

Detection of Most Recent Torjan on Operating System

Yogendra Tiwari, Rajeev Kaushi

Department of Computer Science and Engineering, R D Engineering College, India

DOI: <https://doi.org/10.51583/IJLTEMAS.2025.1407000038>

Abstract - The rapid growth of mobile technologies has expanded the functionality of smartphones, with Android emerging as the dominant platform due to its open ecosystem and massive user base exceeding 2 billion. However, this popularity has also made Android a primary target for cybercriminals, particularly in the distribution of mobile malware. Since 2009, mobile malware has escalated significantly, with a notable rise in new variants in 2017, fueled by profit-driven attackers and underground markets. Advanced evasion techniques such as obfuscation, logic bombs, and runtime payload dropping have made malware detection increasingly difficult. This thesis focuses on analyzing modern Android trojans—specifically, banking trojans that steal user credentials via fake login interfaces. Using both static reverse engineering and dynamic behavioral analysis in emulated environments, the study examines 2,380 samples across 20 trojan (sub)families. The research identifies common traits and evolutionary patterns among these threats, providing valuable insights into their increasing sophistication and the growing challenge they pose to existing mobile security solutions.

Keywords- Trojans, Android, antimalware

I. Introduction

The rapid advancement of microelectronic integration has made mobile devices essential in daily life, largely due to the wide variety of applications they support. Android, as the dominant mobile operating system, reached over 2 billion active devices by 2019 and has maintained a peak market share above 90%. Its open-source nature has enabled extensive app development but has also introduced significant security challenges, particularly due to OS fragmentation. Most users remain on outdated versions, limiting the effectiveness of modern antivirus tools. Additionally, customizations by device manufacturers create inconsistencies that malware authors exploit to bypass detection. These factors collectively contribute to Android's growing vulnerability to sophisticated malware threats.

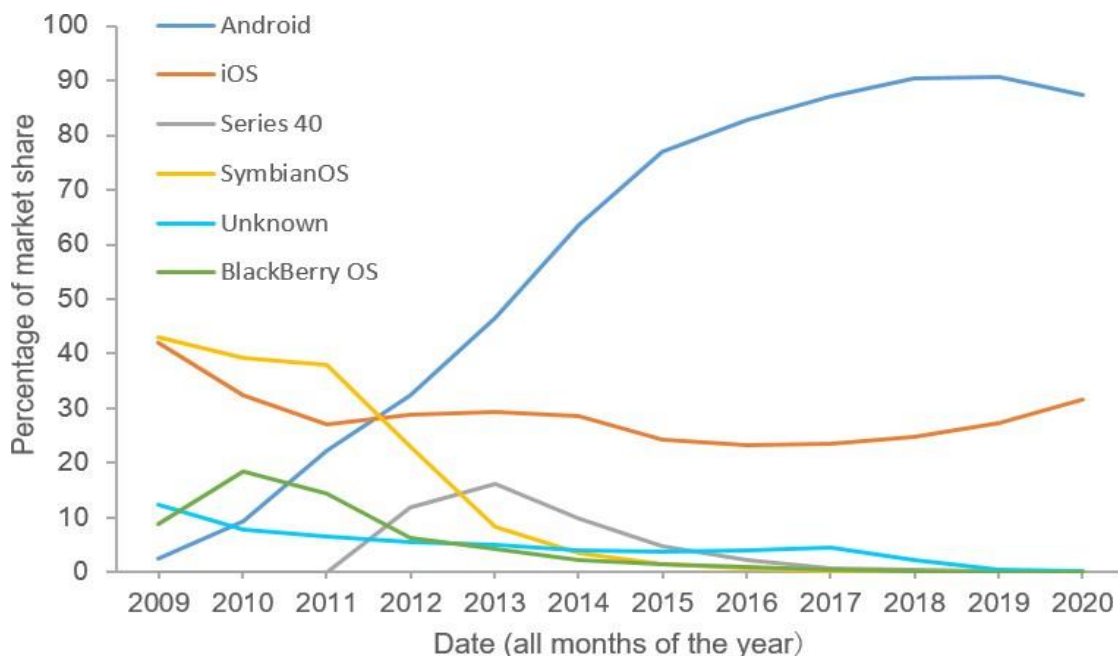


Figure 1: Market shares of different operating systems for mobile phones [2]

