virus BULLETIN Covering the global threat landscape

DEXOFUZZY: ANDROID MALWARE SIMILARITY CLUSTERING METHOD USING OPCODE SEQUENCE

Shinho Lee, Wookhyun Jung, Sangwon Kim, Jihyun Lee, Jun-Seob Kim
ESTsecurity, Korea

ABSTRACT

Recently, *Android* malware has shown a tendency towards large-scale distribution, which reduces the amount of time required for stealing data. Hackers implement such malicious code variants by reusing code. This paper proposes the use of the 'Dalvik EXecutable Opcode Fuzzy' ('Dexofuzzy') hash, which finds similar malware variants without the need for an analyst to have systematic or mathematical knowledge. Dexofuzzy is a method for generating similarity digests with software birthmarking of opcode sequences in Dex files based on ssdeep. The clustering results are obtained by N-gram tokenizing of the Dexofuzzy hash, which is generated from the malware samples. Observations and experimental results have demonstrated the effectiveness of Dexofuzzy's similarity search in analysing the *Android* malware 'Operation Blackbird', which had previously been reported by *ESTsecurity Security Response Center* (*ESRC*). Finally, we have measured the similarity of the indicators of compromise (IOCs) in 74 reports published by *AlienVault* OTX and analysed the association between each report in order to attest the efficiency of Dexofuzzy.

1. INTRODUCTION

Android malware has recently shown a tendency towards large-scale distribution, which reduces the amount of time required for stealing data. Hackers implement such malicious code variants by reusing code. According to a *Kaspersky* report [1], the number of incidences of *Android* malware in 2018 doubled compared with the previous year, despite the presence of security enhancements in the *Android* operating system [2, 3].

Many studies based on static, dynamic and hybrid analysis [4, 5] have been carried out in order to find ways to better protect *Android* devices and users from malware threats. Machine learning and similarity algorithm-based methods have been applied to the previous analysis data [6, 7, 8]. Machine learning-based detection methods enable the clustering and classifying of multiple malware samples, but there is a usage limit on machine learning: high entry barriers make it difficult for general analysts to use such methods with mathematical and statistical approaches and there are considerable costs for the preprocessing and learning of datasets. In contrast, detection methods based on similarity algorithms can evaluate the characteristics of malware samples without the need for additional learning processes. Numerous similarity algorithms have been proposed to improve performance [9]. Although these algorithms require a hash index table for the similarity search, from the point of view of the analyst, the use of similarity algorithms is a much more straightforward method than machine learning.

In particular, the similarity digests algorithm is simple and fast and can be used to compare the similarity of form or objective among multiple samples. Searching for similar malicious codes requires minimum effort, which enables us to cluster and classify a large number of variant malicious codes. Although similarity digests have a limitation in inferring malignancy compared to existing analysis methods, it is simple and quick to cluster, classify and compare the large-scale samples. Because of these advantages, security researchers have introduced analytical methodologies using similarity digests for searching for similar samples – for example, *Trend Micro* proposed a similarity digest using the Locality Sensitive Hash (LSH), called TLSH [10], and JPCERT proposed Impfuzzy, which calculates an ssdeep hash of the import table of a given PE file [11].

In this paper, we propose the Dalvik EXecutable Opcode Fuzzy (Dexofuzzy) hashing method, which enables analysts to search and classify *Android* malware variants in a lightweight way. To do this, first we parse a Dex file in APK (which stands for Android Package and compressed by zip) to extract opcode. Then we calculate the ssdeep value of the opcode generated from the previous step, which is named Dexofuzzy. To search malware variants, we tokenize the hashes generated by Dexofuzzy using N-gram, and then index them for a fast search.

This paper is organized as follows: Section 2 explains the techniques that are used in Dexofuzzy. Section 3 describes a method to search similar malware samples using Dexofuzzy extracted from opcode. In section 4, we verify the performance of Dexofuzzy, in which the malware samples are properly clustered. Finally, section 5 summarizes the results from each part and concludes by highlighting the significance of the methodology that we have proposed in this paper.

2. BACKGROUND

2.1 Android package structure

Android files have an APK extension and utilize the ZIP file format. The most essential components of the APK are AndroidManifest.xml, classes.dex, and resources.arsc. AndroidManifest.xml is an encoded XML file containing information about attributes and permissions of the app, and resources.arsc is an encoded file containing resource information about the app. In order to run the app, classes.dex, known as the Dex file, is used, which contains opcode and is used to generate Dexofuzzy. Figure 1 illustrates the structure of the APK file.

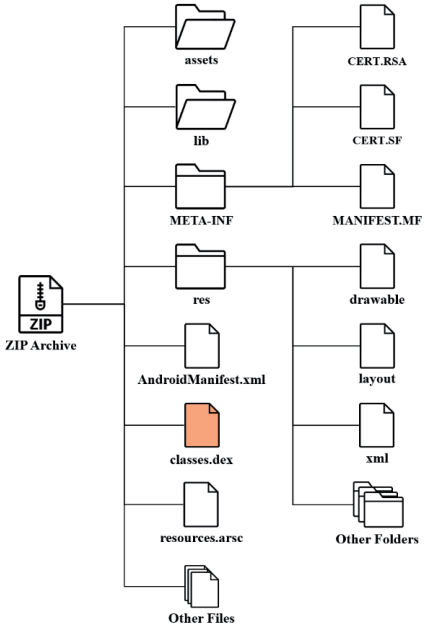


Figure 1: Structure of the APK file.

2.2 Dex format

The Dex file is the most important file that contains the compiled code that runs on *Android*. When decompiling the Dex file (a.k.a. Baksmaling), it returns bytecode called Smali. The structure of the Dex file is shown in Figure 2.

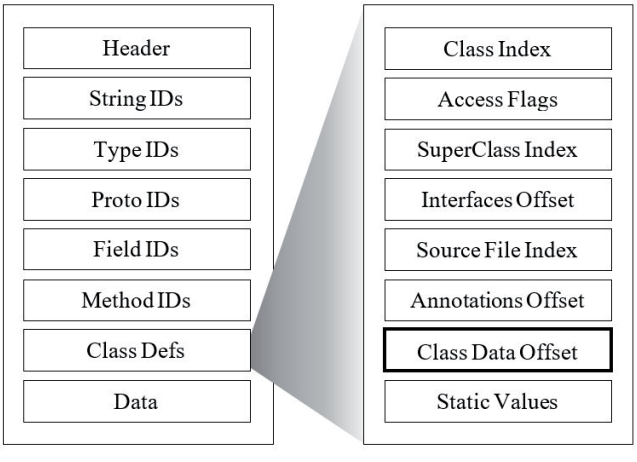


Figure 2: Dex format.

In the Dex file structure, Class Defs is a table that defines code information of *Android* files by Class. Class Data Offset is a set of bytecode offsets except for attribute information of class [12]. We use a method to parse the Class Data Offset of the Dex file to extract the opcode sequence.

2.3 Software birthmarking

Software birthmarking is a promising technique used for the detection of software theft based on the use of unique characteristics to identify programs. A number of studies have been conducted to detect repackaged *Android* apps [13, 14, 15]. G. Myles *et al.* suggested a software birthmark using opcode-level N-gram [16]. In this paper, we use the software birthmark method on the opcode of Dex files to detect *Android* malware variants that reuse existing malicious code.

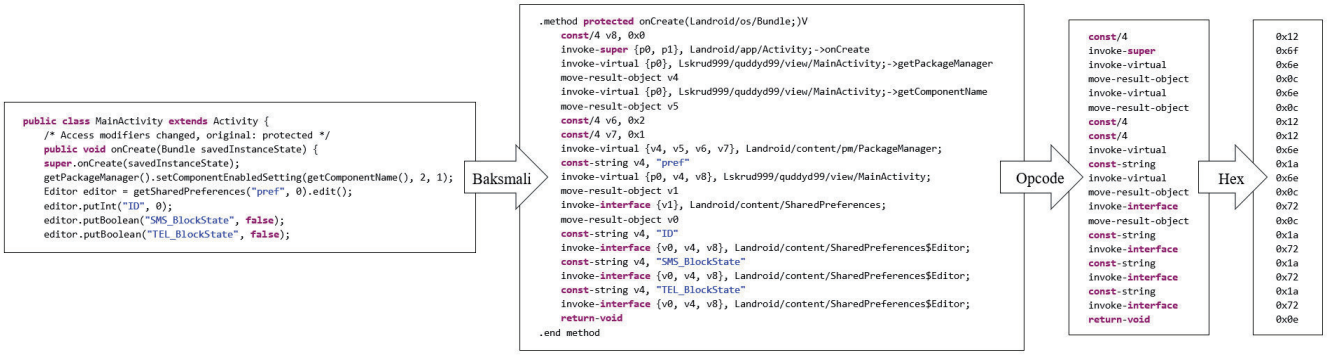


Figure 3: Birthmarking dex opcode by software birthmark.

2.4 Fuzzy hash – ssdeep

The Context Triggered Piecewise Hashes (CTPH) algorithm, proposed by J. Komblum, is a method that generates a similarity digest by dividing a file into segments using the Rolling Hash method, unlike the existing hashing method that guarantees integrity [17]. CTPH is designed to identify the similarity of data, and is currently used for analysing malicious code [10].

2.5 N-Gram tokenizer

N-gram is a contiguous sequence of n items from a given sample of text. The method is widely used for measuring the similarity of sentences in search engines, big data, and computer forensics. B. Wallace presented the results of clustering malware variants by hashing the malware samples to ssdeep and tokenizing them with 7-gram, and proposed the use of the N-Gram tokenizer for searching N-gram as one of the ways to improve the performance of ssdeep [18]. *Elasticsearch* is a search engine that provides a distributed, multitenant-capable, full-text search. Because it supports N-gram tokenizer, it is highly effective for similarity searches [19].

3. METHODOLOGY

This section describes how Dexofuzzy works to measure the similarity between *Android* malware families. Dexofuzzy parses the Dex file inside the *Android* file to extract the opcode, and the extracted features are used to generate the similarity digests using ssdeep. The features are also used to search for similar malicious code by tokenizing the generated Dexofuzzy to N-gram size and using the sliding window technique. Figure 4 is a schematic illustration of how Dexofuzzy is created.

3.1 Opcode sequence extraction

The classes.dex file, which is the core file required to run an *Android* application, is decompiled into Dalvik bytecode using *APKTool* [20]. We will use the opcode defined in Dalvik bytecode for the similarity search. In this paper, opcode has been selected because the opcode sequence will remain unchanged in spite of overall changes in the Dalvik bytecode, even in the case of changing the Package Name, Class Name and Method Name and adding one or more variables in reusable codes. Figure 5 is a representative example to show that Dalvik bytecodes are changed (except for the opcode sequence) in two identical malware samples by the addition of variables in the Method and by changing the Package Name. Therefore, opcode is the most powerful characteristic in the regard that the original remains unchanged despite changes in other elements when measuring the similarity of malware variants implemented by reusing codes.

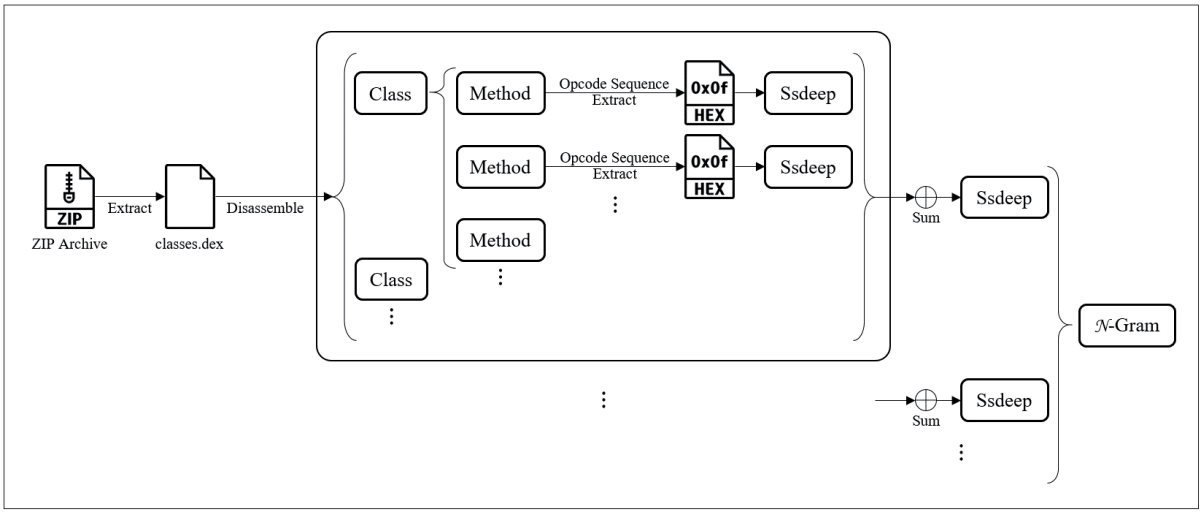


Figure 4: Dexofuzzy's smilarity digests.

