View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by International Journal on Advanced Science, Engineering and Information Technology

International Journal on Advanced Science Engineering Information Technology

Vol.10 (2020) No. 3
ISSN: 2088-5334

A Novel Android Memory Forensics for Discovering Remnant Data

Gandeva Bayu Satrya[a], Febrian Kurniawan[b]

[a] *School of Applied Science, Telkom University, Bandung, 40257, Indonesia*
E-mail:gbs@telkomuniversity.ac.id

[b] *School of Computing, Telkom University, Bandung, 40257, Indonesia*
E-mail: febrian911@gmail.com

***Abstract*—As recently updated on the vulnerability statistics shown in 2019, Android-driven smartphones, tablet PCs, and other Android devices are vulnerable, whether from internal or external threats. Most users store sensitive data like emails, photos, cloud storage access, and contact lists on Android smartphones. This information holds a growing-importance for the digital investigation process of mobile devices, e.g., internal memory or random-access memory (RAM) forensics, or external memory or read-only memory (ROM) forensics on Android smartphones. Internal memory retrieval is considered flawed and difficult by some researchers as it alters the digital evidence in an intrusive way. On the other hand, external memory retrieval also called logical acquisition that implies the image of logical storage items (e.g., files, database, directories, etc.) that locate on logical storage. This research provides a novel methodology that focuses only on internal memory forensic in a forensically sound manner. This research also contributes two algorithms, e.g., collect raw information (CRI) for parsing the raw data, and investigate raw information (IRI) for extracting the digital evidence to be more readable. This research conducted with fourteenth events to be analyzed, and each event was captured by SHA-1 as digital evidence. By using GDrive as the case study, the authors concluded that the proposed methodology could be used as guidance by forensics analyst(s), cyberlaw practitioner(s), and expert witness(es) in the court.**

***Keywords*— vulnerability; investigation; memory forensics; guidance; Android.**

I. INTRODUCTION

Based on the latest update from the global status counter, from July 2018 until July 2019, 76.03% of worldwide mobile phone users are using the Android mobile operating system, 22.04% are using iOS, and the other 1.93% are using other mobile operating systems [1]. Android is a mobile operating system developed by Google in 2005. The first version was released in September 2008 and has been being continuously upgraded since then. The current version, which is Android 9 "Pie," was released in 2019. From this statement, being Android users means that most people in the world also rely on Google account, more specifically, the Gmail account (Google term of service). Hence, much confidential information nowadays is committed to the Gmail account, including messages, chat, browsing histories, GPS data, photos, even data in the cloud storage [2]. Under the increasing number of Gmail accounts worldwide, the number of smartphone cybercrimes is also escalating. Cybercrime refers to criminal activity done by digital technology [3],[4]. This has required an enormous demand for Android forensics, and this research guides by opting for the GDrive as a study case.

Android forensics that was introduced in some studies [5], [6] had already been developed in various methodologies [7], [8]. Android forensics is the successor of digital forensics [9], [10]. The first two rules of digital forensics are handling the original with the minimum and reporting any changes as digital evidence. Android forensics for finding digital evidence can be classified into two domains, which are volatile memory forensics (RAM analysis) and non-volatile memory forensics (internal storage analysis). To the best of authors' knowledge, many researchers have discussed the non-volatile memory forensics, but investigation about the volatile memory forensics still has not been delivering a complete direction. For example, the proposed methodology, pseudocode(s) used during the examination, scenarios (study cases) applied the guidance for the forensics analysts or others.

Some studies in the last five years have provided valuable insight into the pros and cons of each methodology. A survey and analysis of extraction schemes are conducted [11]. Artefacts in ROM memory analysis were produced in Microsoft OneDrive [12]. A study fouces on improving the speed of memory analysis and the access to non-volatile memory [13]. Another study analyzed the malware behaviour from memory and verify the results with three

different sandbox types [14]. Apart from that, other approaches have presented a new methodology to analyze firmware update protocols by using fuzz testing [15]. This approach made it possible to find five unique methods of acquiring data evidence from LG Android smartphones [15], a lightweight live memory forensic framework based on hardware virtualization [16], a Volatility plugin to automate the extraction of data in virtual reality system [17].

