# Poster: HoneyDroid - Creating a Smartphone Honeypot

Collin Mulliner, Steffen Liebergeld, and Matthias Lange
Technische Universität Berlin / Deutsche Telekom Laboratories
{collin,steffen,mlange}@sec.t-labs.tu-berlin.de

## I. Introduction

Attacks against smartphones are becoming commonplace today, especially since they are connected to the Internet at all times. Current attacks range from worms and botnets [9], to user installed Trojans [8]. New vulnerabilities in smartphones emerge fast particularly since today's smartphones are based on common software libraries such as the Linux kernel or the WebKit browser engine. In order to keep up with new attack trends we believe that it makes sense to adopt the honeypot scheme for smartphones.

*Honeypot: a computer system, built and deployed just for the goal of being attacked and compromised in order to study new attacks and to serve as an early warning system.*

In the past honeypots and honeynets (a network of honeypots) [10], [1], [5] have shown to efficiently detect scanning worms and attacks that exploit vulnerabilities.

In this work, we aim to design and build a smartphone honeypot. Our honeypot must catch attacks originating from the Internet, mobile networks as well as through malicious applications.

We started to develop *HoneyDroid*, a smartphone honeypot for the Android [7] operating system. We chose to build the honeypot using real mobile phone hardware as opposed to using the QEMU-based Android emulator. We do this to include access to the mobile network.

The main contributions of our work are:

- **First to address smartphone honeypots.** We identified the main challenges of mobile honeypots.
- **Benefits of real phone hardware.** Using real smartphone hardware has the benefit of being able to use the modem to access the mobile operator network. By using OS virtualization we achieve equal monitoring and containment capabilities compared to an emulator approach.

In the rest of this work we first present the main challenges we identified for building a smartphone honeypot. We briefly discuss related work before giving an overview of our honeypot design in the main part. In the end we briefly discuss ideas to solve the visibility issue and shortly conclude.

## II. Challenges

While investigating the possibilities we identified four major challenges.

**Monitoring:** One major feature of a honeypot is to monitor its activities. The question arises how to monitor system events when taking into account that the monitored operating system kernel might get compromised itself. We aim at a system that can monitor all events needed to recreate the exact execution leading to an successful attack (*Completeness Criterion*).

**Audit Logging:** The audit logs created by monitoring are vital for the reconstruction of an attack. Consequently they need to be stored in a way that their integrity is ensured (*Integrity Criterion*). This property is especially difficult to achieve if we consider the operating system kernel to be vulnerable as well.

**Containment:** Honeypots are made to be compromised. But the compromised honeypot must not provide the attacker with a platform for launching further attacks. We specifically want to avoid premium-number-malware to abuse the honeypot. Therefore, one has to implement a containment mechanism in addition to the monitoring and logging.

**Visibility:** The honeypot needs to be exposed in a way that makes it attractive for attackers. The minimal goal is that the attacker is somehow able to reach the honeypot. Smartphones rarely run network services, therefore, other means have to be developed to increase the visibility of the honeypot system.

## III. Related Work

ReVirt [6] facilitates intrusion detection by running the target operating system in a virtual machine, and logging all asynchronous events. ReVirt is able to replay the virtual machine execution, as thereby helps in understanding an attack. ReVirt takes attacks on the guest kernel into account, and does all monitoring and logging in the virtual machine monitor.

## IV. System Design

We designed HoneyDroid to run on real smartphone hardware with all mobile communication features such as packet-data (GPRS), voice calls, and SMS/MMS messaging enabled. Having access to the mobile phone network will increase the *visibility* of our honeypot.

Our system shown in Figure 1 is based on a micro kernel. A micro kernel provides strong isolation guarantees for its applications. The ability to run monolithic kernels as applications,

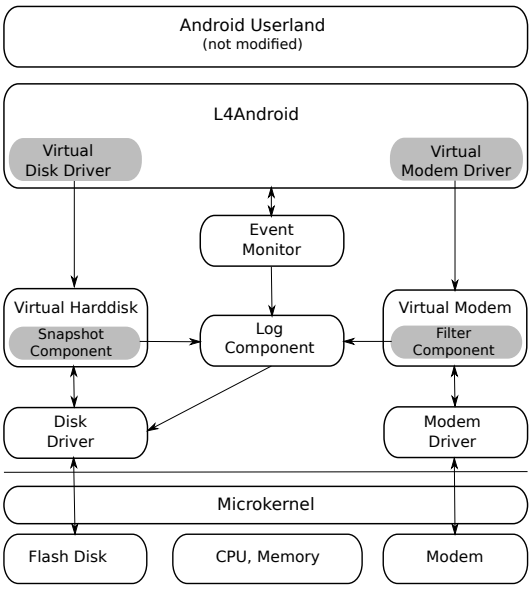


Fig. 1. The HoneyDroid architecture.

enables us to run the complete Android stack side-by-side with secure components on one device.

