Malware Dataset Generation and Evaluation

1st Parthajit Borah
Dept. of CSE
Tezpur University
Tezpur, Assam, India
parthajit@tezu.ernet.in

2nd DK Bhattacharyya
Dept. of CSE
Tezpur University
Tezpur, Assam, India
dkb@tezu.ernet.in

3rd JK Kalita
Dept. of CSE
University of Colorado, Colorado Springs
Colorado, USA
jkalita@uccs.edu

***Abstract*—With the rapid growth of technology and IT-enabled services, the potential damage caused by malware is increasing rapidly. A large number of detection methods have been proposed to arrest the growth of malware attacks. The performance of these detection methods is usually established using raw or feature datasets. The non-availability of adequate datasets often becomes a bottleneck in malware research. To address this issue, this paper presents two malware feature datasets on two different platforms to support validation of the effectiveness of a malware detection method. We evaluate the usefulness of our datasets in a supervised framework.**

***Keywords*: Malware, signature, detection, classfication, static, dynamic**

I. Introducion

The Internet is a complex and global networked system which interconnects various netwroks of networks. The Internet has developed rapidly in the past twenty-five years and is now regarded as a vital organ undergirding the economy of the world. With rapid advances in technology, more than one third of the world's population has now entered the digital world. The exponential growth of the Internet is mainly because of the proliferation of new sophisticated Internet-enabled devices, rapid growth in number of ISPs and increasing affordability of Internet services across the globe. Today, a wide range of devices connected to internet such as cell phones, tablets, phablets, laptops, and netbooks provide hassle free connectivity.

The growth of technology and its services has changed the lifestyles of people. Now, people make use of services available on top of the Internet for numerous purposes. Unfortunately, these services can be exploited by enemies and criminals to fulfill nefarious. They can launch various types of attacks to sweep into interconnected systems by exploiting the vulnerabilities found in different devices and software applications. As a result, they can disrupt ongoing services, steal sensitive information, corrupt important files and exhaust the resources of servers. To counter such attacks, various defence mechanisms have been proposed in recent times. It is also important to note here that over the years, various patterns of malware have been introduced from time to time. So, a detection system is required to adapt itself to detect even the most recent malware attacks.

978-0-7381-2447-6/20/$31.00 © 2020 IEEE

A. Malware and its significance

Any program that *deliberately fulfills the harmful intent of an attacker* is commonly referred to as malicious software or malware [1]. Malware plays an important role in any cyber attack. Such a program is designed by the attacker to create disturbances over the Internet. Malware are used to gain access to different interconnected systems and network resources to steal important information, files and disrupt normal operations without the user's knowledge. Malware programs that have been observed in the wild come in various forms. The most common types of malware are Adaware, Backdoor, Bot, Downloader, Ransomware, Rootkit, Trojan, Virus and Worm.

B. Motivation

To counter malware attacks, development of malware defense systems is on the rise. However, to measure the efficiency and effectiveness of such defense systems, we need featured malware datasets. At this time, only a very few such featured malware datasets are publicly available [2] [3] [4] [5]. It is also necessarily significant to update the malware characteristics in the dataset in accordance with the evolving behaviour patterns of new malware. Otherwise, the defense system trained on an outdated dataset can easily be evaded. Such outdated datasets are never sufficient to evaluate the performance of defense systems. Therefore, it is necessary to generate unbiased datasets of malware from time to time to ensure accurate evaluation of malware detection systems.

C. Contribution

The primary goal of this paper is to report the development of two featured malware datasets for two different platforms, namely Windows and Android. To do so, we develop two different frameworks for dataset creation. Each framework has several phases and modules that work collectively to generate the two datasets, which are refered to as TUMALWD (Tezpur University Windows MALware Dataset) and TUANDROMD (Tezpur University ANDROid Malware Dataset). This paper makes the following contributions.

1) Development of two dataset creation frameworks for the Windows and the Android malware, respectively.
2) Feature extration from malware analysis reports towards creation of two malware feature datasets: TUMALWD and TUANDROMD.

