BRIDEMAID: An Hybrid Tool for Accurate Detection of Android Malware

Fabio Martinelli, Francesco Mercaldo, Andrea Saracino
Istituto di Informatica e Telematica, Consiglio Nazionale delle Ricerche
Pisa, Italy
name.surname@iit.cnr.it

ABSTRACT

This paper presents BRIDEMAID, a framework which exploits an approach static and dynamic for accurate detection of Android malware. The static analysis is based on n-grams matching, whilst the dynamic analysis is based on multi-level monitoring of device, app and user behavior. The framework has been tested against 2794 malicious apps reporting a detection accuracy of 99,7% and a negligible false positive rate, tested on a set of 10k genuine apps.

1. INTRODUCTION

Due to its increasing popularity, Android is currently the target of more than 99% of the security attacks toward mobile platforms. Standard trojanized apps are now also sided by new threats such as polymorphic and composition malware [3], which exploit dynamic code load or modification to reduce the likelihood of being discovered.

Current solutions to protect users from new threats are mainly based on signature detection, which in mobile platform are still inadequate [4]. The main issue is that using signature-based detection, a threat must be widespread for being successfully recognized, and attackers use different techniques to obfuscate code and binaries, making this signature collection task even harder.

To supply this lack of generality, in this paper we discuss BRIDEMAID (Behavior-based Rapid Identifier Detector and Eliminator of Malware for AndroID), a complete and accurate, on device analysis framework for Android apps which combines static and dynamic techniques to discriminate Android malware applications from legitimate ones. The original research bringing to these results has been presented in [4].

The contribution of this paper are (i) the description of BRIDEMAID a novel framework for on-device detection of malicious Android apps, which exploits an hybrid behavior-based approach for detection of Android malware, (ii) a short analysis of typical malware behavior and attack patterns, (iii) the experimental results on a dataset of almost 12k apps, both malicious and benign and the comparison with the VirusTotal framework.

ASIA CCS '17April 02-06, 2017, Abu Dhabi, United Arab Emirates

ACM 978-1-4503-4944-4/17/04.
http://dx.doi.org/10.1145/3052973.3055156

2. DETECTION METHODOLOGY

The analysis performed by BRIDEMAID is constituted of three consequent steps (static, meta-data and dynamic, as explained below) in which different features are retrieved and analyzed together with the two-fold effect of maximizing the malware detection rate and minimizing the amount of false positives. The main phases are illustrated in Figure 1. As shown, the behavior of each app is controlled from the very moment it enters the mobile device. In fact, as soon as the app has been downloaded, the static analysis engine will decompile the apk and analyze the source files looking for similarities in the executed actions exploiting n-grams. If the static analysis marks the app as malicious, the application will be removed, otherwise BRIDEMAID invokes the dynamic analysis to deep investigate about the trustworthiness of the application under analysis.

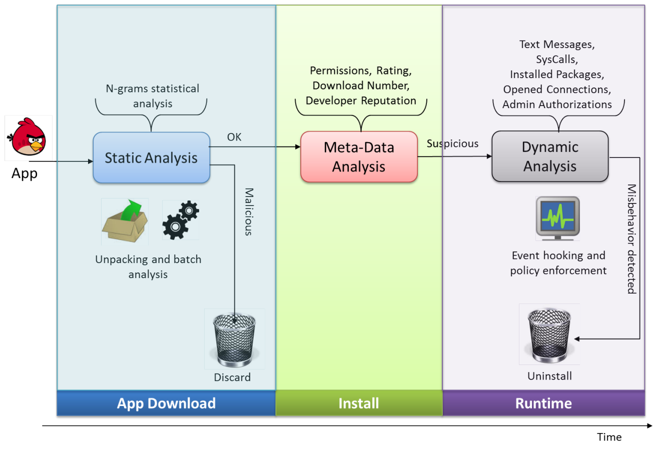
App
N-grams statistical analysis
Static Analysis
OK
Malicious
Unpacking and batch analysis
Discard
Permissions, Rating, Download Number, Developer Reputation
Meta-Data Analysis
Suspicious
Text Messages, SysCalls, Installed Packages, Opened Connections, Admin Authorizations
Dynamic Analysis
Misbehavior detected
Event hooking and policy enforcement
Uninstall
App Download
Install
Runtime
Time

Figure 1: Analysis workflow of the proposed framework.

The static analysis is based on n-grams classification where, the frequency of opcodes is calculated from the decompiled apps, hence analyzed through a binary classifier. The classifier exploited by BRIDEMAID for this static analysis is the SVM (support vector machine), trained on a dataset of malicious and benign apps. Details on the static analysis can be found in [2].

