DETECTION EFFICIENCY OF STATIC ANALYZERS AGAINST OBFUSCATED ANDROID MALWARE

Co-authored by Victor Ajiri

Sergey Butakov

Pavol Zavarsky

Project report

Submitted to the Faculty of Graduate Studies,

Concordia University of Edmonton

in Partial Fulfillment of the

Requirements for the

Final Research Project for the Degree

MASTER OF INFORMATION SYSTEMS SECURITY MANAGEMENT

Concordia University of Edmonton

FACULTY OF GRADUATE STUDIES

Edmonton, Alberta

April 2020

DETECTION EFFICIENCY OF STATIC ANALYZERS AGAINST OBFUSCATED ANDROID MALWARE

Victor Ajiri

Approved:

Sergey Butakov [Original Approval on File]

Sergey Butakov Date: April 15, 2020

Primary Supervisor

Edgar Schmidt [Original Approval on File]

Edgar Schmidt, DSocSci Date: April 20, 2020

Dean, Faculty of Graduate Studies

Detection Efficiency of Static Analyzers against Obfuscated Android Malware

Victor Ajiri
Information Systems Security Management
Concordia University of Edmonton
Edmonton, Canada
vajiri@student.concordia.ab.ca

Sergey Butakov
Information Systems Security Management
Concordia University of Edmonton
Edmonton, Canada
sergey.butakov@concordia.ab.ca

Pavol Zavarsky
Information Systems Security Management
Concordia University of Edmonton
Edmonton, Canada
pavol.zavarsky@concordia.ab.ca

***Abstract*—Mobile antivirus technologies incorporate static analysis which involves the analysis of programs without its execution. This technique relies on pattern matching against a signature repository to identify malware, which can be easily tricked by transformation techniques such as obfuscation. Obfuscation renders character strings disguised and incomprehensive to prevent tampering and reengineering. This paper attempts to study the detection efficiency of static analyzers against obfuscated Android malware. This study is the first step in a larger project attempting to improve the efficiency of malware detectors.**

Keywords—obfuscated malware, static analyzer, Android, malware detection efficiency, mobile antivirus, signature repository

I. INTRODUCTION

Malwares are being produced at an unprecedented scale with hundreds of new entities targeting users across all of technology, as malware developers explore new ways or exploit old ones to evade detection and defeat analysis [1]. The process of obfuscation is originally used to protect benign applications from code alterations, manipulations and reverse engineering, but this mechanic is also a tool malware developers could manipulate to mask the malicious applications they create. In fact, obfuscation has become one of the "popular" techniques used by malware developers. The technique employs processes that make malware code difficult to understand, as the code is being altered in a number of ways with the attempt to make the code look different from the original script while still producing the same malicious actions. Its purpose is to defeat detection by concealing its payload [2]. Obfuscation bypasses static code analyzers to avoid code study as character strings are concealed and made incomprehensive via algorithms that decode the code at execution [3].

In this paper , a publicly available Android malware dataset was subjected to three popular obfuscation techniques [4] [5]. ten random Android malware families were selected from this dataset, with each comprising of five variant samples. The samples were individually subjected to three obfuscation techniques control flow, renaming and string encryption and a final process of combining these individual techniques on the same application. A widespread analysis of these obfuscated and non-obfuscated samples was further carried out against available mobile anti-malware engines provided by virus total platform.

The research summary is described below:

- Obtained and subjected Android malware samples to available obfuscation techniques from [4] [5].
- Subjected these samples to a number of mobile analyzers to assess the detection efficiency of these platforms against obfuscated and non-obfuscated Android malware

This paper reports results of the study of detection efficiency of anti-malware engines against obfuscated Android application malware samples.

II. RELATED WORK

The nature of mobile anti-malware engines being reliant on signature repositories to flag applications as malicious gives room to malicious applications evading detection by the application of evasive techniques such as obfuscation.

