Poster: μBeR: A Microkernel Based Rootkit for Android Smartphones

Joana M. F. da Trindade, Cuong Pham, Nathan Dautenhahn
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
{*trindade, dautenh1, pham9*}*@illinois.edu*

1 Introduction

The growth in adoption of smartphone technology, combined with the development of high speed networking capabilities, has led to an increased reliance on these devices for storage of sensitive data. As an example, a number of enterprises and end users rely on smartphones for performing tasks such as email, scheduling, social networking, and online banking, to name a few.

Nevertheless, this expansion of smartphone activity combined with poor security support makes this type of device an easy target for attackers. A recent search on NIST's National Vulnerability Database for Blackberry and iPhone shows that Blackberry devices currently have over 24 vulnerabilities, and that the Apple iPhone has over 90 vulnerabilities. In addition, due to limiting resource constraints (e.g., limited memory and processing, low power consumption), smartphones are not built with security in mind. As a result, several smartphone models do not come with antiviruses and are subject to a number of threats, including worms, trojans, and rootkits.

Given a smartphone's accessible nature and its potential vulnerability to exploits, we turn our attention to attacks on this platform. Specifically, we focus on demonstrating the feasibility and consequences of a rootkit on a smartphone. In this project we consider a rootkit's evasive potential at staying hidden from Operating Systems (OSes) and future software detection mechanisms on ARM based smartphone devices.

Existing research in rootkits for ARM processors is scarce and limited to Cloaker [3]. The idea behind Cloaker is to run at the same level of a compromised embedded OS by exploiting hardware features of ARM. Furthermore, Cloaker is not easily detected by the host OS, because it does not modify the target OS code.

A direct drawback of this approach, however, is that it does not allow for persistence of the attack through a reboot [3] of the victim's device. The authors of Cloaker claim that their attack can still be effective without persistence because smartphones are not rebooted often, which is not the case for Android phones. First, according to an official developer FAQ, Android devices have to be rebooted everytime a new application is added [1]. Second, a Reddit forum shows that Android users often reboot their phones for a number of reasons, including (i) when they go to sleep, (ii) because they forget to recharge its battery, and (iii) to stop odd phone behavior [2]. With this kind of usage pattern, a stealthy rootkit that efficiently controls Android phones must persist on the system regardless of soft-resets.

One way to accomplish persistence of a rootkit is by having it run below the OS in a fashion similar to a Virtual Machine Monitor (VMM), namely, a Virtual Machine Based Rootkit (VMBR). Existing work in VMBR includes SubVirt [5], BluePill [6] and Vitriol [7]. These rootkits are persistently installed in the machine, and they can evade detection by tricking the guest OS into believing that it is running on top of real hardware.

Although originally proposed for x86 processors, we believe that a software virtualized approach similar to SubVirt is possible in the realm of ARM smartphones. To the extent of our knowledge, no one has conducted a study on the feasibility and security threat of a VMBR-like rootkit for ARM devices, nor has existing research proposed efficient ways to detect and remove such a rootkit.

Therefore, we have designed a low-level microkernel-based rootkit, called μBeR, that dedicatedly supports for persistent malicious services on an Android smartphone. Moreover, as a proof-of-concept, we are implementing μBeR using the OKL4 microkernel as virtualization framework, Android phone as target device, and the ARM Android and ARM11 KZM emulators.

In addition to persistence, a microkernel-based rootkit allows for full control on the targeted system, because most of the communication between guest OS and hardware can be inspected by the microkernel. With more control on the target system, μBeR will be able to effectively hide itself and evade detection, which is a key goal of any rootkit.

2 μBeR Design

2.1 OKL4 architecture

OKL4 is a microkernel-based software framework for embedded systems that has evolved from the L4 microkernel [4]. Its kernel consists of the Pistachio microkernel with three main components: high-performance IPC, scheduler and MMU. OKL4 has a small memory footprint, making it a good fit for mobile devices with resource constraints.

