EVALUATING THE EFFICACY AND CAPABILITY OF DIFFERENT COVERT CHANNEL DETECTION TOOLS IN AN ANDROID APPLICATION

Abhinav K. Shah

Research Scholar, School of Cyber Security & Digital Forensics, National Forensic Sciences University,
Gandhinagar, Gujarat 382007, India,
abhinavk.shah@gmail.com

Dr. Digvijaysinh M. Rathod

Associate Dean, School of Cyber Security & Digital Forensics, National Forensic Sciences University,
Gandhinagar, Gujarat 382007, India,
digvijay.rathod@nfsu.ac.in

Jeet Rami

Threat Analyst, Microsoft,
Hyderabad, Telangana 500032, India
jeetrami39@gmail.com

Abstract

Covert channel attacks have been a significant concern in recent times. With the evolution of the internet and technology, users have adapted to it differently. Cyber security is one of the areas which has opened doors in terms of research and development. Smartphone users have grown considerably in the last decade. With this growth, on the other side, attackers have found an opportunity to create covert channels to leak sensitive information from android devices. Different attacks have been performed in the past, which can leak information such as contact numbers, sms, call logs, battery state, alarm state, and much more details similarly. Various security researchers have already worked in this domain and have built different static and dynamic detection tools to identify covert channels in android devices or applications. In this paper, the author has provided a systematic review of different detection tools used to identify such attacks. The author has executed various detection tools against an application that was able to leak sensitive information through a covert channel. The author also provides a comparative analysis of results by executing different tools in Android. At the end of this paper, the author concludes by giving limitations present in the existing tools.

Keywords**: Cyber Security; Covert Channel; Android Application.**

1. Introduction

Technology has been growing exponentially for the last couple of decades. With this, there has been enormous growth in various domains like artificial intelligence, machine learning, cyber security, quantum computing, blockchain, and virtual and augmented reality [1][6]. As per the report from Oberlo [4], smartphone usage increases by around 4.2% every year. At the end of 2023, the estimated count will jump to about 6.8 billion. COVID-19 brought a massive increase in the usage of smartphones. Heba [3], in her survey, concluded that there was an increase of about 57% in smartphone usage. Within this, 42% of the users used extensively for almost 6 hours daily. In another research carried out by Serra [2], she concluded her work with some school children who used smartphones more actively in the age group between 6-18 years.

As technology has grown, it has opened doors in various fields. People have adapted to this technology in different ways. Some people use it to enhance the current system, known as a researcher and scientists, and others who use it to destroy the current systems are known as attackers. These attackers have performed various attacks, such as DOS, Ransomware, MITM, and many more, resulting in considerable losses to the organizations [5]. Various research is being done in this domain to prevent such types of attacks in the future. With the increase in the usage of smartphones, attackers have found an open pool to create covert channels and leak sensitive information from users' devices. Android was first introduced in 2003 by Andy and his team [11], and the first Android version was in 2007. Then, different versions with performance and security enhancements were released.

The term covert channel was first introduced in 1973 by Lampson [7]. He introduced covert channels with the help of prisoners' problems. He demonstrated how two entities shared information through a covert channel [8][9]. In simpler terms, a covert channel is a way to pass unauthorized information by violating the process of information transfer [10]. Several covert channel attacks have resulted in considerable losses in the last few decades. Chandra [12] in his research presented a study about the different covert channel attacks on smartphones. He demonstrated that covert channels could be at three levels: application, operating system, and hardware. He concluded about the possibility of shared resources, which can lead to creating a covert channel. Different detection systems are already in place, but many gaps need handling to detect such covert channels in android. In another instance, Casolare [13] demonstrated a covert channel with the help of an accelerometer sensor and vibrator motor in android. The author concluded that the given channel is very silent and difficult to detect since it does not leave any communication open between colluding applications.

In this paper, the author has crystallized more on the detection tools used to identify covert channels in android applications and devices. Moving towards section 2, the author has discussed different current detection tools used to identify covert channels. Section 3 talks about the limitations present in the existing tools. In section 4, the author discusses the experiments performed on a malicious application to give concluded results of the different tools and provides a comparative analysis of the outcomes. The later part of the paper ended with a conclusion followed by references.