Pomilia Matteo investigated and implemented a framework to subject applications to obfuscation techniques to avoid detection and tested the samples against nine popular anti-malware tools at the time of research: Avast, AVG, F-secure, Kaspersky, McAfee, Microsoft, Sophos, Symantec and TrendMicro [6]. The malwares included samples preceding the year 2013 from the Drebin and Contagio malware datasets. Results obtained showed obfuscation techniques applied to malware preceding 2013 had averagely a detection rate above 55% opposed to obfuscated malwares of samples succeeding 2012 from the Andrototal and Contagio datasets which reported a detection drop rate below 55%. Furthermore, techniques to reverse engineer obfuscated applications for manual analysis was discussed by the author.

Rastogi et al. proved that the top ten anti-malware products at the time the research was conducted were all vulnerable to common obfuscation techniques [7]. These malware samples were subjected to ten transformation techniques in which most applications were only subjected to not more than two combinations of transformation techniques. Reports from the study showed 43% of signatures identification used by static detection engines were not based on code level objects, stating that simply changing component names in the *AndroidManifest* was more than appropriate to defeat detection. The paper further states that 90% of signatures did not require static analysis of

bytecode because most of the information used for analysis were found in the *class.dex* file of the application which contained the code executed by the Android runtime.

[8] Performed a wide-ranging experimental analysis to estimate the efficacy of top anti-malware products against various obfuscation tools by using twenty nine (29) obfuscation techniques from seven (7) obfuscation tools against 3000 benign and malicious applications, coming to the conclusion that obfuscation impacts Android anti-malware products and the detection of these applications by Android anti-malware products depends on the obfuscation technique adopted and tool used.

III. EXPERIMENT

The Android malware dataset samples needed for the research was obtained from Argus Lab [4]. Subsequently a random selection of ten Android malware families was made, with five variants under each family. The list of selected families comprised of AndroRat, Cova, DroidKungfu, FakeAngry, FakeInst, Finspy, Golddream, Koler, Lootoor, SMSZombie. In total fifty Android malware samples were tested before and after obfuscation. These "original" / un-obfuscated malware samples were passed through the virus total online platform to scan samples through a number of static anti-malware engines. This step created a baseline of pre-obfuscation detection rate. The obtained malware samples are then passed through the DashO obfuscator [5], transforming these samples via three obfuscation techniques ; control flow , renaming and string encryption and then a final transformation is carried out using a combination of all three techniques. Figure 1 shows the logical flow of our experimentation on the Android malware samples and the obfuscation process.

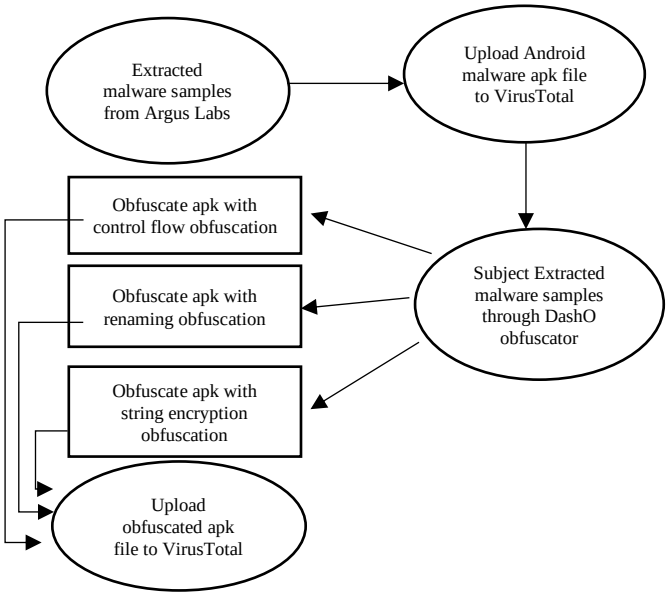

Fig 1. Obfuscation transformation process

The experimental testbed was set with some variations for parameters for conducting the obfuscation process. Parameters for the control flow obfuscation which changes the execution sequence of instructions found in methods to render the flow of control difficult to understand and trace the instructions included the following settings:

(i) Block jumbling - this process randomly organizes how the code blocks are represented.