The dynamic analysis is instead based on a set of monitors placed at kernel, API and Application level, which control the activity of monitored apps and globally of the whole device. The dynamic module exploits both classifiers and security policies to control suspicious activities related to text messages, system call invocations and administrator privilege abuses. For details on the dynamic analysis, the interested reader can refer to [5].

3. EXPERIMENTAL RESULTS

The dataset used for experiments is made of a set of 9804 genuine apps downloaded from Google Play and a set of 2794 malicious apps belonging to 123 malicious families. The malicious apps have been extracted from the Drebin dataset [1], the Genome Dataset [6] and the Contagio Mobile website[1]. Moreover to test the BRIDEMAID capability of not being deceived by obfuscation, we have added two composition malware [3], whose performed misbehavior belongs to the class of SMS-Trojan. For comparison, the malicious apps have been also classified through the VirusTotal service. Table 1 details the detection results on the described set of 2974 malicious apps.

The first column from left represents the malware behavioral classes: Botnet, Installer, Ransomware, Rootkit, SMS Trojan, Spyware, Trojan, Hybrid, Composition and Polymorphic (further details about malware behavioral classes are available in [4]), the second one lists the overall samples, whilst in the third and fourth column there are the number of samples recognized as malware respectively by static and malware analysis. The unified detection result is then expressed in the fourth column, where a malicious app is considered as detected if at least one between static and dynamic approach detects it. As shown in Table 1, the dynamic approach is globally more accurate than the static one, being able to detect an higher number of malware. BRIDEMAID reports an overall detection accuracy of 99.7%, which is 2.5% more accurate than the standalone dynamic approach and 31% more accurate than the standalone static analysis. Moreover, BRIDEMAID is more accurate (1.7%) than VirusTotal, which is not able to detect those malware whose signature is still not known in the antivirus databases. In fact, we can see that VirusTotal is ineffective against the composition malware, which are instead detected by BRIDEMAID. Being based on known signatures, VirusTotal is, in fact, not effective against Zero Day attacks differently from BRIDEMAID, which exploiting computational intelligence and hybrid white-list/black-list approaches, is more resilient to obfuscation techniques and generally able to detect new threats. To evaluate the False Alarm Rate, BRIDEMAID has been used to classify the set of 9804 genuine apps previously described. The FPR is NULL for the static analysis and very low (0.2%) for the dynamic one.

[1]http://contagiominidump.blogspot.com/

4. PERFORMANCE EVALUATION

In this section we discuss the BRIDEMAID performances in terms of the evaluation of static method and from dynamic point of view i.e., the energy consumption and the overhead.

4.1 Static Method Evaluation

In order to measure computational performance related to static method, we used the *System.currentTimeMillis()* Java method that returns the current time in milliseconds. We consider the overall time to analyse a sample as the sum of three different contributions: the average time required to extract the smali classes from the application under analysis ($t_{deassembling}$) , the time required to build the features vector ($t_{features}$), and the time required to test the extracted features vector with the model learned by using the SVM algorithm ($t_{testing}$).

The most intensive task from the computational point of view is represented by $t_{features}$, while $t_{testing}$ requires 0.0235 seconds to evaluate the feature vector: the static approach takes 3,6931 seconds to test a new sample, as shown in Table 2.

4.2 Energy Consumption

To measure the BRIDEMAID energy consumption we have measured the difference in battery consumption over two periods of 24 hours, with and without BRIDEMAID running. To apply the measurements, we have used the Battery Monitor app[2].

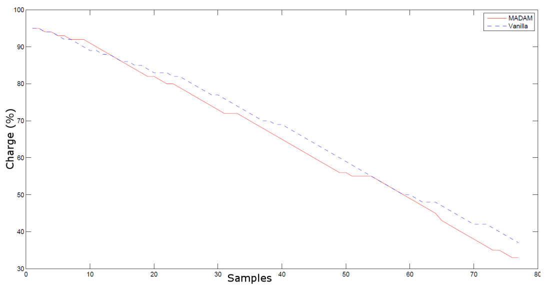

Figure 2: Energy Impact Evaluation of BRIDEMAID

The Samsung Galaxy Nexus smartphone use for the experiments equips a 1750 mAh battery. The results are reported in Figure 2, whose graph reports the two different discharges sampled in a period of 24 hours with 77 sampling intervals (x-axis). The distance between the two discharging curves is always lesser than 4%, which is the maximum value. The average consumption of the BRIDEMAID application reported by Battery Monitor is 82 mAh, accounting to 4.6% of the total battery capacity. Thus, on a period of 24 hours, the device loses approximately one hour of battery time.