Given those previous researches, the interest in the volatile memory forensics of GDrive is evident since there is no publicly known way of rigorous methodology with a comprehensive and acceptable manner in cyberlaw. Based on the initial research from Satrya and Shin that have done thorough non-volatile memory forensics in GDrive, this research enhances a new approach for volatile memory forensics [8]. The research was conducted using two scenarios, e.g., doing Android forensics with the investigator and the victim's smartphone being in the same location and being in different locations. In the victim's smartphone, several scenarios were conducted by using GDrive storage, i.e., signup, sign-in, signout, sharing, deleting, renaming, new folder, new file. This research used two smartphones as clients, Samsung A7 (2016) and Oppo A37F. This research proposed a novel methodology focusing on volatile memory forensics. The methodology has five phases, which are preparation, determination, acquisition, analysis, and presentation. The definition for each phase is explained in the next section.

This research proposes another approach in Android forensics, which is volatile memory forensics. To the best of authors' knowledge, there is still no published work addressing the volatile memory forensics of GDrive on the Android platform with comprehensive analysis and acceptable manner in front of a court. The original contributions of this research can be summarized as follows:

- providing a novel methodology in volatile memory forensics with GDrive as a case study,
- proposing an algorithm for parsing raw data in volatile memory to find the remnant data,
- analyzing fourteenth events that have been conducted to retrieve digital evidence,
- presenting a report of volatile memory forensics that can be used as a reference for investigators, cyberlaw practitioners, and forensics analysts.

As for the rest of this research, Section II discusses related works, research methodology, proposed framework, and the study case. Section III thoroughly explains the memory results and discussion summary. Section V summarizes the conclusions and future work of this research.

II. Material and Method

A. Literature Review

Nisioti et al. have introduced an instant messaging data recovery method from the volatile memory of Android smartphones [7]. The methodology was proved by using a case study of four experiments that provided insights into the data behaviour in memory. The experimental results showed that copious data could be retrieved from the memory, even after the device's battery was removed for a short time. Even though the authors have provided a methodology for data retrieval, but the chronologies of each step have not been addressed well.

Vella and Cilia examined the possibility of detecting insecure inter-app communications inside memory dumps [18]. The forensic analysis results revealed the possibility of doing it across the various layers of Android's architecture. Android's Binder implemented the Android inter-app communication with support from Android shared memory. Despite the framework provided by the authors, the Binder still has many limitations. The current detector is valuable, but it cannot identify specific targets of implicit intents while the presence of multiple target apps.

Yang et al. developed an automated tool, called AMD, that can acquire the entire content of the main memory from Android smartphones and smartwatches [19]. The research analyzed the firmware update protocols of the devices by reverse-engineering the Android bootloader to develop AMD. It also designed a method allowing access to the main memory data through the firmware update protocols. The results showed that AMD surmounted the usability constraints of previous main memory acquisition approaches and that the main memory data acquired from a smartphone or smartwatch could be accurately used in forensic investigations. Comparison to existing acquisition methods showed that the proposed method could acquire main memory data without a system restart, root privilege escalation, custom kernel, and screen lock bypass. To the best of authors' knowledge, that AMD still needs to be fetched and executed in firmware update mode. In the evaluation of AMD, it was also stated that AMD was still not identical by comparing it with LiME.

Ali-Gombe et al. have suggested a new memory forensics technique called DroidScraper [20]. It recovers and reconstructs in-memory runtime artefacts by relying on the design of Android's ART region-based memory allocation. DroidScraper can extract running threads, enumerate objects allocated in the heap region, and then decode objects based on their class definitions. Albeit the workflow of DroidScrapper provided by the authors, e.g., acquisition, recovery, and reconstruction, there remains some constraint on the experiments.

Feng et al. have proposed a method for data acquisition in Android application memory which is called PASM [21]. It can be applied to unprepared Android devices. PASM utilizes system-level data migration function provided by Android manufacturers to migrate and load the private application data into an intermediate device which has been pre-flashed with a custom kernel providing the function of volatile memory forensics. It makes the private application data are possible to be acquired from the volatile memory of the intermediate device. However, PASM still has a limitation, i.e., the target device supports system-level data migration.

B. Research Methodology

This research concerned with the previous researchers [5], [9] that introduced a four-phase methodology for a digital forensics investigation, i.e., identification, preservation, analysis, and presentation. Identification is for knowing what digital evidence presents, where it is stored, and how it is

stored. Preservation is for counting and justifying any kind of alteration to the data during the digital evidence examination. The analysis is for interpreting the digital data information so that it can be readable by other people. The presentation is for representing the whole processes taken during the investigation to become a final report that is legally acceptable.