(ii) Dalvik compatibility - this allows the support of the Dalvik virtual machine to emulate Android platform

(iii) Try/Catch- this process adds try/catch handlers to code which confuses decompilers.

(iv) Block splitting- this process splits blocks into smaller blocks and adds switch-based control flow

(v) Target block size- This process indicates the minimum number of bytecode instructions to remain in each block.

Parameters for the renaming obfuscation which changes the class names, methods, fields and packages to render bytecode compact included

i) Overload induction- which comprises of processes to induce maximum reuse of methods.

Parameters for string encryption obfuscation which is a technique that replaces strings with encrypted values decrypted at execution included

(i) String encryption level - this process controls the strength and performance of the encryption process.

(ii) Decrypter- this parameter controls the decryptor methods that will be created and added to each output.

IV. EXPERIMENTAL RESULTS

During experimentations with obfuscated Android malware samples it was observed that a number of anti-malware engines did not show so much resilience in detecting obfuscated samples as opposed to detecting samples prior to obfuscation. Table I shows a sample section of the detection results of a family variant of Android malware prior and subjected to obfuscation transformations against the virus total platform which houses a number of anti-malware engines, rendering updated insights on the state-of-the-art detection of obfuscated malware in the first quarter of 2020.

TABLE I. DETECTION RATE OF A SAMPLE OBFUSCATED ANDROID MALWARE

Variant 1	**Androrat**			
	Number of engines	***Samples Detected***	***Samples Undetected***	***Detection Ratio %***
Without Obfuscation	60	29	31	48.33%
Control flow obfuscation	61	21	40	34.42%
Renaming obfuscation	60	19	41	31.66%
String encryption obfuscation	61	17	44	27.86%
Combination	59	9	50	15.25%

Results obtained from the total experimentation done showed that detection efficiency of anti-malware engines reduces

drastically when malicious software are subjected to obfuscation. It is also clear that a software subjected to a combination of obfuscation techniques reduces the likelihood of detection as seen in Table I.

Detailed results of detection for each variation with various obfuscation techniques can be found on the following website: *https://sites.google.com/view/malware-analysis-spreedsheet/home*

Table II shows the average detection rating of each 5 samples under individual malware families. This is achieved by taking the detection ratio values of each five variants of a family and deriving the average of those five results. From Table II it can be observed that there's a significant drop in detection rate between the obfuscated and non-obfuscated malware samples, as seen from the results the average rating is slightly dropping with each technique integrated and even lower when a combination of techniques is applied, proving that obfuscation if done properly as an evasion technique is a significant threat to the detection efficiency of static analyzers. The final values are an average of all 10 family derived values.

TABLE II. AVERAGE DETECTION RATING OF EACH OBFUSCATED FIVE MALWARE SAMPLES OF EACH FAMILY

***Family** **	***Without obfuscation***	***Control flow***	***Renaming***	***String encryption***	***Combination***
1	52.13 %	34.54 %	27.06 %	33.78 %	16.59 %
2	50.32 %	18.55 %	14.05 %	11.51 %	12.09 %
3	63.06 %	53.62 %	52.93 %	48.06 %	46.08 %
4	48.04 %	23.44 %	28.84 %	6.65 %	7.03 %
5	54.08 %	37.49 %	27.03 %	15.22 %	9.96 %
6	57.18 %	28.42 %	27.89 %	21.41 %	20.23 %
7	59.66 %	47.34 %	46.03 %	22.68 %	23.52 %
8	45.92 %	17.05 %	11.21 %	10.19 %	5.94 %
9	63.79 %	51.14 %	50.35 %	47.84 %	46.89 %
10	51.69 %	41.45 %	42.48 %	40.26 %	43.6 %
AVG	**54.58 %**	**35.30 %**	**32.78 %**	**25.76 %**	**23.19 %**

