Smartphone Memory Forensics

David Andersen
Gjøvik University College
Email: 140409@hig.no

Tom Roar Furunes
Gjøvik University College
Email: 140467@hig.no

Geir Haugen
Gjøvik University College
Email: 140464@hig.no

Torbjørn Jørgensen
Gjøvik University College
Email: 140459@hig.no

Kevin Mikkelsen
Gjøvik University College
Email: 100889@hig.no

Sindre Smistad
Gjøvik University College
Email: 140468@hig.no

***Abstract*—We will emulate a Android environment, we choose this approach instead of extracting the memory from a real device to reduce the complexity of the task, and literature shows that there is not a big difference between the dumped memory from a real device, and a emulator.**

We will analyse the dumped memory, and focus our efforts on analysing the memory of the Dalvik JVM. We hope to develop a tool that can be used to analyse the different userland applications, such as mail/sms/cloud.

- **Use emulator (Android SDK), pmemsave to get memory.**
- **Look into Dalvik, how can this be analyzed? Tool?**
- **x86 vs ARM emulator.**

I. Introduction

¡¡¡¡¡¡¡ HEAD The research for this project is from selected research papers, [paper1] has a thorough explanation on how to aquire memory from android devices using a kernel module (lime.ko) that satisfies forensic soundnes [kilde til paper som viser hvor like de er]. Other research papers [liste dem opp, med kilde] show other means of acquiring memory from live systems, one paper [link til frost, ref] got our attention since it shows how it is possible to acquire memory from a smartphone forcing coldboot by cooling the smartphone down in a freezer. However this solution has a high risk of memory corruption [bilde
figur fra paperet som viser hvor mye man mistet over n tid, frost paperet afaik].

A. Preparations

When we started this project, the first step was to research papers in this subject. We started of searching on google to find tools and other information that could be important for our project. To find related research papers, we used the google scholar search engine. When everyone in our group had read and understood the papers we started to investigate how to capture the memory from the smartphone. After inquiring our teacher and other resources available at the university, the most feasible solution would be to use an emulator since none of us had any phones that would support the frost[ref til frost paper rec] method or lime kernel module [ref til lime kernel] without kernel sources for our phones. The warranty on our phones would be voided if we were to "root" them and since it is recommended to remove all content on the phone when flashing it, we choose to use an emulator.

B. Building the emulator environment

After we had chosen to go for the emulator route, we had to choose how we could get memory from the phone. After doing some reasearch [ref til pmemdum og lime], pmemdump seemed like the easiest solution since the Android SDK has a virtual machine running on qemu [ref til der vi så det med monitor og qemu]. However we had a few "bumps in the road" where the monitor flag did not give the expected output and we had to research alternative methods. In our research the paper on lime [ref til lime] had good results in forensic use where the memory dumps where close to identical (99%ish?, må få eksakt fra paper + ref) and could be used in a forensic setting (rart skrevet..). The first step to create the android emulator using the lime kernel module was to compile a custom kernel. As a prerequisite of this, we had to download the android SDK, the android NDK and the android source code. After the prerequisite for the project where complete, we followed the wiki on volatility [link, ref] and loaded the custom kernel into a Android Virtual Device. During this process we had a number of issues, the guide on volatility's google codes page was followed first, however there was some confusion on what we had to follow from googles own guidelines and what volatility required.

C. Memory Analysis

This chapter explains how we analyzed the memory of the android device that was emulated. We used multiple tools, where most are open source. ======= In computer forensics, memory/RAM have been common source of digital evidence. Modern smartphones also use memory for running OS, application and user data which might be of value in forensics investigations. Especially when encryption keys are required to decrypt critical digital evidence and when passwords for web and cloud services are needed to access more sources of evidence. How to preserve digital evidence when analysing smartphone memory/RAM?

Expected output: One or more out of these:

- A framework for, or overview of, forensics methods and techniques (based on existing tools and methods) to conduct forensically sound analysis of smartphone memory

- A tool/toolkit for gathering digital evidence from smartphone memory
- Proposed method for how to extract smartphone memory as evidence data in a forensically sound manner.

D. Background

The research for this project is from selected research papers, [paper1] has a thorough explanation on how to aquire memory from android devices using a kernel module (lime.ko) that satisfies forensic soundnes [kilde til paper som viser hvor like de er]. Other research papers [liste dem opp, med kilde] show other means of acquiring memory from live systems, one paper [link til frost, ref] got our attention since it shows how it is possible to acquire memory from a smartphone forcing coldboot by cooling the smartphone down in a freezer. However this solution has a high risk of memory corruption [bilde/figur fra paperet som viser hvor mye man mistet over n tid, frost paperet afaik].