Researches in recent years, particularly focusing on Android forensics methodology, have been broadening to various aspects. To the best of authors' knowledge, Table I shows the recent updates in the relevant studies about Android memory forensics. Along with a comprehensive survey methodology, this research proposed a novel Android memory forensics for finding the remnant data in Gdrive cloud storage.

C. *Android Memory Forensics*

As illustrated in Fig.1, a new methodology for Android memory forensics is proposed. The method is developed from [8] and consists of preparation, preservation, analysis, and presentation. The proposed methodology supports three algorithms, namely, optimal script to assist the dumping process, Algorithm 1 for parsing the raw data, and Algorithm 2 for extracting the digital evidence to be more readable.

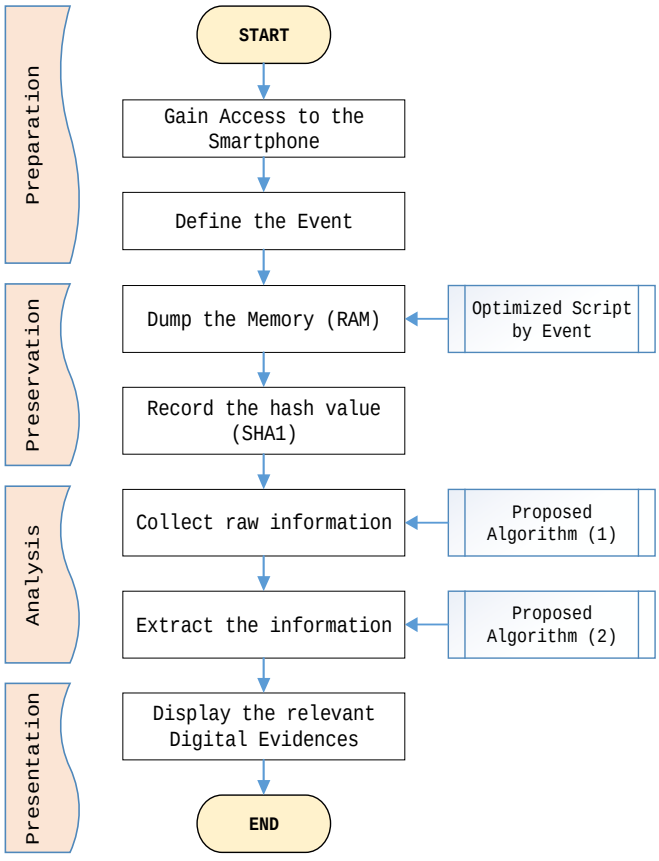

Fig. 1 Proposed Android memory forensics methodology.

The dump acquired from fry dump contains many dump files with a "`.data`" format related to the app. The files are large and still hard to read. In order to overcome this, the algorithm takes each dump file and removes the bytes in hex so the unprintable characters in ASCII or utf-8 encodings were removed, and it reduces the file size significantly. Algorithm 1 used a sequence for the hex were removed, so the structure of the data inside the dump remained intact. The object's structure was readable in ASCII encoding as it was used by the app while in the memory (space and line structures remained intact). Google drive uses JSON to store data in the memory, so the critical information must be read in the same format as it is in the memory in case JSON formatted its content is structured spaces and lines.

Algorithm 1. *Collect raw information (CRI)*

```
1:  FUNCTION Collect()
2:  f1 ← IMPORT "dump" as File
3:  r1 ← All lines on f1
4:  t1 ← Empty string
5:  f2 ← Create new file in "parsed"
6:  FOR i in r1 THEN
7:          t2 ← Empty String
8:          FOR x in i THEN
9:                  Remove hex \0 to \x08 from x
10:                 Remove hex \x0C to \x1f from x
11                  Remove hex \x7f to \xff from x
12                  t2 ← t2 + x
13          ENDFOR
14          Write t2 to t1
15  ENDFOR
16  Write t1 to f2
17  ENDFUNCTION
```

Considering plenty of files were produced, Algorithm 2 focuses on finding the right information out of many processed files from Algorithm 1 in detail. The investigator needs to specify what information needs to be found as string input, and Algorithm 2 investigated all the processed files. The processed files were listed as a memory address and even the information is duplicated on another address. However, the information were still be found by the algorithm. Despite the vast amount of information to look from, on which line the algorithm found the information were recorded as the output for investigator so the investigator recognized in detail where the information was contained in the memory address and on what line.