2. Literature Review

In this section, the author has discussed the various detection tools, techniques, and frameworks widely used to identify covert channels in android applications. A study by Qiang [14] proposed a framework for auditing covert communication in Android applications. In this framework, the author concluded that this approach could considerably reduce the data transfer rate with negligible overhead. The author experimented with this framework at the application layer. Another research by Abro [15] illustrated the open challenges in detecting covert channels. The author also compared various approaches which are present in order to detect covert channels. In the end, the author also proposed an approach to identify app collusion. Goher [16] discovered the detection techniques used to identify covert channels for storage and timing-based covert channels; the author also compared detection techniques at TCP / IP layer. At the end of this article, the author compared the results for different detection techniques. In another research, Graa [17] proposed a framework for taint analysis detection. The author concludes that this approach can detect sensitive information and most exploits without giving false positives. In this approach, the author primarily focused on static analysis.

Enck [18] [23], in another research, proposed a framework called TaintDroid. With this approach, the author showcased flow tracking in real-time android applications. This framework can identify misuse of any permissions in other android applications. In this approach author also discovered how android applications can leak sensitive information; the application could be inbuilt or any other third-party application. This approach focuses mainly on dynamic analysis, meaning real-time detection is possible. Sensitive information such as Location, IMEI number, Contacts, and Call logs detection is possible with the help of TaintDroid. The author concluded this research by experimenting with 30 applications and found that 15 of them could leak information detected by the given framework. Another research by Arzt [19] [22] brought up a static framework called FlowDroid. In this research, the author focused on android's static taint analysis and compared this framework with other commercial tools like IBM AppScan and Fortify on Demand. In the end, the author concluded that FlowDroid could give better results in terms of accuracy, thus reducing false positives. The author experimented with this framework on around 500 applications from Google Play Store and 1000 from the VirusShare project.

Recent research carried out by Buddhadev [20] proposed FloVasion. A framework developed to detect evasion through non-sensitive variables in Android applications. The author also compared this framework with tools like Google Play Protect, DroidSafe, FlowDroid, TaintDroid, AVL Antivirus, and other commercial tools. The author presented the results and stated that FloVasion could detect evasion more accurately and cover other sensitive aspects. Moving forward in this segment Chaurasia [21] came up with another study where the author developed a framework called DroidBox [24]. This dynamic malware analysis framework detects information flow or covert channels in android applications. DroidBox also has capabilities to identify and analyze file operations, information leakage at the network level, operations that include encryptions in Android API, and code-level analysis.

Siswanto [25] and his team developed a tool named Apkleaks. It is an open-source command line utility tool primarily used to scan Apk files and identify URLs, endpoints, and secrets within the APK. This tool is developed in Python and used to identify static flow evasion in the android application. In another study carried out by Ajin [26], his team prepared a static and dynamic analysis framework called MobSF. This framework can identify vulnerabilities, code obfuscation, weak hashing algorithm, threat analysis, and permission analysis in the android application. Dynamic analyses in MobSF help in executing runtime assessments and instrumented testing. Another study by Georgiu [27] showcased RiskInDroid, a tool for quantifying risk analysis in android applications. In this tool, the author has implemented machine learning techniques such as Support Vector Machine, Naive Bayes Classifier, Gradient Boosting, and Logistic Regression to identify risk. The author stated that this tool could identify declared, exploitable, and unused permissions [28]. Another study by Vincent [29] proposed an open-source static analyzer named StaCoAn. The author stated that this tool could identify API Keys, HardCoded Information, code issues, sensitive keys, and URL. Super [30] is another open source and command line tool developed to identify vulnerabilities by decompressing the APK files. The author has used Rust programming language in developing this tool.

Pithus [31] is another open-source threat intelligence platform that detects code issues, analyzes manifest files, and checks permission misuse within other applications. This tool is still under development and performs only static analysis. Another open-source tool widely used in analyzing android applications is VirusTotal [32]. Virus Total can do static analysis, and in addition, it can also perform different security vendor analyses. Table 1 gives an overview of different detection tools.

Sr. No.	Tool / Framework	Manifest Analysis	File / Code Analysis	Control Flow Analysis	Threat Analysis
1	Flowdroid [19] [22]	No	Yes	No	No
2	Apkleaks [25]	No	No	No	No
3	MobSF [26]	Yes	Yes	No	No
4	RiskInDroid [27] [28]	Yes	No	No	No
5	StaCoAn [29]	No	No	Yes	No
6	Super [30]	Yes	Yes	Yes	No
7	Pithus [31]	Yes	Yes	Yes	Yes
8	Virus Total [32]	Yes	Yes	No	Yes

Table 1. An overview of different detection tools.

