Familial Classification of Android Malware using Hybrid Analysis

1st Omer Faruk Turan Cavli
WISE Lab., Department of Computer Engineering
Hacettepe University
Ankara, Turkey
turan.cavli@hacettepe.edu.tr

2nd Sevil Sen
WISE Lab., Department of Computer Engineering
Hacettepe University
Ankara, Turkey
ssen@cs.hacettepe.edu.tr

***Abstract*—With the developments in mobile and wireless technology, mobile devices have become important part of our lives. While Android is the leading operating system in the market share, it is also the most targeted platform by attackers. While there have been many solutions proposed for detection of Android malware in the literature, the family classification of detected malicious applications becomes important, especially where the number of mobile malware variants increases every day in the market. In this study, a solution based on machine learning and hybrid analysis is proposed for the Android malware familial classification problem. An extensive feature set including network-related features and activity bigrams is proposed. The effective static and dynamic analysis features are studied thoroughly and evaluated on Malgenome [1], Drebin [2], and UpDroid [3] datasets.**

***Index Terms*—Android, mobile security, malware analysis and detection, malware family classification, machine learning, static/dynamic analysis, hybrid analysis**

I. Introduction

According to the research of International Data Corporation (IDC), Android has constituted the biggest share of mobile market with 86.1% in 2019 and this share is predicted to increase in the following years [4]. Hence it becomes the most targeted mobile platform by attackers. Malicious applications can harm end users in many ways such as stealing their private data, spamming, sending sms to premium rate numbers. Nowadays, the main motivation of attackers are financial gain.

Every day new mobile malware or new variations of existing malware are introduced by malicious attackers and uploaded to market stores. According to the McAfee Mobile Threat Report Q1 2020, the number of mobile malware increases rapidly every year. It is also stated that evasion techniques such as obfuscation, updating in order to bypass security solutions are used more than before [5]. According to another report published by Kaspersky [6], the number of 1,152,662 mobile malware in the first quarter of 2020 increased by 93,232 and reached 1,245,894 in the second quarter of 2020. Because of increasing number of mobile malware that bypass analysis methods, hybrid analysis techniques become more important for malware detection and familial classification, since an evasion technique that evades one type of analysis could be caught by another type of analysis.

There are two main malware analysis and detection techniques in the literature: static and dynamic analysis. In static analysis, a suspicious application is analyzed without running it on any device and only the apk file is analyzed. Source code and other components of the application such as manifest file are analyzed. On one hand, since the application does not need to be run on any devices, static analysis is usually more efficient than dynamic analysis. On the other hand, attackers could easily bypass static analysis by using evasive techniques such as obfuscation, dynamic code loading and encryption. In dynamic analysis, suspicious application is run on a virtual or real device for the purpose of monitoring run time behaviours of the application such as API/system calls, network traffic, file operations. While dynamic analysis is robust to obfuscation techniques, the main problem here is to trigger malicious behaviour during the analysis, which is carried out for a limited time. Attacker could bypass dynamic analysis by waiting for a specific time or an action in order to run the malicious code. Moreover, if the attacker understands that it is run in an emulator, it may not trigger the malicious code. Hybrid analysis aims to take advantage of both techniques.

As a result of the arms race between attackers and security solutions, while attackers always develop their techniques in order to bypass security mechanisms, in a similar way, security developers improve their solutions against new malware. Besides new malware, attackers commonly introduce new variants of existing malware. Therefore, not only detection of malicious applications, but also their familial classification has become significantly important. When a malicious application is detected, if its family is identified, its effect on the device could be minimized with taking predetermined actions. Moreover, the familial classification helps to reduce the manual analysis time of malware.

In this study, a new multi-class classification algorithm is proposed in order to group malware into their families. Although we have applied well-known machine learning approaches such as k-NN, SVM, Random Forest and Decision Tree algorithms, the main contribution of the study is to extensively analyze effective static and dynamic features on mobile malware familial classification. Few studies in the literature, namely UpDroid [3] and Ec2 [7], have also employed hybrid analysis for malware family classification. In this study, in addition to the hybrid features that are employed in those previous studies, new feature groups such as network-related features, activity bigrams are firstly employed for malware family classification and thoroughly analyzed. We focus on update attacks that are emphasized in UpDroid [3] study. Besides UpDroid, the proposed features are evaluated on Malgenome [1] and Drebin [2] datasets. It is shown that even only using network-related features can be very effective for familial classification on the UpDroid [3] dataset and high accuracy is obtained. In general, newly added features have positive effects on malware familial classification and has shown to produce better results than Ec2 [7] and UpDroid [3].