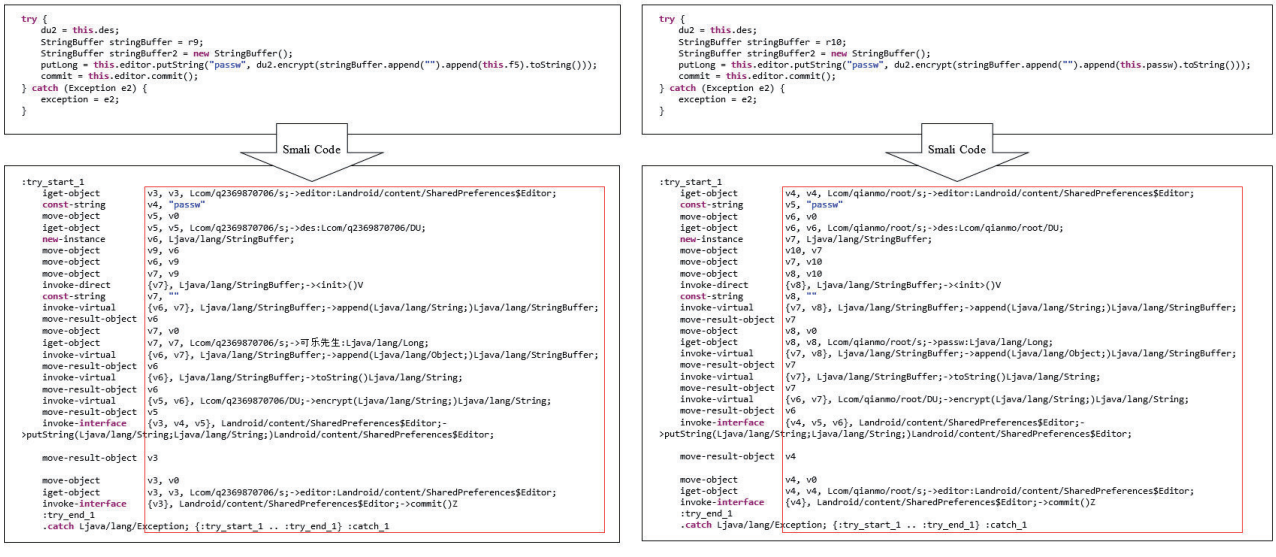


Figure 5: Comparison of Smali code in malware variants.

Figure 6 displays the process of extracting the opcode sequence from each method into Hex values by parsing the classes.dex file. The extracted opcode sequence is bundled into a single string that stores the opcode sequence of the Dex file in list form. Since such an opcode sequence string has unique characteristics of each method, the malware variants reusing the code contain a large number of opcode sequence strings that match in the list, compared to malware which is not implemented by reusing code.

3.2 Fuzzy hashing based on opcode sequence

As shown in Figure 7, the extracted opcode sequence string list performs the first stage of fuzzy hashing for each string. Next, the fuzzy-hashed Methodfuzzy list is fuzzy-hashed again in the second stage to generate Dexofuzzy.

The ssdeep format is composed of BlockSize:Signature1:Signature2, where BlockSize is the minimum block size for rolling hashing. Signature1 is a signature generated by rolling hashing based on previously calculated block size, and Signature2 is a signature generated by rolling hashing with twice the block size. Dexofuzzy is a similarity digest generated based on ssdeep, which enables us simply to compare the signatures using the function ssdeep compare. Figure 8 shows the result of comparing two *Android* malware variants with Dexofuzzy.

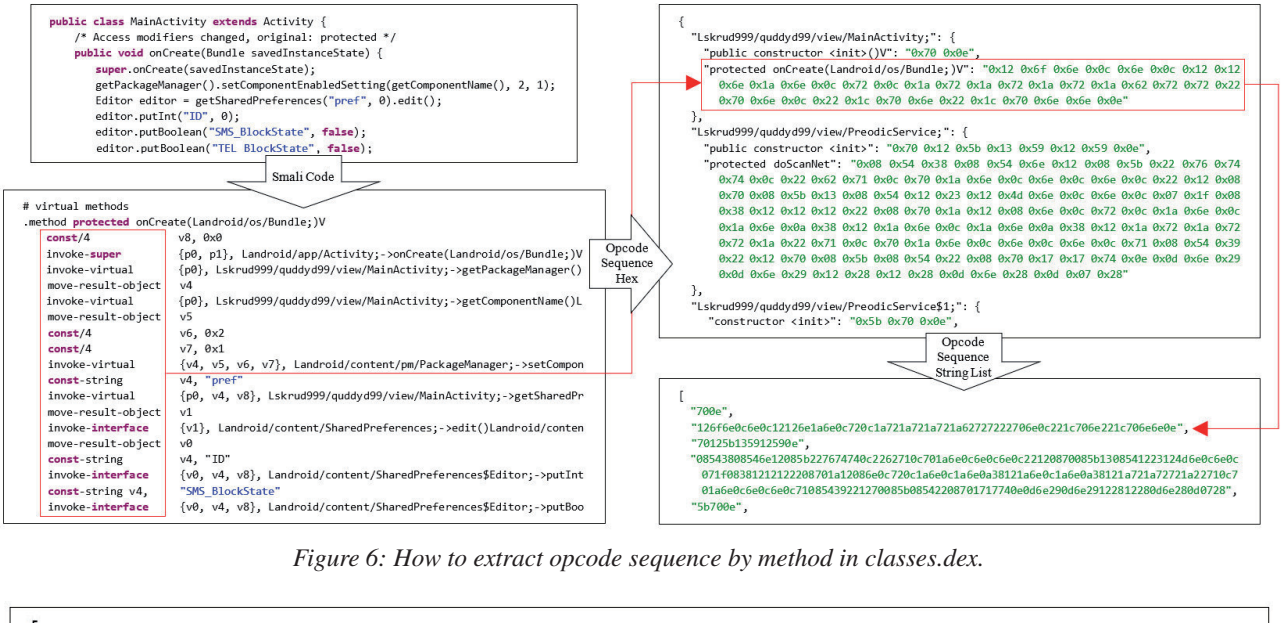


Figure 6: How to extract opcode sequence by method in classes.dex.

Figure 7: Generating process of Dexofuzzy.

```
>>> import Ssdeep
>>> Ssdeep.compare("48:U7uPrEMc0HZj0/zeGnD2KmUCNc2FuGgy9fY:UHMHZ4/zeGD2+Cap3y9Q", "48:U7uPrEMc0HZj0/
zeGnV2KmDmUCNc9tzFuGgLM/xBDyw9V4XIU0pt:UHMHZ4/zeGV2lCa43LM5B/H7U0pt")
72
```

Figure 8: Results of comparing ssdeep using Dexofuzzy.

B. Wallace has confirmed that Signature1 is more efficient than Signature2 for searching malware by performing a test to compare the two signatures with 7-gram Slice with ssdeep [18]. Accordingly, we use Signature1 to search for similar malware variants in this paper.

3.3 N-gram tokenizer sliding window search

We use *Elasticsearch* to search for Dexofuzzy by N-gram. *Elasticsearch* basically provides an N-gram tokenizer and indexes all generated N-length tokens. Figure 9 describes the process of tokenizing Dexofuzzy into 7-grams to detect the indexed seven-character strings with the sliding window technique.

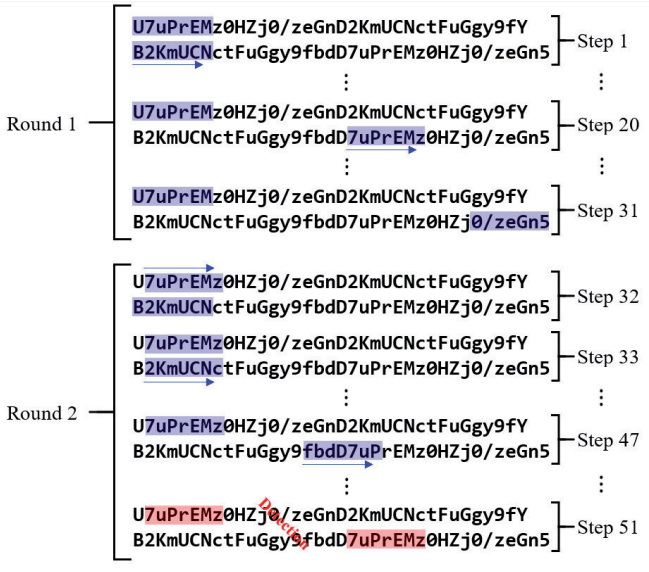


Figure 9: Dexofuzzy N-gram tokenizer's sliding window search.

In addition, *Elasticsearch* is useful for calculating the optimal thresholding value to increase the true positive rate and to reduce the false negative rate of each sample since the method enables the number of M to be specified, where the N-gram strings are matched, as a parameter when searching. Figures 10 and 11 illustrate how to search for a similar Dexofuzzy by setting N-gram and Minimum Should Match (from *Elasticsearch* settings) in a single Dexofuzzy sample.

```
U7uPrEMc0HZj0/zeGnD2KmUCNc2FuGgy9fY
5PizeGnW7uPrEMc0HZj0482KmUCNc3FuGgy9fY → Detection (3 matches)
+CY+LuBl9kiZ8hi1nSaNdDguY+dFR+1PxXrjuNAVpVLCbeBQEWk27uPrEMc0HZjR → Detection (3 matches)
```

Figure 10: Partial matching results of Minimum_Should_Match = 3 with 7-gram.

```
U7uPrEMc0HZj0/zeGnD2KmUCNc2FuGgy9fY
5PizeGnW7uPrEMc0HZj0482KmUCNc3FuGgy9fY → Detection (7 matches)
+CY+LuBl9kiZ8hi1nSaNdDguY+dFR+1PxXrjuNAVpVLCbeBQEWk27uPrEMc0HZjR → Non-Detection (5 matches)
```

Figure 11: Partial matching results of Minimum_Should_Match = 7 with 7-gram.

4. EXPERIMENT

This chapter presents the results of clustering and performance measurements on malware variants that are active in the wild, with the generated Dexofuzzy. We collected 212,955 malware samples and created a list of the ssdeep and Dexofuzzy hash values of a Dex file of the APK respectively, then searched samples using the *Elasticsearch* 7-gram tokenizer and clustered the result values in order to assess the performance of Dexofuzzy.

In this section, we will also demonstrate the effectiveness of the Dexofuzzy similarity search using the visualization tool *Gephi* [21] to analyse the malware 'Operation Blackbird', which has previously been reported. Finally, we collected the malware samples based on indicators of compromise (IOCs) in *AlienVault* OTX reports published by multiple vendors and then created Dexofuzzy and clustered it in order to explain the relationship between the reports. Table 1 shows the configuration of the environment for the experiment.

CPU	Intel(R) Core(TM) i5-7500 CPU @ 3.40GHz
RAM	24GB
Storage	240GB
OS	Ubuntu 18.08 LTS
Search engine	Elasticsearch 7.2

Table 1: The environment for the experiment.

4.1 7-Gram clustering result of Dex ssdeep vs. Dexofuzzy

We clustered the similarity digest hashes of the Dex files, which were computed using Dex ssdeep and Dexofuzzy respectively, with 431 samples detected as Trojan.Android.KRBanker, to evaluate the performance of the two similarity clustering algorithms. Figure 12 shows the clustering results of Dex ssdeep and Dexofuzzy by using the 7-gram search.

