Ceballos Delgado *et al. J Surveill Secur Saf* 2022;3:3-15
DOI: 10.20517/jsss.2021.26

Journal of Surveillance, Security and Safety

Original Article

Open Access

Check for updates

Android App Antiforensics

Alberto Ceballos Delgado, Bing Zhou

Department of Computer Science, Sam Houston State University, Huntsville, TX 77340, USA.

Correspondence to: Alberto Ceballos Delgado, Department of Computer Science, Sam Houston State University, 1905 University Ave, Huntsville, TX 77340, USA. E-mail: aac088@shsu.edu

How to cite this article: Ceballos Delgado A, Zhou B. Android App Antiforensics. *J Surveill Secur Saf* 2022;3:3-15. https://dx.doi.org/10.20517/jsss.2021.26

Received: 5 Dec 2021 **First Decision:** 7 Jan 2022 **Revised:** 24 Jan 2022 **Accepted:** 4 Mar 2022 **Published:** 29 Mar 2022

Academic Editors: Zheng Xu, Pedro Peris-Lopez **Copy Editor:** Xi-Jun Chen **Production Editor:** Xi-Jun Chen

Abstract

Aim: Android is one of the most popular platforms in the market. This popularity has led the operating system to be a potential tool for criminal activities. Law enforcement has noted this development and started incorporating smartphone evidence into their cases. However, digital evidence is susceptible to data modification, and thus anti-forensic techniques have been developed to counter forensic investigations. This research investigates the possibility of generating false data using automation techniques.

Methods: A rooted Android device was acquired. The device screen coordinates were mapped using screenshots and Gimp. The coordinates were used to develop a Python script to automate common user tasks such as making a phone call, sending a text message, or adding a contact. These tasks were performed manually and using the automation script. A system image was acquired of the device before and after data population. The images were analyzed using Autopsy and Cellebrite's Inspector. The forensic artifacts retrieved were compared between the manual and automatic data population.

Results: The artifacts show that the data was added successfully and that forensic tools may not detect that the data was automatically generated.

Conclusion: This research shows that it is possible to populate an Android device with false forensic artifacts using automation scripts. Being able to generate forensic artifacts using automation scripts can allow educators to more easily generate datasets to teach forensic techniques. Additionally, it can also be used by malicious actors to generate false forensic artifacts to mislead an investigator. Future work could improve the proposed data generation technique via machine learning to prevent hardcoding the screen coordinates in the automation script or improve the technique to generate data with old timestamps. Another avenue of future work includes the

development of techniques to identify false data.

Keywords: Anti-forensics, Android, data generation, android antiforensics, mobile data generation

INTRODUCTION

There are many different smartphone operating systems used to access social media platforms, but Android is the dominant operating system in the market[1-5]. Statista[1] reports that Android's worldwide mobile market share is 71.93% as of February 2021. Similar to other popular operating systems, Android collects user data to improve its services[6,7]. Active data collection encompasses data collected when the user is actively interacting with Google's application and websites such as Chrome, PlayStore, Maps, Images, Drive, Sheets, and other applications[8]. Whereas passive data collection includes data acquired from Android devices without actively interacting with Google's products, such as usage information, device configuration, crash reports, backups, and other types of data[8,9].

Law enforcement and prosecutors have taken advantage of smartphone data collection practices to aid their investigations[10,11]. Google's location data has been used by state law enforcement to aid their investigation of murder and potential arson cases[12]. The federal government has also taken an interest in Google's location data as the FBI has used location data to aid in their investigations on several occasions[13]. Since Google's location data is more accurate than the traditional cell tower triangulation, reverse search warrants are becoming an increasingly common practice for law enforcement agencies[14]. More recently, Google's location data has been used to help law enforcement identify perpetrators in the capitol riot on January 6, 2021[15]. Criminals have started developing and employing anti-forensic techniques as a response against law enforcement usage of digital evidence[16-18]. Encrypted devices have been used by criminals to aid in drug trafficking[16,18]. Encryption has also been involved in murder cases, illegal firearm sales, and torture[16,19,20]. Steganography is used by malware developers to hide malicious software[17,21,22]. The anti-forensic technique is also used by spies to covertly share information[23].

Academicians and practitioners have taken an interest in the field of anti-forensics to assist law enforcement in their investigations. Researchers have focused on the development of new anti-forensic techniques, potential countermeasures, and on the investigation of existing anti-forensic methods[24-27]. Android anti-forensics is an area of ongoing research which has resulted in the development of new anti-forensics techniques[24,25]. However, limited research exists on data counterfeit generation on Android applications. This scenario motivates the hypothesis that it is possible to produce false evidence on Android applications. The following research inquiries are proposed to investigate this hypothesis:

• Can you populate an Android device with false information?

• Are there forensic artifacts left on the device that could be used to identify that the data was generated using the proposed data generation technique?

The research contributions of this paper are two-fold. First, this research investigates the generation of false Android application data as an anti-forensic technique. Second, it establishes a foundation to conduct future investigations on the topic. The paper is structured as follows: the next section discusses existing Android anti-forensics research. Section three presents the methodology and experimental design. Section four discusses the implementation and results. Section five discusses the research conclusions and presents future work.

As a response to criminal usage of anti-forensic techniques, researchers and practitioners have investigated this novel field[24-27]. Research on this topic diverges into different concentrations. Some researchers construct novel anti-forensic techniques[24-26,28], while others postulate solutions to existing methods[27] and develop comprehensive taxonomies[29].