The rest of the paper is organized as follows: Section II summarizes the related approaches in the literature. The proposed approach, particularly on how features are selected and extracted, are given in details in Section III. In Section IV, the datasets used in the experiments are presented, and the experimental results are discussed. The effect of each feature group on malware family classification is extensively analyzed. Finally, the study is concluded in Section V.

II. Related Work

Machine learning-based techniques have already been proposed to classify Android malware by using static, dynamic or hybrid approaches. However, most of these approaches employ either only static features or only dynamic features. There are also a few study that employ hybrid features. In this section, these approaches are reviewed.

Malgenome [1] categorizes Android malware under 4 groups according to methods they applied: repackaging, update attack, drive-by download and standalone. They have collected 1260 Android malware sample belonging to 49 different family from different app markets. They analyzed these 1260 Android malware samples manually over a year. In Drebin [2], machine learning algorithms were used for classifying Android malware into families. They have used only static features because of the limited resources for dynamic analysis. The feature vector that they have used consisted of Hardware Components, Permissions, Intents, Activities, Services, Content Providers and Broadcast Receivers that was given in AndroidManifest.xml file. In addition, they have extracted restricted API calls, IP addresses, URLs, hostnames from the source code of an application. The Malgenome and Drebin datasets are extensively used in the studies in order to compare their approaches with other proposals in the literature.

Dendroid [8] employs machine learning algorithms on Android code blocks/structures. These code blocks are represented as control flow graphs. For each malware family, the commonly used code blocks and their count in the application code were extracted. Droidminer [9] uses behavioural graphs of mobile applications. DroidSIFT [10] is a semantic-based approach that classifies Android malware via dependency graphs obtained from source codes of applications. Another work on familial classification study is Droidlegacy [11]. They extract the most common malicious code blocks and Android API calls in a family. It is tested only with 11 families and achieved to get an accuracy of 98%. On the other hand, attackers can mislead the model by using the most common libraries.

As attackers use code obfuscation and code loading techniques more than ever, for some cases static analysis approach has remained incapable of classifying or detecting malware samples accurately. Therefore, a dynamic analysis based approach has been proposed to address the issue. In DroidScribe [12], the features such as total network traffic size, accessed IP/Port numbers, files and Binder-IPC methods are extracted for malware familial classification. To the best of our knowledge, Ec2 [7] and UpDroid [3] are the only studies that employ hybrid approach for analyzing and classifying Android malware samples by using machine learning techniques. UpDroid also presents a dataset of malware that use updating techniques for downloading malicious code at runtime.

There are also few studies that analyze network activities of malicious applications. CREDROID [13] analyzes the network activities of the samples in the Malgenome dataset. They extract information related to the following protocols: HTTP, DNS and SSL protocols. It is noticed that 63% of the samples generate network traffic. Rest of them either do not generate network traffic or are not compatible with the environment. In the current study, we employ also HTTP and DNS traffic related features for malware familial classification. Chen et al. [14] analyzes the network activities of the samples in the Drebin dataset. They obtain similar results with the current study that more than 70% malware samples generates malicious traffic in their first five minutes. By using information collected from DNS query and HTTP request packets, they obtained 69.55% and 40.89% detection rates respectively.

III. Hybrid Analysis

The increasing variety of Android malware has caused to increase the number of families. So researchers have recently focused on the malware family classification. In these approaches, mostly static analysis are employed, there are only few studies based on dynamic analysis [12]. However, as noted above, both analysis have pros and cons, therefore hybrid analysis is employed here. Choosing the right features is very important for producing effective classifiers and hence distinguishing the characteristics of malware families. This is the main of this study. By using features collected from both static and dynamic analysis, the effective features on malware familial classification are explored. The general overview of the proposed approach is given in Figure 1.