Algorithm 2. *Investigate raw information (IRI)*

```
1:  FUNCTION investigate()
2:  found ← False
3:  files ← Current Directory + "/parsed"
4:  INPUT: String to be find
5:  s1 ← INPUT
6:  FOR i in files DO
7:          f1 ← IMPORT i as File
8:          r1 ← All lines on i
9:          count ← 0
10:         FOR x in r1 DO
11                  count ← count + 1
12                  IF s1 in x THEN
13                          found ← True
14                          OUTPUT
15  Notification
16                  ENDIF
17          ENDFOR
18  ENDFOR
    ENDFUNCTION
```

TABLE I
COMPARISON TO RELATED STUDIES

Research Work	Memory Forensic Analyses		
	Method	**Experiment device**	**Tool & Result**
Vella and Cilia [18]	Unclear procedures about Android's Binder	System Image (Android 6.0 API level 23)	Android's Binder implemented the Android inter-app communication with support from Android shared memory
Nisioti et al. [7]	Providing a methodology for data retrieval, but the chronologies of each step have not been addressed well.	Samsung Galaxy III (GT-I9300)	By using LiME, the methodology was proved by using a case study of four experiments that provided insights into the data behavior in memory.
Yang et al. [19]	acquiring the main memory data through a firmware update command in firmware update mode without kernel replacement, without device restart, and without root privilege	SHV-E210S, SHV-E210K, SHV-E330S, SHV-E330K, SM-N900S, SM-N9005, LGF180S, LG-F180L, LG-F240S, LG-F240K, LG-E960, SM-V700.	Acquisition using AMD and analysis using LiME, a tool for acquiring main memory data from smart devices which can easily be deployed in forensic investigations
Ali-Gombe et al. [20]	Unclear procedures for conducting the DroidScrapper memory analyses.	Dalvik Virtual Machine (from Android 5.0 and beyond)	Using DroidScrapper for showing acquisition, runtime data structure recovery and object recovery and reconstruction modules.
Feng et al. [21]	Providing only the framework of proposed method (PASM) and still has a limitation.	Samsung Galaxy S7 (G9300) and Samsung Galaxy S8 (G9500)	By using PASM, makes the application private data are possible to be acquired from the volatile memory of the intermediate device.
Proposed	Preparation, determination, acquisition, analysis, and presentation	Android devices: Samsung Galaxy A7 (2016) 3GB RAM and Nexus 5 2GB RAM	By using Fridump for finding remnant data related to the client's activities focusing on Volatile Memory

D. Study Cases

Initially, there are two Gmail accounts, i.e., first account or cariduitayo12@gmail.com and second account or cariduitayo13@gmail.com. This test was carried out using three types of scenarios, namely:

- logging in using the first account on the first smartphone test,
- logging in using a second account on the first smartphone after the first account was logged out,
- logging in using the first account and second account on the first smartphone simultaneously.

During the process, every change in the scenarios or events was carried out by recording the hash value using SHA-128 and then using the factory reset. For the validity of the tests in this research, the three scenarios above were repeated on the second smartphone using the same proposed method. It was expected that identical digital evidence could be generated.

The tools used in this research were: Python 3.7.3 to perform scripting and analyzing, Frida-server 12.6.5 should be switched on to gain access to Android memory level. Fridump v0.1 to perform dumping of the memory contents. Nano 4.2 to do a quick examination of the raw data. Magisk manager v18.1 to gain rooted access. This research opted for Google Drive v2.19.192.05.35 as the study case. By following the rules of digital forensics practice, the two devices used as experimental objects in this research were put through a rooting process. The first tests were conducted on Samsung A7-2016 SM-A710L (Android 7.0) for the remnant data that was leftover. The second test was carried out on OPPO A37f (Android 5.1.1) for the validation of the whole processes in Samsung A7-2016.

III. RESULT AND DISCUSSION

A. Memory Forensics Analysis

1) Dealing with the first account

In the first scenario, three events were carried out, namely login, logout, and logout with a restart. The first event, login, was carried out using the first account on Samsung A7-2016. Fig. 2 shows the activities of the user after logging in, with the condition in which the smartphone was still switched on. By using Algorithm 1 and 2, the digital evidence that can be obtained are user information, i.e., Gmail account, name, created date, modified data.