3. Limitations in Existing Tools

In the previous section, the author discussed the different detection tools, frameworks and methods used in performing static and dynamic analysis, which further help detect covert channels. In this section, the author describes the limitations present in the existing tools. In android applications, data flow happens at various layers, such as code, network, and application levels. When we talk about covert channels in the past various attacks have been performed, which resulted in leaking sensitive information.

Various tools and techniques are already in use to detect such information flow. However, when we went deep into the tools, it was identified that only some tools could detect covert audio and video channels. All the tools do different kinds of analysis, such as Manifest, Permission, Code, and Control Flow, but when it comes to audio and video, these tools cannot detect such phenomena. Apart from this, results generated from the tools are often considered false positives, so to improve efficiency, different researchers are already working on it. The second limitation is that some of the covert channels are time-based. Therefore tools cannot be run at different time intervals; otherwise, they could not identify patterns created in time-based covert channels.

Another limitation of the static analysis tool is that it can only identify leaks at the APK level. However, since covert channel formation is at runtime, it is mandatory to do dynamic analysis to get results in time. In the previous research, the author found that most of the research considered only static, and some had considered only dynamic analysis for covert channel detection. In order to get accurate results, we should consider static and dynamic analysis. With the help of static analysis, one can know about permission usage and any hardcoded information stored at the code level. The second phase is performing dynamic analysis, where one can see code-level analysis in runtime, control flow analysis, network analysis, and application analysis.

4. Experiments and Results

In the previous section, the author discussed existing tools' limitations. The author has performed various experiments with the existing tools in this section and further analyzed the results. The author has prepared a file-sharing android application that can share audio files from one user to another. This application creates a covert audio channel that sends sensitive information from one user to another without taking permission. The author has considered this application a benchmark to execute all the tools. Table 2 describes the configurations of the android testing device and windows laptop.

Sr. No.	Testing Device	Type of Analysis	Configuration
1	OnePlus 3T Android Phone	Dynamic	Ram: 6GB CPU: SnapDragon 821
2	Dell Latitude 6430u Laptop	Static	Ram: 16GB CPU: i7 3rd Gen

Table 2. Testing Device Configurations.

With the help of a windows laptop, we performed static analysis. The author had to pass the APK file for static analysis, and the tool completed the rest of the analysis. For performing dynamic analysis author used both an android device and a laptop to perform the analysis in runtime. Table 3 describes the requirements and execution time taken by each tool, along with detection details.

Sr. No.	Tool Name	Installation Requirements	Type of Tool	Execution Time	Type of Analysis	Is Detection Possible?
1	Flowdroid [19] [22]	Java	Open Source	3 Mins	Static	No
2	Apkleaks [25]	Python	Open Source	2 Mins	Static	No
3	MobSF [26]	Python, JDK, OpenSSL, Microsoft Visual C++ Build tools	Open Source	15 Mins	Static	No
				30 Mins	Dynamic	No
4	RiskInDroid [27] [28]	Python	Open Source	5 Mins	Static	No
5	StaCoAn [29]	Python, Git	Open Source	2 Mins	Static	No
6	Super [30]	Java	Open Source	3 Mins	Static	No
7	Pithus [31]	Browser Based	Open Source	10 Mins	Static	No
8	Virus Total [32]	Browser Based	Open Source	5 Mins	Static	No
9	AppKnox [34]	Browser Based	Commercial	5 Mins	Static	No
				15 Mins	Dynamic	No

Table 3. Requirements, Execution Time, and Type of Analysis for different detection tools.

The author has experimented with different detection tools and has presented the results in the above table. The results show that most of the tools were open source, and some of them could perform both static and dynamic analysis. Each of the tools has taken a different time respectively. The author also tried to detect the covert audio channel with these tools. However, the tools were not able to detect the same. The author also tried experimenting with tools like TaintDroid and DroidBox. However, in Android Version 9, those tools have been deprecated.

5. Conclusion

Covert channels are a current threat, and passing sensitive information from one user to another is possible. Various tools and techniques are already in place. However, they must be more comprehensive to detect such leakages in android applications. To dig deeper into this author here did a review of the different detection tools by experimenting with them against an android application that can covertly share information.