The features employed in this study is represented in detail in the subsequent section. Well-known machine learning algorithms, namely k-nearest neighbors (k-NN), decision tree, random forest and Support Vector Machine (SVM), has been applied on these features. Here, Weka tool [15] is used to implement the machine learning algorithms and the default setting of these algorithms as given in Weka is employed.

A. *Feature extraction*

In static analysis, features are usually extracted from the manifest file or directly from the code. Firstly, a reverse engi-

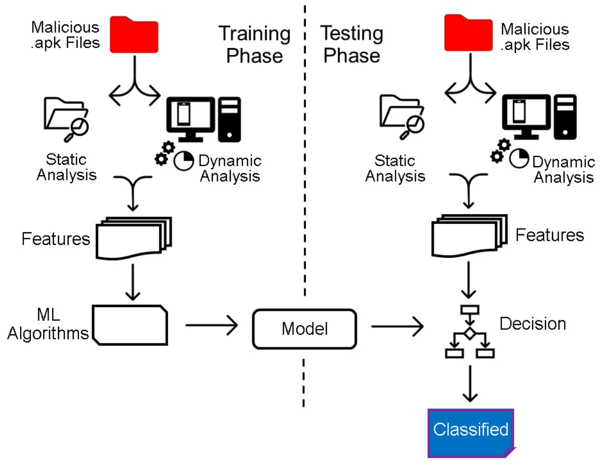

Fig. 1. Simplified schema of the proposed approach

neering tool called Apktool [16] is used to obtain the source files of an application such as assests, source code, manifest file. Here, the static features collected from the manifest file is employed as given in UpDroid [3]. Accessing to personal data such as call list, sms, gallery, hardware components (e.g. camera, bluetooth, microphone) can only be used by granting permission by the user. Therefore, the permissions, which are among the most used features in Android malware detection [17], are employed in this study. Other static features are the number of services, activities, receivers, extra components collected from the manifest file and the apk size.

As noted above, features collected from both static and dynamic analysis are explored in this study. Besides features given in UpDroid [3], two new feature groups collected from dynamic analysis are introduced in this study: network-related features and activity bigrams. In UpDroid [3], 175 features are employed in total, where most of them (110) are dynamic features since the main focus is to classify update attacks. The majority of static features are obtained from the manifest file such as number of services, activities, receivers and permissions. Dynamic features obtained from the output of DroidBox [18] mainly consist of the number of operations at runtime such as file operations, sent SMSs, phone conversations, cryptographic and dynamic code loading operations, data leakage.

Besides general information about network traffic such as the number of sent and received network packets given in Table III, most of the network-related features used in this study have been extracted from HTTP and DNS protocol headers. HTTP traffic could be used for leaking some important data such as SDK version, IMEI and IMSI numbers. For example, Kmin family could leak the SDK version over HTTP traffic [14]. Moreover, it is observed that samples in a same family can show similarity in their HTTP and DNS traffic patterns. For example, most of the samples belonging to the shedun family has the same URL length in GET requests. Some malware families try to leak fixed sized IMSI and IMEI numbers of victim devices in HTTP GET requests. In this study, the existence of the following extensions .apk and .jar in HTTP request/reply messages is also included in the features. Especially update attacks can download their malicious code at runtime, these features could be indicators for malicious activity. Therefore, these newly added features control whether the application has downloaded a file successfully that has the extension of .jar or .apk. The content types of HTTP responses (app, text, video, audio, multi, and image) are also considered. Besides IMSI/IMEI information of a device, malicious applications can send other important data such contact lists, phone numbers via HTTP requests as URL parameters. Therefore, the average number of parameters in a URL is added into the features. Besides HTTP traffic, features related to DNS requests/responses are included in the feature set as listed in Table II. To sum up, there are overlapping network-related features with UpDroid [3] such as total number of opened/closed connections, total number of network connections, total size of network packets and number of sent/received network packets. In addition, we analyse network packets more deeply and extract new features from the headers of the following protocols: IP, TCP, UDP, HTTP, DNS.