3) Evaluation of the usefulness of the datasets in a supervised framework.

The rest of the paper is organized as follows. Section 2 presents background. In section 3, we discuss the TUMALWD dataset creation framework and its characteristics. Section 4 discusses the TUANDROMD dataset generation framework and its characteristics. In Section 5, we present the performance evaluation of our datasets using several classifiers. Lastly, in Section 6 we wind up with the conclusion.

II. Background

Malware plays an important role in all kinds of network intrusions and security attacks. Any software that disrupts user data, computers or networks can be considered malware, including viruses, trojans, worms, rootkits, scareware, and spyware. The concept of a malicious program is not new. The theoretical idea behind malware was introduced by Jon von Neumann in his article *"Theory of self-reproducing automata"* [6]. Till the early 80s, malware was created with benign intent. Most malware was written just for fun or to try out experiments or just to annoy others. Starting from the late 80s, malware have been created with malicious intent, and since then incidents of malware attacks have grown tremendously, and in the present era cyber threats have evolved to a feverish level, hazardously infecting systems and platforms which were not known to be vulnerable earlier. With each passing day, the structures and methods of cyber-attacks are becoming more complex, working with increasing stealth and frequencies of attack. These include clickless threats, and Internet-of-Things (IoT) attacks. For example, the Mirai[1] malware was created to infect Internet-of-Things (IoT) devices like thermostats, webcams, home security systems and routers.

A. Representation of Malware and Their behaviour

To study the functionality of malware, we must first understand the kinds of data or file types used by the malware to launch attacks. File attachments in email are common threat vectors for malware. Executable files are not the only file type that can include threats. For example, Microsoft Office documents (e.g. doc, docx, xlx, xlsx, ppt, pptx, etc) can contain macros or scripts that include threats. The data associated with malware can be of two types, Raw data and Feature level data, both of which are described below.

1) Raw Data: Data that has not been processed for use is called raw data. In our case, different Windows file types constitute raw data. The most common Windows file types that are used by malware can be found in [2].
   - Malware File: Executable files are widely used by malware to infect victim systems. There are many types of executable files, scripts, programs and malicious shortcuts that can be manipulated. However, this method has become outdated since most e-mail providers and firewalls block these kind of attachments. Another particular type of files, i.e., Microsoft Office files, have been very popular and effective in infecting victims. The primary reason for this is the usage of malicious macros that are embedded within the documents themselves. This makes them bypass any antivirus software and e-mail attachment protection software.
   - Malware in execution: To invade the victim machine, different malware use different transport mechanisms. These mechanisms enable malware propagation among information devices and systems, including mobile devices it seeks to infect. Attackers often employ techniques of social engineering to distribute malware over the Internet. They use different elements of social engineering to lure victims to click on malicious links or to launch an infected file or to open a malicious email attachment. For example, the LoveLetter [3] worm uses the social engineering tactics to propagate and compromise the victim's system.
2) Feature Level data: Features are characteristics that are extracted from raw data because they are relevant for a problem at hand. Features or attributes or characteristics have a significant role to play because they affect the data's complexity, readability and functionality. Following are the categories of features in our case.
   - Host based features: Host based features can be extracted from the malware or benign programs present in the system logs or the malware analysis reports. These features enumerate the activities that the applications can perform in order to interact with the system.
   - Network traffic features: Network traffic features are extracted from network traffic logs that are generated by programs under analysis. These features help distinguish between malware and benign programs.

III. The Proposed Dataset Creation Framework

The dataset creation framework describes the process and the computing environment need to create the dataset, which we call TUMALWD. The preparation framework handles three phases, namely data collection and storage, data analysis, and feature engineering. Each phase has several components as described below. In Phase 1, data are collected and stored. The Second Phase is responsible for data analysis. In Phase 3, we get the dataset TUMALWD as output.