Research on data generation focuses mostly on Windows and Android platforms[24,30-37]. Windows data generation involves the usage of virtual machines to generate false data[36] or forensic artifact generation via data injection[37]. However, these techniques only work on Windows environments, these techniques fail to generate older data, and they may leave forensic artifacts that could be used to detect them[36-37]. On the other hand, Android data generation focuses on the usage of Android terminal tools such as the "sendevent" and "getevent" commands[24] or relies on other tools such as SQLite or Android's "touch" and "date" commands[24,35]. However, these techniques may generate forensic artifacts that could be used to identify them; they are limited and may not work on novel Android devices.

Research on Android anti-forensics focuses on hindering forensic processes or compromising the data's veracity[26,30,31]. Literature on the topic includes the development of data modification techniques on Android devices with the objective of modifying the data partition and deleting evidence[26]. However, this technique may lead to additional forensic artifacts that could be used for identification and may not work on newer Android devices. Other techniques include the modification of the Android content providers to hamper forensic acquisition[31]. However, this technique ignores data generation and may not work on novel devices. Researchers have also examined the anti-forensics potential of application features such as WhatsApp[30] to delete forensic artifacts. However, this technique ignores data generation. Other researchers have examined anti-forensic techniques to develop detection mechanisms[27]. Existing literature has examined the possibility of detecting database manipulation on Android and iOS devices via a series of forensic artifacts[27]. However, their research focuses mostly on detection and not data generation.

Existing Android anti-forensic research concentrates on the development of anti-forensic techniques[24-26,28], the evaluation of existing techniques[32,33], and the development of a comprehensive taxonomy[29]. Research on data generation on an Android device as an anti-forensic technique is both limited and minimal[35-37]. To the best of our knowledge, there is little research on false data generation of Android applications as an anti-forensics technique.

METHODOLOGY

The study presented in this article implements a controlled experiment methodology as described by[38]. This research design employs experiments to assess a hypothesis. The hypothesis measured in this experiment is that the artificial creation of realistic Android application forensic artifacts is possible. To test this hypothesis, an Android application data population technique is developed. Artificial data generation is achieved employing Python, Android Debug Bridge, and Android's input command.

The development and testing environment are illustrated in Table 1. The population technique uses Android's input command to simulate user's swipe and touch. Different screen element coordinates are used in conjunction with Android's input utility to interact with various applications.

Figure 1 details the proposed data population technique. First, each target automation activity is performed manually, such as creating a text message or adding a phone contact. This step is performed to identify smaller tasks required to complete the activity, such as touching a text box, typing the desired message, and clicking the "Send" button. Once all smaller tasks are identified, a device's screenshot is captured at each

Table 1. Instruments used

Hardware & Software	Version
Android Debug Bridge (ADB)	v1.0.41
Python[39]	v3.9.2
Gimp[40]	v2.10.22
Autopsy[41]	v4.11.0
Inspector[42]	v10.4
Rooted Android Device	SM-G935V with Android 8.0.0
Dell Desktop Computer	Intel Xeon W-2102 CPU @ 290 GHz. 32 GB RAM. 500 GB Hard Drive. Running Windows 10 Pro
SQLite DB Browser[43]	v3.12.0

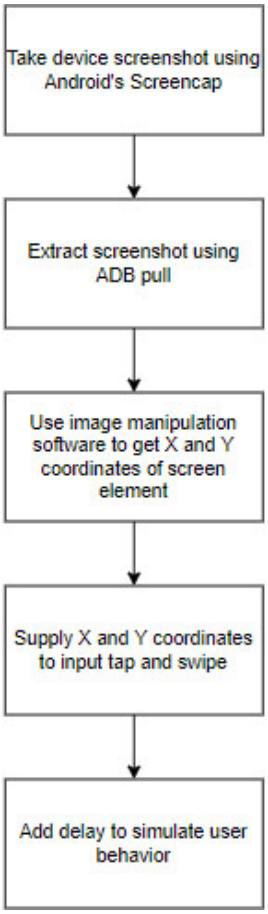

Figure 1. Android App population diagram.

task using Android Debug Bridge and Android's screencap utility[44]. This is accomplished using the command "adb shell screencap -p <download location on device>"[44]. The screenshot is extracted using ADB pull command to a remote computer. The command executed at this step is "adb pull <device location> <remote computer location>"[44]. The extracted image file is then examined using image manipulation software, namely GIMP[40], to identify the X and Y coordinates of different screen elements used by the subtask. These coordinates are supplied to Android's input utility to emulate user's screen touch. The commands used for this task are "adb shell tap <X coordinate> <Y coordinate>"[45].

Android's input swipe command is used to simulate long touch and swipe actions. This command also uses the previously mentioned X and Y coordinates. The command used to achieve this task is "adb shell input swipe <X1 coordinate> <Y1 coordinate> <X2 coordinate> <Y2 coordinate> <duration in milliseconds>"[45]. To simulate long screen taps, the X and Y coordinates are kept constant, and the duration is set to a random value between 1 and 3 s. The X coordinate is kept constant, the Y coordinate changes, and the duration is set to value between 1 and 3 s to emulate vertical screen swipes. The same process is used but with Y constant and X varied for horizontal screen swipes.

Android's input utility is also used to simulate user typing behavior. A screenshot of the different states of the keyboard is acquired. This screenshot is examined using GIMP[40] to map each key to a set of X and Y coordinates. A small random delay between 0.5 and 1 s is added to emulate human typing behavior. The aforementioned functions are compiled into a Python script for ease of execution.