TABLE I
FEATURES RELATED TO HTTP PROTOCOL

HTTP Features	Type	Count
Content type of HTTP traffic	Boolean	6
Size of HTTP traffic in each time interval (byte)	Numeric	3
Total size of HTTP traffic (byte)	Numeric	1
Number of HTTP requests/responses	Numeric	2
Number of HTTP frames	Numeric	1
Average content length in HTTP responses (byte)	Numeric	1
Time of the first/last HTTP packet	Numeric	2
Total time period of HTTP traffic	Numeric	1
Number of unique HTTP servers	Numeric	1
Average URL length in GET/POST messages	Numeric	2
Average number of parameters in GET/POST messages	Numeric	2
Existence of .apk/.jar extension in GET/POST messages	Boolean	4
Number of GET/POST requests	Numeric	2
Number of unique URLs in GET/POST messages	Numeric	2
Size of POST messages (byte)	Numeric	1

TABLE II
FEATURES RELATED TO DNS PROTOCOL

DNS Features	Type	Count
Time of the first/last DNS packet	Numeric	2
Total time period of DNS traffic	Numeric	1
Total number of unique domains in DNS queries	Numeric	1
Average length of DNS domain names	Numeric	1
Total number of DNS queries/responses	Numeric	2
Total size of DNS packets (byte)	Numeric	1

Malicious code part of an Android application could be triggered at different times [19]. A malware can run its malicious code when the mobile device is booted or after a certain period of time passed. The dynamic analysis tools in Android generally run for 10 minutes [20]. Hence, attackers can postpone triggering malicious code in a virtual/real device. Therefore, in this study, we run each application for 15 minutes and divide the analysis time into three equal parts (each for

5 minutes). Some features collected separately for each time period, since malicious code might be active in only one period. Here, Droidbox is used for collecting data produced at runtime.

TABLE III
OTHER NETWORK-RELATED FEATURES

Other Network-Based Features	Type	Count
Total network traffic size in each time period (byte)	Numeric	3
Total network traffic size (byte)	Numeric	2
Total UDP traffic size (byte)	Numeric	1
Number of UDP frames	Numeric	1
Number of SYN packets	Numeric	1
Number of unique IP/Port	Numeric	2
Number of opened/closed network connections	Numeric	2
Number of sent/received network packets	Numeric	2
SSL usage in network traffic	Boolean	1
Average size of IP packets (byte)	Numeric	1

Malware can leak personal and/or device information with different activity sequences (e.g. reading a specific file on the device device and sending this information over the network). Therefore, consecutive activities performed by applications are taken into account in this study. Here, the bigrams of the activities given in Table IV are used and the number of bigrams encountered in the application are given to the model as features. Hence, 100 (10x10) features are collected for this feature group. All these 100 features are firstly employed in this study. So, together with 175 features given in UpDroid, in total, 329 features are used in this study.

TABLE IV
ACTIVITIES USED IN BIGRAMS

Activity Name	Description
cryptousage	Usage of cryptographic alg. with support of Android API
dataleaks	Information leakage via SMS, file or network
dexclass	Code loading by using DexClassLoader
fread	Reading files
fwrite	Writing files
recvnet	Network connections received
sendnet	Network connections requested
runtimesystemevents	Usage of Android system events
sendsms	Sending SMS
servicestart	Starting services

IV. EXPERIMENTAL RESULTS

In this section, features on malware family classification are evaluated on three datasets, namely Malgenome [1], Drebin [2] and UpDroid [3], and discussed. Malgenome is the first dataset introduced and employed for comparison in the literature. Malgenome has 1,260 applications belonging to 49 families. This dataset is extended in Drebin [2], which has 5554 applications belonging to 178 families. UpDroid is one of the recently proposed datasets. It consists of 2535 update attacks belonging to 21 malware families. An update attack load its malicious code at runtime. Since network-based features could be more effective on update attacks, this dataset is included in the experiments. Moreover, the feature set employed in the UpDroid study is extended in the current study.

In the experiments, families that have larger samples have more impact on results. Therefore, families that have more than 10 samples are included in the evaluations. 10 fold cross-validation method is employed for comparisons as in UpDroid. Table V shows the results. Three metrics are employed for comparison: accuracy, true positives (TP) or detection rate in other words and, false positives (FP). In this table, all features are included. The best results are obtained with the kNN algorithm, so unless specified differently, the results of this algorithm are given in the subsequent results.