System Description

Android malware and Android banking trojan

Malware, a broad term for malicious software, refers to programs designed to harm systems or disrupt user experience. However, distinguishing between benign and malicious behavior is often ambiguous, as some applications covertly collect sensitive data—such as device identifiers, keystrokes, or audio—and transmit it to remote servers. In the Android ecosystem, malware is typically categorized into seven types, each differing in behavior and impact. These categories are defined based on specific properties, which help in identifying and analyzing various malware threats more effectively:

Table 1: Differences between different types of malware

		Virus	Worm	Trojan	Backdoor	Spyware	Rootkit	Botnet
Existing Form	Parasitic	C						
	Masquerade		C	C	C	C	C	C
	Independent identity		C					
Propagation mode	Repackaging	C		C	C	C	C	C
	Update attack			C	C	C	C	C
	Sideload	C	C	C	C	C	C	C
	Self-replicate	C	C					
Attack target	Local files	C	C	C		C		C
	Network traffic	C	C					C
	Operating system		C	C	C	C	C	C
Major risks	Data theft	C	C	C	C	C	C	C
	Network paralysis	C	C					C
	System damage	C	C					C
Spreading speed		--	----	-		-	--	----
Difficulty of being detected		-	----	--	--	--	--	--

II. Methodology

Up to now, the researches on Android malware mostly concentrate on generic analysis, for instance, collecting different sets of samples and trying to categorize them into groups (detection, clustering or classification), which neglects the specifics that a single family can present. Even though these applications themselves do not target the device itself, they are used for bad purposes. N.

I. AMINuddIN et al. detect Android trojan based on dynamically extracted system calls [46]. However, some trojans that implement environment-aware techniques are able to evade such approach. A detection framework regarding Android Banking Trojans (ABTs) is proposed by Other literature is more focused on detection (i.e. a binary decision system that tells whether a given sample is malicious or not), clustering (i.e. a decision system that can aggregate similar samples into a group without any label from prior knowledge) or classification (i.e. a decision system that can aggregate similar samples into a group with labels from initial knowledge).

We can partition the related work of Android malware detection in two classes: (i) run-time monitoring of the invoked events, (ii) static analysis of the code to detect known patterns of misbehaviors.

Data Collection and Extraction

To collect newer samples of Android trojans, a simple script for crawling some public security repositories has been written, especially the platform of Koodous [93] and Apkdetect [1]. To illustrate, these platforms have already output labels or classification results given a sample.

The sample `fatigue.hazard.horror` has the same tags with `spider.cheese.sketch`, thus generate the same label. Although `alone.network.curve` and `behind.limit.track` possess one more label, **banker**, they also generate the same label **BlackRock**, as the label **banker** refers to a general type of malware that targets the victim's bank card information. Compared with the label **BlackRock**, it is less specific and unique in terms of the coverage.