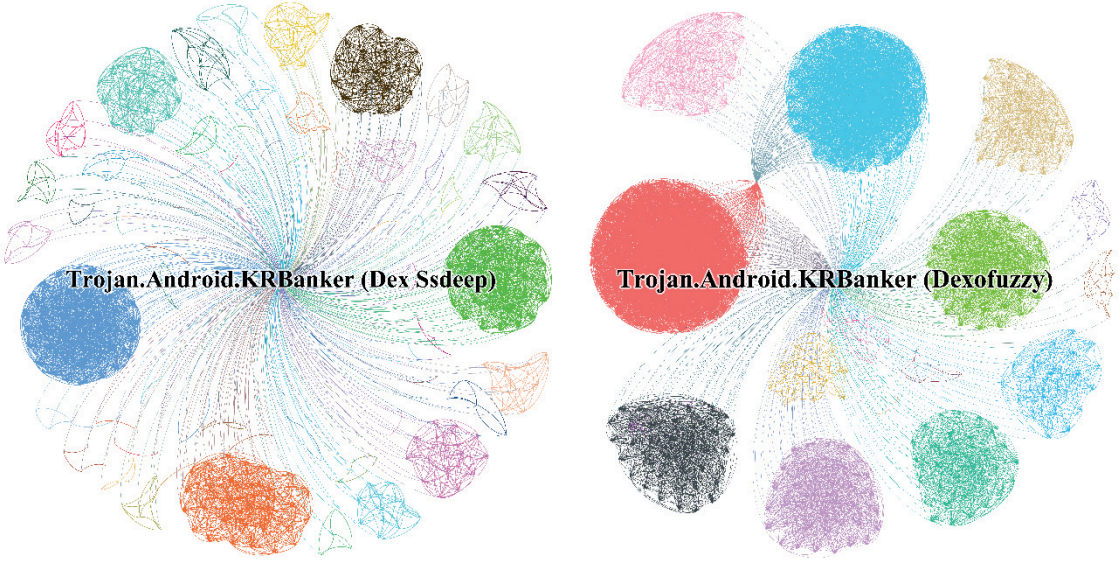


Figure 12: Clustering result from Dex ssdeep vs. Dexofuzzy 7-gram.

The experimental results show that Dex ssdeep hashes are clustered into 144 groups and 21 groups are clustered by Dexofuzzy – the clustering size of Dexofuzzy is seven times smaller than that of ssdeep, which means that Dexofuzzy is more effective than ssdeep. Next, we conducted an experiment to search for multiple Dexofuzzy hashes simultaneously to measure the search performance of the *Elasticsearch* tokenizer. Figure 13 shows the operation time of Dexofuzzy when searching 212,955 samples at the same time.

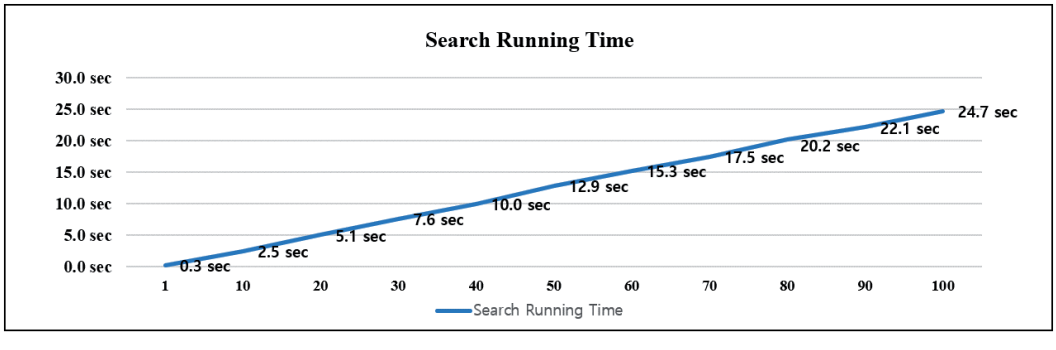


Figure 13: Search running time of Elasticsearch 7-gram tokenizer.

4.2 'Operation Blackbird' sample clustering

In the following experiment, we searched for similar variants using 7-gram tokens of Dexofuzzy generated in 10 malware samples from the IOCs [22] of 'Operation Blackbird' reported by *ESTsecurity*. Figure 14 shows the clustering result of the additionally found malware variants among 212,955 samples.

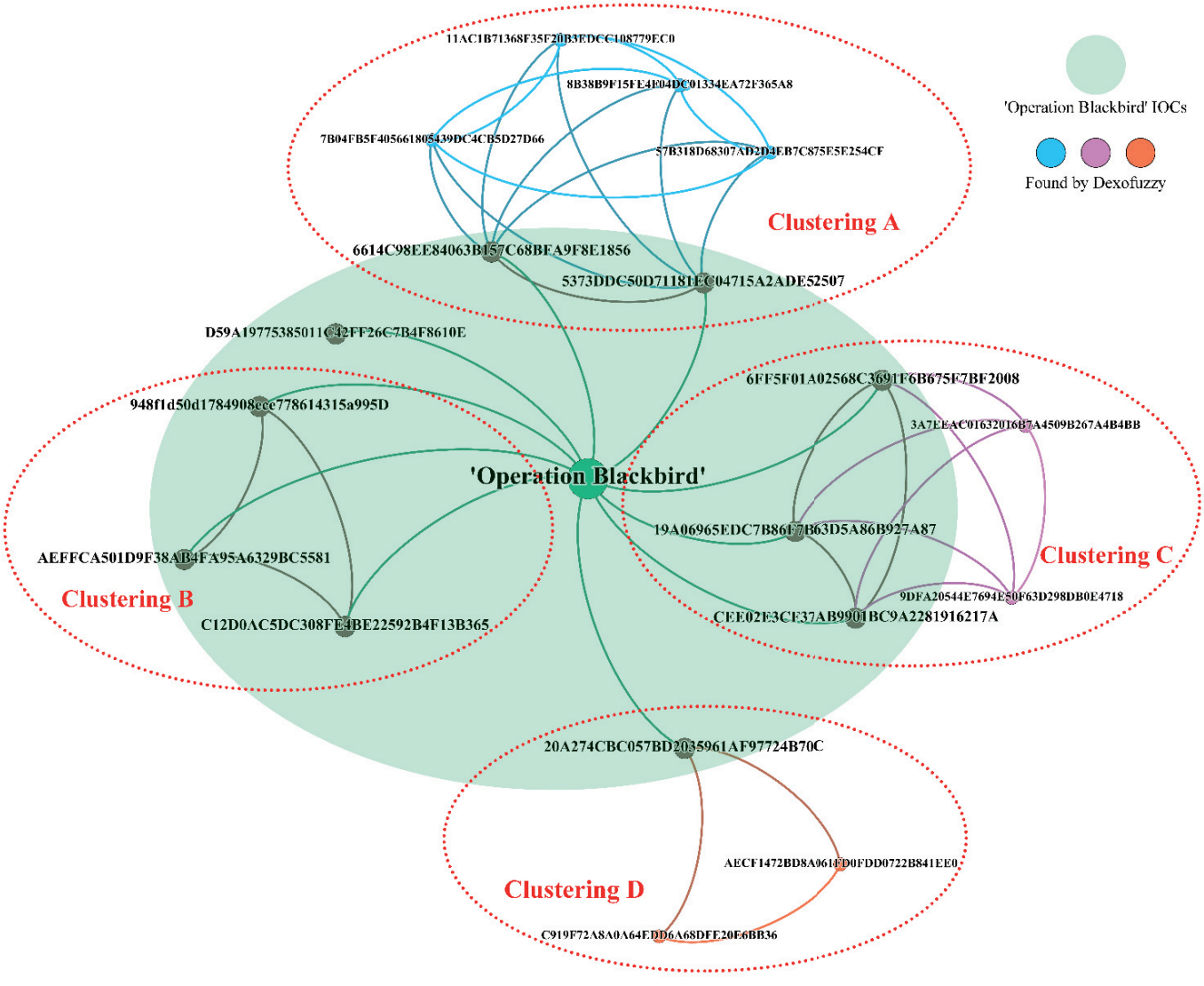


Figure 14: Dexofuzzy 7-gram search of 'Operation Blackbird' samples.

A total of eight variants (excluding the original malware) were found from the 10 malware samples from the IOCs [22] of the 'Operation Blackbird' report. Table 2 presents the MD5s of the searched malware samples and the detected Dexofuzzy 7-gram tokens, which are highlighted in bold.

We have analysed the samples in Clustering D based on the IOC (MD5: 20a274cbc057bd2035961af97724b70c), which is indicated in the report 'Operation Blackbird', to identify the similarity of the samples from the clustering results. Table 3 shows the MD5 and Dexofuzzy hashes which are created from samples in 'Operation Blackbird' and Clustering D, and Figure 15 describes the class structure of three malware samples.

Label	MD5	Dexofuzzy	Size (bytes)
Clustering A	11ac1b71368f35f20b3edcc108779ec0	768:xxRjpEiGPZJXw1rekq**FPwmk/b**Gagrfk7mFaoEcz5T3sciP48k3BxcW:xjpEpRJXw1CzIZBRiF3d3NiQ8k3oW	437,435
	57b318d68307ad2d4eb7c875e5e254cf	768:xxRI434VRH5**FPwmk/b**Gagrfk7mFaoEcz5TIsciP48k3Bxcx:xH34VRHLIZBRiF3dINiQ8k3ox	1,903,214
	7b04fb5f405661805439dc4cb5d27d66	768:xxRzQsj7UZJWmGm09**FPwmk/b**Gagrfk7mFaoEcz5T3sciP48k3BxcW:xzQsvGJWmGjXIZBRiF3d3NiQ8k3oW	420,495
	8b38b9f15fe4f04dc01334ea72f365a8	768:xxRzQsj7pZJWmGm09**FPwmk/b**Gagrfk7mFaoEcz5T3sciP48k3BxcW:xzQsvPJWmGjXIZBRiF3d3NiQ8k3oW	420,443
Clustering C	3a7eeac01632016b7a4509b267a4b4bb	12:**l6rjux+**B9xa3GUGrW/JU/fX4bznCrrNNlu8LL+GhUtKjxH+YrbB99QuJVQQQQL5b:0ux+B9MxGrW/JU//4bzerNNlu8LLhsKv	93,272
	9dfa20544e7694e50f63d298db0e4718	12:**l6rjux+**B9xa3GUGrW/JU/fX4bznCrrNNlu8LL+GhUtKjxH+YrbB99QuJVQQQQL5b:0ux+B9MxGrW/JU//4bzerNNlu8LLhsKv	24,514
Clustering D	aecf1472bd8a061fd0fdd0722b841ee0	24:CVb9X92**WLLLLLL**LTgGyplomnwlomx0+NssWwftQgK7ZHOV82:wd9BAlwGsWwlcYD	726,261
	c919f72a8a0a64edd6a68dfe20e6bb36	24:Wx4KAKtRQbIk4ldSZ2**WLLLLLL**L0NENCSVtxgG5:kHvwMk4ldSZWGxZ5	11,094,249

Table 2: Sample IOCs and Dexofuzzy.

Label	MD5	Dexofuzzy	Size (bytes)
Operation Blackbird IOC	20a274cbc057bd2035961af97724b70c	12:H7Oe0yovJkUTSX9Xh75L7xZ2Pe**LLLLLLL**0NEN/aVtXp6gg695:Ha9vJkU2tXhzZ2WLLLLLL0NENetxgG5	1,897,320
Clustering D	aecf1472bd8a061fd0fdd0722b841ee0	24:CVb9X92W**LLLLLLL**TgGyplomnwlomx0+NssWwftQgK7ZHOV82:wd9BAlwGsWwlcYD	726,261
	c919f72a8a0a64edd6a68dfe20e6bb36	24:Wx4KAKtRQbIk4ldSZ2W**LLLLLLL**0NENCSVtxgG5:kHvwMk4ldSZWGxZ5	11,094,249

Table 3: IOCs and Dexofuzzy.

