Unmasking the Malware Using Android Debug Bridge

Himanshi, Harjas Kalsi, Annu, and Akanksha Dhamija

Abstract The growing number of malware puts security in peril, several studies highlight the stark consequences of malware, so the abstraction of malware is crucial to keep personal information confidential. The motive of this research is the detection and abstraction of malware present on an android device using the Android Debug Bridge (ADB), Android debug bridge is utilized for communicating with an android device and acquiring information about the device. The detection process is achieved by creating a shell script and executing it on a Unix terminal. Hereafter, if malware is detected the script will automatically abstract them and exhibit the number of malware detected. Along with detection, a comparative analysis is performed which indicates various advantages over free anti-malware software one of them is a sanction, the antiviruses present on the Google Play Store can share or store data and shows superfluous ads, these antiviruses require many sanctions from the user before scanning, while in this research no third party software is required user can itself run the script without involving any third party tool or without compromising security. This research unveils malware from android devices in a security-friendly way.

Keywords ADB · USB · Shell script · Terminal

1 Introduction

The ADB is a multipurpose command-line tool which when connected with an android device through USB cable allows communication and various other actions

Himanshi (✉) · H. Kalsi · Annu · A. Dhamija
BPIT, GGSIPU, New Delhi, India
e-mail: himanshi0875@gmail.com

H. Kalsi
e-mail: harjaskalsi12@gmail.com

Annu
e-mail: annugoel1234@gmail.com

A. Dhamija
e-mail: akankshadhamija12@gmail.com

S. Maurya et al. (eds.), *Cyber Technologies and Emerging Sciences*, Lecture Notes in Networks and Systems 467, https://doi.org/10.1007/978-981-19-2538-2_9

like installing and debugging applications using a Unix shell. Its three components are:

1. Client—It sends and runs commands on a development computer. By issuing a command from a command-line terminal, one can invoke the client.
2. Daemon—It runs in the background, executes commands on an android device.
3. Server—It runs as a background process on the development computer and establishes communication between the daemon and the client [1].

A study conducted by International Data Corporation (IDC) states that the Android platform is still dominating the smartphone industry with a share of more than 86%. Its popularity is incrementing rapidly with more developing sensitive operations and features being integrated [2]. But along with this, Android malware is also growing, and that too with more complex anti-analysis techniques and logic [3]. Mobile malware is a malicious software which specifically targets mobile operating systems. Nowadays, malware can be utilized for numerous purposes like for tracking user activity, spreading spam, stealing data, etc. [4]. It is generally used for more than 1 purpose. To explain malware classification, we require two terms: Malware type (Based on its General functionality, what it does) and Malware family (based on its particular functionality, how it acts) [5, 6].

2 Literature Review

The paper named "M0droid: An android behavioural-based malware detection model" [7] represents a model used for detection of malware codes or harmful scripts on an Android device. The process flow of the model given in the paper was to install an application on a mobile device and then analyze the data on the server. The results from the experiments conducted demonstrates that the detection rate of the model used is 60.16% where false positives percentage was 39.43%.

SAndroid [8], the tool which enhanced the malware and harmful scripts detection by application signatures [9, 10], over detection and tracking of malicious and harmful process signatures [11, 12]. Though, there are some disadvantages of this method like high amount of battery consumption.

Canfora et al. [13] estimated some techniques to detect malevolent apps. Their perspective is potent for desktop malwares and categorize the ill-natured applications. Practically, they attain a precision "0.96" to differentiate the malicious applications, and "0.978" to determine malware family.

Feizollah et al. [14] came up with AndroDialysis, to judge how efficient the android application intent: explicit and implicit, like specification to check the ill-natured applications. They convey that the intents have semantically better structures as compared to other attributes [15]. Though, they said that not all these features are the final solution, and also it can be used with other well known positive features [16, 17]. It's outcome depends on the probe of data present of 7406 apps (5560 infected apps and 1846 clean). They attain 91% accuracy by operating the Android Intents, while 83% uses Android permissions and the merging of these characteristics they get

the spotting rate as 95.5%. They declare that in the process of malware identification, Intents are more worthwhile than the permissions [18].

Then, another paper named "Comparative study of mobile forensic tools" [19] described a method termed as FAMOUS stands for Forensic Analysis of Mobile Devices. This method examines app permission to determine whether the app is a malware or not. Tools like Droid Scope [20] and Profile Droid for analyzing apk files so that there is any scope of malware in them. The paper named "On the Efficacy of Using Android Debugging Bridge for Android Device Forensics" [21], this paper concluded that ADB is unable to flag any file but can effectively find and extract the present malware files [22].