A. Phase 1: Data collection and Storage

For collecting malware, an automated honeynet system is set up to automatically store the collected malware binaries in a centralized server. The whole system is based on a client-server architecture where the client module is responsible for data collection and the server module is used for data storage.

[1]https://github.com/jgamblin/Mirai-Source-Code

[2]https://www.howtogeek.com/137270/50-file-extensions-that-are-potentially-dangerous-on-windows/

[3]http://virus.wikidot.com/loveletter

A generic architecture of the data collection framework is shown in the Figure 1.

1) *Honeynet Client*: On the client side, two virtual honeypots are set up and configured in a workstation with Linux based OS using a Virtual box. This virtualization offers an emulated environment and also mimics the operating systems services and ports. The honeypots used in the system are run on the Windows platform. As shown in the figure, a honeywall is used as gateway for the honeypots. This allows for complete control of all the inbound and outbound traffic. During the operation period of five moths i.e., August 2019 to December 2019, we collected 4000 malware binaries.
2) *Honeynet Server*: This module stores all the malware binaries and network traffic files collected at the honeynet clients. Further, all the collected binaries are transfered to an Analysis Server for further processing and analysis.

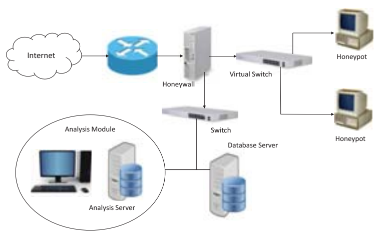
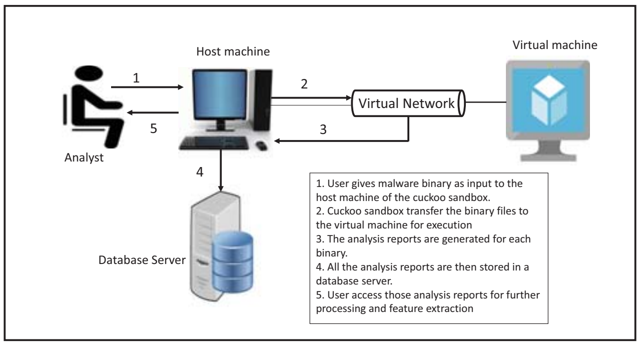


Fig. 1. Testbed Architecture for malware collection

B. Phase 2: Data Analysis

After the Data collection and Storage phase, the collected malware binaries are transfered to the Data Analysis phase where thorough malware analysis is conducted. To accomplish the tasks of Phase 2, two modules come into action, namely Analysis Client and Analysis Server respectively responsible for malware analysis and storage of all analysis reports in a database server for further processing and feature extraction. Steps followed to accomplish the tasks in Phase 2 are shown in figure 2.

1) *Analysis Client* : This module receives the input from the Honeynet Server module in the form of binary programs. For the analysis of the collected malware binaries, we set up and configured an open source malware analysis tool called the Cuckoo Sandbox [4]. The Cuckoo Sandbox is an automated system which can analyze different types of files in a realistic but isolated environment. Initially, we installed the Cuckoo Sandbox in a worksation where Ubuntu 16 is used as the host operating system. For the creation of an isolated environment, we used Virtual Box where Winodws7 is installed as a guest operating system to carry out all analysis tasks. For the analysis, the Cuckoo Sandbox is fed with malware binaries and in return, it genertaes a complete behavioral report for each malware binary. In addition to Cuckoo Sandbox, we also installed an API monitor [5] in the guest operating system to monitor and log the API calls made by malware applications.
2) *Analysis Server* : This component receives all the analysis reports as input from the *Analysis Client* component and stores them in a database server for further processing and feature extraction. It also receives all log files of the API monitor as input and stores them in the database server.