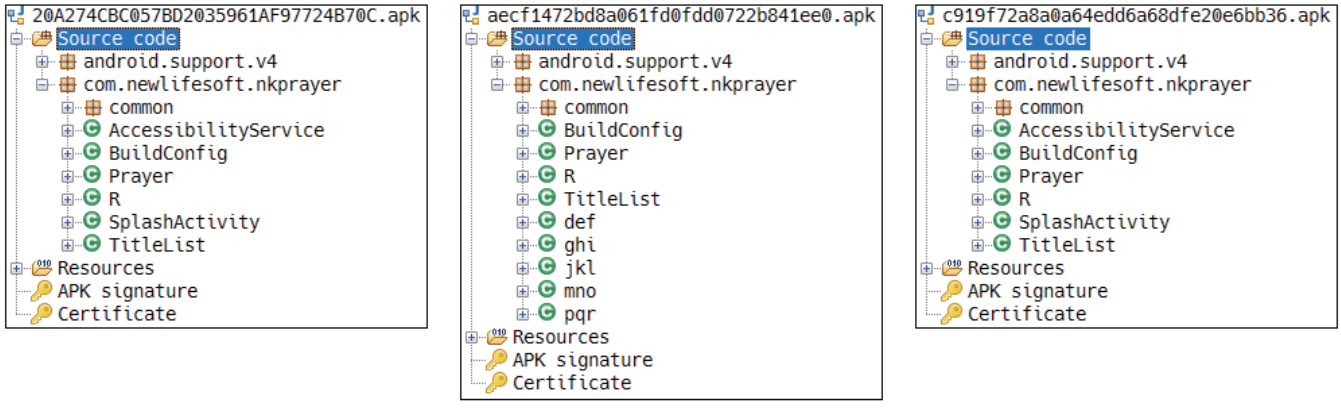


Figure 15: Class structure of malware in 'Operation Blackbird' and Clustering D.

Analysing three malware samples revealed that the same source code was partially reused in '20a274cbc057bd2035961af97724b70c' and 'c919f72a8a0a64edd6a68dfe20e6bb36', and that 'aecf1472bd8a061fd0fdd0722b841ee0' was created by adding the logic to specify the malicious behaviours. Figure 16 illustrates the logic added in malware samples.

Figure 16: Logic added in malware variants.

4.3 AlienVault OTX reports relationship

Finally, we have investigated the correlation using Dexofuzzy based on 2,494 IOCs extracted from 74 *AlienVault* OTX reports released by multiple vendors. Appendix A represents the correlation of 74 *AlienVault* OTX reports. We have selected 465 IOCs from 15 of the *AlienVault* OTX reports and clustered the samples to analyse the correlation between the reports in this experiment. Figure 17 shows that there are four types of relationship in the results of analysing the 15 reports, which were clustered into six groups.

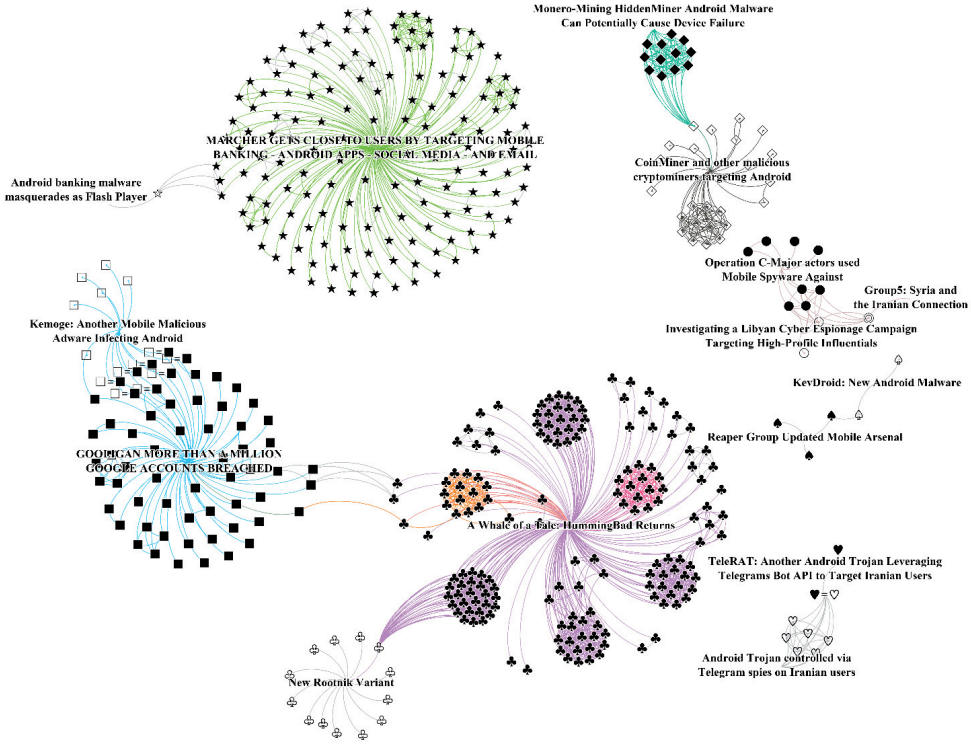


Figure 17: Relationship of 15 AlienVault OTX reports.

4.3.1 Relationship of samples containing similar opcode sequence

The first type is the relationship, in which opcode sequence is similar, clustering nine reports into four reports, as shown in Figure 18.

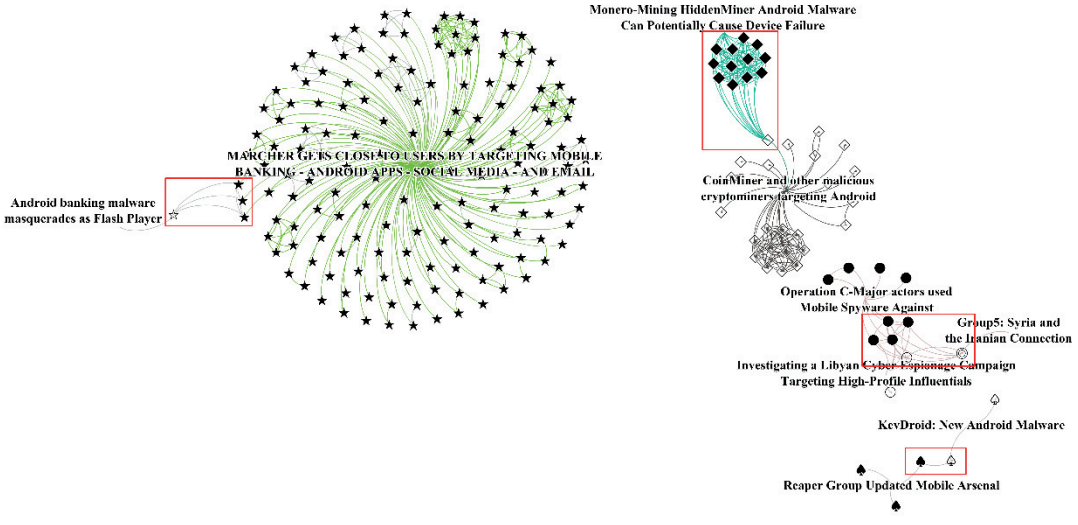


Figure 18: Clustering reports containing similar opcode sequence.

Tables 4 to 7 show clustering results (MD5, Dexofuzzy) of the set of malware samples containing a similar opcode sequence. The samples in each report are clustered by similar logic, and the clustered reports are correlated according to the themes.

Title	MD5	Dexofuzzy	Size (bytes)
Android banking malware masquerades as Flash Player	78c2444fe15a8e58c629076781d9442a	96:VsMRXFTmw+cFcpF1THYb6Hi4W**493J05z**Ro/+8257Nx1FwxocRVT:VsMOKSe6HRWc0By+8211FEocRVT	212,981
MARCHER GETS CLOSE TO USERS BY TARGETING MOBILE BANKING - ANDROID APPS - SOCIAL MEDIA - AND EMAIL	140687aa4d4fc70175c7df1d737d5515	96:5yMzj+xFTmXrcupF1THvb6Hi47**493J05z**MFomUZ91k7G1bZPPfpRea:5yMzjjIuZ6HR7c0BPmUZ9N1FPXpRea	212,985
	c918c977d48855d115527eddde7dbc99	96:zP6jkAKcWFTmopF1THPb6Hi47**493J05z**bJ6O+6197d1bZ3JhBe4:zP6jkAZzoJ6HR7c0BbT+6h1F5hBe4	212,995
	f741d7f608a826e96d06a549602b1ce2	96:VWjY++FTm4cFcpF1THPb6Hi47**493J05z**nMmqzaKQk7d1bZq+xrBeH:VWjYWvSJ6HR7c0BnIzaKH1FlxrBeH	212,973

Table 4: Sample IOCs and Dexofuzzy.

Title	MD5	Dexofuzzy	Size (bytes)
Monero-Mining HiddenMiner Android Malware Can Potentially Cause Device Failure	530bd6c95c3a79c04f49880a44c348db	384:H3Ddb**m8whYHV**xIiKXB+O51BVapknL+G2Po1V2CxETzHz:xfB1xIiKXB+gV6knL+GOo1V2CxETzT	3,281,844
	a13126ed31b3a7982133ff57e6f9676d		3,281,844
	659909c20269c630372eac4878e679ca	384:H3DEb**m8whYHV**xIiKUO51BVapknL+G2Po7V2CxETzHz:YfB1xIiKUgV6knL+GOo7V2CxETzT	2,725,223
	73415fbf16952894e0620b40766d9e2f		2,725,223
	a765d2829b80d812b321c663d8d8320e		2,725,223
	c18f39c4b09e542926d728195b88e418		2,725,223
	c36475ede88631a74f046bd2d4c96405		2,725,223
	ef161923c7a6f99d134467ca21e34410		2,725,223
	fffb8d51838af6bb742e84b8b16239bb		2,725,223
	642bef4824d549ac56520657a1868913	384:i/C**m8whYHV**xIiKSO51BVapknL+G2Po7V2CxETzHz:4yB1xIiKSgV6knL+GOo7V2CxETzT	4,848,132
	a0f776e61cf4ddc55c28051583fbb28e		4,848,132
	e24a0d6b17a9dbf0456bbf4bb93adb25		4,848,132
CoinMiner and other malicious cryptominers targeting Android	766055b991805fe8ef0a1c96643a98a1	384:H3Ddb**m8whYHV**xIiKXB+O51BVapknL+G2Po1V2CxETzHz:xfB1xIiKXB+gV6knL+GOo1V2CxETzT	3,281,844

Table 5: Sample IOCs and Dexofuzzy.

Title	MD5	Dexofuzzy	Size (bytes)
Operation C-Major actors used Mobile Spyware Against Targets	11ba93d968bd96e9e9c9418ea1fdcbbc	384:**qK9ydh5i**x2PaQdyUrrrr6CpXX5asWMoumGTdhJl6w39qfcknpWCYgapOaUyHYrrZ:V9ShwxeQCpXX57IfckkPgap5Zs	202,743
	af046d94f254a3f85a0ba731562a05c5		216,084
	ce59958c01e437f4bdc68b4896222b8e		196,513
	dfd2eca84919418da2fa617fc51e9de5		216,081
Group5: Syria and the Iranian Connection	8ebeb3f91cda8e985a9c61beb8cdde9d	384:**qK9ydh5**ix2PaQdyUrrrr6CpXX5asWMoumGTdhJl6w39qfcknpWCYgapOaUyHYrrZ:V9ShwxeQCpXX57IfckkPgap5Zs	206,939
Investigating a Libyan Cyber Espionage Campaign Targeting High-Profile Influentials	93ebc337c5fe4794d33df155986a284d	384:**qK9ydh5**ix2PaQdyUrrrr9CpXRgasWMoumGTdhJl6w39qfcknpWCYgaLkaUyH52W2:V9ShwxerCpXRg7IfckkPgaLrgeQ	260,991

Table 6: Sample IOCs and Dexofuzzy.

Title	MD5	Dexofuzzy	Size (bytes)
KevDroid: New Android Malware	56b1f4800fa0e083caf0526c3de26059	3072:K**EKQgvB+**ZNXB/ZTUKvcb0crwIbRo TX5DySGpUfpnKd6p76pw66M6pB66p8tAh pop:jvgv+NXHBSwIwDyjCYeTyFnZoixug5	3,820,830
Reaper Group Updated Mobile Arsenal	d6abaa07f7e525153116c98412115b2e	3072:E**EKQgvB+**ZNXB/ZTUKvcb0crwIbRkT X5DySGpUfpnKd6p76pw66M6pB66p8tAhpop: Nvgv+NXHBSwIkDyjCYeTyFnZoixug5	3,895,493

Table 7: Sample IOCs and Dexofuzzy.

