracy for LR was low (≈ 85%). The reasons for this are that LR is suitable to process linear problems and simple datasets with separable data (uncorrelated features), because LR has a linear decision surface.

In comparison with previous works, Table VI presents the detection accuracy for the proposed system and the works [7] and [16] which used the same dataset (the CICAndMal2017 dataset) for android ransomware detection. Also, the table shows a comparison with the work [15] which used the dataset with the same features that were extracted from network traffic for android ransomware detection.

Table VI. Comparison between the proposed system and previous works

	Works			
Methods	**[15]**	**[16]**	**[7]**	**The proposed system**
DT	97.10	-	77.70	**99.30**
RF	96.10	97.45	82.80	97.02
k-NN	95.30	-	74.76	**98.16**
XGB	-	-	-	**99.20**
MLP	95.20	-	-	91.64

From Table VI, it can be concluded that the proposed system gives a higher accuracy rate for ransomware detection (in bold font) than other works.

According to the experiments, the benefits of dynamic analysis for ransomware detection from its early phases of entering the device over the network become clear. Also, it can be concluded that using a number of network traffic features (e.g., 21 features by RF selector, 20 by ET selector ... etc.) is considered effective in distinguishing ransomware from benign. The proposed system gives high accuracy (more than 99%) for both DT and XGB classifiers, and low FPR (0.010) and (0.011) for DT and XGB respectively. So, it can be stated that the proposed ML methods are effective for Android ransomware detection when applied to network traffic features. Thus, it can be adopted as a method for detecting ransomware applications on smartphone.

7. Conclusion and Future works

Some attackers can recognize patterns of malicious and benign applications and then attempts to simulate a specific class of traffic (for example the duration of the flow) or use fake IP address, but it is impossible to change the variance of some or all features for evading the detection system as in the proposed system. For this reason, it is important to highlight that this research differs from other works that utilize one category of network traffic. This research focused on selecting the best features from the total feature set. The analysis results of the experiments show that the features of network traffic are very suitable when utilizing them to detect the ransomware, as the dataset used in this analysis has been extracted from online network traffic. Several techniques were used to select best features, and six ML algorithms were applied for android ransomware detection. Finally, seven performance metrics were used to evaluate ML classifiers. The results showed that the average detection accuracy was (more than 99%) for DT and XGB, and FPR is (0.016% and 0.029%) for DT and XGB respectively. For future work, the suggestion is using network traffic features for detection and classification of other android malware types. Also, the aim is to choose other types of features (e.g., utilizing memory dump, permission, logs, or API calls) to develop a widespread Android ransomware detection framework.

8. Acknowledgements

The authors of this research would like to thank the University of Mosul / College of Computer Science and Mathematics for their facilities, support, and guidance throughout the project, which have helped to enhance the quality of this work. The authors would also like to thank their friends and families for believing in them and continuously supporting them.

9. References

[1] A. Bhattacharya, R. T. Goswami, “Community Based Feature Selection Method for Detection of Android Malware,” *Journal of Global Information Management (JGIM)*, 2018.

[2] M. W. Azeem, M. S. Sarfraz, U. Shoaib, "Comparison of Different Techniques for Detecting Malware in Smartphones," *International Journal of Computer Science and Information Security 14.5*. (2016).

[3] C. Chio, D. Freeman, "Machine learning and security: Protecting systems with data and algorithms," *O'Reilly Media, Inc.*, 2018.

[4] D. Maiorca, et al. "R-PackDroid: API package-based characterization and detection of mobile ransomware," *Proceedings of the symposium on applied computing*, 2017.

[5] Cyber Security Services in Washington, DC. *The 2021 CYBER SECURITY STATISTICS, DATA, & TRENDS*. Available: https://purplesec.us/cyber-security-trends-2021/

[6] CISCO. *The Cisco Annual Internet Report (2018–2023) White Paper*. March 9, 2020. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[7] F. Noorbehbahani, F. Rasouli, M. Saberi, "Analysis of machine learning techniques for ransomware detection," *2019 16th International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology (ISCISC), IEEE*, 2019.