In this paper author executed various tools, including static and dynamic analysis. The author has worked intensely with each tool and showcased the results. From the results, the author concludes that none of the tools could detect a covert audio channel in the android application. However, MobSF and Pithus could give a good amount of information from the application, like permission details, manifest analysis, code analysis, and application analysis. The rest of the tools could only identify the permission declared in the manifest file.

Detection of a covert audio channel is still an open issue. It can be an open area for the research community. The references follow the later section.

References

[1] Viswanathan A G (2022) 5 Trends Transforming Information Technology Industry In 2022 https://www.computer.org/publications/tech-news/trends/5-trends-transforming-information-technology-industry (last accessed 14 March 2023)

[2] Serra, G., Lo Scalzo, L., Giuffrè, M., Ferrara, P., & Corsello, G. (2021). Smartphone use and addiction during the coronavirus disease 2019 (COVID-19) pandemic: cohort study on 184 Italian children and adolescents. Italian Journal of Pediatrics, 47(1), 1-10.

[3] Saadeh, H., Al Fayez, R. Q., Al Refaei, A., Shewaikani, N., & Khawaldah, H. (2021). Smartphone Use Among University Students During COVID-19 Quarantine: An Ethical Trigger. Frontiers in Public Health, 9. https://doi.org/10.3389/fpubh.2021.600134

[4] Oberlo (2023) HOW MANY PEOPLE HAVE SMARTPHONES IN 2023? https://www.oberlo.in/statistics/how-many-people-have-smartphones (last accessed 14 March 2023)

[5] Shah, Abhinav, Digvijaysinh Rathod, and Dharmesh Dave. "DDoS Attack Detection Using Artificial Neural Network." International Conference on Computing Science, Communication and Security. Springer, Cham, 2021.

[6] Raval, Helly Yogeshkumar, Satyen M. Parikh, and Hiral R. Patel. "Self-Maintained Health Surveillance Artificial Intelligence Assistant." Handbook of Research on Lifestyle Sustainability and Management Solutions Using AI, Big Data Analytics, and Visualization. IGI Global, 2022. 168-184.

[7] Lampson, Butler W. "A note on the confinement problem." Communications of the ACM 16.10 (1973): 613-615.

[8] Anderson, Ross J., and Fabien AP Petitcolas. "On the limits of steganography." IEEE Journal on selected areas in communications 16.4 (1998): 474-481.

[9] John McFarland (2017) Covert Channels: An Overview https://www.researchgate.net/publication/330875417_Covert_Channels_An_Overview (last accessed 14 March 2023)

[10] Prof. Claudio Cilli (2017) Understanding Covert Channels of Communication https://www.isaca.org/resources/news-and-trends/isaca-now-blog/2017/understanding-covert-channels-of-communication (last accessed 14 March 2023)

[11] Aayushi (2022) History of Android https://www.geeksforgeeks.org/history-of-android/ (last accessed 14 March 2023)

[12] Chandra, S., Lin, Z., Kundu, A., & Khan, L. (2015). Towards a systematic study of the covert channel attacks in smartphones. In International Conference on Security and Privacy in Communication Networks: 10th International ICST Conference, SecureComm 2014, Beijing, China, September 24-26, 2014, Revised Selected Papers, Part I 10 (pp. 427-435). Springer International Publishing.