A set of rooted and non-rooted emulators were configured with Android API level versions ranging from 16 to 28 to identify Android's input syntax differences between versions. Using ADB, Android's input utility was executed on each emulator to determine if the command exists on that API version. If the utility is found, the help function is executed to verify command syntax differences. The previously mentioned process is repeated for all emulators. Table 2 details the results of this evaluation. These experiments confirm that Android's input command exists on the tested Android API level for both rooted and non-rooted devices. This situation suggests that device population using Android's input command should work for a large variety of Android versions.

The automation technique is developed using a rooted Android device. The device specifications are detailed in Table 1.

A factory reset is performed on the device to delete all previous data. The device is connected to the internet. A new Google account is created for the device. The following activities are performed on the device:
1. Add a phone contact;
2. Send a text message to a phone number;
3. Call a phone number.

The aforementioned tasks were performed manually, and a system image was extracted after the manual population. The tasks were also performed using the automation script, and a system image was extracted after data population. The forensic artifacts were identified and examined using Autopsy and Cellebrite's Inspector. The forensic artifacts for each system image were compared to verify the effectiveness of the proposed data generation technique.

The technique used in this research relies on screen coordinates. This technique uses the screen layout and icons to interact with different applications and generate data. The environment used to develop and evaluate the technique is described in Table 1. The technique was evaluated on rooted Android emulators with different API versions described in Table 2. This technique was implemented on a rooted Android device running Android 8.0.0. The technique was evaluated using Autopsy and Cellebrite's Inspector. For the purposes of this research, all other Android versions and population techniques are considered out of scope.

Table 2. Android input evaluation

API level	Rooted	Syntax difference
16	Yes	Input swipe command does not contain duration
17	Yes	Input swipe and tap command have an optional [touscreen] and [touchpad] parameter. swipe command does not contain duration
18	Yes	Input swipe and tap command contain optional [touchscreen|touchpad|touchnavigation] parameter
19	Yes	Input command has optional source argument. Default source is touchscreen for both tap and swipe command
21	Yes	Same as 19 rooted
22	Yes	Same as 19 rooted
23	Yes	Same as 19 rooted
24	Yes	Same as 19 rooted
25	Yes	Same as 19 rooted
26	Yes	Same as 19 rooted
27	Yes	Same as 19 rooted
28	Yes	Same as 19 rooted
16	No	Same as 16 non-rooted
17	No	Same as 17 non-rooted
18	No	Same as 18 non-rooted
19	No	Same as 19 non-rooted
21	No	Same as 21 non-rooted
22	No	Same as 22 non-rooted
23	No	Same as 23 non-rooted
24	No	Same as 25 non-rooted
25	No	Same as 25 non-rooted
28	No	Same as 28 non-rooted

RESULTS

Forensic examination

The forensic analysis was performed using Autopsy and Cellebrite's Inspector. Autopsy required a forensic image to analyze, so the image was acquired in a separate step. First, Android Studio was installed on the forensic workstation to acquire Android developer tools to extract the forensic image from the device. Next, the device was powered on, logged in, and Android's developer mode was activated, as shown in Figure 2.

Next, the device was connected to the forensic workstation as a USB device to be able to execute commands on the device. In the forensic workstation, two command lines prompts were started and navigated to the Android developer tools folder in the Android Studio folder. In the first command prompt, the following command was executed: "adb shell". Next, using the "adb shell" the following command was executed to identify the "data" partition of the device: "mount | grep /data". The output of the command was examined to identify the partition that contains the "/data" folder. Next, the "adb shell" was exited, and the following command was executed "adb forward tcp:8888 tcp:8888". This process is shown in Figure 3.

The "Disk Dump" utility (dd) and the "netcat" tool were installed on the forensic workstation to extract the partition of the Android device. The following command were typed on the second command prompt: "nc 172.0.0.1 8888 > image.dd". This command extracts the information received in the localhost port into the "image.dd" file. In the first command prompt the following command was typed in the "adb shell": "dd if=<data partition path> | busybox nc -l -p 8888". This command extracts the content of the data partition and outputs it to the "netcat" port. The commands in both command prompts were executed at the same time to extract the information from the device, as shown in Figure 4.

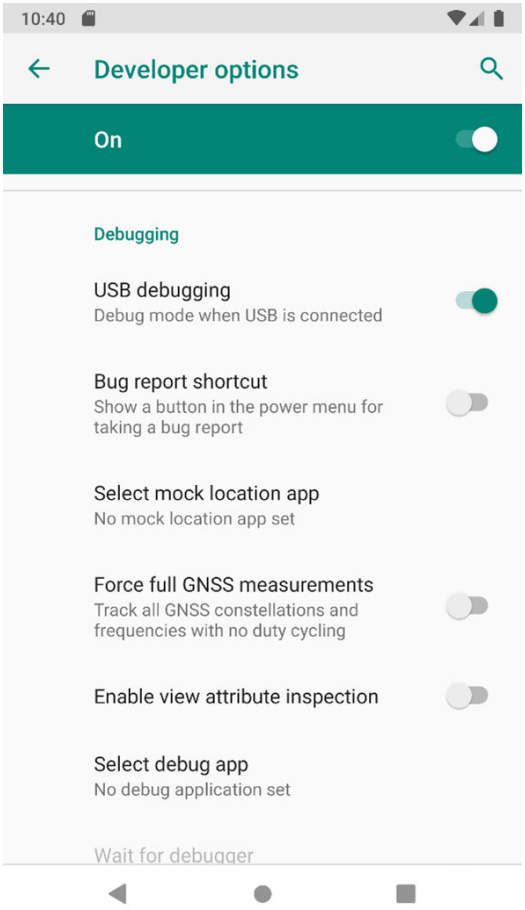

Figure 2. Android developer mode enabled.