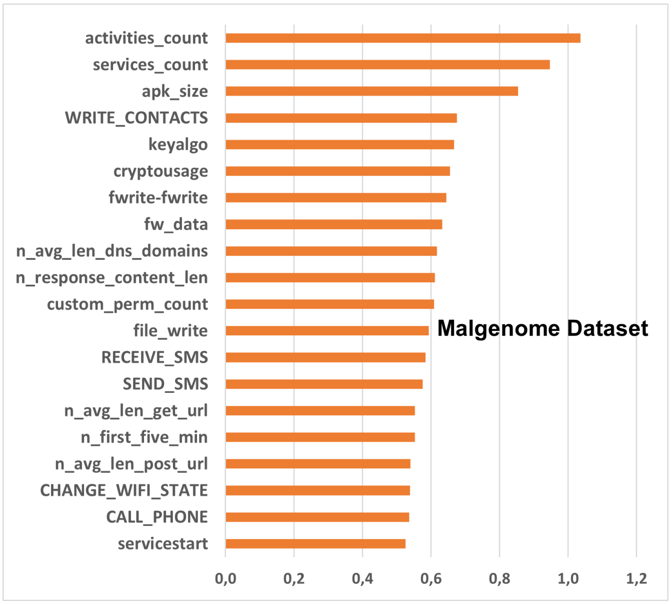

Fig. 2. Most effective 20 features for the Malgenome dataset

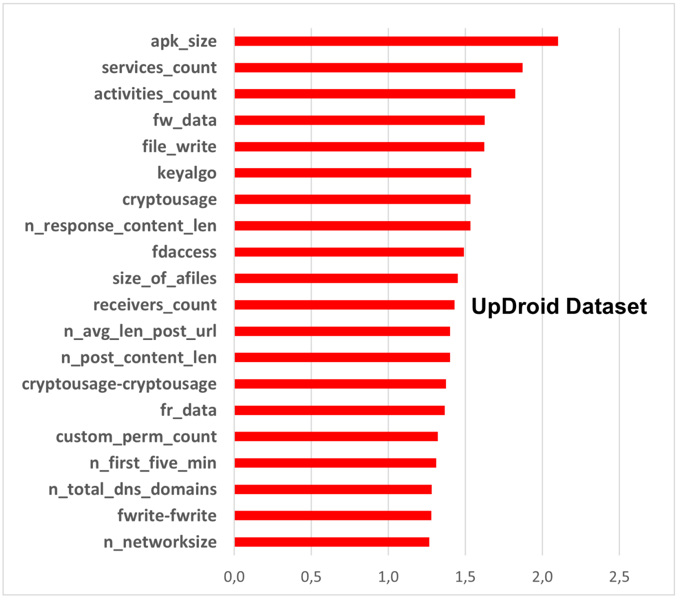

Fig. 3. Most effective 20 features for the UpDroid dataset

Figure 2, Figure 3 and Figure 4 show the most effective top twenty features on datasets. Features starting with ”n” are representing network-related features. It is observed that quite

TABLE V
EFFECTS OF ALL FEATURES ON MALGENOME [1], DREBIN [2] AND UPDROID [3]

Algorithm	Accuracy %			TP %			FP %		
	Malgenome	UpDroid	Drebin	Malgenome	UpDroid	Drebin	Malgenome	UpDroid	Drebin
kNN	97.38	98.04	96.40	97.4	98.0	96.4	1.4	0.4	0.3
Random Forest	93.49	97.44	92.33	93.5	97.4	92.3	3.9	0.6	0.9
Decision Tree	96.89	97.53	93.58	96.9	97.5	93.6	1.3	0.4	0.5
SVM	97.83	97.44	95.77	97.8	97.4	95.8	0.8	0.8	0.4

number of network-related features are in top 20 for all three datasets. As seen in the figure, the size of the network traffic collected in the first five minutes is the common feature in all datasets. It is observed that malicious applications mostly generate network traffic in their first five minutes. In addition, the average length of URL of GET and POST requests are among the most distinguishing features in all datasets.