4.3.2 Relationship of identical samples in reports

The second is the group of samples including the same MD5, which clusters similar samples, leading to the result that there is a relationship among the analysed reports. Figure 19 shows the relationship between the reports, in which similar malware samples are clustered.

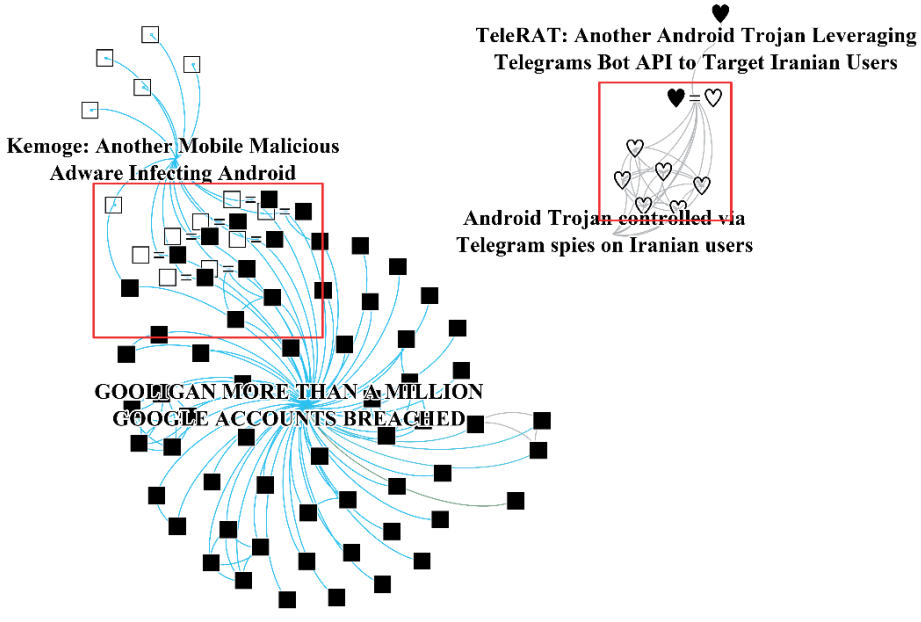


Figure 19: Clustering identical samples in reports.

Table 8 (on the following page) describes two reports where similar Telegram malware was examined, enabling us to clearly identify if those samples are similar using Dexofuzzy.

Table 9 (on the following page) shows that there are eight malware samples containing the same SHA-256 in the reports 'Kemoge: Another Mobile Malicious Adware Infecting Android' and 'GOOLIGAN MORE THAN A MILLION GOOGLE ACCOUNTS BREACHED'. Additionally, three more similar samples were found in the IOCs of the report 'GOOLIGAN MORE THAN A MILLION GOOGLE ACCOUNTS BREACHED' based on the identified samples.

4.3.3 Clustering malware samples using packer

The third type clusters malware samples that are packed using the same logic. In this case, however, there is no correlation between the reports. Figure 20 shows the clustering results of the malware samples that are packed by similar packers *Qihoo* and *Tencent*.

Title	MD5	Dexofuzzy	Size (bytes)
TeleRAT: Another Android Trojan Leveraging Telegrams Bot API to Target Iranian Users	9d23f7688a82d487a8bb87df19cb2426	768:**L6T++XM**pGVznLwCtyaipLbV8wyNN7HNNzNNZczC:L6TeG1v2Laf77HX	412,903
Android Trojan controlled via Telegram spies on Iranian users	34be73f9fdccc152530f2d6cc26cc640		413,917
	356f50c4202d6e96462484004d06f25e		413,943
	6a5f850d5f6a319bba2326a7e015dc97		417,678
	7399e38c0729c122d02a6085391cbb5a		467,413
	a6c6daed941a33248c5232a4507ee726		413,915
	3f13c5c6de3139ecf86120df58cc4b53	768:**L6T++XM**pGVznLwCtyaipLbV8wyNN7HNNzNNZcztMrAnwMUr/ZL:L6TeG1v2Laf77HWMrAnHUr/d	563,806

Table 8: Sample IOCs and Dexofuzzy.

Title	MD5	Dexofuzzy	Size (bytes)
Kemoge: Another Mobile Malicious Adware Infecting Android & GOOLIGAN MORE THAN A MILLION GOOGLE ACCOUNTS BREACHED	0c67d0919e574a6876c73118260368ee	6144:y2EM9Uz**oyIqvv5**RqdGMhukivPW2UX4LtT2VSL+J:xphuWdXcTb+J	4,305,803
	162cb09e2eebd595eae2617cd3af1d0d	1536:**2mjgynB**C0okyF32Bc688O/daKTmYv01pTP7TiePwGbQCfY7b/0IY9jXFyv:2mjjn4D12TO/daKTf6p7tf16D0H9jXF4	5,679,125
	1be29a6622543f6f5063eda1d83a4e49	768:+dtDwG46p13cVpVCORmXrqRu/LWzxntTUdcFXjT96ppFtpHb/0Dd5Y9j8m+FZx54:aW6pMKOgqzbw37b/0bY9jDFya	7,932,730
	7cd86d83d916dbd9b04d0e7e4f9ff6e8	1536:7U1c27yCCb7WnWW8mtxXW79INd9BZC9jDQPdn:I227yCQSbn/WAd09jDQ1n	4,257,499
	abaf6cb1972d55702b559725983e134a	3072:r2L6sy4CDTN0QFKY9jeFn9wC2/7kqTN18CcGpunZZUWjm6dUY**vWy6Gw1**tqPqjdo5:r2usyHTN0MGxFIN18O9W7Xq2wovvavs	7,010,193
	b36a751d72e2bdea80e7ff72b6fb3a41	6144:y280dGmoXd9id49zvvRck2XMxBP5HhA5J:KhPPnCJ	9,008,353
	bf6dc2f78baed212f6aa4268da086e09	768:LdlewFo1cxyn3vjSo1Z0mAwazUGK6HHHH51F5Ga5QZWN4LBjNFE3laFbIlLXBF6j:3i1ci7faOSW79INQBZ09jDwPdn	3,068,850
	cec85188308644273332d00d633ab875	1536:**2mjgynB**C0okyF32Bc688O/KOhTS01pTP7TFePwGbQJYnfY7S:2mjjn4D12TO/KOhTS6p7Ifb6S	5,000,787

Table 9: Sample IOCs and Dexofuzzy.

Title	MD5	Dexofuzzy	Size (bytes)
GOOLIGAN MORE THAN A MILLION GOOGLE ACCOUNTS BREACHED	5b446ec92f1cf0a2a06fbe66a95a6c89	3072:2+y4CDTN0Qw7Nw968CcGpunZZUWjm6dUY**vWy6Gw1**tqPqjtXYklAIvvX4vW:2+yHTN0n5x8O9W7XqlzvvovW	4,134,488
	d7b8e2001ea50c008a6ed068cdbb716c	6144:w**oyIqvv5**RqdGShukivPW2UX4LtTmGtZK2:1PhuWdXcTNtr	3,120,712
	ef835c570bed7d36b8a935a6b7d85b8a	3072:LsyNRmWsOpC/PvvPpAadG8r767yEjRmovi6p0I**vWy6Gw1**twIPoV+dLnSmitW8Xl3:Lsyv5SvvPFdG1fjUov/0pocdbS/t3	2,471,905

Table 9 contd.: Sample IOCs and Dexofuzzy.

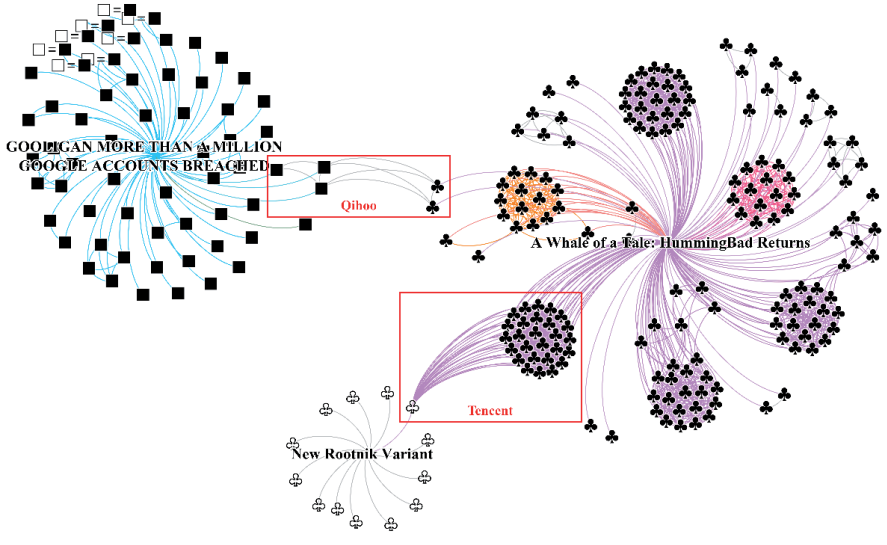


Figure 20: Clustering malware samples by types of packers.

Figure 21 displays the clustered malware samples using the packer *Qihoo*, and Table 10 indicates the details of the five clustered malware samples from the reports from 'GOOLIGAN MORE THAN A MILLION GOOGLE ACCOUNTS BREACHED' and 'A Whale of a Tale: HummingBad Returns.'

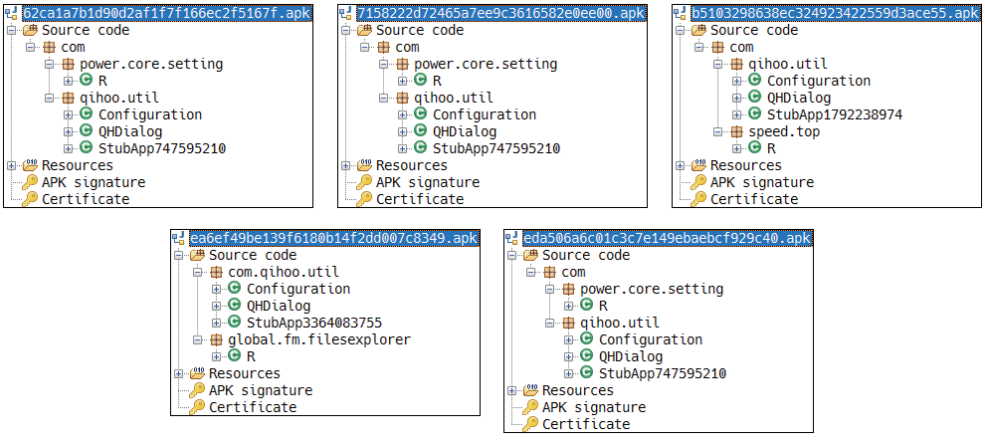


Figure 21: Class structure of sample using Qihoo packer.

Title	MD5	Dexofuzzy	Size (bytes)
GOOLIGAN MORE THAN A MILLION GOOGLE ACCOUNTS BREACHED	eda506a6c01c3c7e149ebaebcf929c40	6:z9m3**KnKA4fd**QV/EughUtDK1LAYDK4HSMEs+cSVPZw:z9m364fyVsu5tDU12aSNcS/w	1,366,251
	62ca1a7b1d90d2af1f7f166ec2f5167f	6:**zLMXv56**Pikl7**KnKA4fd**QV/EughUtDK1LAYDK4HSMEs+cSVPZw:zQh6Pikh64fyVsu5tDU12aSNcS/w	1,379,340
	7158222d72465a7ee9c3616582e0ee00		1,369,341
A Whale of a Tale: HummingBad Returns	b5103298638ec324923422559d3ace55	6:**zLMXv56**Pikl7KnKztsk4fdQV/PfwVrUtDsqxhYDK4HSerDc+coHbPWk:zQh6Pikhjs/fyV4VAtDLQ2aSefcy1	3,919,190
	ea6ef49be139f6180b14f2dd007c8349		4,014,418

Table 10: Sample IOCs and Dexofuzzy.

The 28 samples in the reports 'A Whale of a Tale: HummingBad Returns' and 'New Rootnik Variant' were clustered by the similarity of the logic used in the *Tencent* packer. Figure 22 presents the result of analysing the class structure of four representative malicious codes from each cluster based on Dexofuzzy, and Table 11 shows a list of malware samples packed by *Tencent* using 7-gram search of Dexofuzzy.