* Families- (1) AndroRat (2) Cova (3) DroidKungfu (4) FakeAngry (5) FakeInst (6) Finspy (7) Golddream (8) Koler (9) Lootoor (10) SMSZombie

Table III shows some anti-malware tools ranked by levels of performance in showing resilience against the obfuscated Android malware samples used during experimentation. This ranking was based on the number of detected obfuscated samples across all obfuscation methods applied. The optimality of the anti-malware engines provided by the Virus Total platform is unknown, therefore the results obtained are subject to change and does in no way discredit the antimalware engines listed. In total the consideration was 200 obfuscated samples excluding 50 unobfuscated samples. The table below shows a ranking based on percentages obtained from the total number of detected obfuscated instances of a sample variant by each analyzer. An example would be the Dr web anti malware engine which detected 196 obfuscated instances of all variants out of the 200 obfuscated samples.

TABLE III. DETECTION RATING OF ANTI MALWARE ENGINES BASED ON DETECTED OBFUSCATED APPLICATIONS

SN	**Anti-malware**	**SN**	**Anti-malware**
1.	Dr web c. **98%**	32.	Fireeye **16%**
2.	ESET-NOD32 **86.5%**	32.	Antiy avl **16%**
3.	K7GW **82.5%**	33.	Sangfor Engine Zero **11.5%**
4.	Zone Alarm **75.5 %**	34.	Max-secure **10%**
5.	Sophos-AV **74.5 %**	35.	eScan **10%**
6.	Kaspersky **74%**	36.	Symantec **8.5%**
7.	Ikarus **66%**	37.	Yandex **6%**
8.	Qihoo **65.5%**	38.	Zoner **6%**
9.	F-secure **65%**	39.	Alibaba **4%**
9.	Ahnlab-v3 **65%**	40.	Aegislab **0.5%**
10.	AVG **64.5%**	40.	Tencent Habo **0.5%**
11.	Avast **64%**	41.	Alyac **0%**
11.	Avast mobile **64%**	42.	Vipre **0%**
12.	CAT Quickheal **63%**	43	TotalDefense **0%**
13.	Symantec mobile insight **56.5%**	44.	SuperAntispyware **0%**
14.	Fortinet **54.5%**	45.	Panda **0%**
15.	Tencent **44.5%**	46.	Kingsoft **0%**
16.	Nano-Antivirus **43.5%**	47.	CMC **0%**
17.	Avira **39%**	48.	BitDefenderTheta **0%**
18.	Cyren **36%**	49.	Baidu **0%**
19.	Microsoft **35.5 %**	50	Ad aware **0%**
20.	Trustlook **34.5%**	51.	Virobot **0%**
21.	McAfee **34 %**	52.	VBA32 **0%**
22.	Clam AV **32 %**	53.	Tachyon **0%**
23.	McAfee-GW-Edition **30%**	54.	Malwarebytes **0%**
24.	Jiangmin **26%**	55.	K7antivirus **0%**
25.	Comodo **25.5%**	56.	BKAV **0%**
26.	Rising **21.5%**	57.	Acronis **0%**
27.	Gdata **20.5%**	58.	Crowdstrike falcon **0%**
28.	F-prot **19%**	59.	Cylance **0%**
29.	Zilya **17%**	60.	Endgame **0%**
30.	Max **16.5%**	61.	Sentinel one **0%**
31.	Trendmicro housecall **16.5%**	62.	Cyber reason **0%**
31.	Emsisoft **16.5%**	63.	eGambit **0%**
32.	Trendmicro 1**6%**	64.	Palo Alto Network **0%**
32.	Arcabit **16%**	65.	Sophos ML **0%**
32.	Bitdefender **16%**	66.	Webroot **0%**