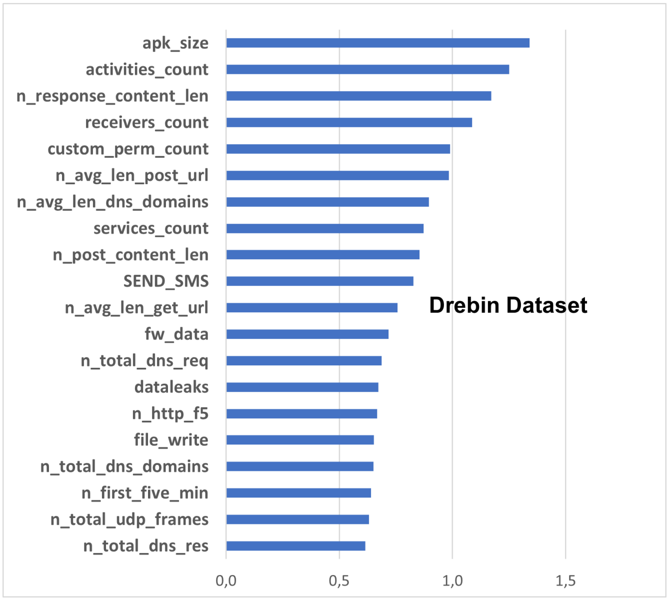

Fig. 4. Most effective 20 features for the Drebin dataset

The comparison with UpDroid is given in Table VI. As shown in the results, the proposed features have positive effects on the results. By using all three machine learning algorithms employed in UpDroid [3], the proposed approach produces higher accuracy. Please note that the results are evaluated on the UpDroid dataset.

TABLE VI
COMPARISON WITH UPDROID [3] ON THE UPDROID DATASET

Study	Algorithm	Accuracy %	TP %	FP %
Our Approach	kNN	**98.04**	98.0	0.4
UpDroid	kNN	96.85	96.4	0.4

Figure 5 shows the confusion matrix of the results given in Table VI. The comparison results on the Drebin dataset [2] are given in Table VII. Here, the proposed approach produces comparable results with UpDroid.

TABLE VII
COMPARISON WITH UPDROID [3] ON THE DREBIN DATASET

Study	Algorithm	Accuracy %	TP %	FP %
UpDroid	kNN	**96.85**	**96.8**	**0.3**
Our Approach	kNN	96.39	96.4	0.3

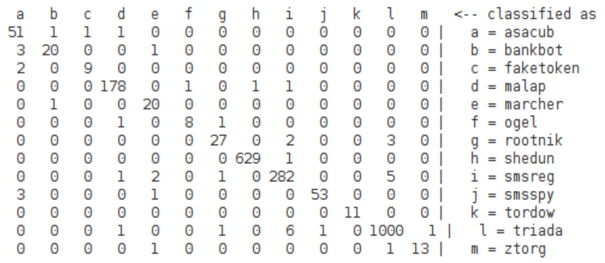
```
  a    b    c    d    e    f    g    h    i    j    k    l    m   <-- classified as
 51    1    1    1    0    0    0    0    0    0    0    0    0 |   a = asacub
  3   20    0    0    1    0    0    0    0    0    0    0    0 |   b = bankbot
  2    0    9    0    0    0    0    0    0    0    0    0    0 |   c = faketoken
  0    0    0  178    0    1    0    1    1    0    0    0    0 |   d = malap
  0    1    0    0   20    0    0    0    0    0    0    0    0 |   e = marcher
  0    0    0    1    0    8    1    0    0    0    0    0    0 |   f = ogel
  0    0    0    0    0    0   27    0    2    0    0    3    0 |   g = rootnik
  0    0    0    0    0    0    0  629    1    0    0    0    0 |   h = shedun
  0    0    0    1    2    0    1    0  282    0    0    5    0 |   i = smsreg
  3    0    0    0    1    0    0    0    0   53    0    0    0 |   j = smsspy
  0    0    0    0    0    0    0    0    0    0   11    0    0 |   k = tordow
  0    0    0    1    0    0    1    0    6    1    0 1000    1 |   l = triada
  0    0    0    0    1    0    0    0    0    0    0    1   13 |   m = ztorg
```

Fig. 5. Confusion matrix for the UpDroid Dataset

As it is shown in Figure 5, the highest TP rate is obtained without misdetection in Tordow malware family which has only 11 samples. Malicious applications belong to Tordow family usually download other application files to the device at runtime [21]. Triada ans Shedun families, whose have high number of samples in the dataset, also have high TP rates. Despite their high TP rates, most of confusions are obtained between Triada and Smsreg families. As it is stated in [22] and [23] that both families have adware purposes and, download harmless adware applications after being installed on a victim device.