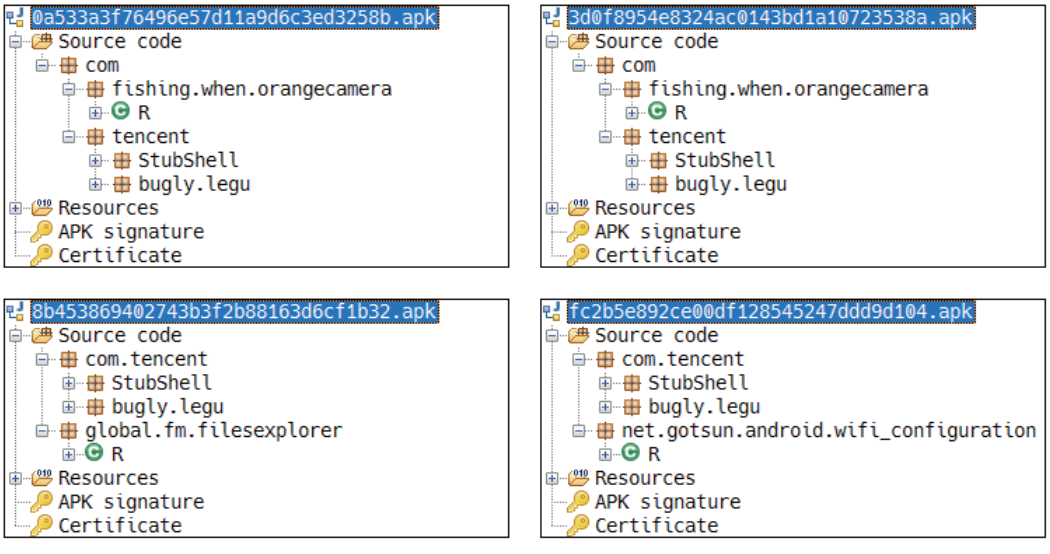


Figure 22: Class structure of sample using Tencent packer.

4.3.4 Incorrect clustering caused by excessive usage of SDK

The fourth cluster of malware samples contains more SDK opcode sequence than malicious opcode sequence due to excessive usage of SDK, as shown in Figure 23.

Figure 24 shows an example of two samples which are incorrectly clustered due to the reason mentioned above, and Table 12 details the two samples that are detected by the 7-gram search of Dexofuzzy. As the 7-gram search of Dexofuzzy is based on the similarity of opcode sequence, malware samples are clustered according to the SDK usage rather than the logic similarity in case the proportion of malicious logic is relatively lower than that of SDK usage.

Title	MD5	Dexofuzzy	Size (bytes)
A Whale of a Tale: HummingBad Returns	5f512bf1f51141d4201dcfe819dc2165	384:9IIi**CtwXLNR**twkqrEEDjyqJP21bS8FhYLZbL9:9IIiCtmPwjyq521WLZbh	13,814,432
	8b453869402743b3f2b88163d6cf1b32		3,891,624
	0cc5d5436d7ff42886b74e89cf6f7047	384:9IIK**CtwXLNR**twkqrEEDjyqJP21bS8FhYLZbL9:9IIKCtmPwjyq521WLZbh	3,192,437
	15be23d3724fafaa16c7e68f1f6466f6		3,201,547
	2e3990fd4af3ea26066a7180b24bb435		15,303,085
	3d0f8954e8324ac0143bd1a10723538a		15,261,131
	4c635fcce49743de86d8f9cc58d2de8b		3,671,850
	5ee2367fa2c4f8dc79a9d466148b3819		15,259,287
	69e30a40e68d85140bd881f195bc791a		3,671,824
	7c7b32233f94e850703880caee1bac15		2,593,776
	81426b5812f164f16daf0c59e0593dbe		3,853,283
	9388b89593e515e89263c113d1245e04		14,858,701
	9e8b27b00da7f56371125c5659b09f20		15,316,677
	b7c173fa6b86ba87f13a4b6221646b49		14,781,770
	ceab2234b547df62747d901397b419d2		2,615,708
	dc34055f88595063cc66baf238486919		15,245,522
	e3c22b146d4cf6aa70292ee12622afeb		12,136,614
	0a533a3f76496e57d11a9d6c3ed3258b	384:9IIV**CtwXLNR**twkqrEEDjyqJP21bS8FhYLZbL9:9IIVCtmPwjyq521WLZbh	15,303,017
	1aeb25ac71b8fc1b76f87e2db5f7d650		16,093,071
	296bed0e48929cd83b84624239683ded		12,978,770
	533fa599f95864701025b205cd24226e		14,762,818
	77cf656556bfdcd0bbdfd7a8d48702de		11,831,615
	9d7adfe4e98ed8dc0623c6a6bed85adf		12,970,387
	a7917eacaf02c715a8e232ae18551a09		14,805,954
	deca693848b8926a32ae1048e02d5b52		4,045,436
	e69ca52ff99ac45c30a7eca833bf17c0		3,943,398
	eaf5620c94ca479f49593350e0e53052		16,090,695
New Rootnik Variant	fc2b5e892ce00df128545247ddd9d104	384:QIIdV**CtwXLNR**twkqrEEDjyqJP21bS8FhYLZbL9:QIIdVCtmPwjyq521WLZbh	1,810,587

Table 11: Sample IOCs and Dexofuzzy.

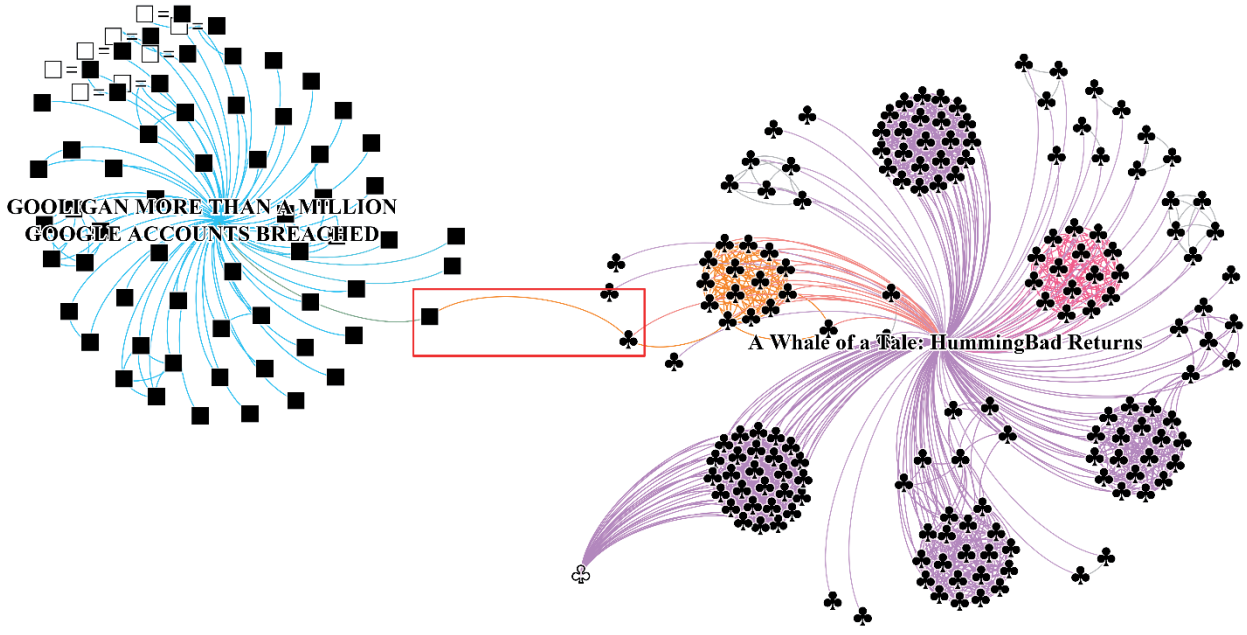


Figure 23: Incorrect clustering caused by excessive usage of SDK.

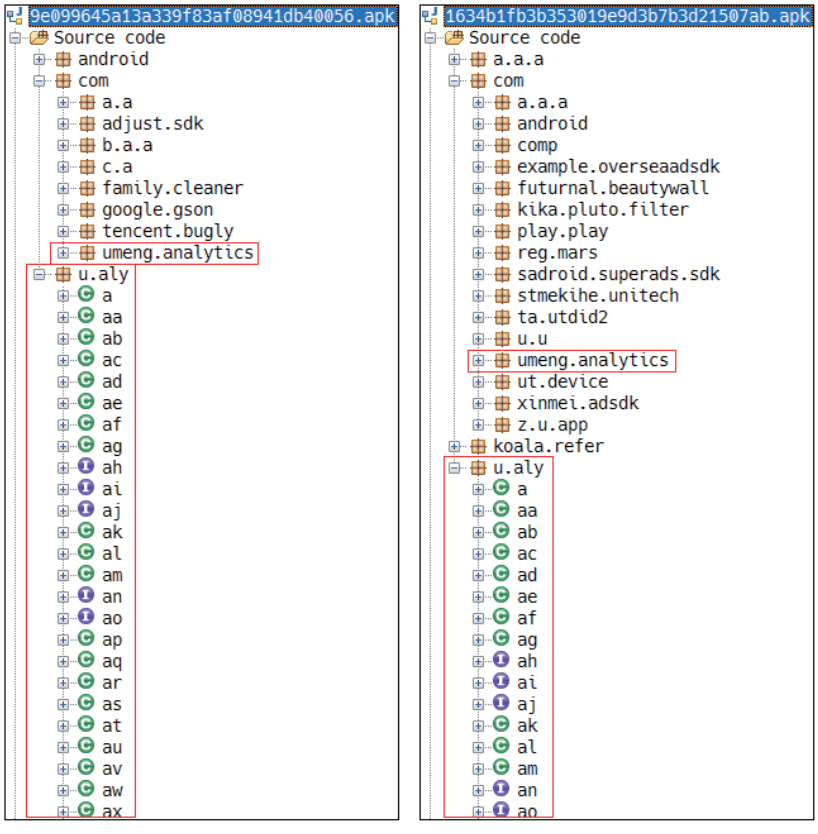


Figure 24: Examples detected as similar samples using the SDK.

Title	MD5	Dexofuzzy	Size (bytes)
GOOLIGAN MORE THAN A MILLION GOOGLE ACCOUNTS BREACHED	1634b1fb3b353019e9d3b7b3d21507ab	768:XgwscmCnqs+13uXFAF5UPgoj8iS**fU5Ngk5**bqlA9D/Hx:X+cmCnMBkPgojh1NfbqlABJ	429,516
A Whale of a Tale: HummingBad Returns	9e099645a13a339f83af08941db40056	768:BJlKnhSX7RF96rXlOaLQWmjI0T21utTj8i0**fU5Ngk5**bqlA9D/Hx:BJl3L8jLyM01Bjh/NfbqlABJ	1,693,075

Table 12: Sample IOCs and Dexofuzzy.

5. CONCLUSION

As *Android* malware has shown a tendency toward large-scale distribution, users need more efficient and reliable analysis information to cope with the rapidly changing security environment. This paper proposes the methodology 'Dexofuzzy' to overcome the limitations of hardware resources or analysis environment. Dexofuzzy measures the similarity among a large number of *Android* malware variants by using opcode of the Dex file. In addition, Dexofuzzy utilizes *Elasticsearch*'s N-gram tokenizer, making it easier to perform a quick analysis of similarities.

In this paper, we have used Dexofuzzy to figure out the correlation based on 2,494 IOCs from 74 *AlienVault* OTX reports written by various vendors (see Appendix) and identified the correlation by mainly utilizing 15 reports and 465 samples among those reports. In addition, the eight more unknown samples were found among the 212,955 samples by utilizing the similarity search 'Dexofuzzy' with the IOCs of the report 'Operation Blackbird.' However, Dexofuzzy has limitations in that samples are incorrectly clustered when they are packed with the same packer or the proportion of SDK usage is excessive. It is recommended to use existing static and dynamic analysis methods along with Dexofuzzy to overcome these limitations, which helps us to more effectively respond to the large and rapidly evolving sets of *Android*-based malware. This tool is an open-source project that can be downloaded from the *GitHub* repository [23] and is also provided in the *Pypi* repository [24].