1. User gives malware binary as input to the host machine of the cuckoo sandbox.
2. Cuckoo sandbox transfer the binary files to the virtual machine for execution
3. The analysis reports are generated for each binary.
4. All the analysis reports are then stored in a database server.
5. User access those analysis reports for further processing and feature extraction

Fig. 2. Steps in data analysis phase

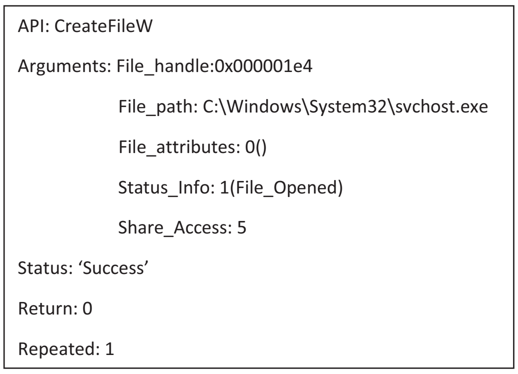
C. Phase 3: Feature Engineering

Phase 3 is the final and main part of our dataset generation framework. To accomplish the tasks of Phase 3, two components come into action, namely preprocessing and feature extraction.

1) *Preprocessing*: In the preprocessing component, we discard all the analysis report whose content is empty, i.e., those malware binaries that failed to execute during analysis. For our conveninece and to carry forward the tasks of preprocessing, we extract meaningful contents from the analysis reports and stores them in a separate file. To accomplish this task, we wrote C programs to extract information from the API calls for which the status value is success. In other words, our programs takes the text file of the analysis report as an input and searches for the status value of each API block. Once it encounters the status tag and if its value is "SUCCESS", it writes all the API call information to an another output file and this process is goes on until the end of the file. Figure 3 is an example of API call information of a binary file obtained during analysis. As seen in the figure, the binary file successfully created a file in the system during execution.
2) *Feature Extraction* : In the Feature Extraction phase, two categories of features, namely, host based and network based features, are extracted from the preprocessed analyis reports. Host based features contain three feature sets

[4]https://www.cuckoosandbox.org/

[5]https://www.apimonitor.com/



```
API: CreateFileW
Arguments: File_handle:0x000001e4
              File_path: C:\Windows\System32\svchost.exe
              File_attributes: 0()
              Status_Info: 1(File_Opened)
              Share_Access: 5
Status: 'Success'
Return: 0
Repeated: 1
```

Fig. 3. Example of API call information

where the first set contains the API call sequences, the senond set comprises the successful API call status and the third set contains the frequency of each API call. On the other hand, network based feature set contain all the flow information of malware binary applications.

D. TUMALWD: The dataset and its characteristics

The processes and phases involved in the dataset creation framework have been discussed in detail in the previous sections. The output of the framework is the dataset TUMALWD comprising of two feature sets, namely host based and network based features. Host based and network based features are for the system based and the network based indicators, respectively.

1) Host based features:: For host based features, we consider Windows API functionality that helps distinguish between malware and benign programs. We extract all critical API functions that malware used successfully in order to interact with the operating system's libraries. These features are extracted from the API call information. The list of top ranked feature categories are enlisted in Table I. The ranks are calculated using three feature selectors [7] [8] [9].

TABLE I
LIST OF TOP RANKED FEATURE CATEGORIES FOR TUMALWD

Feature Rank	Feature Category
1	$RegistryKeysOperations$
2	$ProcessCalls$
3	$Strings$
4	$Filesoperations$
5	$Fileextensions$
6	$Directoryoperations$
7	$Droppedfiles$

2) Network traffic features:: To extract network level features, a malware program needs to perform activities over the network during execution. The network activities of malware are captured by executing them in the Cuckoo sandbox. Additionally, we collect network traces that are captured in the honeynet. The traces are analyzed using an open source tool called Wireshark. The raw traces are then preprocessed and filtered to extract different features. We extract 27 different flow-based features from the traces. To extract these features we wrtie Python scripts and also use an open source tool called Flowtbag [6]. These features can be specifically classified into basic features, time-based and byte-based features.