[8] M. K. Alzaylaee, S. Y. Yerima, S. Sezer, "DL-Droid: Deep learning based android malware detection using real devices," *Computers & Security 89*, 2020.

[9] J. Saxe, H. Sanders, "Malware Data Science: Attack Detection and Attribution," No Starch Press, 2018.

[10] B. A. Al-rimy, M. A. Maarof, S. Z. Shaid, "Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions," *Computers & Security*, 2018.

[11] Jessica Ellis. December 15, 2020. *Year In Review: Ransomware*. Available: https://securityboulevard.com/2020/12/year-in-review-ransomware/

[12] G. Cusack, O. Michel, E. Keller, "Machine learning-based detection of ransomware using SDN," *Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, 2018.

[13] Z. G. Chen, H. S. Kang, S. N. Yin, S. R. Kim, "Automatic Ransomware Detection and Analysis Based on Dynamic API Calls Flow Graph," *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*, 2017.

[14] H. Zhang, X. Xiao, F. Mercaldo, S. Ni, "Classification of ransomware families with machine learning based on N-gram of opcodes," *Future Generation Computer Systems*, 2018.

[15] O. M. Alhawi, J. Baldwin, A. Dehghantanha, "Leveraging machine learning techniques for windows ransomware network traffic detection," *Cyber Threat Intelligence. Springer, Cham*, 2018.

[16] A. O. Almashhadani, M. Kaiiali, "A multi-classifier network-based crypto ransomware detection system: A case study of locky ransomware," *IEEE Access*, 2019.

[17] R. Moussaileb, N. Cuppens, J. L. Lanet, H. Bouder, "Ransomware Network Traffic Analysis for Pre-encryption Alert," *International Symposium on Foundations and Practice of Security. Springer, Cham*, 2019.

[18] A. Sangal, H. K. Verma, "A Static Feature Selection-based Android Malware Detection Using Machine Learning Techniques," *2020 International Conference on Smart Electronics and Communication (ICOSEC). IEEE*, 2020.

[19] A. Mahindru, A.L. Sangal, "DLDroid: Feature Selection based Malware Detection Framework for Android Apps developed during COVID-19," *International Journal on Emerging Technologies*, 2020.

[20] D. Gibert, C. Mateu, J. Planes, "The rise of machine learning for detection and classification of malware: Research developments, trends and challenges," *Journal of Network and Computer Applications 153*, 2020.

[21] S. J. Lee, H. Y. Shim, et al. "Study on Systematic Ransomware Detection Techniques," *2021 23rd International Conference on Advanced Communication Technology (ICACT). IEEE*, 2021.

[22] Canadian Institute for Cybersecurity. *Canada's research leader in cybersecurity*. accessed by Jan 2020. Available: https://www.unb.ca/cic/

[23] The CICFlowMeter packet capture tool. Available: https://www.unb.ca/cic/research/applications.html#CICFlowMeter

[24] Virustotal website for security services. (2020). Available: https://www.virustotal.com/en

[25] Abdullah, Mohammed Hamid Abdulraheem. *Designing Deep Learning Based Network Intrusion Detection System for Software Defined Network*. Diss. University of Mosul, 2020.

[26] Scikit learn. *Variance threshold technique for feature selection, by scikit learning*. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.VarianceThreshold.html

[27] M. Kuhn, K. Johnson, "Feature engineering and selection: A practical approach for predictive models," *CRC Press*, 2019.

[28] Jason Brownlee. *Machine Learning Mastery. The feature selection methods*. November 27, 2019. Available: https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/

[29] Scikit learn. f_classif technique for feature selection, by scikit learning. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_classif.html

[30] Scikit learn. *k_best technique for feature selection*. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html

[31] Scikit learn. Select from model technique for feature selection, by scikit learning. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectFromModel.html

[32] F. A. Narudin, A. Feizollah, N. B. Anuar, A. Gani, "Evaluation of machine learning classifiers for mobile malware detection," *Soft Computing*, 2016.