The moment after the first account has been logged out, the first smartphone (Samsung A7-2016) was dumped by fry dump to get information related to what happened after the account was logged out. The initial dump automatically dumped all addresses of the memory-related to it by specifying the Google Drive application's name *(com.google.android.apps.docs)* on the fry dump. The result of this phase was that the 1st account information still exists on the memory, as shown in Fig. 3.

After the logout has been carried out and the device has been restarted, the event was to examine the data inside the memory to find out whether data related to the already

logged out account were still can be traced or not. Fig. 4 showed the results of the examination, which was the information related to the account have still existed. This was done to prove either the memory volatility does matter or not in this case or application (e.g., Gdrive). It turned out that the memory was still stored with information from the previously loaded data. Following the preservation phase, the hash values (see Appendix A) were taken from the three events.

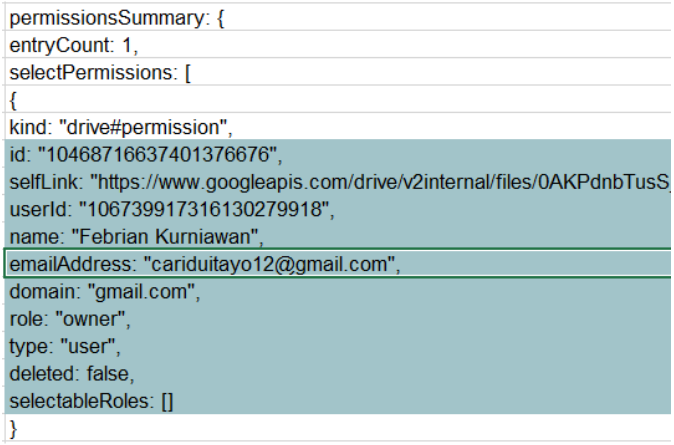

Fig. 2 Digital evidence for first account login activity.

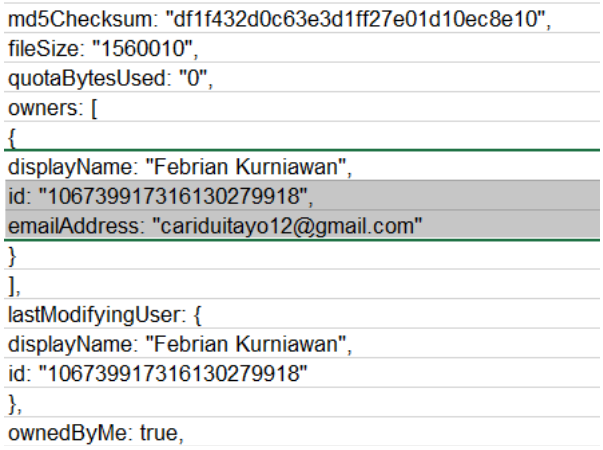

Fig. 3 Digital evidence for first account logout activity.

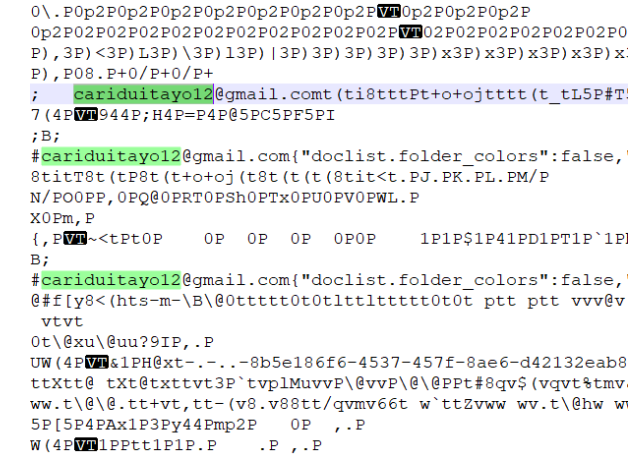

Fig. 4 Digital evidence for first account logout activity in 0xcb700000 memory address.

2) Dealing with file operations

For the second scenario, six events were carried out, namely file upload, file download, file share, file share (revoke), new file, delete the file. The hash values were taken per event and attached in Appendix A. Fig. 4 shows a file-sharing event where the first account shares files to the second account. Based on the digital evidence, both accounts can be seen along with the shared files. In this scenario, we discussed two file operations as a representative, and further complete analyses are provided in Table II that is presented in the next section.