1) Basic features: Unlike the standalone Malware, some variations perform malicous activities based on the command they receive from a master program. Consequently, the infected system needs to establish a connection to the master machine. Basic network features help identify if the compromised victim system performs any malicious activity over the network. For example, if the victim system tries to establish an unwanted connection with a blacklisted website, it can easily be detected based on the destination IP of the blacklisted site.
2) Time-based features: Time-based features help distinguish malicious network connections from legitimate ones. For example, the duration of the flow, i.e., the start and end time of the flow can play a useful role in identifying malware compromised hosts as most follow similar communication patterns with their masters, and the respective durations are almost similar.
3) Byte-based features: By monitoring the amount of data exchange between two hosts, it possible to infer whether the communication link is malicious or not.

TABLE II
NETWORK LEVEL FEATURES

Category	Feature name	Feature description
Basic features	Src IP	Source IP
	Src port	Source port
	Dst IP	Destination IP
	Dst port	Destination port no
	Total_pkts	Total packets in the forward direction
	Proto	The protocol
Byte based features	total_fvolume	Total bytes in the forward direction
	total_bvolume	Total bytes in the backward direction
	min_fpktl	Smallest packet size (forward direction)
	mean_fpktl	Average size of the packets (forward direction)
	max_fpktl	Maximum size of the packets (forward direction)
	min_bpktl	Least packet size (backward direction)
	mean_bpktl	Average size of the packets (backward direction)
	max_bpktl	Maximum size of the packets (backward direction)
Time based features	Duration	The flow duration
	min_fiat	The least amount of times b/w two packets (forward direction).
	mean_fiat	The average amount of times b/w two packets (forward direction).
	max_fiat	The highest amount of times b/w two packets (forward direction).
	std_fiat	The standard deviation from the mean amount of times b/w two packets sent in the forward direction
	min_biat	The least amount of times b/w two packets (backward direction)
	mean biat	The average amount of times b/w two packets (backward direction)
	max_biat	The max amount of times b/w two packets sent in the backward direction
	std_biat	The standard deviation from the average amount of times b/w two packets (backward direction)
	min_active	The least amount of time of the active flow
	mean_active	The average amount of time of the active flow
	max_active	The max amount of time of the active flow
	std_active	The standard deviation of the active flow

3) Characteristics of TUMALWD: As discussed in Section 3, the following are the particular characteristics of the TUMALWD dataset that we create.

[6]https://github.com/DanielArndt/flowtbag

(a) *Labels in the data:* The TUMALWD dataset has two labels, namely benign and malware. The benign instances make up the benign class, while the malcious instances make up the malware class.

(b) *Number of instances:* There are a total 5400 instances for the host-based features, of which 4000 instances belong to the malware class and the rest 1400 instances belong to the benign class. For network-based features, there are a total 3000 instances, of which, 1600 instances are malware and the rest 1400 instances are benign.

(c) *Number of features:* For host-based features, all the successful API calls are extracted as features. A total of 7545 calls are extracted. Additionally, 27 network based features are extracted.

(d) *Balance between the classes:* The dataset is not perfectly balanced i.e., the dataset does not have an equal number of instances for both classes. This slight imblance can be easily handled by collecting more instances or sampling techniques. The proposed dataset creation framework can handle such adaptaions.

(e) *Recency:* The data collected for creation of TUMALWD are recent (dated to late 2019). As and when new malicious instances are available, the dataset can be updated accordingly.

(f) *Relevance:* The extracted features helps distinguish between malware and benign programs. To the best of our knowledge, these features are relevant differentiating between the two.

IV. TUANDROMD: The Proposed Android Malware Dataset Creation Framework

The dataset creation framework describes the whole preparation process for the created dataset: TUANDROMD. The preparation framework is divided into three phases, namely data collection, data analysis and feature engineering. In Phase 1, benign and malware Android applications are collected for further analysis. In the second phase, data analysis is carried out on the data collected in Phase 1. Finally, Phase 3 takes output of Phase 2 as input and performs feature engineering to ouput the final dataset, which we call TUANDROMD.