E. Preparations

When we started this project, the first step was to research papers in this subject. We started of searching on google to find tools and other information that could be important for our project. To find related research papers, we used the google scholar search engine. When everyone in our group had read and understood the papers we started to investigate how to capture the memory from the smartphone. After inquiring our teacher and other resources available at the university, the most feasible solution would be to use an emulator since none of us had any phones that would support the frost[ref til frost paper/rec] method or lime kernel module [ref til lime kernel] without kernel sources for our phones. The warranty on our phones would be voided if we were to "root" them and since it is recommended to remove all content on the phone when flashing it, we choose to use an emulator.

F. Building the emulator environment

After we had chosen to go for the emulator route, we had to choose how we could get memory from the phone. After doing some reasearch [ref til pmemdum og lime], pmemdump seemed like the easiest solution since the Android SDK has a virtual machine running on qemu [ref til der vi så det med monitor og qemu]. However we had a few "bumps in the road" where the monitor flag did not give the expected output and we had to research alternative methods. In our research the paper on lime [ref til lime] had good results in forensic use where the memory dumps where close to identical (99%ish?, må få eksakt fra paper + ref) and could be used in a forensic setting (rart skrevet..). The first step to create the android emulator using the lime kernel module was to compile a custom kernel. As a prerequisite of this, we had to download the android SDK, the android NDK and the android source code. After the prerequisite for the project where complete, we followed the wiki on volatility [link, ref] and loaded the custom kernel into a Android Virtual Device. During this process we had a number of issues, the guide on volatility's google codes page was followed first, however there was some confusion on what we had to follow from googles own guidelines and what volatility required.

¿¿¿¿¿¿ 2ec2bff6dcbed21969b3b462e22f1a79d5f9e01e

II. Related Work

III. Method

IV. Results

V. Memory Analysis

This chapter explains how we analyzed the memory of the android device that was emulated. We used multiple tools, where most are open source.

VI. Conclusion

References

[1] J. Sylve, A. Case, L. Marziale, and G. G. Richard, "Acquisition and analysis of volatile memory from android devices," *Digital Investigation*, vol. 8, no. 3-4, pp. 175–184, Feb. 2012. [Online]. Available: http://dx.doi.org/10.1016/j.diin.2011.10.003

[2] H. Sun, K. Sun, Y. Wang, J. Jing, and S. Jajodia, "TrustDump: Reliable Memory Acquisition on Smartphones," in *Computer Security - ESORICS 2014*, ser. Lecture Notes in Computer Science, M. Kutyłowski and J. Vaidya, Eds. Springer International Publishing, 2014, vol. 8712, pp. 202–218. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-11203-9_12

[3] T. Müller and M. Spreitzenbarth, "FROST: Forensic Recovery of Scrambled Telephones," in *Proceedings of the 11th International Conference on Applied Cryptography and Network Security*, ser. ACNS'13. Berlin, Heidelberg: Springer-Verlag, 2013, pp. 373–388. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-38980-1_23

[4] L. K. Yan and H. Yin, "DroidScope: Seamlessly Reconstructing the OS and Dalvik Semantic Views for Dynamic Android Malware Analysis," in *Proceedings of the 21st USENIX Conference on Security Symposium*, ser. Security'12. Berkeley, CA, USA: USENIX Association, 2012, p. 29. [Online]. Available: http://portal.acm.org/citation.cfm?id=2362822

[5] P. Albano, A. Castiglione, G. Cattaneo, and A. De Santis, "A Novel Anti-forensics Technique for the Android OS," in *Broadband and Wireless Computing, Communication and Applications (BWCCA), 2011 International Conference on*. IEEE, Oct. 2011, pp. 380–385. [Online]. Available: http://dx.doi.org/10.1109/bwcca.2011.62

[6] V. L. L. Thing, K.-Y. Ng, and E.-C. Chang, "Live memory forensics of mobile phones," *Digital Investigation*, vol. 7, pp. S74–S82, Aug. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.diin.2010.05.010

[7] T. Vidas, C. Zhang, and N. Christin, "Toward a General Collection Methodology for Android Devices," *Digit. Investig.*, vol. 8, pp. S14–S24, Aug. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.diin.2011.05.003

[8] V. Thing and Z.-L. Chua, "Smartphone Volatile Memory Acquisition for Security Analysis and Forensics Investigation," in *Security and Privacy Protection in Information Processing Systems*, ser. IFIP Advances in Information and Communication Technology, L. Janczewski, H. Wolfe, and S. Shenoi, Eds. Springer Berlin Heidelberg, 2013, vol. 405, pp. 217–230. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-39218-4_17

[9] H. Pieterse and M. Olivier, "Smartphones as Distributed Witnesses for Digital Forensics," in *Advances in Digital Forensics X*, ser. IFIP Advances in Information and Communication Technology, G. Peterson and S. Shenoi, Eds. Springer Berlin Heidelberg, 2014, vol. 433, pp. 237–251. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-44952-3_16