V. CONCLUSION

This study is a step in the development of improved malware tools. It shows that obfuscation has a serious impact to the detection efficiency of existing anti-malware analyzers. Some obfuscation techniques show strong detection resilience, but a mixture of obfuscation techniques show even stronger detection resilience. Publicly available dataset was examined and utilized for this study, while being subjected to obfuscation techniques offered by the PreEmptive DashO obfuscator. The results obtained shows there needs to be an improvement in mobile security as access to these obfuscation mechanics are available to the public. There is also a large existence of Anti-malware engines in mobile stores that do not perform any form of analysis, these engines claim to do some form of checks but in reality do next to nothing as they are developed by individuals for monetary purposes and offer no protection which begs to question which mobile anti malware software actually offers a satisfactory amount of protection and is resilient to masqueraded malware. Anti-malware static analysis engines should be trained with obfuscated samples so signatures and prints these techniques leave behind should serve to aid blacklisting of similar applications that portray these same behaviors. Additional research may be required for tools that look at system calls for detection purpose.

VI. ACKNOWLEDGEMENTS

Authors would like to acknowledge the help from Argus Labs for providing access to Android malware database (*http://amd.arguslab.org/*) and PreEmptive solutions (*https://www.preemptive.com/*) for the obfuscation tool used during the research.

VII. REFERENCES

[1] S. M, L. A, G. Jonathan and L. Wenke , "Impeding Malware Analysis Using Conditional Code Obfuscation," in *16th Annual Network & Distributed System Security Symposium Proceedings*, 2008.

[2] K. Iliev, "Top 6 Advanced Obfuscation Techniques Hiding Malware on Your Device," 31 August 2017. [Online]. Available: https://sensorstechforum.com/advanced-obfuscation-techniques-malware/. [Accessed 8 June 2019].

[3] T. Gendron, "Malware Analysis, Part 1: Understanding Code Obfuscation Techniques," 16 May 2019. [Online]. Available: https://www.vadesecure.com/en/malware-analysis-understanding-code-obfuscation-techniques/. [Accessed 12 June 2019].

[4] A. Labs, "Android Malware Dataset," Argus Lab, January 2020. [Online]. Available: amd.arguslab.org. [Accessed September 2019].

[5] P. solutions, "PreEmptive protection," PreEmptive , [Online]. Available: Preemptive.com. [Accessed September 2019].

[6] M. Pomilia, "A STUDY ON POPULAR OBFUSCATION TECHNIQUES FOR ANDROID MALWARE," March 2016. [Online]. Available: https://pdfs.semanticscholar.org/2ff1/9ac2087bcc8fbbe1d11b9b49c8e8d2486964.pdf. [Accessed 17 June 2019].

[7] V. Rastogi, Y. Chen and X. Jiang, "Droidchameleon: Evaluating Android Anti-malware against Transformation Attacks," *Information Forensics and Security IEEE Transactions*, vol. 9, no. 1, pp. 99-108, 2014.

[8] H. Mahmoud, J. Garcia and S. Malek, "A Large-Scale Emprical Study on the Effects of Code Obfuscations on Android Apps and Anti-Malware Products," in *International Conference on Software Engineering*, New York, 2018.