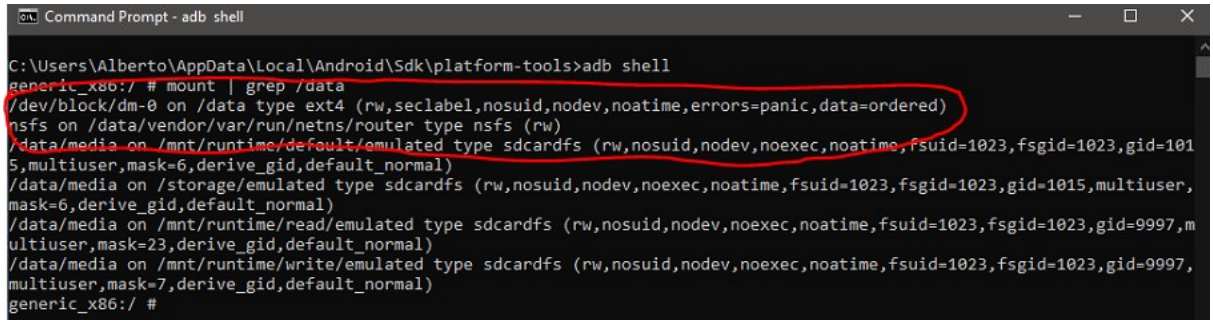

Figure 3. Data partition location on device.

This image extraction process was performed before and after the execution of the proposed data generation technique to acquire two forensic images for comparison. Both images were loaded into Autopsy by creating a new case and loading each forensic image file. The "hash calculation" module was executed on each image to compute an MD5 hash of each file. Autopsy's report feature was utilized to create a comma-separated value (CSV) file with the filenames, file paths, and file hashes to a CSV file for easier analysis. This process is summarized in Figures 5 and 6.

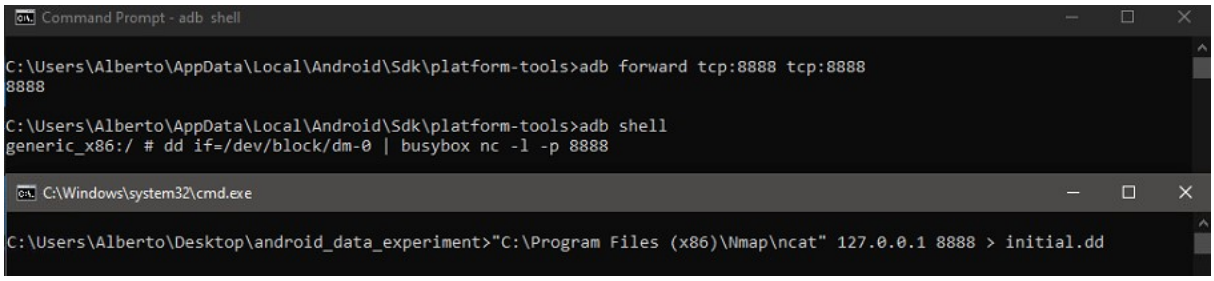

Figure 4. Data extraction process.

