DroidGraph: Discovering Android Malware by Analyzing Semantic Behavior

Jonghoon Kwon, Jihwan Jeong, Jehyun Lee, Heejo Lee[1]

Dept. of Computer Science and Engineering, Korea University
Seoul, Republic of Korea
{signalnine, askjjh, arondit, heejo}@korea.ac.kr

***Abstract*—Mobile malware has been recently recognized as a significant problem in accordance with the rapid growth of the market share for smartphones. Despite of the numerous efforts to thwart the growth of mobile malware, the number of mobile malware is getting increased by evolving themselves. By applying, for example, code obfuscation or junk code insertion, mobile malware is able to manipulate its appearance while maintains the same functionality, thus mobile malware can easily evade the existing anti-mobile-malware solutions. In this paper, we focus on Android malware and propose a new method called DroidGraph to discover the evolved Android malware. DroidGraph leverages the semantics of Android malware. More precisely, we transform an APK file for Android malware to hierarchical behavior graphs that represent with 136 identical nodes based on the semantics of Android API calls. Then, we select unique behavior graphs as semantic signatures describing common behaviors for Android malware. In evaluation, DroidGraph shows approximately 87% of detection accuracy with only 40 semantic signatures against 260 real-world Android malware, and no false positives for 3,623 benign applications.**

***Index Terms*—Android Malware, Semantic Analysis**

I. Introduction

The population of mobile malware has been dramatically increased nowadays due to the growth of adoption of smartphones. Since the smartphone is recognized as a private property, people used to store the sensitive information without awareness of the threat of mobile malware. The fact gives a great motivation to attackers to target the smartphone, and in the fact that mobile malware is rampant now, especially on the most popular mobile platform Android. To respond to the growing threats of Android malware, numerous countermeasures have been suggested, but none of them is able to be a fundamental solution.

The major difficulty of the response is the rapid increase of Android malware variants. According to the Symantec report [5], the average number of variants per family is getting increased more than 50% between 2012 and 2013, while the number of families records 45% of decrease in same period. The main reason for the trend is that Android malware is continuously evolving with evasion techniques such as the code obfuscation, or the junk code insertion. The techniques authorize the attackers to easily build new malware variants, thus the existing anti-mobile-malware solutions which rely on the binary signature [1][2] or appeared characteristic like granted permissions [4], class names, or package names [3][6] are not available to detect current Android malware.

[1]This research was supported by the Public Welfare & Safety Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2012M3A2A1051118).

Motivated by this, we propose DroidGraph, a new detection mechanism based on the semantic behavior analysis for Android malware. DroidGraph deals with API calls, since in Android platform, the API calls stand on a significant position for Android operation, and each API call has a distinct semantic meaning. Therefore, analyzing Android API calls leads us to easily understand the attempt of Android malware. We transform the API calls extracted from an APK file to hierarchical behavior graphs. The graphs represent semantic meaning in point of view of every methods, classes, packages and an application level respectively, by decomposing into the APK hierarchy. This hierarchical approach allows us to avoid false alarm against repackaged malware. After extracting the behavior graphs, we select graphs as semantic signatures to detect unknown malware variants, which represent common behaviors of malware families.

To evaluate DroidGraph, we used the real-world data including a total of 260 Android malware which collected from the Contagio mobile Web site, and a total of 3,623 benign applications gathered from Google play store and alternative Android Markets. In an experiment, DroidGraph shows 87% of detection accuracy by the use of only 40 semantic signatures.

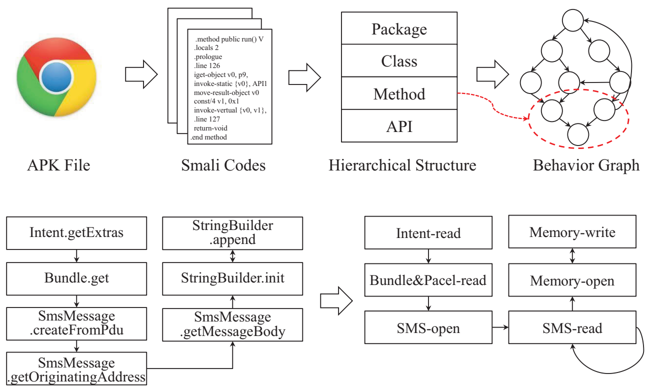
II. Android Malware

To design an effective mitigation solution, firstly we need an insightful understanding of Android malware and its phenomenon. In this section, we describe two main phenomenons of recent Android malware to be considered, which are repackaging and metamorphism. Android malware used to apply the repackaging in purposes of easy creation and distribution. Attackers build decoy applications by simply adding attack modules into a well known application, and upload to public Android Markets to lure people to install their legacy. It means that, when we analyze such a repackaged malware, we should be able to recognize which part is an attacker's property.

Another phenomenon to be considered is metamorphism. Current Android malware is known to utilize metamorphism to avoid existing anti-mobile-malware solutions [7]. To accomplish the metamorphism, attackers used to adopt several techniques such as the code obfuscation, junk code insertion, and API substitution. Therefore, considering only the appearance of Android malware may cause serious false detection. To this end, we need to analyze the semantics of Android malware rather than its appearance.