APPENDIX

Detection rating of all samples under each Android malware family

	Engines	Samples Detected	Samples Undetected	Detection Ratio %		Engines	Samples Detected	Samples Undetected	Detection Ratio %
	AndroRat					**COVA**			
Without Obfuscation	60	29	31	**48.33 %**		62	32	30	**51.61 %**
Control flow obfuscation	61	21	40	**34.42 %**		62	13	49	**20.96 %**
Renaming obfuscation	60	19	41	**31.66 %**		63	10	53	**15.87 %**
String Encryption	61	17	44	**27.86 %**		59	5	54	**8.47 %**
Combined Obfuscation Techniques	59	9	50	**15.25 %**		61	9	52	**14.75 %**
Without Obfuscation	62	31	31	**50 %**		62	31	31	**50 %**
Control flow obfuscation	61	21	40	**34.42 %**		62	12	50	**19.35 %**
Renaming obfuscation	62	19	43	**30.64 %**		63	9	54	**14.28 %**
String Encryption	59	23	36	**38.98 %**		64	17	47	**26.56 %**
Combined Obfuscation Techniques	62	10	52	**16.12 %**		60	7	53	**11.66 %**
Without Obfuscation	61	33	28	**54.09 %**		61	31	30	**50.81 %**
Control flow obfuscation	59	20	39	**33.89 %**		61	10	51	**16.39 %**
Renaming obfuscation	60	15	45	**25 %**		61	8	53	**13.11 %**
String Encryption	61	18	43	**29.5 %**		61	4	57	**6.55 %**
Combined Obfuscation Techniques	63	11	52	**17.46 %**		61	7	54	**11.47 %**
Without Obfuscation	58	31	27	**53.44 %**		60	30	30	**50 %**
Control flow obfuscation	60	21	39	**35 %**		60	11	49	**18.33 %**
Renaming obfuscation	59	15	44	**25.42 %**		61	9	52	**14.75 %**
String Encryption	62	19	43	**30.64 %**		62	5	57	**8.06 %**
Combined Obfuscation Techniques	61	10	51	**16.39 %**		62	7	55	**11.29 %**
Without Obfuscation	62	34	28	**54.83 %**		61	30	31	**49.18 %**
Control flow obfuscation	60	21	39	**35 %**		62	11	51	**17.74 %**
Renaming obfuscation	59	15	44	**25.42 %**		62	9	53	**14.51 %**
String Encryption	62	26	36	**41.93 %**		63	5	58	**7.93 %**
Combined Obfuscation Techniques	62	11	51	**17.74 %**		62	7	55	**11.29 %**
	DROIDKUNGFU					**FAKEANGRY**			
Without Obfuscation	62	37	25	**59.67 %**		62	31	31	**50 %**
Control flow obfuscation	62	25	37	**40.32 %**		60	13	47	**21.66 %**
Renaming obfuscation	60	25	35	**41.66 %**		60	15	45	**25 %**
String Encryption	62	18	44	**29.03 %**		60	7	53	**11.66 %**
Combined Obfuscation Techniques	62	16	46	**25.8 %**		60	5	55	**8.33 %**
Without Obfuscation	61	33	28	**54.09 %**		63	31	32	**49.2 %**
Control flow obfuscation	62	23	39	**37.09 %**		58	17	41	**29.31 %**
Renaming obfuscation	61	22	39	**36.06 %**		60	18	42	**30 %**
String Encryption	60	17	43	**28.33 %**		61	2	59	**3.27 %**
Combined Obfuscation Techniques	62	16	46	**25.8 %**		59	4	55	**6.77 %**
Without Obfuscation	62	41	21	**66.12 %**		60	27	33	**45 %**
Control flow obfuscation	63	41	22	**65.07 %**		61	6	55	**9.83 %**
Renaming obfuscation	60	39	21	**65 %**		60	20	40	**33.33 %**