The effects of new features are represented in Table VIII. Since the samples in the UpDroid dataset generates more network traffic than the samples in the Drebin dataset, network-based features have more positive impact on the UpDroid dataset as shown in Table VIII. It can be inferred from the table that over 90% of accuracy is achieved with only 55 different network-related feature on the UpDroid dataset. 10-fold cross validation is used for obtaining results in Table VIII Majority of malicious applications in the UpDroid dataset have higher network activity than applications in Drebin and Malgenome datasets. Similarly, newly added activity bigrams features have more positive effect on the UpDroid dataset.

Finally, the proposed approach is compared with Ec2 [7] and UpDroid [3], two studies that use hybrid features for malware familial classification. Drebin dataset [2] is used for a fair comparison. In Ec2 [7], the following metrics are used for the performance evaluation: MiF and MiAUC scores. Therefore, the same metrics are employed in Table IX. It is shown that our

TABLE VIII
EFFECTS OF NEW FEATURES

	Accuracy %			TP %			FP %		
FEATURES / DATASETS	**Malgenome**	**UpDroid**	**Drebin**	**Malgenome**	**UpDroid**	**Drebin**	**Malgenome**	**UpDroid**	**Drebin**
Network-Related	73.99	91.72	61.38	74.0	91.7	61.4	8.8	2.3	4.3
Activity Bigrams	75.86	92.61	70.59	75.9	92.6	70.6	8.6	2.0	2.6
Network-Related + Activity Bigrams	79.59	92.78	77.11	79.6	92.8	77.1	8.1	2.0	2.1
UpDroid Features + Network-Related	97.45	98.17	95.92	97.5	98.2	95.9	1.3	0.4	0.3
UpDroid Features + Activity Bigrams	97.73	98.04	96.32	97.7	98.0	96.3	1.1	0.3	0.3
All Features	97.36	98.04	96.40	97.4	98.0	96.4	1.4	0.4	0.3

approach outperforms than Ec2 [7] and shows similar results with UpDroid [3] on these metrics.

TABLE IX
COMPARISON WITH EC2 [7] AND UPDROID [3] ON THE DREBIN DATASET

Study	**Algorithm**	**MiF**	**MiAUC**
Ec2	kNN	0,47	0,73
UpDroid	kNN	0,96	0,98
Our Approach	kNN	0,96	0,98
Ec2	Random Forest	0,95	0,97
UpDroid	Random Forest	0,94	0,99
Our Approach	Random Forest	0,94	0,99

V. CONCLUSION

In this study, a hybrid approach is proposed for malware familial classification. Besides features given in UpDroid [3], the effects of two new feature groups, network-based features and activity bigrams, are explored. To the best of the authors' knowledge, these two feature groups are firstly used on malware familial classification. The results show the positive effect of these features, particularly on the UpDroid dataset [3], since malicious applications in this dataset have more network activity than samples in other datasets, Malgenome and Drebin.

Acknowledgements This study is partly supported by the Scientific and Technological Research Council of Turkey (TUBITAK-115E150). We thank TUBITAK for its support. We also would like to thank Kursat Aktas for his help on the study.