REFERENCES

[1] Mobile malware evolution 2018. https://securelist.com/mobile-malware-evolution-2018/89689/. Cited May, 2019.

[2] Cyber attacks on Android devices on the rise. https://www.gdatasoftware.com/blog/2018/11/31255-cyber-attacks-on-android-devices-on-the-rise. Cited May, 2019.

[3] 2018 Mobile Threat Landscape. https://www.trendmicro.com/vinfo/au/security/research-and-analysis/threat-reports/roundup/2018-mobile-threat-landscape. Cited May, 2019.

[4] Rashidi B.; Fung, C. A Survey of Android Security Threats and Defenses. J. Wirel. Mob. Networks, Ubiquitous Comput. Dependable Appl., vol. 6, no. 3, pp. 3–35, 2015.

[5] Faruki, P.; Bharmal, A.; Laxmi, V.; Ganmoor, V.; Gaur, M. S.; Conti, M.; Rajarajan, M. Android security: A survey of issues, malware penetration, and defenses. IEEE Commun. Surveys Tutorials, vol. 17, no. 2, pp. 998–1022, Mar.–Apr. 2015.

[6] Sufatrio, D. J.; Tan, J.; Chua, T.-W.; Thing, V. L. L. Securing Android: A Survey, Taxonomy, and Challenges. ACM Comput. Surveys, vol. 47, no. 4, pp. 1–45, 2015.

[7] Tam, K.; Feizollah, A.; Anuar, N. B.; Salleh, R.; Cavallaro, L. The evolution of Android malware and Android analysis techniques. ACM Comput. Surveys, vol. 49, no. 4, pp. 1–41, 2017.

[8] Yu, B.; Fang, Y.; Yang, Q.; Tang, Y.; Liu, L. et al. A survey of malware behavior description and analysis. Frontiers of Information Technology and Electronic Engineering, Volume 19(5), 583-603, 2018.

[9] Wang, J.; Shen, H. T.; Song, J.; Ji, J. Hashing for similarity search: A survey. CoRR, abs/1408.2927, 2014.

[10] Oliver, J.; Chun, C.; Yanggui, C. TLSH -- A Locality Sensitive Hash. In Cybercrime and Trustworthy Computing Workshop (CTC), 2013 Fourth, 2013, pp.7–13.

[11] Classifying Malware using Import API and Fuzzy Hashing – impfuzzy. https://blogs.jpcert.or.jp/en/2016/05/classifying-mal-a988.html. Cited May, 2019.

[12] Android Runtime (ART) and Dalvik. https://source.android.com/devices/tech/dalvik. Cited May, 2019.

[13] Zhou, W.; Zhou, Y.; Jiang, X.; Ning, P. Detecting Repackaged Smartphone Applications in Third-Party Android Marketplaces. In Proceedings of the 2nd ACM Conference on Data and Application Security and Privacy (CODASPY), 2012.

[14] Ko, J.; Shim, H.; Kim, D.; Jeong, Y.-S.; Cho, S.-j.; Park, M.; Han, S.; Kim, S. B. Measuring similarity of Android applications via reversing and k-gram birthmarking. In Proc. of the 2013 Research in Adaptive and Convergent Systems (RACS'13), Montreal, Quebec, Canada. ACM, October 2013, pp.336–341.

[15] Zhang, F.; Huang, H.; Zhu, S.; Wu, D.; Liu, P. Viewdroid: Towards obfuscation-resilient mobile application repackaging detection. In Proc. 7th ACM Conf. Secur. Privacy Wireless Mobile Netw., 2014, pp.25–36.

[16] Myles G.; Collberg, C. K-gram based software birthmarks. In Proceedings of the 2005 ACM symposium on Applied computing, 2005.

[17] Kornblum, J. Identifying almost identical files using context triggered piecewise hashing. Digital investigation, vol. 3, pp.91–97, Sep. 2006.

[18] Wallace, B. Optimizing ssDeep for use at scale. Virus Bulletin. Cited Nov. 2015.

[19] Intezer Community Tip: How to Optimize ssdeep Comparisons with ElasticSearch. https://www.intezer.com/intezer-community-tip-ssdeep-comparisons-with-elasticsearch. Cited May, 2019.

[20] Apktool. https://ibotpeaches.github.io/Apktool. Cited Jul., 2019.

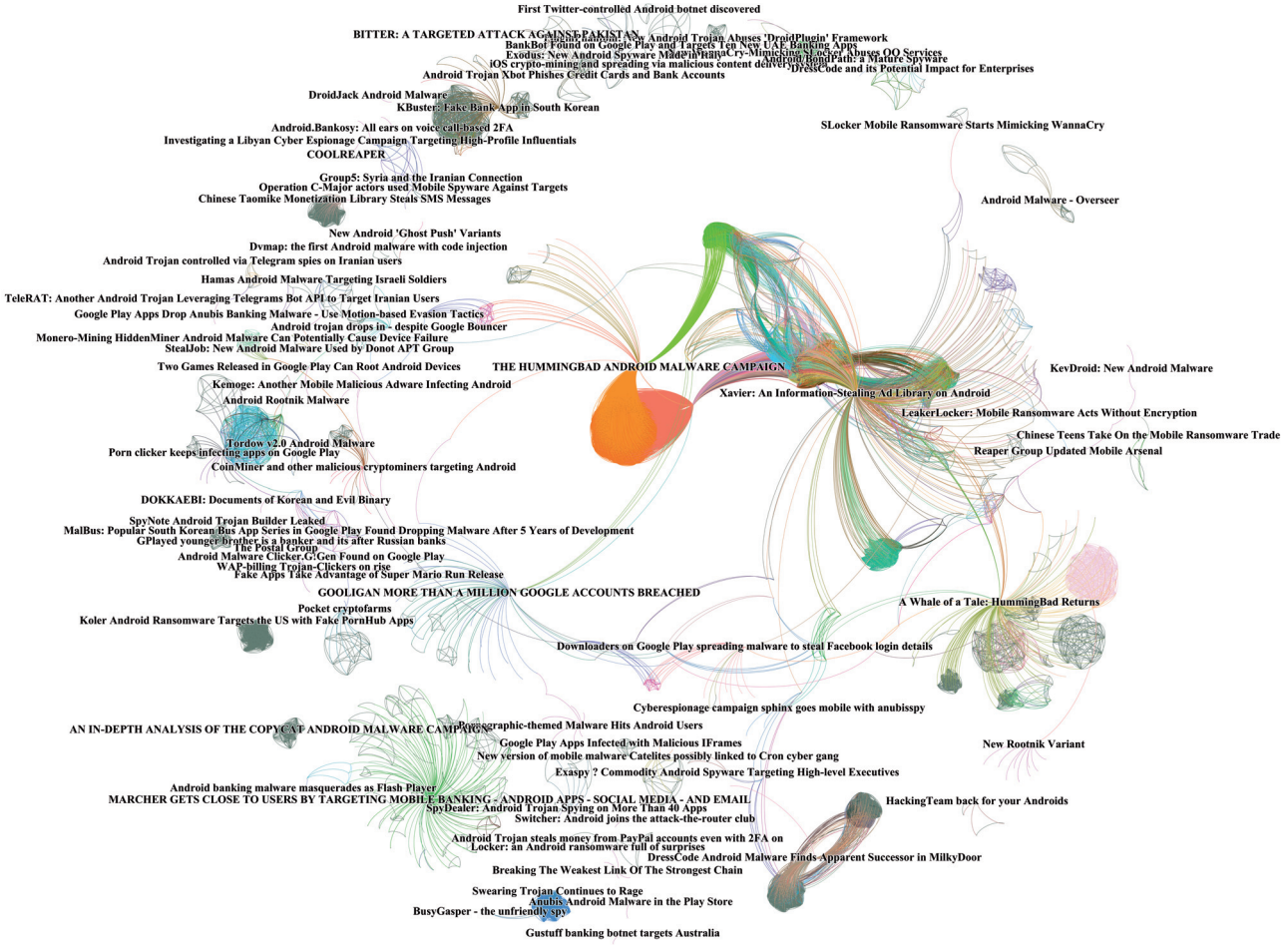
[21] Bastian, M.; Heymann, S.; Jacomy, M. Gephi: An open source software for exploring and manipulating networks. In ICWSM. The AAAI Press, 2009.

[22] 'Operation Blackbird'. https://blog.alyac.co.kr/2035. Cited Jul., 2019.

[23] Open Source Project – 'Dexofuzzy: Dalvik EXecutable Opcode Fuzzyhash'. https://github.com/ESTsecurity/Dexofuzzy.

[24] Python Package Index – 'Dexofuzzy: Dalvik EXecutable Opcode Fuzzyhash'. https://pypi.org/project/dexofuzzy.

APPENDIX A: CORRELATION OF 74 ALIENVAULT OTX REPORTS



[1] StealJob: New Android Malware Used by Donot APT Group. https://ti.360.net/blog/articles/stealjob-new-android-malware-used-by-donot-apt-group-en/. Cited Jul., 2019.

[2] Gustuff banking botnet targets Australia. https://blog.talosintelligence.com/2019/04/gustuff-targets-australia.html. Cited Jul., 2019.

[3] Exodus: New Android Spyware Made in Italy. https://securitywithoutborders.org/blog/2019/03/29/exodus.html. Cited Jul., 2019.

[4] KBuster: Fake Bank App in South Korean. https://ti.360.net/blog/articles/kbuster-fake-bank-app-in-south-korean-en/. Cited Jul., 2019.

[5] MalBus: Popular South Korean Bus App Series in Google Play Found Dropping Malware After 5 Years of Development. https://securingtomorrow.mcafee.com/other-blogs/mcafee-labs/malbus-popular-south-korean-bus-app-series-in-google-play-found-dropping-malware-after-5-years-of-development/. Cited Jul., 2019.

[6] Google Play Apps Drop Anubis Banking Malware, Use Motion-based Evasion Tactics. https://blog.trendmicro.com/trendlabs-security-intelligence/google-play-apps-drop-anubis-banking-malware-use-motion-based-evasion-tactics/. Cited Jul., 2019.

[7] Android Trojan steals money from PayPal accounts even with 2FA on. https://www.welivesecurity.com/2018/12/11/android-trojan-steals-money-paypal-accounts-2fa/. Cited Jul., 2019.

[8] Gplayed's younger brother is a banker and its after Russian banks. https://blog.talosintelligence.com/2018/10/gplayerbanker.html. Cited Jul., 2019.

[9] Roaming Mantis, part III: iOS crypto-mining and spreading via malicious content delivery system. https://securelist.com/roaming-mantis-part-3/88071/. Cited Jul., 2019.

[10] BusyGasper - the unfriendly spy. https://securelist.com/busygasper-the-unfriendly-spy/87627/. Cited Jul., 2019.

[11] Android/BondPath: a Mature Spyware. https://www.fortinet.com/blog/threat-research/android-bondpath--a-mature-spyware.html. Cited Jul., 2019.

[12] Anubis is Back: Are You Prepared? https://news.sophos.com/en-us/2018/08/14/anubis-is-back-are-you-prepared/. Cited Jul., 2019.

[13] DOKKAEBI: Documents of Korean and Evil Binary. https://www.virusbulletin.com/conference/vb2018/abstracts/dokkaebi-documents-korean-and-evil-binary/. Cited Jul., 2019.

[14] Infrastructure and Samples of Hamas' Android Malware Targeting Israeli Soldiers. https://www.clearskysec.com/glancelove/. Cited Jul., 2019.

[15] CoinMiner and other malicious cryptominers targeting Android. https://www.sophos.com/en-us/medialibrary/PDFs/technical-papers/sophos-coinminer-and-other-malicious-cryptominers-tpna.aspx. Cited Jul., 2019.

[16] Reaper Group's Updated Mobile Arsenal. https://researchcenter.paloaltonetworks.com/2018/04/unit42-reaper-groups-updated-mobile-arsenal/. Cited Jul., 2019.

[17] Pocket cryptofarms: Investigating mobile apps for hidden mining. https://securelist.com/pocket-cryptofarms/85137/. Cited Jul., 2019.

[18] Fake AV Investigation Unearths KevDroid. New Android Malware. https://blog.talosintelligence.com/2018/04/fake-av-investigation-unearths-kevdroid.html. Cited Jul., 2019.