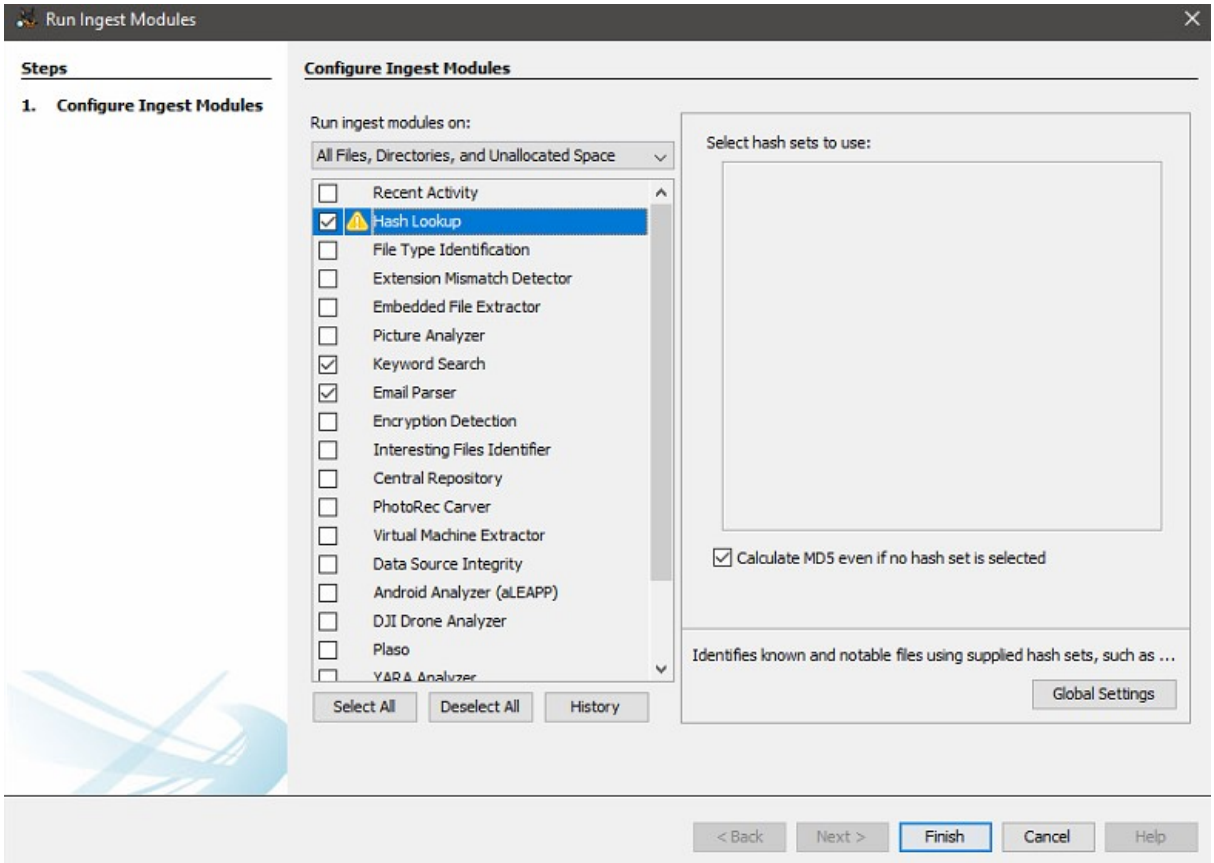

Figure 5. Autopsy modules.

The process was repeated for each forensic image. Afterward, the data in the CSV files were compared using a Python script to identify differences between both images. These differences were then further examined in Autopsy to verify their data contents and confirm their forensic relevance.

Next, Cellebrite's Inspector was started, and a new case was created. The Android device was connected, and data were acquired from the device using Inspector's data acquisition features. The forensic artifacts identified in the comparison were verified using Inspector's data analysis feature, and their forensic relevance was confirmed. The identified forensic artifacts are summarized in Table 3.

Table 3. Artifacts retrieved

File	Location
contacts2.db	/data/com.android.providers.contacts/databases
mmssms.db	/user_de/0/com.android.providers.telephony/databases
calllog.db	/data/com.android.providers.contacts/databases

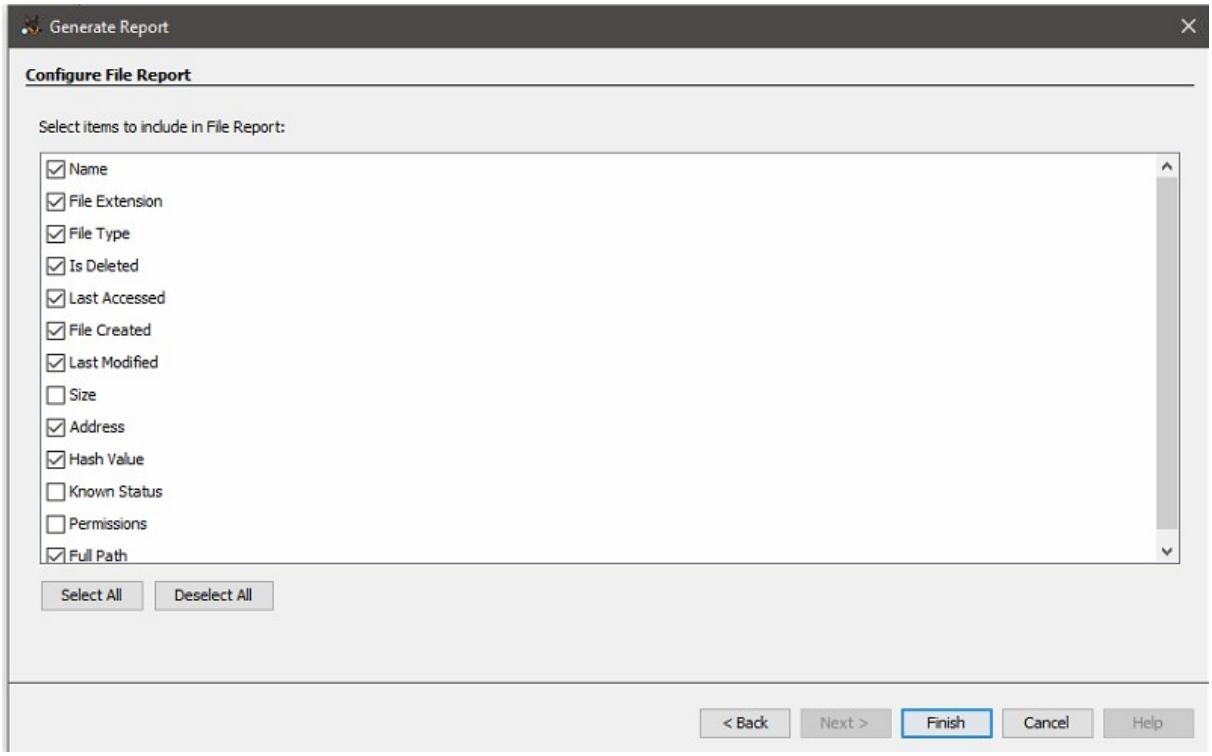

Figure 6. Report generation.

The forensic image extraction and analysis process was repeated for both manual data generation and using the technique proposed in this research. The relevant forensic artifacts were compared, and no significant difference was found.