Fig. 5 Digital evidence for first account uploading activity

The first account has uploaded a file named `forensic_evidence.txt`. By using Algorithm 2, it helpedthe investigator to find the uploaded file, created file, modified date, and last viewed by expressing the input related to the targeted information. From the in-depth analysis, digital evidence related to the uploaded file were remained in the smartphone (see Fig. 5). The other digital evidence for that uploaded file that was in the directory or the link can also be used as references in the investigation.

Initiating this event, the "`Sample Sheet`" file had been created in the GDrive of the first account. This event addressed the deleted file in the Gdrive account. As can be seen in Fig. 6, it showed that information, e.g., title, version, created date, modified date, last viewed can still be acquired from deleted files. Other important evidence related to file location was alternate link and default open link that also can be referred to as digital evidence. However, recovering the deleted file still needs to be investigated in further research.

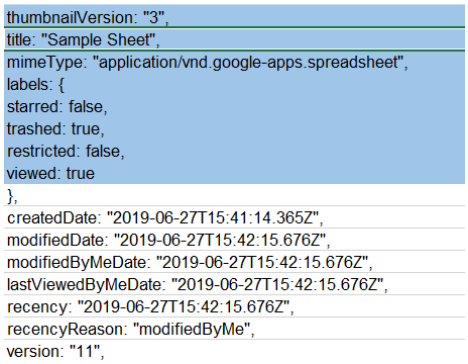

Fig. 6 Digital evidence for first account deleted file activity

TABLE II
EXPERIMENT RESULTS FOR FINDING REMNANT DATA

Event	Found	Result	Information Found
After Login (first account)	Yes	Found Details of the Account	First Logged in of first account
After Logout (first account)	Yes	Details of the Account still can be found	Logged out from first account
After Logout then Reboot	Yes	Found Email address (no details found)	Reboot after Logging out from first account (then dump)
Create File (Sheet)	Yes	Found File details	Created a Sheet file (named "Sample Sheet") and Dump
Update File	Yes	Second account listed as the Last Modifier	File (Secondary Sheet) updated by second account on another device, and dump
Delete File	Yes	Deleted file (Sample Sheet) Information still exists along with the details	Deleted a file (Sample Sheet) and dump
File Upload	Yes	Uploaded file details found with Date Information and Uploader Information	Uploaded a file (forensic_evidence.txt) to first account
File Download	Yes	The downloaded files information still exists along with its download path on the phone	Downloaded a file and dump
File Share (first account)	Yes	Permission Details found, Share Date found, User and Uploader Information found	First account shared a file (forensic_evidence-1.txt) with second account
File Share Revoked (second account)	Yes	No readable Permission Details Information, Account and file info found	First account revoked access file of second account
File Share Revoked (Reboot)	Yes	There is no readable Permission Details Information, Account and file info found	First account revoked access file of second account then reboot
Logout after Upload	Yes	The information of Files and Account still exist	Logged out from first account after uploading a file (forensic_evidence.txt)
Switch After Upload	Yes	The first account and its file information exist	- Logged in with multiple accounts (first account and second account) - Uploaded a file to first account (forensic_evidence.txt) - Switched to second account, and dump
Switch After Upload (Reboot)	Yes	The first account and its files information exist but scrambled	- Logged in with Multiple Accounts (first account and second account) - Uploaded a file to first account (forensic_evidence.txt) - Switched to second account -Reboot -Opened Gdrive (Gdrive would open last switched account by default)

3) Dealing with multiple accounts

In the third scenario, the first account and second account simultaneously logged in on the first smartphone. Firstly, user-a logged in without logging out, then user-b also logged in without logging out. Secondly, there were two events in this scenario, i.e. files sharing from first account to second account without rebooting and with rebooting. Because the Gdrive supports logging in using multiple accounts, this test was conducted to find out the digital evidence left behind when that happens.