[19] Monero-Mining HiddenMiner Android Malware Can Potentially Cause Device Failure. https://blog.trendmicro.com/trendlabs-security-intelligence/monero-mining-hiddenminer-android-malware-can-potentially-cause-device-failure/. Cited Jul., 2019.

[20] TeleRAT: Another Android Trojan Leveraging Telegrams Bot API to Target Iranian Users. https://researchcenter.paloaltonetworks.com/2018/03/unit42-telerat-another-android-trojan-leveraging-telegrams-bot-api-to-target-iranian-users/. Cited Jul., 2019.

[21] Downloaders on Google Play spreading malware to steal Facebook login details. https://blog.avast.com/downloaders-on-google-play-spreading-malware-to-steal-facebook-login-details. Cited Jul., 2019.

[22] New version of mobile malware Catelites possibly linked to Cron cyber gang. https://blog.avast.com/new-version-of-mobile-malware-catelites-possibly-linked-to-cron-cyber-gang. Cited Jul., 2019.

[23] BankBot Found on Google Play and Targets Ten New UAE Banking Apps. http://blog.trendmicro.com/trendlabs-security-intelligence/bankbot-found-google-play-targets-ten-new-uae-banking-apps/. Cited Jul., 2019.

[24] WAP-billing Trojan-Clickers on rise. https://securelist.com/wap-billing-trojan-clickers-on-rise/81576/. Cited Jul., 2019.

[25] New WannaCry-Mimicking SLocker Abuses QQ Services. http://blog.trendmicro.com/trendlabs-security-intelligence/new-wannacry-mimicking-slocker-abuses-qq-services/?linkId=40489549. Cited Jul., 2019.

[26] Unmasking Android Malware: A Deep Dive into a New Rootnik Variant, Part III. http://blog.fortinet.com/2017/07/09/unmasking-android-malware-a-deep-dive-into-a-new-rootnik-variant-part-iii. Cited Jul., 2019.

[27] LeakerLocker: Mobile Ransomware Acts Without Encryption. https://securingtomorrow.mcafee.com/other-blogs/mcafee-labs/leakerlocker-mobile-ransomware-acts-without-encryption/. Cited Jul., 2019.

[28] SpyDealer: Android Trojan Spying on More Than 40 Apps. https://unit42.paloaltonetworks.com/unit42-spydealer-android-trojan-spying-40-apps/. Cited Jul., 2019.

[29] CopyCay: AN IN-DEPTH ANALYSIS OF THE COPYCAT ANDROID MALWARE CAMPAIGN. https://www.checkpoint.com/downloads/resources/copycat-research-report.pdf. Cited Jul., 2019.

[30] SLocker Mobile Ransomware Starts Mimicking WannaCry. http://blog.trendmicro.com/trendlabs-security-intelligence/slocker-mobile-ransomware-starts-mimicking-wannacry/. Cited Jul., 2019.

[31] Koler Android Ransomware Targets the US with Fake PornHub Apps. https://www.bleepingcomputer.com/news/security/koler-android-ransomware-targets-the-us-with-fake-pornhub-apps/. Cited Jul., 2019.

[32] Spyware targets Iranian Android users by abusing messaging app Telegram's Bot API. https://blog.avast.com/spyware-targets-iranian-android-users-by-abusing-messaging-app-telegram-bot-api#hs_cos_wrapper_blog_comments. Cited Jul., 2019.

[33] Analyzing Xavier: An InformationStealing Ad Library on Android. https://documents.trendmicro.com/assets/appendix--analyzing-xavier-an-information-stealing-ad-library-on-android.pdf. Cited Jul., 2019.

[34] Dvmap: the first Android malware with code injection. https://securelist.com/78648/dvmap-the-first-android-malware-with-code-injection/. Cited Jul., 2019.

[35] Marcher Gets Close to Users by Targeting Mobile Banking, Android Apps, Social Media, and Email. https://f5.com/labs/articles/threat-intelligence/malware/marcher-gets-close-to-users-by-targeting-mobile-banking-android-apps-social-media-and-email-26004. Cited Jul., 2019.

[36] DressCode Android Malware Finds Apparent Successor in MilkyDoor. http://blog.trendmicro.com/trendlabs-security-intelligence/dresscode-android-malware-finds-successor-milkydoor/. Cited Jul., 2019.

[37] Swearing Trojan Continues to Rage, Even After Authors' Arrest. https://blog.checkpoint.com/2017/03/21/swearing-trojan-continues-rage-even-authors-arrest/. Cited Jul., 2019.

[38] Google Play Apps Infected with Malicious IFrames. http://researchcenter.paloaltonetworks.com/2017/03/unit42-google-play-apps-infected-malicious-iframes/. Cited Jul., 2019.

[39] Breaking The Weakest Link Of The Strongest Chain. https://securelist.com/blog/incidents/77562/breaking-the-weakest-link-of-the-strongest-chain/. Cited Jul., 2019.

[40] Deep Analysis of Android Rootnik Malware Using Advanced Anti-Debug and Anti-Hook, Part II: Analysis of The Scope of Java. http://blog.fortinet.com/2017/01/26/deep-analysis-of-android-rootnik-malware-using-advanced-anti-debug-and-anti-hook-part-ii-analysis-of-the-scope-of-java. Cited Jul., 2019.

[41] A Whale of a Tale: HummingBad Returns. http://blog.checkpoint.com/2017/01/23/hummingbad-returns/. Cited Jul., 2019.

[42] Switcher: Android joins the 'attack-the-router' club. https://securelist.com/switcher-android-joins-the-attack-the-router-club/76969/. Cited Jul., 2019.

[43] Fake Apps Take Advantage of Super Mario Run Release. http://blog.trendmicro.com/trendlabs-security-intelligence/fake-apps-take-advantage-mario-run-release/. Cited Jul., 2019.

[44] 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/. Cited Jul., 2019.

[45] GOOLIGAN: MORE THAN A MILLION GOOGLE ACCOUNTS BREACHED. http://blog.checkpoint.com/wp-content/uploads/2016/12/Gooligan-Research-Report.pdf. Cited Jul., 2019.

[46] PluginPhantom: New Android Trojan Abuses 'DroidPlugin' Framework. http://researchcenter.paloaltonetworks.com/2016/11/unit42-pluginphantom-new-android-trojan-abuses-droidplugin-framework/. Cited Jul., 2019.

[47] HackingTeam back for your Androids, now extra insecure! http://rednaga.io/2016/11/14/hackingteam_back_for_your_androids/. Cited Jul., 2019.

[48] Exaspy - Commodity Android Spyware Targeting High-level Executives. https://www.symantec.com/connect/blogs/exaspy-commodity-android-spyware-targeting-high-level-executives. Cited Jul., 2019.

[49] Android banking malware masquerades as Flash Player, targeting large banks and popular social media apps. https://blog.fortinet.com/2016/11/01/android-banking-malware-masquerades-as-flash-player-targeting-large-banks-and-popular-social-media-apps. Cited Jul., 2019.

[50] BITTER: a targeted attack against Pakistan. https://www.forcepoint.com/blog/security-labs/bitter-targeted-attack-against-pakistan. Cited Jul., 2019.

[51] DressCode and its Potential Impact for Enterprises. http://blog.trendmicro.com/trendlabs-security-intelligence/dresscode-potential-impact-enterprises/. Cited Jul., 2019.

[52] Investigating a Libyan Cyber Espionage Campaign Targeting High-Profile Influentials. https://cyberkov.com/wp-content/uploads/2016/09/Hunting-Libyan-Scorpions-EN.pdf. Cited Jul., 2019.

[53] Four spyware apps removed from Google Play. https://blog.lookout.com/blog/2016/09/16/embassy-spyware-google-play/. Cited Jul., 2019.

[54] First Twitter-controlled Android botnet discovered. http://www.welivesecurity.com/2016/08/24/first-twitter-controlled-android-botnet-discovered/. Cited Jul., 2019.

[55] Group5: Syria and the Iranian Connection. https://citizenlab.ca/2016/08/group5-syria/. Cited Jul., 2019.

[56] SpyNote Android Trojan Builder Leaked. http://researchcenter.paloaltonetworks.com/2016/07/unit42-spynote-android-trojan-builder-leaked/. Cited Jul., 2019.

[57] DroidJack Uses Side-Load…It's Super Effective! Backdoored Pokemon GO Android App Found. https://www.proofpoint.com/us/threat-insight/post/droidjack-uses-side-load-backdoored-pokemon-go-android-app. Cited Jul., 2019.

[58] From HummingBad to Worse. http://blog.checkpoint.com/wp-content/uploads/2016/07/HummingBad-Research-report_FINAL-62916.pdf. Cited Jul., 2019.

[59] Android Malware Clicker.G!Gen Found on Google Play. https://blogs.mcafee.com/mcafee-labs/android-malware-clicker-dgen-found-google-play/. Cited Jul., 2019.

[60] 'Operation C-Major' Actors Also Used Android, BlackBerry Mobile Spyware Against Targets. http://blog.trendmicro.com/trendlabs-security-intelligence/operation-c-major-actors-also-used-android-blackberry-mobile-spyware-targets/. Cited Jul., 2019.

[61] New Android Trojan 'Xbot' Phishes Credit Cards and Bank Accounts, Encrypts Devices for Ransom. http://researchcenter.paloaltonetworks.com/2016/02/new-android-trojan-xbot-phishes-credit-cards-and-bank-accounts-encrypts-devices-for-ransom/. Cited Jul., 2019.

[62] Android.Bankosy: All ears on voice call-based 2FA. http://www.symantec.com/connect/blogs/androidbankosy-all-ears-voice-call-based-2fa. Cited Jul., 2019.

[63] Android.ZBot banking Trojan uses 'web injections' to steal confidential data. http://news.drweb.com/show/?i=9754&lng=en&c=14. Cited Jul., 2019.

[64] Pornographic-themed Malware Hits Android Users in China, Taiwan, Japan. http://blog.trendmicro.com/trendlabs-security-intelligence/pornographic-themed-malware-hits-android-users-in-china-taiwan-japan/. Cited Jul., 2019.

[65] Chinese Taomike Monetization Library Steals SMS Messages. http://researchcenter.paloaltonetworks.com/2015/10/chinese-taomike-monetization-library-steals-sms-messages/. Cited Jul., 2019.

[66] The Postal Group. https://www.cert.pl/wp-content/uploads/2015/12/The_Postal_Group.pdf. Cited Jul., 2019.

[67] Kemoge: Another Mobile Malicious Adware Infecting Over 20 Countries. https://www.fireeye.com/blog/threat-research/2015/10/kemoge_another_mobi.html. Cited Jul., 2019.

[68] Two Games Released in Google Play Can Root Android Devices. http://blog.trendmicro.com/trendlabs-security-intelligence/two-games-released-in-google-play-can-root-android-devices/. Cited Jul., 2019.

[69] New 'Ghost Push' Variants Sport Guard Code; Malware Creator Published Over 600 Bad Android Apps. http://blog.trendmicro.com/trendlabs-security-intelligence/new-ghost-push-variants-sport-guard-code-malware-creator-published-over-600-bad-android-apps/. Cited Jul., 2019.

[70] Android trojan drops in, despite Google Bouncer. http://www.welivesecurity.com/2015/09/22/android-trojan-drops-in-despite-googles-bouncer/. Cited Jul., 2019.

[71] Locker: an Android ransomware full of surprises. https://www.fortinet.com/blog/threat-research/locker-an-android-ransomware-full-of-surprises.html. Cited Jul., 2019.

[72] Porn clicker keeps infecting apps on Google Play. http://www.welivesecurity.com/2015/07/23/porn-clicker-keeps-infecting-apps-on-google-play/. Cited Jul., 2019.

[73] Attack of the 90s Kids: Chinese Teens Take On the Mobile Ransomware Trade. http://blog.trendmicro.com/trendlabs-security-intelligence/attack-of-the-90s-kids-chinese-teens-take-on-the-mobile-ransomware-trade/. Cited Jul., 2019.

[74] COOLREAPER: The Coolpad Backdoor. https://www.paloaltonetworks.com/content/dam/paloaltonetworks-com/en_US/assets/pdf/reports/Unit_42/unit42-cool-reaper.pdf. Cited Jul., 2019.