4.3 Performance Overhead

The performance overhead of BRIDEMAID has been measured through a standard benchmark tool, i.e. the Quad-

[2]https://play.google.com/store/apps/details?id=ccc71.bmw

Table 1: Detection results for analyzed malicious apps.

Malware Type	Families	Samples	Static Fam	Static Sam	Dynamic Fam	Dynamic Sam	BRIDEMAID	VirusTotal
Botnet	2	7	1	2	0	0	2	7
Installer	6	406	3	236	6	406	406	400
Ransomware	3	30	2	11	3	30	30	30
Rootkit	13	543	10	436	13	543	543	541
SMS Trojan	40	1295	33	771	40	1295	1295	1276
Spyware	38	231	38	231	21	161	231	220
Trojan	5	23	5	20	2	19	20	22
Hybrid	14	243	10	189	14	243	243	243
Composition (SMS Trojan)	1	2	0	0	1	2	2	0
Polymorphic (SMS Trojan)	1	14	1	14	1	14	14	14
Total	123	2794	103	1910	101	2713	2784	2753
Accuracy			68,4 %		97,2 %		99.7%	98%

Table 2: Static method evaluation

$t_{deassembling}$	$t_{features}$	$t_{testing}$	t_{total}
1.5407 s	2.1289 s	0.0235 s	3,6931 s

rant Standard Edition app, which is distributed through Google Play[3]. Table 3 reports the benchmark for the system when BRIDEMAID was running (third column from left, "BRIDEMAID") and when it was not (second column left, "Vanilla"). The last column reports the overhead computed as a percentage overhead between the two performances. Benchmarks are provided as indexes, where a highest value means a better performance. Benchmarks reported have been computed as the average of five experiments, both in "Vanilla" and "BRIDEMAID" configuration. The overhead of BRIDEMAID is caused by both the kernel module, which hijacks system calls, and a *Global Monitor* service that runs in background when the system is active. As shown, the performance impact of BRIDEMAID is acceptable; in fact, the overall performance impact (Total) is 1.4%. On three devices, we measured that on average the *App-Evaluator* increases by 3 to 7 seconds the app installation phase.

Table 3: Benchmark Tests

Test	Vanilla	BRIDEMAID	Overhead
Total	2911	2868	1,4%
CPU	5509	5459	0,9%
Memory	2660	2409	9,4%
I/O	3860	3705	4%
2D	327	327	0%
3D	2250	2250	0 %

5. CONCLUSION

In this paper we present BRIDEMAID, a framework combining static and dynamic analysis to detect Android malware. We obtain an accuracy in Android malware detection equal to 99,7%, overcoming the current signature-based antimalware technologies.

[3]http://play.google.com/store/apps/details?id=com.aurorasoftworks.quadrant.ui.standard

Acknowledgements

This work has been partially supported by H2020 EU-funded projects NeCS and C3ISP and EIT-Digital Project HII.

References

[1] D. Arp, M. Spreitzenbarth, M. Huebner, H. Gascon, and K. Rieck. Drebin: Efficient and explainable detection of android malware in your pocket. In *Proceedings of 21th Annual Network and Distributed System Security Symposium (NDSS)*, 2014.

[2] G. Canfora, A. De Lorenzo, E. Medvet, F. Mercaldo, and C. A. Visaggio. Effectiveness of opcode ngrams for detection of multi family android malware. In *Availability, Reliability and Security (ARES), 2015 10th International Conference on*, pages 333–340. IEEE, 2015.

[3] G. Canfora, F. Mercaldo, G. Moriano, and C. A. Visaggio. Composition-malware: building android malware at run time. In *Availability, Reliability and Security (ARES), 2015 10th International Conference on*, pages 318–326. IEEE, 2015.

[4] F. Martinelli, F. Mercaldo, A. Saracino, and A. Visaggio. I find your behavior disturbing: Static and dynamic app behavioral analysis for detection of android malware. In *14th annual conference on Privacy Security and Trust*. IEEE, 2016. To Appear.

[5] A. Saracino, D. Sgandurra, G. Dini, and F. Martinelli. Madam: Effective and efficient behavior-based android malware detection and prevention. *IEEE Transactions on Dependable and Secure Computing*, PP(99):1–1, 2016.

[6] Y. Zhou and X. Jiang. Dissecting android malware: Characterization and evolution. In *Proceedings of 33rd IEEE Symposium on Security and Privacy (Oakland 2012)*, 2012.