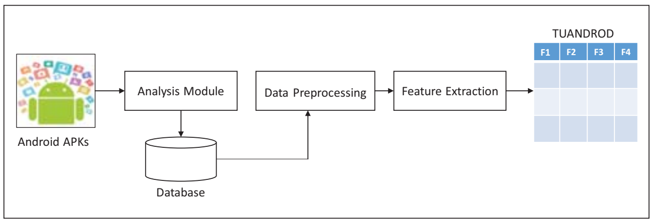


Fig. 4. TUANDROMD: Dataset creation framework

A. Phase 1: Data collection and Storage

For developing a featured dataset of Android malware, we need raw malware and benign applicatons. For this, we collected 24,553 raw malware binaries from [10]. These malware binaries are further categorized in 135 varieties among 71 malware families. Additionally, we gathered top 1000 Android applications from Google Play as benign applications. Finally, all the collected Android applications are stored in a database server for further processing and analysis.

B. Phase 2: Data analysis

Unlike PC based malware analysis, we have performed manual analysis on the collected Android malware using a set of tools–Androguard[7], ApkInspector[8], ApkAnalyser[9] and Smali-CFGs[10]. For analysis, we consider each variant of malware in each family. All analysis tasks are carried out in an isolated environment. For the creation of isolated environment, we used Virtual Box where Winodws7 is installed as a guest operating system in which all the analysis tasks are carried out.

C. Phase 3: Feature Engineering

In the feature engineering phase, two categories of features namely permission based and API based features are extracted from the analyis results. For permission based features, we extract all the install time and runtime permissions required for an android application to run smoothly on the device. For API based features, we extract all the APIs used by an application to carry out all the tasks for the users.

D. TUANDROMD: The proposed dataset and its characteristics

The processes and phases involved in the dataset creation framework has been discussed in details in the previous sections. The output of the framework is the dataset TUANDROMD comprising of two feature sets namely permission based and API based features.

1) Permission based features:
   Android has a built in defense system which is termed as permission based system. This system defines a set of actions that an app is allowed or not allowed to perform. Those permission that requires users confirmation can be exploited by the attackers to harm or compromise the system. These permission based features helps to distinguish between malicious and benign app. These features are extracted from our analysis report using several C routines. A total of 178 features are extracted out of which 15 top most features are enlisted in table III.
2) API based features:
   The Android platform provides a framework API which consists of a core set of packages and classes. Since most of the applications are dependent on large number of API calls, it seems all the more practical and sensible to use API calls of each application as feature to characterize and differentiate malware from benign applications.

[7]https://github.com/androguard/androguard
[8]https://github.com/honeynet/apkinspector/
[9]https://github.com/sonyxperiadev/ApkAnalyser
[10]https://github.com/EugenioDelfa/Smali-CFGs

TABLE III
LIST OF TOP RANKED FEATURES FOR TUANDROMD

Feature Rank	Feature Name
1	*SEND_SMS*
2	*RECEIVE_BOOT_COMPLETED*
3	*GET_TASKS*
4	$Ljava/net/URL; -> openConnection$
5	*VIBRATE*
6	*WAKE_LOCK*
7	*KILL_BACKGROUND_PROCESSES*
8	*SYSTEM_ALERT_WINDOW*
9	*ACCESS_WIFI_STATE*
10	*DISABLE_KEYGUARD*
11	$Landroid/location/LocationManager;$ $-> getLastKnownLocation$
12	*READ_PHONE_STATE*
13	*RECEIVE_SMS*
14	*CHANGE_WIFI_STATE*
15	*WRITE_EXTERNAL_STORAGE*

These features are also extracted from the analysis reports using several C routines. A total 310 unique API calls are extracted.

1) Characteristics of TUANDROMD: As discussed in Section 3, the following are the particular characteristics of the TUANDROMD dataset that we create.