The literature review includes Malware detection using System Calls, Intents, Manifest file, Permissions required by the application and ADB (Android Debug Bridge). Very few studies have been done using ADB and more work can be performed by using Android Debug Bridge as a malware detector [23–25].

3 Implementation

For the detection of malware in any android phone, a shell script and a text file are created. The name of the shell script is checkapk.sh and that of the text file is malwarehashes.txt. checkapk.sh consists of a script which is used for detecting and abstracting the malware in the android phones and malwarehashes.txt file consists of hash codes of malware.

3.1 Steps of Implementation

Step 1: Connect the phone with a laptop or computer running the ADB command line. Turn on the USB debugging from the developer options and sanction the permissions. For the developer options click on the build number 7 times in the about phone menu.
Step 2: Type commands in the command line:

1. "adb shell"—to grant permission for connecting the android phone with ADB
2. "exit"

Step 3: Type command "bash checkapk.sh" and the file will start running.

The process of detection and abstraction of malware takes place until the "Success" message appears.

Backend working:

1. The shell script was written to detect and remove malware consists of ADB and Linux commands, which is used for fetching the packages on the android phone and removing it if malware is detected
2. When the shell script is executed, the packages present on the android phone are fetched with their consummate name and location. By using the location of the

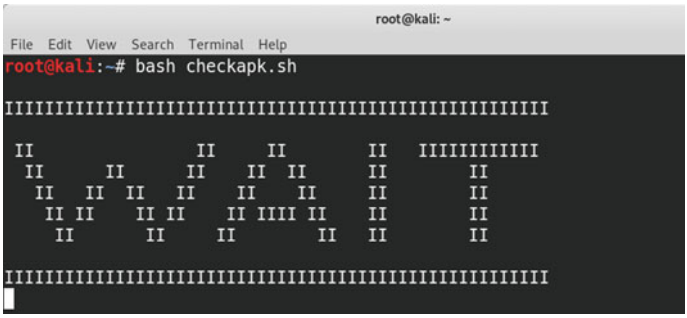

Fig. 1 Running shell script

package, md5sum of that package is generated and stored in a file designated as apphashes.txt.

3. After generating the file apphashes.txt, the hash codes of the file are compared with the hash codes of malwares which are already stored in a file designated as malwarehashes.txt.

After comparing both the files with each other, If any hash code is obtained which is mundane in both the files, then the package of that hashcode will be permanently deleted and uninstalled And, if no hash code is mundane then simply a message appears on the screen "No malware detected".

In the terminus, "rm" command is used in a script to abstract all the files generated during implementation to minimize the internal system storage utilized by the script (Fig. 1).

4 Result

The script created in this research can detect and remove malware utilizing the android debug bridge. There are two scenarios since an android device may or may not have any malware. The Figs. 2 and 3 shows the output when the device doesn't have any type of malware and when the system has only one malware respectively.

4.1 Comparative Analysis of Our Study with Pre-existing Tools

For the comparative analysis, malware is designed using the MSF venom and injected within an android device then some antivirus is used to detect that malware, and likewise the script created in this research is used and the analysis is shown.

Table 1 shows the comparison between different antiviruses available. Despite these tools, certain applications present on the play store required credit card information for a free tribulation of some days and then pleaded for payment. This table

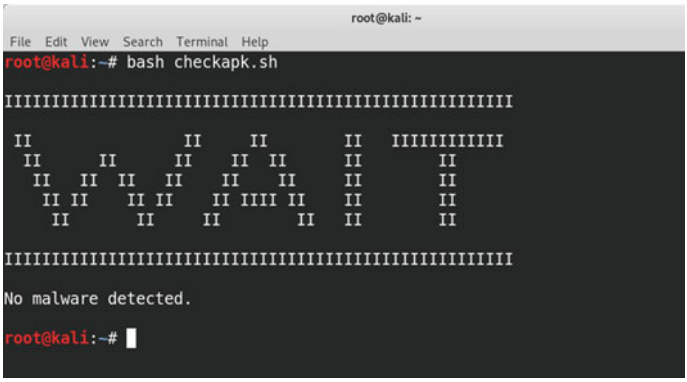

Fig. 2 No malware detected

Fig. 3 One malware detected

Table 1 Comparative analysis

Antivirus	Permission required					
	Photos and media files	Modify system settings (erase data, change screen look etc.)	Storage	Camera	Location	SMS
Samsung inbuilt scanner	☑	☑	☑	☑	☑	☑
AVG	☑	☑	☑			
Avast	☑		☑	☑		
Spyware detector		☑ modify SD card data	☑			
Kaspersky internet security			☑	☑	☑	☑
Spy apps finder			☑			☑
Antispy and spyware scanner	☑		☑			

compare them on the basis of permission required, whenever we use any third-party tool then there is always a security concern due to the fact that the tool can access some confidential information due to the various permissions required to run the software but while using ADB only the owner of the device has the permission and there is no involvement of any third party tool and can efficiently detect and remove malware.