Analysis

The "mmssms.db" is an SQlite database which contains text message data. This database shows the text messages that were sent along with their sent date, receiver, and the message body. The database shows the messages that were sent both with the automation script and manually in the "SMS" table. The "contacts2.db" is an SQlite database which contains the contacts data. The contact information is stored across different tables which include the "raw_contacts", "contacts", "data", "phone_lookup", "name_lookup", and other tables. This database shows that the contacts that were added manually and with the automation script were added successfully to the Android device. The "calllog.db" is an SQlite database which contains the call log information. The "calls" table in this database stores information about the calls, which includes its state, duration, number, and other data. This database shows that the automation script was successful in placing a call.

The results of this research suggest that an Android device can be populated with false data. The case study presented in this research using contacts, SMS, and calls shows that it is possible to populate an application with fake data using an automation technique. However, this technique is screen-specific since it relies on screen elements X and Y coordinates. The method will be impacted by layout changes that come with application updates or website updates. The technique proposed in this research does not require a device to be rooted to work correctly as Android's input and Android's screencap are both available using Android Debug Bridge. Despite its shortcomings, the technique is able to interact with a variety of Android applications. If a user is able to accurately map the X and Y coordinates of their actions in target applications, they could generate an extended amount of false data. This technique can be implemented using a scripting language like Python to insert delays to emulate human interaction with Android applications.

Regarding the second research question proposed in the introduction, the research results suggest that there are no relevant forensic artifacts that could be used to detect that the data was generated using the proposed technique. The rooted android device was populated with false data using the anti-forensic technique proposed in this research and analyzed using two popular forensic tools: Autopsy and Cellebrite's Inspector. A forensic analysis was performed based on a list of known forensic artifacts and techniques. This analysis showed that multiple desired artifacts were generated, such as database entries. The forensic investigation did not find any signs of evidence tampering or data generation that would raise suspicion towards the forensic artifact's validity.

Validation

The proposed technique was compared with existing data generation techniques for Android devices. More precisely, the proposed technique was compared with Ceballos Delgado *et al.*[35] and Albano *et al.*[24] techniques. Ceballos Delgado *et al.*[35] technique utilizes Android Debug Bridge (ADB), Android's "date", Android's "time", ADB's "push", and Android's "SQLite" utility to populate the device with false data. Differential analysis of two images acquired before and after the usage of their technique shows that the device holds forensic artifacts that could be used to verify the veracity of the data. First, the database journal file does not match the database file. There is also a substantial timestamp difference between the journal file and the database, which can indicate that the database was proctored using the technique proposed by Pieterse *et al.*[27]. Furthermore, the timestamp modification capabilities of their tool allow the user to modify the affected file timestamps to a time older than the filesystem creation time, which could be used to detect the data generation technique. Additionally, this data generation technique fails to generate events related to user activities such as event logs since it modifies the database directly. The technique proposed in this research generates said event logs since it recreates the actions performed by the user via interaction with the user interface. While this technique is unable to generate data with an older timestamp, it maintains the consistency between journal file and database files which reduces detection probability using Pieterse *et al.*[27] technique.

The data generation technique proposed by Albano *et al.*[24] uses Android's "sendevent" and "getevent" command to interact with the user interface and generate false data. Their technique works by mapping the events related to a specific activity in an application using the "getevent" command and repeating them using the "sendevent" command to recreate the activity. The result is that the device generates the events, which emulate user activities. However, their proposed technique requires a significant number of events and parameters to generate a single activity. Furthermore, their tool is limited to a small set of automation activities, and their research lacks replication details on the forensic examination which hinders replication. The technique proposed in this research can generate a diverse set of automation activities depending on the screen coordinates of the activity to replicate. This proposed technique requires less information to

recreate an emulation activity than Albano *et al.*[24] method. Lastly, this research provides more details on the forensic examination to aid experiment replication.

Other data generation techniques focus on Windows platforms such as Scanlon *et al.*[37] and Du *et al.*[36] methods. Their proposed techniques are more versatile and can generate a wide variety of false data. However, their techniques are unable to generate data on Android devices, which the technique proposed in this research can accomplish.

CONCLUSION AND FUTURE WORKS

The research presented in this document shows an investigation of Android application anti-forensics. The software developed in this research can populate an Android application with false data indistinguishable from real data. The software uses Android Debug Bridge and Android's input to simulate screen touches and interact with a target application. The X and Y coordinates used by Android's input are acquired using Android's screencap and an image editor. The proposed population technique is evaluated using contacts, call logs, and SMS as a case study. A forensic analysis was performed on a rooted Android device. A variety of general and application-specific artifacts were retrieved and analyzed. The forensic investigation was not able to detect false data generation.

The scope of this research is limited to Android devices, all other operating systems are considered out of scope. This research evaluates data generation of text messages, phone calls, and phone contacts as an anti-forensic technique, other possible techniques are considered out of scope.

A potential improvement in this research would be the automation of screen coordinate identification. Image recognition technology could be applied to detect the different elements' screen coordinates and update them automatically. Thus, addressing one of the shortcomings of the application. Additionally, this improvement would reduce the time required to automate false data generation and improve the technique's efficiency. Another possible future avenue includes the automatic detection of artificially generated data. This could be accomplished using machine learning models and event log timestamp differences. Lastly, a potential avenue for future research is the generation of false data with older timestamps.

DECLARATIONS

Authors' contributions

Made substantial contributions to conception and design of the study and performed data analysis and interpretation: Ceballos Delgado A
Provided administrative, technical, and material support: Zhou B

Availability of data and materials

Not applicable.

Financial support and sponsorship

None.