Our *event monitor* interposes between the Android kernel and its applications to monitor events such as system calls and signals. In HoneyDroid, Android is not allowed to access hardware directly. Instead we virtualize all relevant devices e.g. modem, wifi, and mass storage. This puts us in control of all of Android's hardware interaction, which can then be monitored. Further, it allows us to take snapshots of the file system and store them in a place not accessible to Android. Our virtual modem implements filters that mitigate any attempts of malware committing fraud and thus implements the *containment* functionality.

The log software is a separate component running directly on the micro kernel. It receives log information from different components in the system. Log messages are tagged with a time stamp to maintain their correct order. All logging is transparent to Android. This ensures that logging cannot be disabled or bypassed by an attacker. To ensure their *integrity* log files are inaccessible from Android.

With our event monitor and the virtual devices we can collect enough information to replay Android's execution to gain a better understanding of the attack.

This setup is similar to ReVirt [6]. In difference to ReVirt which is based on a monolithic kernel, we build on a micro kernel, which reduces the trusted computing base of the honeypot by orders of magnitude.

### A. Visibility

Visibility is a major issue. Implementing the best monitoring does not buy anything if the honeypot is not attacked. To increase the visibility we plan to:

- Have a public IP address for the honeypot.
- Automate installation and execution of applications.
- Automate loading of mobile websites.
- Spread name of Google account linked with the honeypot.

### B. Drawbacks of Chosen Approach

The most significant drawback is that HoneyDroid does not behave exactly the same way the original Android system does, as some virtualization overhead has to be taken into account. This might be detected by malware, which could then stop its attack and thus escape our honeypot.

We are aware that our virtualization software presents a new attack vector. The micro kernel is of low complexity, which leaves little room for serious kernel flaws. Thus, we are confident that its isolation boundaries hold, and an attack remains confined.

## V. Prototype Implementation

For this project we chose *Fiasco.OC* [4], a modern third generation micro kernel, as the basis of our architecture. It features an object capability mechanism to implement access control. Object capabilities facilitate the creation of systems implementing the principle of least authority.

We leverage *L4Android* [2] (L4Android is based on *L4Linux* [3]), a system that runs the Android kernel as an application on top of Fiasco.OC. L4Android remains binary compatible with the Android kernel and therefore runs the Android software stack without modification. L4Android allows us to run multiple instances of Android in parallel on one device. It supports different versions of the Android stack such as Froyo and Gingerbread.

## VI. Conclusions

With HoneyDroid we are going to show that a mobile honeypot is feasible. We employ virtualization to create system logs that are complete enough to replay an attack. With our architecture, we are the first to create a honeypot on a mobile device.

## References

[1] The Honeynet Project. http://www.honeynet.org/.
[2] L4Android: Android on top of L4. http://www.l4android.org, February 2011.
[3] L4Linux - Running Linux on top of L4. http://os.inf.tu-dresden.de/L4/LinuxOnL4/, January 2011.
[4] The Fiasco microkernel. http://os.inf.tu-dresden.de/fiasco/, January 2011.
[5] D. Dagon, X. Qin, G. Gu, W. Lee, J. Grizzard, J. Levine, and H. Owen. Honeystat: Local worm detection using honepots. In *in Proceedings of the 7 th International Symposium on Recent Advances in Intrusion Detection (RAID*, pages 39–58, 2004.
[6] G. W. Dunlap, S. T. King, S. Cinar, M. A. Basrai, and P. M. Chen. Revirt: enabling intrusion analysis through virtual-machine logging and replay. *SIGOPS Oper. Syst. Rev.*, 36:211–224, December 2002.
[7] Google Inc. Android. http://www.android.com/.
[8] Lookout Inc. DroidDream Malware Found in Official Android Market. http://blog.mylookout.com/2011/03/security-alert-malware-found-in-official-android-market-droiddream/, March 2011.
[9] P.A. Porras, H. Saidi, V. Yegneswaran. An Analysis of the iKee.B iPhone Botnet. In *Proceedings of the 2nd International ICST Conference on Security and Privacy on Mobile Information and Communications Systems (Mobisec)*, May 2010.
[10] N. Provos. A virtual honeypot framework. In *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 1–1, Berkeley, CA, USA, 2004. USENIX Association.