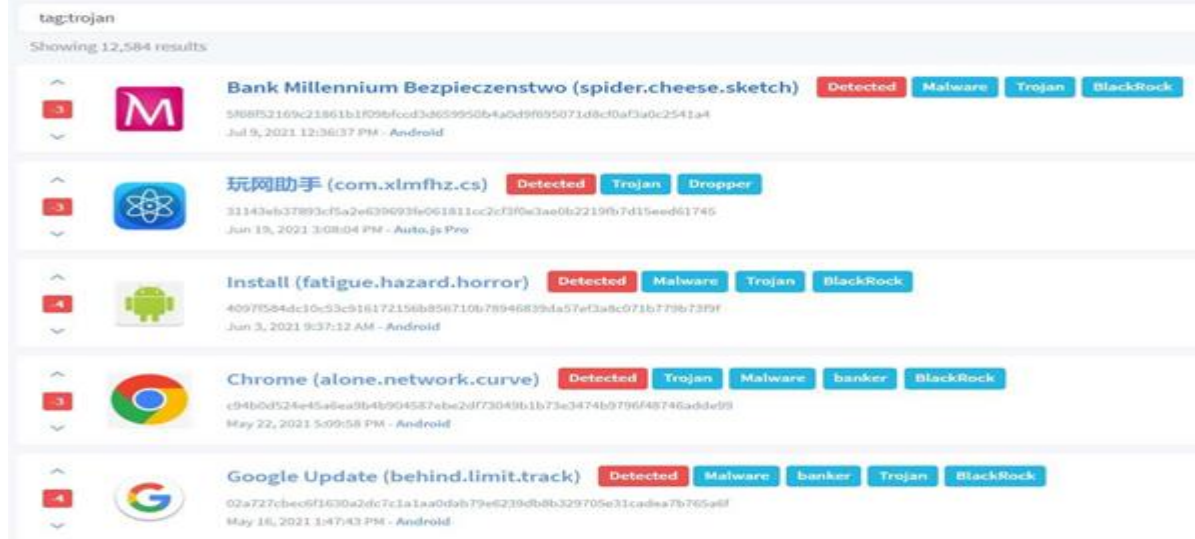


Figure 2: The screenshot that the hashtags added by an analyst's comment

Apkdetect is similar with Koodous. The differences are that Apkdetect does not provide any convenient API for queries and that the classification name produced by Apkdetect is based on the matching of the configurations of different malware, which are observed and submitted by security researchers. Due to the fact that Apkdetect does not possess a large volume of samples, a manual collection method is adopted, i.e. before loading the webpage, a chrome browser where the webpage is loaded is used. Specially, the network recording function is activated, which is able to capture and preserve all requests and responses in logs. Upon typing in the familyname that is to be collected, the returned responses mainly include two parts, i.e. one with brief information including uploaded file name, file type, its MD5 hash and the most probable name produced (Figure 2); the other with more details, i.e. the configuration matched and the control & command server extracted from the sample (Figure 3). The second part is parsed to collect the MD5 hash, family name and matched configuration.

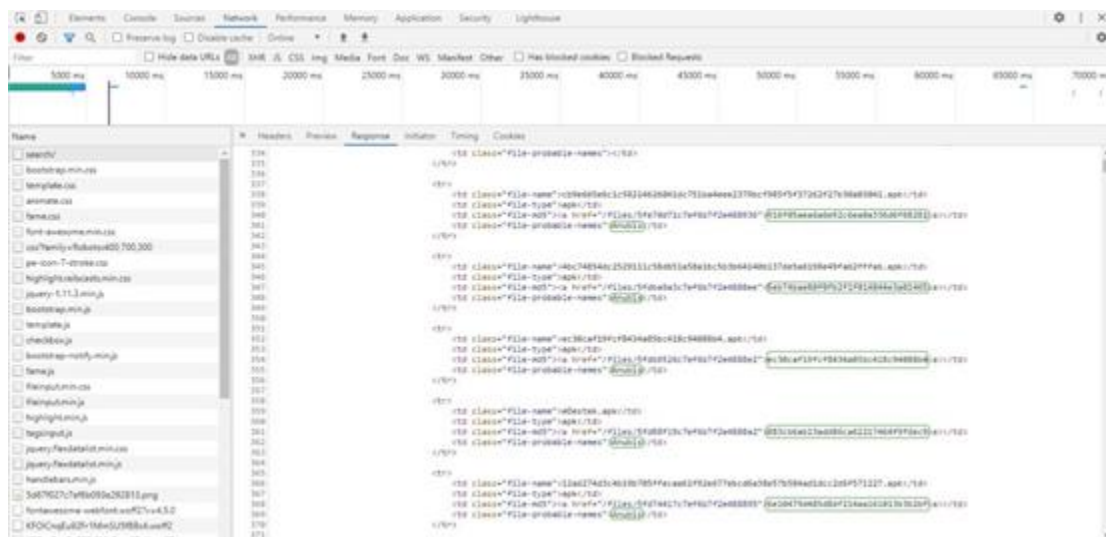


Figure 3: The captured brief information that matches the family name (e.g. Anubis) input in the search box