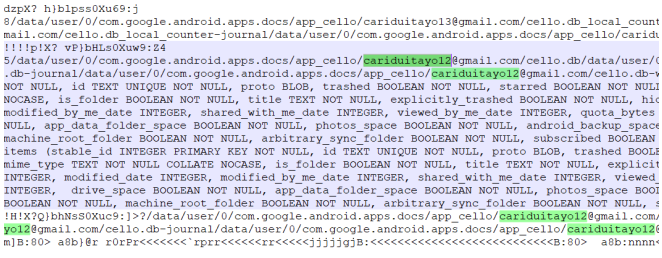

Fig. 7 Digital evidence for the first account and the second account is logging in at the same smartphone.

In both events for this scenario, the two accounts were logged in in the same smartphone by using two events, i.e., multiple accounts without restart and multiple accounts with a restart. By continuing the previous scenario (uploaded file), a sample file has been given as verification that the file was uploaded by the first account. After some time, the Gdrive second account was logging in without logging out the first account. This scenario gave two kinds of results, i.e., multiple accounts without a restart (as shown in Fig. 7), and multiple accounts with a restart (as depicted in Fig. 8). From those two events, the investigator can still extract prominent digital evidence.

Fig. 8 Digital evidence for the first account, and the second account is logging in at the same smartphone after the restart.

B. Discussion for Presentation

The memory data structures acquired in this research were all resumed based on the events that have been conducted as

can be seen in Table II. This played an important role in locating the proper information and results. For instance, translating raw data memory from bytes to ASCII (American Standard Code for Information Interchange) that contain critical information for the investigator yielded an enormous amount of potential memory addresses. The proposed methodology can be used as guidance for the investigator to find the digital evidence related to Gdrive cloud storage.

To the best of authors' knowledge, technological limitation of fry dump might not meet all the expectations of the investigator related to Gdrive cloud storage, e.g., recovery of deleted files, password of the users, or some missing after rebooted. On the other hand, it is also possible to support the anti-forensics because of the incomplete information retrieved from the memory data structure. Even though this research has provided fourteen events (as digital evidence), many further features or incoming versions of Gdrive cloud storage applications need to be evaluated.

IV. CONCLUSIONS

Based on the three scenarios consist of fourteenth events that have been conducted, a novel methodology for Android memory forensics has been proposed, and the conclusion of this research was drawn. This paper presented two algorithms for supporting the analysis phase. Those algorithms were justified as guidance during the investigation processes. Furthermore, the experiment results have provided essential digital evidence to support the investigator and to provide knowledge for cyberlaw practitioner(s) about Android memory forensic, especially on the cloud storage client. Consequently, it is important in the future to develop Android memory forensics methodology by using the network connection with the victim's smartphone being a different place from where the investigator(s) is.

ACKNOWLEDGMENT

This research was a collaboration between School of Applied Science and School of Computing, Telkom University. This research was also funded by PPM, Telkom University.

REFERENCES

[1] Statcounter. (2019) Mobile Operating System Market Share Worldwide. [Online]. Available: http://gs.statcounter.com/os-market-share/mobile/worldwide.

[2] Google Drive. (2019) Google Drive Terms of Service. [Online]. Available: https://www.google.com/drive/terms-of-service/.

[3] Holt, Thomas J., Adam M. Bossler, and Kathryn C. Seigfried-Spellar. Cybercrime and digital forensics: An introduction. Routledge, 2017.

[4] Caviglione, Luca, Steffen Wendzel, and Wojciech Mazurczyk. "The future of digital forensics: Challenges and the road ahead," IEEE Security & Privacy, vol. 15, issue 6, pp. 12-17, 2017.

[5] Ogazi-Onyemaechi, Bernard Chukwuemeka, Ali Dehghantanha, and K-KR Choo. "Performance of android forensics data recovery tools," Contemporary Digital Forensic Investigations of Cloud and Mobile Applications, Syngress, pp. 91-110, 2017.

[6] Lin, Xiaodong. "Android Forensics." Introductory Computer Forensics. Springer, Cham, pp. 335-371, 2018.

[7] Nisioti, Antonia, et al. "You can run but you cannot hide from memory: Extracting IM evidence of Android apps," 2017 IEEE Symposium on Computers and Communications (ISCC). IEEE, 2017.

[8] Satrya, Gandeva Bayu, and Soo Young Shin. "Proposed Method for Mobile Forensics Investigation Analysis of Remnant Data on Google Drive Client," Journal of Internet Technology, vol. 19, issue 6, pp. 1741-1751, 2018.

[9] McKemmish, Rodney. What is forensic computing? Canberra: Australian Institute of Criminology, 1999.