String Encryption	60	37	23	**61.66 %**		60	5	55	**8.33 %**
Combined Obfuscation Techniques	60	39	21	**65 %**		61	4	57	**6.55 %**
Without Obfuscation	62	44	18	**70.96 %**		63	29	34	**46 %**
Control flow obfuscation	60	38	22	**63.33 %**		60	17	43	**28.33 %**
Renaming obfuscation	61	37	24	**60.65 %**		60	17	43	**28.33 %**
String Encryption	61	37	24	**60.65 %**		60	3	57	**5 %**
Combined Obfuscation Techniques	61	35	26	**57.37 %**		59	4	55	**6.77 %**
Without Obfuscation	64	43	21	**67.18 %**		62	31	31	**50 %**
Control flow obfuscation	61	38	23	**62.29 %**		57	16	41	**28.07 %**
Renaming obfuscation	62	38	24	**61.29 %**		58	16	42	**27.58 %**
String Encryption	61	37	24	**60.65 %**		60	3	57	**5 %**
Combined Obfuscation Techniques	62	35	27	**56.45 %**		59	4	55	**6.77 %**
	FAKEINST					**FINSPY**			
Without Obfuscation	63	34	29	**53.96 %**		61	35	26	**57.37 %**
Control flow obfuscation	58	22	36	**37.93 %**		59	16	43	**27.11 %**
Renaming obfuscation	58	18	40	**31.03 %**		59	16	43	**27.11 %**
String Encryption	60	11	49	**18.33 %**		60	11	49	**18.33 %**
Combined Obfuscation Techniques	61	11	50	**18.03 %**		60	9	51	**15 %**
Without Obfuscation	60	29	31	**48.33 %**		61	36	25	**59.01 %**
Control flow obfuscation	59	23	36	**38.98 %**		60	14	46	**23.33 %**
Renaming obfuscation	60	16	44	**26.66 %**		60	15	45	**25 %**
String Encryption	61	10	51	**16.39 %**		59	9	50	**15.25 %**
Combined Obfuscation Techniques	59	5	53	**8.47 %**		60	9	51	**15 %**
Without Obfuscation	60	34	26	**56.66 %**		61	30	31	**49.18 %**
Control flow obfuscation	59	21	38	**35.59 %**		59	20	39	**33.89 %**
Renaming obfuscation	61	15	46	**24.59 %**		58	18	40	**31.03 %**
String Encryption	60	6	54	**10 %**		60	19	41	**31.66 %**
Combined Obfuscation Techniques	60	4	56	**6..66 %**		61	18	43	**29.5 %**
Without Obfuscation	59	30	29	**50.84 %**		61	38	23	**62.29 %**
Control flow obfuscation	60	22	38	**36.66 %**		58	20	38	**34.48 %**
Renaming obfuscation	58	16	42	**27.58 %**		59	19	40	**32.2 %**
String Encryption	61	10	51	**16.39 %**		60	19	41	**31.66 %**
Combined Obfuscation Techniques	60	5	55	**8.33 %**		60	17	43	**28.33 %**
Without Obfuscation	61	37	24	**60.65 %**		62	36	26	**58.06 %**
Control flow obfuscation	60	23	37	**38.33 %**		60	14	46	**23.33 %**
Renaming obfuscation	60	16	44	**26.66 %**		58	14	44	**24.13 %**
String Encryption	60	9	51	**15 %**		59	6	53	**10.16 %**
Combined Obfuscation Techniques	60	5	55	**8.33 %**		60	8	52	**13.33 %**
	GOLDDREAM					**KOLER**			
Without Obfuscation	60	35	25	**58.33 %**		63	27	36	**42.85 %**
Control flow obfuscation	60	27	33	**45 %**		60	10	50	**16.66 %**
Renaming obfuscation	60	27	33	**45 %**		60	6	54	**10 %**
String Encryption	61	7	54	**11.47 %**		60	3	57	**5 %**
Combined Obfuscation Techniques	60	8	52	**13.33 %**		61	11	50	**16.66 %**