III. Conclusion and Future Work

This paper investigates recent Android trojans through both static and dynamic analysis to reveal the techniques used in modern malware attacks. Most trojans rely on an active command-and-control (C&C) server to function effectively. Without it, even with elevated privileges, attackers gain little benefit, as financial motives typically drive trojan distribution. While aggressive adware may yield minor profits, it risks immediate detection and removal. In contrast, well-crafted trojans that maintain user trust are more profitable, emphasizing the need to evade detection as long as possible. The thesis also examines *Anubis*, a prominent Android trojan family, analyzing its evolution and technical variations. Despite changes in implementation, core code blocks remain consistent. Its phishing tactics, overlay techniques, and obfuscation methods are detailed to aid understanding of both past and emerging threats. This work was presented at the 7th International Conference on Dependable Systems and Their Applications. For future research, enhancing hybrid (static + dynamic) analysis is crucial, especially against advanced payload-loading methods like

dynamic and native loading. Tools like Qihoo 360 use multi-layered encryption to obscure malicious payloads, rendering static analysis ineffective. Meanwhile, dynamic analysis must adapt by varying infrastructure parameters to avoid detection by malware that checks for emulator artifacts before executing.

References

1. apkdetect, <https://app.apkdetect.com/>, @pr3wtd, accessed: February 2020.
2. "Mobile operating system market share worldwide, 2020," <https://gs.statcounter.com/os-market-share/mobile/worldwide>, 2020.
3. E. Alepis and C. Patsakis, "Hey doc, is this normal?: exploring android permissions in the post marshmallow era," in *International Conference on Security, Privacy, and Applied Cryptography Engineering*. Springer, 2017, pp. 53–73.
4. J. Gamba, M. Rashed, A. Razaghpanah, J. Tapiador, and N. Vallina-Rodriguez, "An analysis of pre-installed android software," *arXiv preprint arXiv:1905.02713*, 2019.
5. L. Wei, Y. Liu, and S.-C. Cheung, "Taming android fragmentation: Characterizing and detecting compatibility issues for android apps," in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, 2016, pp. 226–237.
6. M. Zheng, P. P. Lee, and J. C. Lui, "Adam: an automatic and extensible platform to stress test android anti-virus systems," in *International conference on detection of intrusions and malware, and vulnerability assessment*. Springer, 2012, pp. 82–101.
7. V. Chebyshev, "Mobile malware evolution 2019," <https://securelist.com/mobile-malware-evolution-2019/96280/>, 2020.
8. C. Bai, Q. Han, G. Mezzour, F. Pierazzi, and V. Subrahmanian, "Dbank: Predictive behavioral analysis of recent android banking trojans," *IEEE Transactions on Dependable and Secure Computing*, 2019.
9. Y. Zhang, G. Xiao, Z. Zheng, T. Zhu, I. Tsang, and Y. Sui, "An empirical study of code deobfuscations on detecting obfuscated android piggybacked apps," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 914–926.
10. Y. Tang, H. Wang, X. Zhan, X. Luo, Y. Zhou, H. Zhou, Q. Yan, Y. Sui, and J. W. Keung, "A systematical study on application performance management libraries for apps," *IEEE Transactions on Software Engineering*, 2021.
11. L. Wang, R. He, H. Wang, P. Xia, Y. Li, L. Wu, Y. Zhou, X. Luo, Y. Sui, Y. Guo et al., "Beyond the virus: a first look at coronavirus-themed android malware," *Empirical Software Engineering*, vol. 26, no. 4, pp. 1–38, 2021.
12. Saurabh Chauhan, Dharamveer Singh, Atul Kumar Singh (2022) "Artificial Intelligence In The Military: An Overview Of The Capabilities, Applications, And Challenges", *Journal of Survey in Fisheries Sciences*, Vol 9 (2) pp 984-991. <https://doi.org/10.53555/sfs.v9i2.2911>
13. Kiran, Dharamveer Singh, Nitin Goyal, (2023) "Analysis Of How Digital Marketing Affect By Voice Search", *Journal of Survey in Fisheries Sciences*, Vol. 30 (2) 407-412. <https://doi.org/10.53555/sfs.v10i3.2890>
14. Yukti Tyagi, Dharamveer Singh, Ramander Singh, Sudhir Dawra (2024) "Analysis Of The Most Recent Trojans On The Android Operating System", *Educational Administration: Theory and Practice*, Vol. 30(2) 1320-1327. <https://doi.org/10.53555/kuey.v30i2.6846>
15. Shivance Singh, Dharamveer Singh, Ravindra Chauhan (2023) "Manufacturing Industry: A Sustainability Perspective On Cloud And Edge Computing", *Journal of Survey in Fisheries Sciences*, pp 1592-1598. <https://doi.org/10.53555/sfs.v10i2.2889>