(a) *Labels in the data:* The TUANDROMD dataset has 72 labels where 71 labels represents the whole malware family and the remaining one label belong to the normal class.
(b) *Number of instances:* There are a total 25,553 instances for both the permission and API-based features, of which 24,553 instances belong to the malware class and the rest 1000 instances belong to the benign class.
(c) *Number of features:* For permission-based features, all the permissions used by the applications are extracted as features. A total of 178 features are extracted. Similarly, for API based features, a total of 186 features are extracted.
(d) *Balance between the classes:* The dataset is not perfectly balanced i.e., the dataset does not have an equal number of instances for both classes. This slight imblance can be easily handled by collecting more instances or sampling techniques. The proposed dataset creation framework can handle such adaptaions.
(e) *Recency:* The data collected for creation of TUANDROMD are recent. As and when new malicious and normal android applications are available, the dataset can be updated accordingly.
(f) *Relevance:* The extracted features helps distinguish between malware and benign android applications. To the best of our knowledge, these features are relevant differentiating between the two.

V. PERFORMANCE EVALUATION AND VALIDATION

For evaluating the performance and also to form a benchmark on these datasets, we used a total of five classifiers. For each classifier, the K-Fold cross validation is used where the value of K=10. The classification results are enlisted in table IV. The reason we use classification algorithms for evaluation in order to assess the effectiveness of the datasets so that machine learning based malware defense system can be developed using these datasets.

TABLE IV
BENCHMARK ON TUMALWD AND TUANDROMD

Classifier	Test Accuracy	
	TUMALWD	TUANDROMD
Random Forest	98.4	98.7
Extra Tree	97.6	98.8
Ada Boost	98.5	97.9
Xg Boost	97.8	97.8
Gradient boosting	96.3	97.4

VI. CONCLUSION

In this paper, we discuss about the importance of malware datasets and its role in evaluating malware detection systems. For the proper assesment of malware detection system, we also introduce two new malware datasets for two different platforms: Windows and Android. For windows platform, we have created the TUMALWD dataset based on host level and network level features and for android platform, we have created the TUANDROMD dataset based on permission and API based features.

REFERENCES

[1] A. Moser, C. Kruegel, and E. Kirda, "Exploring multiple execution paths for malware analysis," in *Security and Privacy, 2007. SP'07. IEEE Symposium on*. IEEE, 2007, pp. 231–245.
[2] A. Sami, B. Yadegari, H. Rahimi, N. Peiravian, S. Hashemi, and A. Hamze, "Malware detection based on mining api calls," in *Proceedings of the 2010 ACM symposium on applied computing*. ACM, 2010, pp. 1020–1025.
[3] "Malicious software datasets," http://csmining.org/index.php/malicious-software-datasets-.html.
[4] A. H. Lashkari, A. F. A. Kadir, L. Taheri, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark android malware datasets and classification," in *2018 International Carnahan Conference on Security Technology (ICCST)*. IEEE, 2018, pp. 1–7.
[5] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, and D. Garant, "Botnet detection based on traffic behavior analysis and flow intervals," *Computers & Security*, vol. 39, pp. 2–16, 2013.
[6] J. Von Neumann, A. W. Burks *et al.*, "Theory of self-reproducing automata," *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 3–14, 1966.
[7] F. Fleuret, "Fast binary feature selection with conditional mutual information," *Journal of Machine Learning Research*, vol. 5, no. Nov, pp. 1531–1555, 2004.
[8] R. Battiti, "Using mutual information for selecting features in supervised neural net learning," *IEEE Transactions on neural networks*, vol. 5, no. 4, pp. 537–550, 1994.
[9] M. Robnik-Šikonja and I. Kononenko, "Theoretical and empirical analysis of relieff and rrelieff," *Machine learning*, vol. 53, no. 1-2, pp. 23–69, 2003.
[10] F. Wei, Y. Li, S. Roy, X. Ou, and W. Zhou, "Deep ground truth analysis of current android malware," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA'17)*. Bonn, Germany: Springer, 2017, pp. 252–276.