5 Conclusion

This paper concludes that users can themselves detect and remove malware present inside an android device without compromising security by using android debug bridge. It is efficacious for detecting and abstracting the malicious software or application with the avail of shell script without involving any third party application. Most of the free third-party applications are not able to detect and abstract the malware injected on the testing phone. These third-party applications require sanction for accessing the internal storage and the external SD card or recollection card connected and additionally the credit card and debit card information for free trial. While in this research the shell script generated can be used by the user itself without involving any third-party application or software. Users can execute the script on their system and can abstract the malware if detected.

References

1. Amarante J, Barros JP (2017) Exploring USB connection vulnerabilities on android devices breaches using the android debug bridge. In: Proceedings of the 14th international joint conference on e-business and telecommunications (ICETE 2017)
2. Lee Y, Larsen KR (2017) Threat or coping appraisal: determinants of SMB executives' decision to adopt anti-malware software
3. Wei F, Li Y, Roy S, Ou X, Zhou W (2017) Deep ground truth analysis of current android malware
4. Salah A, Shalabi E, Khedr W (2020) A lightweight android malware classifier using novel feature selection methods
5. Banina S, Dyrkolbotnab GO (2018) Multinomial malware classification via low-level features
6. Al-rimy BAS, Maarof MA, Shaid SZM (2018) Ransomware threat success factors, taxonomy, and countermeasures: a survey and research directions
7. Damshenas M, Dehghantanha A, Choo KKR, Mahmud R (2015) M0droid: an android behavioural-based malware detection model. J Inf Privacy Secur 11(3):141–157
8. Niazi RH, Shamsi JA, Waseem T, Khan MM (2015) Signature-based detection of privilege-escalation attacks on Android. In: 2015 Conference on information assurance and cyber security (CIACS), pp 44–49, Dec 2015
9. Yang C et al. (2015) Using provenance patterns to vet sensitive behaviors in Android apps. In: International conference on security and privacy in communication systems. Springer International Publishing
10. Duc NV, Giang PT, Vi PM, Bhatt MS et al (2015) Int J Comp Technol Appl 6(5):852–856. Conference Paper, November 2015

11. Kumar M, Mishra BK, Panda TC (2016) Predator-prey models on interaction between computer worms, trojan horse and antivirus software inside a computer system
12. Rastogi V, Chen Y, Jiang X (2014) Catch me if you can: evaluating android anti-malware against transformation attacks
13. Canfora G, Mercaldo F, Visaggio CA (2016) An HMM and structural entropy based detector for android malware: an empirical study. Comput Secur 61:1–18
14. Feizollah A, Anuar NB, Salleh R, Suarez-Tangil G, Furnell S (2017) Androdialysis: analysis of android intent effectiveness in malware detection. Comput Secur 65:121–134
15. Zhang M, Song G, Chen L (2016) A state feedback impulse model for computer worm control
16. Eugene Schultz E Dr (2003) Pandora's box: spyware, adware, auto execution, and NGSCB
17. English ED (2014) Detection of bot-infected Computers using a web browser
18. Dawson JA, McDonald JT, Shropshire J, Andel TR, Luckett P, Hively L (2017) Rootkit detection through phase-space analysis of power voltage measurements
19. Agrawal AK, Khatri P, Sinha SR (2018) Comparative study of mobile forensic tools. In: Advances in data and information sciences. Springer, Singapore, pp 39–47
20. Amer N, Al-Halabi YS (2018) Android forensics tools and security mechanism: survey paper. In: Proceedings ACM the fourth international conference on engineering & MIS, p 12
21. Easttom C, Sanders W (2019) On the efficacy of using android debugging bridge for android device forensics. In: IEEE 10th annual ubiquitous computing, electronics and mobile communication conference, pp 0734
22. Zheng M, Sun M, Lui JCS (2013) DroidAnalytics: a signature based analytic system to collect, extract, analyze and associate android malware
23. Tan L, Liu C, Li Z, Wang X, Zhou Y, Zhai C (2013) Bug characteristics in open source software
24. Chatterjee R, Doerfler P, Orgad H, Havron S, Palmer J, Freed D, Levy K, Dell N, McCoy D, Ristenpart T (2018) The spyware used in intimate partner violence
25. Fatima U, Ali M, Ahmed N, Rafiq M (2018) Numerical modeling of susceptible latent breaking-out quarantine computer virus epidemic dynamics