# 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

## II. Related Work

## III. Volatility

Volatility is a open source collection of tools used to extract digital artifacts from memory.[1] The framework is not meant for a live forensic analysis but an offline analysis on memory aquired from a live machine or device, see Section **??** for memory aqusition. Volatility has support for Windows, Mac OS, and Linux, and from version 2.3 they also supported the ARM architecture, this is essential for analysing memory on Android devices, as most devices that run Android has a ARM architecture. With a memory dump Volatility can extract information such as running processes, network connections, mounted devices and view the mapped memory for processes and kernel modules. Volatility is especially useful when analysing a system compromised by malware, as malware often hides itself in memory to avoid detection. With Votality the artifacts of the malware can be found, and be analysed.

Volatility can also help when encryption is used on the device, if the device is powered down the keys will be lost and you might be unable to use the data collected from a less volitaile source such as a SD card or a hard drive. With Volatility the keys can be extracted from the memory (if present), Volatility comes with a plugin that can be used to extract the key used by Truecrypt for disk encryption. In our case we wanted to see if we could find the encryption key used by a popular SMS app to encrypt messages.

The easiest approach to the problem of finding data used by Android applications is to go through the Dalvik VM as proposed by Case[1]. The Dalvik VM is the *process virtual machine*[2] for Android that is used to run Android application, applications are translated to Dalvik bytecode. This makes the Dalvik VM a good entry point for analysing applications. The first application that starts and runs in a Dalvik VM is the *zygote* process. It preloades and initializes classes that are shared between different processes, it is also the process that will fork additional Dalvik VMs if requierd. The Dalvik VM is implemented as a shared library which is loaded dynamically by applications. This can be seen by the pstree and proc_maps plugins, see Appendics **??** and **??**.

Since all the user applications on Android runs in a Dalvik VM it makes a good entry point for forensic analysis, the source code is also publicly available. It is written in C++, but makes use of simple C structures, which makes it easier to locate them in memory compared to more complex structures such as classes. Case and [2] shows that the *DvmGlobals* structure which is declared in *dalvik/vm/Globals.h* contains a lot of information about the application it is running. The *loadedClasses* member is a pointer to a hash table which contains the all the system classes, this makes it a valueable structure because it can point us to where in memory information can be found.

## IV. Method

## V. Results

## VI. Memory Analysis

## VII. Conclusion

## VIII. Further Work

## References

[1] A. Case. (2011) Memory analysis of the dalvik virtual machine.
[2] H. Macht. (2013) Live memory forensics on android with volatility - dipolma thesis. [Online]. Available: http://www.homac.de/publications/Live_Memory_Forensics_on_Android_with_Volatilit

[1] https://github.com/volatilityfoundation/volatility

[2] https://en.wikipedia.org/wiki/Dalvik_%28software%29

[3] 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

[4] 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

[5] 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

[6] 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

[7] 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

[8] 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

[9] 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

[10] 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

[11] 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