[13] Casolare, R., Martinelli, F., Mercaldo, F., & Santone, A. (2021). Colluding Covert Channel for Malicious Information Exfiltration in Android Environment. In ICISSP (pp. 811-818).
[14] Qiang, W., Xin, S., Jin, H., & Sun, G. (2017). DroidAuditor: A framework for auditing covert communication on Android. Concurrency and Computation: Practice and Experience, 29(19), e4205.
[15] Abro, F. I., Rajarajan, M., Chen, T. M., & Rahulamathavan, Y. (2017). Android application collusion demystified. In Future Network Systems and Security: Third International Conference, FNSS 2017, Gainesville, FL, USA, August 31-September 2, 2017, Proceedings (pp. 176-187). Springer International Publishing.
[16] Goher, S. Z., Javed, B., & Saqib, N. A. (2012, December). Covert channel detection: A survey based analysis. In High Capacity Optical Networks and Emerging/Enabling Technologies (pp. 057-065). IEEE.
[17] Graa, M., Cuppens-Boulahia, N., Cuppens, F., & Cavalli, A. (2012). Detecting control flow in smarphones: Combining static and dynamic analyses. In Cyberspace Safety and Security: 4th International Symposium, CSS 2012, Melbourne, Australia, December 12-13, 2012. Proceedings 4 (pp. 33-47). Springer Berlin Heidelberg.
[18] Enck, W., Gilbert, P., Han, S., Tendulkar, V., Chun, B. G., Cox, L. P., ... & Sheth, A. N. (2014). Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. ACM Transactions on Computer Systems (TOCS), 32(2), 1-29.
[19] Arzt, Steven, et al. "Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps." Acm Sigplan Notices 49.6 (2014): 259-269.
[20] Buddhadev, Bharat, et al. "FloVasion: towards detection of non-sensitive variable based evasive information-flow in android apps." IETE Journal of Research 68.4 (2022): 2580-2594.
[21] Chaurasia, P. (2015). Dynamic analysis of Android malware using DroidBox (Doctoral dissertation, Tennessee State University).
[22] Github https://github.com/secure-software-engineering/FlowDroid (last accessed 14 March 2023)
[23] Github https://github.com/TaintDroid/taintdroid-project (last accessed 14 March 2023)
[24] Github https://github.com/pjlantz/droidbox (last accessed 14 March 2023)
[25] Dwi Siswanto (2021) Github https://github.com/dwisiswant0/apkleaks (last accessed 14 March 2023)
[26] Ajin Abraham (2022) Github https://github.com/MobSF/Mobile-Security-Framework- MobSF (last accessed 14 March 2023)
[27] Merlo, A., & Georgiu, G. C. (2017). Riskindroid: Machine learning-based risk analysis on android. In ICT Systems Security and Privacy Protection: 32nd IFIP TC 11 International Conference, SEC 2017, Rome, Italy, May 29-31, 2017, Proceedings 32 (pp. 538-552). Springer International Publishing.
[28] Claudiu Github (2022) https://github.com/ClaudiuGeorgiu/RiskInDroid (last accessed 14 March 2023)
[29] Vincent Cox (2022) Github https://github.com/vincentcox/StaCoAn (last accessed 14 March 2023)
[30] Github https://github.com/SUPERAndroidAnalyzer/super (last accessed 14 March 2023)
[31] Pithus (2022) https://beta.pithus.org/about/ (last accessed 14 March 2023)
[32] VirtusTotal https://www.virustotal.com/gui/home/upload (last accessed 14 March 2023)
[33] Android (2023) https://www.android.com/versions/pie-9-0/ (last accessed 14 March 2023)
[34] appKnox (2023) https://www.appknox.com/vulnerability-assessment/dynamic-application-security-testing-dast (last accessed 14 March 2023)

Authors Profile

Abhinav K. Shah, he is currently pursuing his Ph.D. from National Forensic Sciences University. Along with this he is working as a Security Analyst at ArmorCode Inc. He has approximately 7 years of experience in the field of security and development for both the private and public sectors, which include Application Security Assessment, Vulnerability Assessment, Penetration Testing, Configuration Review, Vulnerability Management, Risk Reporting, Software Development, Web Application Development, Mobile Application Development, and so on.

Dr. Digvijaysinh M. Rathod, Dean of School of Open Learning and Associate Dean of School of Cyber Security and Digital Forensics and Head of Center of Excellence in Cyber Security, National Forensic Sciences University, Institution of National Importance, MHA, Govt. of India. He has 18 years of teaching, consultancy and research experience in the domain of Cyber Security and Digital Forensic. He has provided his service as a cyber security consultant to the Govt. of Saudi Arabia and Bangladesh and also trained more than 800 law enforcement officers from the India and outside India also. He is also involved in the high profile and important cases of digital forensic analysis in the India. His area of interest is Industrial Control System Security, Web and Mobile Security, Dark and Deep web investigation.

Jeet Rami, he is working as a Threat Hunter & Researcher at Microsoft. He is a recent graduate, pursued his master's in Cyber Security from National Forensic Sciences University. He loves low end stuff development plus infosec things. So he is more into Windows Exploit Development. He worked on stack based exploits, heap exploits and currently tried to get his hands on kernel exploit development. In malware development he worked on some of the malware development techniques. Previously he was working on his own Linux Distro Development.