Without Obfuscation	61	36	25	**59.01 %**		61	30	31	**49.18 %**

Control flow obfuscation	60	29	31	**48.33 %**		62	12	50	**19.35 %**
Renaming obfuscation	60	30	30	**50 %**		60	7	53	**11.66 %**
String Encryption	59	18	41	**30.5 %**		61	16	45	**26.22 %**
Combined Obfuscation Techniques	60	19	41	**31.66 %**		61	2	59	**3.27 %**
Without Obfuscation	61	38	23	**62.29 %**		62	30	32	**48.38 %**
Control flow obfuscation	60	31	29	**51.66 %**		59	10	49	**16.94 %**
Renaming obfuscation	60	28	32	**46.66 %**		61	7	54	**11.47 %**
String Encryption	61	18	43	**29.5 %**		61	4	57	**6.55 %**
Combined Obfuscation Techniques	60	19	41	**31.66 %**		61	2	59	**3.27 %**
Without Obfuscation	62	37	25	**59.67 %**		61	29	32	**47.54 %**
Control flow obfuscation	60	26	34	**43.33 %**		61	11	50	**18.03 %**
Renaming obfuscation	61	27	34	**44.26 %**		61	7	54	**11.47 %**
String Encryption	61	7	54	**11.47 %**		60	4	56	**6.66 %**
Combined Obfuscation Techniques	61	6	55	**9.83 %**		61	2	59	**3.27 %**
Without Obfuscation	61	36	25	**59.01 %**		60	25	35	**41.66 %**
Control flow obfuscation	62	30	32	**48.38 %**		61	11	50	**16.66 %**
Renaming obfuscation	61	27	34	**44.26 %**		61	7	54	**11.47 %**
String Encryption	59	18	41	**30.5 %**		61	4	57	**6.55 %**
Combined Obfuscation Techniques	61	19	42	**31.14 %**		61	2	59	**3.27 %**
	LOOTOOR					**SMSZOMBIE**			
Without Obfuscation	61	42	19	**68.85 %**		62	32	30	**51.61 %**
Control flow obfuscation	61	36	25	**59.01 %**		62	27	35	**43.54 %**
Renaming obfuscation	61	36	25	**59.01 %**		61	26	35	**42.62 %**
String Encryption	62	36	26	**58.06 %**		60	25	35	**41.66 %**
Combined Obfuscation Techniques	61	35	26	**57.37 %**		61	24	37	**39.34 %**
Without Obfuscation	61	41	20	**67.21 %**		64	35	29	**54.68 %**
Control flow obfuscation	61	36	25	**59.01 %**		59	26	33	**44.06 %**
Renaming obfuscation	59	33	26	**55.93 %**		60	26	34	**43.33 %**
String Encryption	60	34	26	**56.66 %**		61	25	36	**40.98 %**
Combined Obfuscation Techniques	61	35	26	**57.37 %**		61	25	36	**40.98 %**
Without Obfuscation	61	40	21	**65.57 %**		63	34	29	**53.96 %**
Control flow obfuscation	62	29	33	**46.77 %**		61	25	36	**40.98 %**
Renaming obfuscation	61	28	33	**45.9 %**		60	25	35	**41.66 %**
String Encryption	61	26	35	**42.62 %**		60	25	35	**41.66 %**
Combined Obfuscation Techniques	61	26	35	**42.62 %**		61	26	35	**42.62 %**
Without Obfuscation	61	36	25	**59.01 %**		59	27	32	**45.76 %**
Control flow obfuscation	61	27	34	**44.26 %**		61	23	38	**37.7 %**
Renaming obfuscation	61	27	34	**44.26 %**		59	22	37	**37.28 %**
String Encryption	60	23	37	**38.33 %**		61	19	42	**31.14 %**
Combined Obfuscation Techniques	62	23	39	**37.09 %**		61	19	42	**31.14 %**
Without Obfuscation	60	35	25	**58.33 %**		61	32	29	**52.45 %**
Control flow obfuscation	60	28	32	**46.66 %**		61	25	36	**40.98 %**
Renaming obfuscation	60	28	32	**46.66 %**		61	29	32	**47.54 %**

String Encryption	59	24	35	**40.67 %**		61	28	33	**45.9 %**
Combined Obfuscation Techniques	60	24	36	**40 %**		61	39	22	**63.93 %**

GRAPH SHOWING DETECTION RANKING