REFERENCES

[1] Zhou, Y., Jiang, X.: Dissecting android malware: Characterization and evolution. In: 2012 IEEE symposium on security and privacy. pp. 95–109. IEEE (2012)
[2] Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., Rieck, K., Siemens, C.: Drebin: Effective and explainable detection of android malware in your pocket. In: Ndss. vol. 14, pp. 23–26 (2014)
[3] Aktas, K., Sen, S.: Updroid: Updated android malware and its familial classification. In: Nordic Conference on Secure IT Systems. pp. 352–368. Springer (2018)
[4] IDC: Smartphone market share. https://www.idc.com/promo/smartphone-market-share/os (2020)
[5] McAfee: Mcafee mobile threat report q1-2020. https://www.mcafee.com/content/dam/consumer/en-us/docs/2020-Mobile-Threat-Report.pdf (2020)
[6] Chebyshev, V.: It threat evolution q2 2020. mobile statistics. https://securelist.com/it-threat-evolution-q2-2020-mobile-statistics/98337/ (2020)
[7] Chakraborty, T., Pierazzi, F., Subrahmanian, V.: Ec2: Ensemble clustering and classification for predicting android malware families. IEEE Transactions on Dependable and Secure Computing 17(2), 262–277 (2017)
[8] Suarez-Tangil, G., Tapiador, J.E., Peris-Lopez, P., Blasco, J.: Dendroid: A text mining approach to analyzing and classifying code structures in android malware families. Expert Systems with Applications 41(4), 1104–1117 (2014)
[9] Yang, C., Xu, Z., Gu, G., Yegneswaran, V., Porras, P.: Droidminer: Automated mining and characterization of fine-grained malicious behaviors in android applications. In: European symposium on research in computer security. pp. 163–182. Springer (2014)
[10] Zhang, M., Duan, Y., Yin, H., Zhao, Z.: Semantics-aware android malware classification using weighted contextual api dependency graphs. In: Proceedings of the 2014 ACM SIGSAC conference on computer and communications security. pp. 1105–1116 (2014)
[11] Deshotels, L., Notani, V., Lakhotia, A.: Droidlegacy: Automated familial classification of android malware. In: Proceedings of ACM SIGPLAN on program protection and reverse engineering workshop 2014. pp. 1–12 (2014)
[12] Dash, S.K., Suarez-Tangil, G., Khan, S., Tam, K., Ahmadi, M., Kinder, J., Cavallaro, L.: Droidscribe: Classifying android malware based on runtime behavior. In: 2016 IEEE Security and Privacy Workshops (SPW). pp. 252–261. IEEE (2016)
[13] Malik, J., Kaushal, R.: Credroid: Android malware detection by network traffic analysis. In: Proceedings of the 1st acm workshop on privacy-aware mobile computing. pp. 28–36 (2016)
[14] Chen, Z., Han, H., Yan, Q., Yang, B., Peng, L., Zhang, L., Li, J.: A first look at android malware traffic in first few minutes. In: 2015 IEEE Trustcom/BigDataSE/ISPA. vol. 1, pp. 206–213. IEEE (2015)
[15] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. ACM SIGKDD explorations newsletter 11(1), 10–18 (2009)
[16] Connor Tumbleson, R.W.: A tool for reverse engineering android apk files. https://ibotpeaches.github.io/Apktool/ (2010)
[17] Feizollah, A., Anuar, N.B., Salleh, R., Wahab, A.W.A.: A review on feature selection in mobile malware detection. Digital investigation 13, 22–37 (2015)
[18] pjlantz: Droidbox: Dynamic analysis of android apps. https://github.com/pjlantz/droidbox (2018)
[19] Garcia, J., Hammad, M., Malek, S.: Lightweight, obfuscation-resilient detection and family identification of android malware. ACM Transactions on Software Engineering and Methodology (TOSEM) 26(3), 1–29 (2018)
[20] Maier, D., Protsenko, M., Müller, T.: A game of droid and mouse: The threat of split-personality malware on android. Computers & Security 54, 2–15 (2015)
[21] Comodo: Comodo threat research labs warns android users of tordow v2.0 outbreak. https://blog.comodo.com/comodo-news/comodo-warns-android-users-of-tordow-v2-0-outbreak/ (March 2018)
[22] Data, G.: Analysis of xafecopy android.application.smsreg.a. https://file.gdatasoftware.com/web/en/documents/whitepaper/G_Data_Analysis_Xafecopy.pdf (2017)
[23] Snow, J.: Triada: organized crime on android. https://www.kaspersky.com/blog/triada-trojan/11481/ (2016)