2.2 Keylogger Attack

In our attack we leverage OKL4's access to keyboard interrupts and memory sharing functionality to observe the desired actions performed by the victim OS. Specifically, we have chosen the interrupt handling code within OKL4 microkernel as the primary location for our keylogger to reside. There are mainly two components to our keylogger attack: interception of keystrokes, and keystroke data processing before it is read by the victim OS. To identify when a key is pressed we modify low level OKL4 (Figure 1) interruption handling code to filter out the specific interrupt that signals keypad related events. Once a keystroke is detected, it must be processed by a *semantic aware* function that is capable of interpreting the hardware provided character (e.g., to which application the pressed key belongs). In Figure 2, we outline two possible designs that we have devised for the location of this introspection functionality.

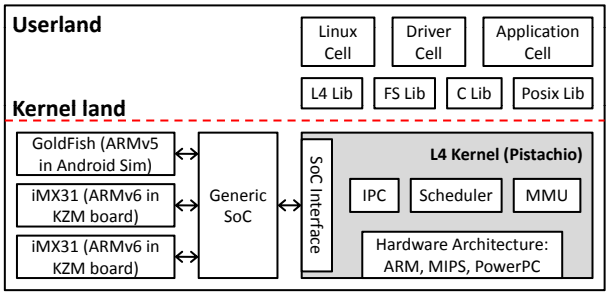

Figure 1: OKL4 architecture. A Cell, which can vary from an individual process up to a complete virtual machine (para-virtualized OS), runs on unprivileged mode.

The *semantic aware* function has two potential locations that we display in dotted boxes. The functionality can either be left in the OKL4 kernel itself, or implemented in a Cell residing in userland. Implementing the code inside of the OKL4 kernel (Attack Style 2) has the advantage of simplicity. Nevertheless, it incurs the performance overhead of processing a character in the direct execution line of interrupt handler. On the other hand, if we deploy the attack in an external dedicated Cell (Attack Style 1) we can leverage OKL4's IPC mechanism and allow for faster response times to the victim OS.

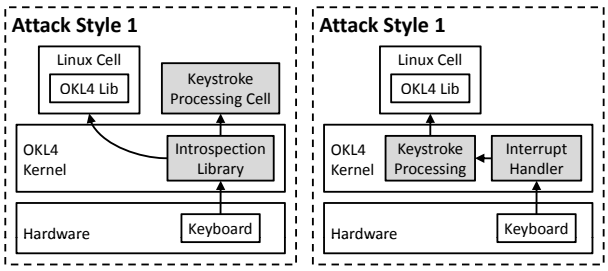

Figure 2: μBeR keylogger attack architecture. Highlighted in dotted boxes are two possible attack styles. In Attack Style 1, OKL4 forwards necessary information to a separate userland Cell through VM introspection. In Attack Style 2, keystroke data processing is performed entirely in the OKL4 microkernel.

3 Future Work

As future work, we plan to (i) complete implementation of the keylogger attack, (ii) finish design and implementation of a file exfiltration attack that will allow for stealing arbitrary files from the target smartphone (iii) conduct a set of experiments to evaluate the spatial and temporal intrusion of μBeR, which can indicate its detectability, and (iv) propose mechanisms for detecting and preventing such a rootkit in ARM-based smartphones.

References

[1] Android developer FAQ: My new application activity isn't showing up in the applications list. `http://developer.android.com/guide/appendix/faq/troubleshooting.html`.

[2] How often do you turn off or restart your android. Reddit Forum. `http://www.reddit.com/r/Android/comments/ar71i/randroid_how_often_do_you_turn_off_or_restart/`.

[3] F. M. David et al. Cloaker: Hardware supported rootkit concealment. In Proc. of the 2008 IEEE Symposium on Security and Privacy.

[4] H. Härtig et al. The performance of u-kernel-based systems. In Proc. of the 1997 16th ACM symposium on Operating systems principles.

[5] S. T. King et al. Subvirt: Implementing malware with virtual machines. In Proc. of the 2006 IEEE Symposium on Security and Privacy.

[6] J. Rutkowska. Blue pill project. `http://www.bluepillproject.org`.

[7] D. A. Zovi. Hardware virtualization rootkits. Black Hat 2006.