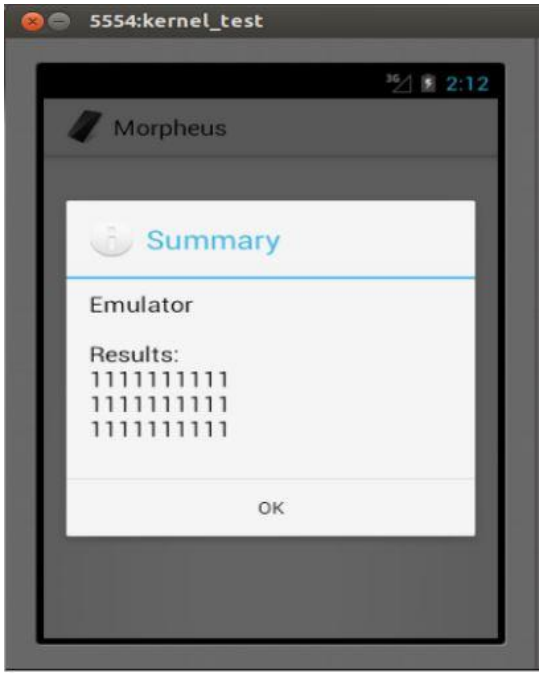

Figure 3: Android emulator: Morpheus running in Droidbox with the framework

The framework has bypassed all the file heuristics and more than 50% of the property heuristics of Morpheus, implying that the emulator can be safely categorized as a real device. Moreover our framework has bypassed the file heuristics which had the accuracy of 100% [18]. Table 5 lists the heuristics and their accuracy bypassed by the framework.

7. Limitations

The framework doesn't deal with the internal JAVA APIs as it is assumed that the dynamic analysis tools will incorporate the necessary changes in it while compiling the android emulator image. Although detection through JAVA APIs can be eliminated if the emulator is compiled from scratch with relevant changes in the API calls such as, Droidbox's getDeviceId() returns non-zero IMEI number ("357242043237511") however, if it is called via binder, it will return a "000000000000000" [18]. If the emulator image is compiled carefully taking into consideration, what a real device would return, it would definitely enhance the system. User input function is not comprehensive at this stage and will have to be updated as the application UI trend changes in future. Screen matrix will have to be re-evaluated periodically for updated probabilities. Some hardware, such as Headphones and Bluetooth are not emulated by the emulator [4]. This can be a setback if the malware is detecting the device using Bluetooth API calls. However, the emulator image can be compiled to give desirable results for particular API calls made by the application.

Table 4. Morpheus Artifacts and their accuracy, bypassed by the Framework [18]

Artifact	**Type**	**Accuracy (%)**
/proc/misc	file	98.5
/proc/ioports	file	100.0
/proc/uid stat	file	94.9
/sys/devices/virtual/misc/ cpu dma latency/uevent	file	97.1
/sys/devices/virtual/ppp	file	99.3
/sys/devices/virtual/switch	file	99.3
/sys/module/alarm/parameters	file	94.9
/sys/devices/system/cpu/ cpu0/cpufreq	file	100.0
/sys/devices/virtual/misc/ android adb	file	100.0
/proc/sys/net/ipv4/ tcp syncookies	file	94.2
ro.build.description	property	86.9
ro.build.fingerprint	property	83.9
rild.libpath	property	98.5
gsm.version.baseband	property	89.8
ro.build.tags	property	92.0
ro.build.display.id	property	99.3

Implementation of real device like hypervisor heuristics [19] has not been taken into consideration as it is a wide domain.

8. Conclusion

In this paper we have proposed a emulator anti-detection framework for QEMU-based emulators, which works with dynamic analysis environments. Our results show that the system's static attributes can be exploited to make it similar to a real device. Also the dynamic attributes can be simulated from real data-sets to make it look realistic. Adding the framework to any dynamic analysis environment will considerably improve its performance in terms of detection. However, despite the authors' effort, there are certain loopholes which still need to be fixed, such as Bluetooth emulation. This field is ever changing and will require constant modification in the framework as soon as changes are made in emulator functioning. In our future work, we plan to work on VirtualBox based emulators and develop a

mechanism for Hardware emulation, which doesn't exist in current emulators. Also, we'll work on hardware based detection heuristics to make the emulator resilient towards hypervisor based detection.

9. References

[1] Build: Class overview. http://developer.android.com/reference/android/os/Build.html, Accessed: September 2015.

[2] Display metrics.http://developer.android.com/reference/android/util/DisplayMetrics.html, Accessed: September 2015.

[3] Droidbox: An android application sandbox for dynamic analysis.https://code.google.com/p/droidbo x/, Accessed: September 2015.

[4] Emulator limitations, http://developer.android.com/tools/devices/emulator. html#starting, Accessed: September 2015.

[5] Hierarchy viewer, http://developer.android.com/tools/help/hierarchyviewer.html, Accessed: Septem-ber 2015.

[6] Kernel/goldfish, https://android.googlesource.com/kernel/goldfish/+/androidgoldfish-2.6.29, Access date: September 2015.

[7] Monkeyrunner.http://developer.android.com/tools/help/monkeyrunner concepts.html, Access date: September 2015.

[8] Network speed emulation, http://developer.android.com/tools/devices/emulator.html#netspeed, Access date: September 2015.

[9] Public gps traces.http://www.openstreetmap.org/traces, Access date: September 2015.

[10]Rotterdam open data store, http://rotterdamopendata.nl/dataset, Access date: September 2015.

[11] Ui automator.https://developer.android.com/tools/testing-supportlibrary/index.html, Access date: September 2015.

[12]Ui/application exerciser monkey, http://developer.android.com/tools/help/monkey.html, Access date: September 2015.

[13] Uiselector. Android Git Repository Access date: September 2015.

[14] Wisdm: Wireless sensor data mining. http://www.cis.fordham.edu/wisdm/dataset.php, Accessed September 2015, Access date: September 2015.

[15] Maier, D., Protsenko, M., Müller, T. (2015). A game of droid and mouse: The threat of split-personality malware on android. http://dx.doi.org/10.1016/j.cose.2015.05.001.

[16] C. Ionescu. Obfuscating embedded malware on android. http://www.symantec.com/connect/blogs/obfuscating-embeddedmalware-android, Access date: September 2015.

[17] M. Jain, A. P. Singh, S. Bali, and S. Kaul. CRAWDAD data set jiit/accelerometer (v. 2012-11-03). Downloaded from http://crawdad.org/jiit/accelerometer/, Nov. 2012.

[18] Y. Jing, Z. Zhao, G.-J. Ahn, and H. Hu. Morpheus: Automatically generating heuristics to detect android emulators. In Proceedings of the 30th Annual Computer Security Applications Conference, ACSAC '14, pages 216–225, New York, NY, USA, 2014. ACM.

[19] T. Petsas, G. Voyatzis, E. Athanasopoulos, M. Polychronakis, and S. Ioannidis. Rage against the virtual machine: Hindering dynamic analysis of android malware. In Proceedings of the Seventh European Workshop on System Security, EuroSec '14, pages 5:1–5:6, New York, NY, USA, 2014. ACM.

[20] T. Vidas and N. Christin. Evading android runtime analysis via sandbox detection. In Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '14, pages 447–458, New York, NY, USA, 2014. ACM.