Conflicts of interest

Both authors declared that there are no conflicts of interest.

Ethical approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Copyright

REFERENCES

1. O'Dea S. Statista. Market share of mobile operating systems worldwide 2012-2021. Available from: https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/ [Last accessed on 28 Mar 2022].
2. Statcounter. Mobile Operating System Market Share Worldwide Feb 2020-Feb 2021. Available from: https://gs.statcounter.com/os-market-share/mobile/worldwide [Last accessed on 28 Mar 2022].
3. Statcounter. Operating System Market Share Worldwide Feb 2020-Feb 2021. Available from: https://gs.statcounter.com/os-market-share [Last accessed on 28 Mar 2022].
4. NetMarketShare. Operating System Market Share. Mobile 2019-11 to 2021-02. Available from: https://netmarketshare.com/operating-system-market-share.aspx?options=%7B%22filter%22%3A%7B%22%24and%22%3A%5B%7B%22deviceType%22%3A%7B%22%24in%22%3A%5B%22Mobile%22%5D%7D%7D%5D%7D%2C%22dateLabel%22%3A%22Trend%22%2C%22attributes%22%3A%22share%22%2C%22group%22%3A%22platform%22%2C%22sort%22%3A%7B%22share%22%3A-1%7D%2C%22id%22%3A%22platformsMobile%22%2C%22dateInterval%22%3A%22Monthly%22%2C%22dateStart%22%3A%222019-11%22%2C%22dateEnd%22%3A%222020-10%22%2C%22segments%22%3A%22-1000%22%7D [Last accessed on 28 Mar 2022].
5. Popal N, Reith R. IDC. Smartphone market share. Available from: https://www.idc.com/promo/smartphone-market-share [Last accessed on 28 Mar 2022].
6. Google. Data Practices & Transparency. Available from: https://safety.google/privacy/data/ [Last accessed on 28 Mar 2022].
7. Google. How Google uses information from sites or apps that use our services. Available from: https://policies.google.com/technologies/partner-sites?hl=en-US [Last accessed on 28 Mar 2022].
8. Schmidt DC. Google data collection research. Available from: https://digitalcontentnext.org/blog/2018/08/21/google-data-collection-research/ [Last accessed on 28 Mar 2022].
9. Nakashima R. AP Exclusive: Google tracks your movements, like it or not. Available from: https://apnews.com/article/north-america-science-technology-business-ap-top-news-828aefab64d4411bac257a07c1af0ecb [Last accessed on 28 Mar 2022].
10. Baker C. What would your phone tell police? Data plays growing role in investigations. Available from: https://www.sctimes.com/story/news/local/2018/08/02/cell-phone-data-plays-growing-role-st-cloud-police-investigations-digitalfootprint/759365002/ [Last accessed on 28 Mar 2022].
11. McKellop M. Why smartphone data warrants are the future of criminal investigation. Available from: https://www.theburnin.com/category/featured/page/58/ [Last accessed on 28 Mar 2022].
12. Dukes T. To find suspects, police quietly turn to Google. Available from: https://www.wral.com/Raleigh-polic e-search-google-location-history/17377435/ [Last accessed on 28 Mar 2022].
13. Brewster T. Feds Order Google To Hand Over A Load Of Innocent Americans' Locations. Available from: https://www.forbes.com/sites/thomasbrewster/2018/10/23/feds-are-ordering-google-to-hand-over-a-load-of-innocent-peoples-locations/?sh=6d1ff6995a0d [Last accessed on 28 Mar 2022].
14. Mak A. Police departments are using "reverse location search warrants" to force Google to hand over data on anyone near a crime scene. Available from: https://slate.com/technology/2019/02/reverse-location-search-warrants-google-police.html [Last accessed on 28 Mar 2022].
15. Brewster T. FBI Uses Google Location Data To Ensnare Alleged Capitol Hill Rioter. Forbes; 2021. Available from: https://www.forbes.com/sites/thomasbrewster/2021/03/10/fbi-uses-google-location-data-to-ensnare-alleged-capitol-hill-rioter/?sh=61c073c8104e [Last accessed on 28 Mar 2022].
16. Corder M. European police crack encrypted phones, arrest hundreds. Available from: https://apnews.com/article/international-news-technology-netherlands-crime-arrests-a9916c80e8493c9a26412f45919d7a96 [Last accessed on 28 Mar 2022].
17. Ang MC, Mendoza E, Yeneza J. LokiBot Gains New Persistence Mechanism, Steganography. Trend Micro; 2019. Available from: https://www.trendmicro.com/en_us/research/19/h/lokibot-gains-new-persistence-mechanism-uses-steganography-to-hide-its-tracks.html [Last accessed on 28 Mar 2022].
18. Davis K. Global investigation into drug traffickers' use of encrypted phones lands in San Diego. Available from: https://www.sandiegouniontribune.com/news/courts/story/2021-03-13/sky-global-eap-encryption-san-diego-indictment [Last accessed on 28 Mar 2022].