III. DroidGraph

DroidGraph is a system to detect Android malware using semantic signatures. Fig. 1 (top) depicts how DroidGraph extracts



```
 .method public run() V
.locals 2
.prologue
.line 126
iget-object v0, p9,
invoke-static {v0}, API1
move-result-object v0
const/4 v1, 0x1
invoke-vertual {v0, v1},
.line 127
return-void
.end method
```

Fig. 1: An overview of DroidGraph (top) and an example of semantic abstraction with a real malicious method (bottom).

semantics of Android malware. At the first step, we extract smali codes from an APK file using well known repackaging tool APKtool. Note that the smali code forms an interpreted language that syntactically closes to pure source codes, thus it is useful to construct the control flow of the original codes. In second step, we build API call graphs following the control flow for each smali codes in accordance with APK hierarchy. The API call graph can be represented as a directed graph $G = (V, E)$, where V is a set of API calls, and E is a set of calling relationship of the API calls, $E = \{(v_i, v_j)|v_i, v_j \in V\}$, where v_i denotes the caller and v_j denotes the callee. The API call graphs representing the behavior of each method, class, and package as well.

The main novelty of DroidGraph is that how we represent the semantics of Android malware using API calls. Since thousands of API calls are defined for Android developers, it brings huge graphs. Analyzing the naive API call graph that contains thousands of API nodes is not an effective way, and it may not allow us to clearly understand the semantic of the graph. Furthermore, considering the situation in which we need to compare the graphs to figure out Android malware, the graph isomorphism is commonly known as a NP-complete problem. Driven by this, we finally transform the API call graph to a semantic graph that each API call is substitute into semantic nodes according to their semantics. Each API call is classified into 34 objects, where the objects are process, network, account, and so on. The objects are classified again into 4 behaviors such as open, read, write, and close. Consequently, the thousands of API calls are transformed to a total of 136 identical semantic nodes. We call this transformation as the semantic abstraction. Fig. 1 (bottom) exhibits a simple example of the abstraction with a real malicious activity which attempts SMS message theft. Through the semantic abstraction step, finally we obtain the semantic graphs for each APK.

The semantic graph gives us next three benefits. First, the semantic graph is represented in every hierarchical levels, thus even if the attackers use repackaging, DroidGraph determines that which modules, or classes have the malicious codes. Second, the semantic graph considers the semantics of API calls, therefore if the attacker attempts to substitute API calls, or put some junk codes, or even obfuscate code itself, DroidGraph extracts same semantic graphs. At last, DroidGraph reduces the graph comparison overhead to a constant time. This is a great advantage in terms of practicality, compared to the naive call-graphs which have polynomial time overhead.

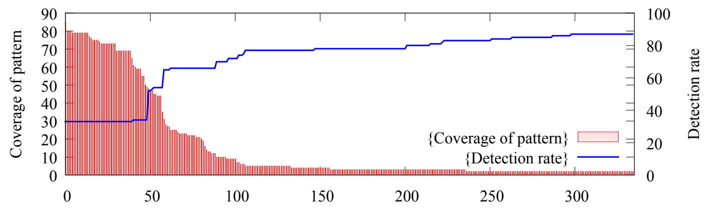


Fig. 2: The coverage of each semantic graph and the accumulative detection rate.

For the detection of Android malware, DroidGraph needs the semantic signatures which are extracted from existing known Android malware. To this end, the graph mining approach in which the signature graphs are selected from a graph pool that only contains malicious semantic graphs which commonly occurred in Android malware families but normal apps, is performed. Using the signature graphs, DroidGraph determines whether a test application contains malicious semantic behavior or not.

IV. Experimental Results

To evaluate the performance of DroidGraph, we have an experiment with real-world Android applications including 260 Android malware which collected from the Contagio mobile Web site, and 3,623 benign apps which collected from third party Android Markets and Google play store.

In experiments, we obtained 16,080 graphs from the benign applications and 1,863 graphs from the Android malware samples. By applying the graph mining, finally we obtained 335 unique graphs which only appear in Android malware samples. Fig. 2 shows the coverage of graphs and accumulated detection rate for Android malware. Maximum coverage of single graph is 85, which means the single graph can be used to detect 1/3 of the malware samples. Detection accuracy with 335 semantic behavior graphs is 87%, but in the fact, we could get a same detection results with only 40 graphs. And there was no false positive for the benign applications.

V. Conclusion

DroidGraph is an effective countermeature for the sophisticated Android malware. For further research, we plan to improve the detection accuracy with graph matchings in different level of the hierarchy, and analysis to find the most efficient semantic signatures and strategies as well.

References

[1] L. Deshotels, V. Notani, and A. Lakhotia, "Droidlegacy: Automated familial classification of android malware," in *ACM SIGPLAN on Program Protection and Reverse Engineering Workshop*, 2014, p. 3.
[2] H. Gascon, F. Yamaguchi, D. Arp, and K. Rieck, "Structural detection of android malware using embedded call graphs," in *ACM Workshop on Artificial Intelligence and Security*, 2013, pp. 45–54.
[3] S. Lee, J. Lee, and H. Lee, "Screening smartphone applications using behavioral signatures," in *28th IFIP Int'l Conf. on Computer Security, IFIP/SEC*, 2013, pp. 14–27.
[4] H. Peng, C. Gates, B. Sarma, N. Li, Y. Qi, R. Potharaju, C. Nita-Rotaru, and I. Molloy, "Using probabilistic generative models for ranking risks of android apps," in *ACM Conf. on Computer and Communications Security*, 2012, pp. 241–252.
[5] P. Wood, B. Nahorney, K. Chandrasekar, S. Wallace, and K. Haley, "Symantec internet security threat report," *Trends for 2013*, vol. 1, 2014.
[6] W. Zhou, Y. Zhou, X. Jiang, and P. Ning, "Detecting repackaged smartphone applications in third-party android marketplaces," in *ACM Conf. on Data and Application Security and Privacy*, 2012, pp. 317–326.
[7] Y. Zhou and X. Jiang, "Dissecting android malware: Characterization and evolution," in *IEEE Symp. on Security and Privacy*, 2012, pp. 95–109.