[10] Årnes, André, ed. Digital forensics. John Wiley & Sons, 2017.

[11] Scrivens, Nathan, and Xiaodong Lin. "Android digital forensics: data, extraction and analysis." Proceedings of the ACM Turing 50th Celebration Conference, China, 2017, pp. 1-10.

[12] Gandeva Bayu Satrya, A. Ahmad Nasrullah, and Soo Young Shin. "Identifying artefact on Microsoft OneDrive client to support Android forensics", International Journal of Electronic Security and Digital Forensics, vol 9, issue 3, 269-291, 2017.

[13] Sylve, Joseph T. "Towards real-time volatile memory forensics: frameworks, methods, and analysis." Dissertation Thesis. University of New Orleans, 2017.

[14] C. Tien, J. Liao, S. Chang and S. Kuo, "Memory forensics using virtual machine introspection for Malware analysis," 2017 IEEE Conference on Dependable and Secure Computing, Taipei, 2017, pp. 518-519.

[15] Park, Juhyun, Yun-Hwan Jang, and Yongsu Park. "New flash memory acquisition methods based on firmware update protocols for LG Android smartphones," Digital Investigation, vol. 25, pp. 42-54, 2018.

[16] Cheng, Yingxin, et al. "A lightweight live memory forensic approach based on hardware virtualization," Information Sciences, vol. 379, pp. 23-41, 2017.

[17] Casey, Peter, et al. "Inception: Virtual Space in Memory Space in Real Space–Memory Forensics of Immersive Virtual Reality with the HTC Vive," Digital Investigation, vol. 29, pp. S13-S21, 2019.

[18] Vella, Mark, and Rachel Cilia. "Memory Forensics of Insecure Android Inter-app Communications." ICISSP, Porto, 2017, pp.481-486.

[19] Yang, Seung Jei, et al. "Live acquisition of main memory data from Android smartphones and smartwatches," Digital Investigation, vol. 23, pp. 50-62, 2017.

[20] Ali-Gombe, Aisha, et al. "DroidScraper: A Tool for Android In-Memory Object Recovery and Reconstruction." 22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID). Beijing, 2019, pp. 547-559.

[21] P. Feng, Q. Li, P. Zhang and Z. Chen, "Private Data Acquisition Method Based on System-Level Data Migration and Volatile Memory Forensics for Android Applications," in IEEE Access, vol. 7, pp. 16695-16703, 2019.

APPENDIX A
DETAILS OF ACQUISITION DUMPING MEMORY HASH VALUES

Scenario	Event	Acquisitioned File Name	SHA-1 Hash Value (128 bits)
1 (Single Account) Primary: First account	After Login (First account)	0xb7e00000.data	151801bdbc658feeee64ac539ffda39df4d3c0dd
	After Logout (First account)	0xb7c80000.data	0f8dc18cd30195cbc1b8188e05c16aa5c74df363
	After Logout then Reboot	0xcb700000. data	4a72b21210f2e4cf827e1daabe697e09b21a893b
2 (File Operation) Primary: First account Secondary: Second account	Create File (Sheet)	0x12d21000. data	4023e24c92af78979b517b2c27d409b5d74d4790
	Update File	0xc0e00000. data	26aa57d01b7ac628a3d706f1f6a5448b9e353cee
	Delete File	0xd0100000. data	4b124b393e775d21bcb1ee273837567736f9c6b3
	File Upload	0x12d28000. data	4375fcfcb886ee56f2aeb6332f200647a31a940c
	File Download	0xc1e00000. data	83e7beed4ef142022c61f58219a4bf35fee51e9b
	File Share (First account)	0xc2000000. data	3ad5b75722a776333c7a46eb680ef3665053bc66
	File Share Revoked (Second account)	0xc1780000. data	f337dcbb68481a873876fa4dcb3143518a72a14d
	File Share Revoked (Reboot)	0xca080000. data	64e5f346c55ac871e262cb66d773929056202870
	Logout after Upload	0xc9c80000. data	bc7d5c5229d70f7ebbefb685e4d76838c803ac99
3 (Multiple Account) Both users are logging in	Switch After Upload	0xc8b00000. data	2f75ccfc9a944c2738793ed6ee0dc5be253dded7
	Switch After Upload (Reboot)	0xc6b00000. data	4ab5b1ac40d83dd5cf623fdcf5f29667746177ef