19. Bond S. Apple Declines DOJ Request To Unlock Pensacola Gunman's Phones. Available from: https://www.npr.org/2020/01/14/796160524/apple-declines-doj-request-to-unlock-pensacola-gunmans-phones#:~:text=Apple%20Declines%20DOJ%20Request%20To%20Unlock%20Pensacola%20Gunman's%20Phones%20Apple,government%20and%20Apple%20over%20privacy [Last accessed on 28 Mar 2022].
20. Greenblatt M, Cribb R. Encrypted evidence is increasingly hampering criminal investigations, police say. Available from: https://www.wptv.com/news/national/encrypted-evidence-is-increasingly-hampering-criminal-investigations-police-say [Last accessed on 28 Mar 2022].
21. McMillen D. Steganography: A Safe Haven for Malware. Available from: https://securityintelligence.com/steganography-a-safe-haven-for-malware/ [Last accessed on 28 Mar 2022].
22. Stone-Gross B. Malware Analysis of the Lurk Downloader. Available from: https://www.secureworks.com/research/malware-analysis-of-the-lurk-downloader [Last accessed on 28 Mar 2022].
23. Fox S. FBI: Russian spies hid codes in online photos. Available from: https://www.nbcnews.com/id/wbna38028696 [Last accessed on 28 Mar 2022].
24. Albano P, Castiglione A, Cattaneo G, De Maio G, De Santis A. On the construction of a false digital alibi on the Android OS. 2011 Third International Conference on Intelligent Networking and Collaborative Systems; 2011 Nov 30-Dec 2; Fukuoka, Japan. IEEE; 2011. p. 685-90. DOI
25. Leguesse Y, Vella M, Colombo C, Hernandez-castro J. Reducing the forensic footprint with Android accessibility attacks. In: Markantonakis K, Petrocchi M, editors. Security and trust management. Cham: Springer International Publishing; 2020. p. 22-38. DOI
26. Albano P, Castiglione A, Cattaneo G, De Santis A. A novel anti-forensics technique for the Android OS. 2011 International Conference on Broadband and Wireless Computing, Communication and Applications; 2020 Oct 26-28; Barcelona, Spain. IEEE; 2011. p. 380-85. DOI
27. Pieterse H, Olivier M, van Heerden R. Detecting manipulated smartphone data on Android and iOS devices. In: Venter H, Loock M, Coetzee M, Eloff M, Eloff J, editors. Information Security. Cham: Springer International Publishing; 2019. p. 89-103. DOI
28. Zheng J, Tan YA, Zhang X, Liang C, Zhang C, Zheng J. An anti-forensics method against memory acquiring for Android devices. 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC); 2017 Jul 21-24; Guangzhou, China. IEEE; 2017. p. 214-8. DOI
29. Conlan K, Baggili I, Breitinger F. Anti-forensics: furthering digital forensic science through a new extended, granular taxonomy. *Digital Investigation* 2016;18:S66-75. DOI
30. Mirza MM, Salamh FE, Karabiyik U. An Android case study on technical anti-forensic challenges of WhatsApp application. 2020 8th International Symposium on Digital Forensics and Security (ISDFS); 2020 Jun 1-2; Beirut, Lebanon. IEEE; 2020. p. 1-6. DOI
31. Karlsson KJ, Glisson WB. Android anti-forensics: modifying cyanogenmod. 2014 47th Hawaii International Conference on System Sciences; 2014 Jan 6-9; Waikoloa, HI, USA. IEEE; 2014. p. 4828-37. DOI
32. Azadegan S, Yu W, Liu H, Sistani M, Acharya S. Novel anti-forensics approaches for smart phones. 2012 45th Hawaii International Conference on System Sciences; 2012 Jan 4-7; Maui, HI, USA. IEEE; 2012. p. 5424-31. DOI
33. Sporea I, Aziz B, McIntyre Z. On the availability of anti-forensic tools for smartphones. *International Journal of Security* 2012;6:58-64.
34. Asbeh S, Al-Sewadi H, Hammoudeh S, Hammoudeh A. Hex symbols algorithm for anti-forensic artifacts on android devices. *IJACSA* 2016;7:4. DOI
35. Ceballos Delgado AA, Glisson WB, Grispos G, Choo KR. FADE: a forensic image generator for android device education. *WIREs Forensic Science* 2022;4:e1432. DOI
36. Du X, Hargreaves C, Sheppard J, Scanlon M. TraceGen: user activity emulation for digital forensic test image generation. *Forensic Science International: Digital Investigation* 2021;38:301133. DOI
37. Scanlon M, Du X, Lillis D. EviPlant: an efficient digital forensic challenge creation, manipulation and distribution solution. *Digital Investigation* 2017;20:S29-36. DOI
38. Oates BJ. Researching information systems and computing. Sage; 2005. Available from: https://uk.sagepub.com/en-gb/eur/researching-information-systems-and-computing/book226898 [Last accessed on 28 Mar 2022].
39. Van Rossum G, Drake Jr FL. Python reference manual. Centrum voor Wiskunde en Informatica Amsterdam; 1995. Available from: https://dl.acm.org/doi/10.5555/869369 [Last accessed on 28 Mar 2022].
40. GIMP. Available from: https://www.gimp.org/ [Last accessed on 28 Mar 2022].
41. Carrier B. Autopsy. Available from: https://www.sleuthkit.org/autopsy/ [Last accessed on 28 Mar 2022].
42. Cellebrite. Inspector. Available from: https://www.cellebrite.com/en/inspector/ [Last accessed on 28 Mar 2022].
43. Hipp R. SQLite. Available from: https://www.sqlite.org/index.html [Last accessed on 28 Mar 2022].
44. Google. Android Debug Bridge (adb). Available from: https://developer.android.com/studio/command-line/adb [Last accessed on 28 Mar 2022].
45. Lucas R. Automating Input Events on Android. Available from: https://www.rightpoint.com/rplabs/automating-input-events-abd-keyevent [Last accessed on 28 Mar 2022].