Virtualized In-Cloud Security Services for Mobile Devices

Jon Oberheide, Kaushik Veeraraghavan, Evan Cooke, Jason Flinn, Farnam Jahanian
Electrical Engineering and Computer Science Department
University of Michigan, Ann Arbor, MI 48109
{jonojono, kaushikv, emcooke, jflinn, farnam}@umich.edu

Abstract

Modern mobile devices continue to approach the capabilities and extensibility of standard desktop PCs. Unfortunately, these devices are also beginning to face many of the same security threats as desktops. Currently, mobile security solutions mirror the traditional desktop model in which they run detection services on the device. This approach is complex and resource intensive in both computation and power. This paper proposes a new model whereby mobile antivirus functionality is moved to an off-device network service employing multiple virtualized malware detection engines. Our argument is that it is possible to spend bandwidth resources to significantly reduce on-device CPU, memory, and power resources. We demonstrate how our in-cloud model enhances mobile security and reduces on-device software complexity, while allowing for new services such as platform-specific behavioral analysis engines. Our benchmarks on Nokia's N800 and N95 mobile devices show that our mobile agent consumes an order of magnitude less CPU and memory while also consuming less power in common scenarios compared to existing on-device antivirus software.

1 Introduction

Modern mobile devices such as the Apple iPhone and Nokia N800 run near-complete versions of commodity operating systems like BSD and Linux. Functionality like complete multi-protocol networking stacks, UI toolkits, and file systems provide developers with a rich environment to quickly build applications but open up devices to the same wide range of threats that target desktops. Over a thousand native third-party applications were developed for the iPhone platform before the official SDK was even released [11], several hundred have been developed for Nokia's Maemo platform [10], and thousands of developers are creating applications for Google's new Android platform [6].

To date, security vendors have marketed mobile-specific versions of antivirus software [8, 17, 3]. However, as the complexity of mobile platforms and threats increase, we argue that mobile antivirus solutions will look more like their desktop variants. The functionality required to detect sophisticated malware can have significant power and resource overhead – critical resources on mobile devices.

To conserve scarce mobile resources and improve detection of modern threats this paper advocates moving mobile antivirus functionality to an off-device in-cloud network service. The core of this approach is expending bandwidth to reduce on-device CPU and memory resources and thereby save power. We foresee three important benefits:

Better detection of malicious software: Once detection functionality is offloaded to a network service, significantly more resources can be dedicated to evaluating each suspicious file. Our approach uses fully virtualized detection engines running in parallel inside a network service providing mobile devices with the protection capabilities of multiple detection engines.

Reduced on-device resource consumption: By transferring files to an in-cloud network service for analysis, we argue that overall CPU use, memory use, and power can be reduced compared to performing the analysis on-device. Even more important, the network service can scale and be extended with new signatures and detection engines without using additional resources on mobile devices.

Reduced on-device software complexity: Modern threats have become extremely sophisticated, requiring complex antivirus software to detect and mitigate [12]. By deploying a relatively simple agent on mobile devices and pushing complex detection software into the network, the complexity of mobile software can be

minimized. This reduces the on-device attack surface and the effort required to port the agent to new platforms.

To explore the idea of a virtualized malware detection service for mobile devices, we extend the CloudAV platform [13] with an on-device mobile agent and an off-device mobile-specific behavioral detection engine. Through a series of benchmarks comparing CloudAV to existing on-device antivirus software, we find that our mobile agent consumes an order of magnitude less CPU and memory, consumes less power in common scenarios, and offers greater protection capabilities that scale against future threats.

2 Approach

We propose an architecture that consists of two primary components: a lightweight *host agent* that runs on mobile devices, acquires files, and sends them into the network for analysis; and a *network service* that receives files from the agent and identifies malicious or unwanted content. The proposed architecture could be deployed by a mobile service provider or third-party vendor.

This approach is an extension of the existing CloudAV platform [13]. In this section, we first provide background material on the fundamental CloudAV architecture and then discuss the extensions required to facilitate the approach in a mobile environment.

2.1 CloudAV Background

We now provide a brief overview of CloudAV [12, 13], which provides an in-cloud service for malware detection and consists of both host agent and network service components.

2.1.1 Host Agent

Just like existing antivirus software, the host agent is a lightweight process that runs on each device and inspects file activity on the system. Access to each file is trapped and diverted to a handling routine which begins by generating a unique identifier (such as a hash) of the file and comparing that identifier against a cache of previously analyzed files. If a file identifier is not present in the cache, then the file is sent to the in-cloud network service for analysis.

The threat model for the host agent is similar to that of existing software protection mechanisms such as antivirus. As with these host-based systems, if an attacker has already achieved code execution privileges, it may be possible to evade or disable the host agent. However, by reducing the complexity of the host agent by moving detection into the network, it is possible to reduce the vulnerability footprint of host software that may lead to elevated privileges or code execution.

Engine Combination	Detected	Coverage
CM	229/469	48.82%
CM, SM	290/469	61.83%
CM, SM, MA	358/469	76.33%
CM, SM, MA, BD	417/469	88.91%
CM, SM, MA, BD, FS	430/469	91.68%

Table 1: An example of the increased detection coverage against a dataset of recent month's worth of desktop malware samples when using multiple engines in parallel: ClamAV (CM), Symantec (SM), McAfee (MA), BitDefender (BD), and F-Secure (FS).

2.1.2 Network Service

The second major component of the architecture is a network service responsible for file analysis. The task of the network service is to determine whether a file is malicious or unwanted. Unlike existing antivirus software that cannot run multiple detection engines on a single device due to technical conflicts and resource constraints, moving detection capabilities to a network service allows the use of multiple antivirus engines in parallel by hosting them in virtualized containers. That is, each candidate file is analyzed by multiple detection engines to determine whether a file is malicious or unwanted. The use of virtualization allows the network service to scale to large numbers of engines and users. If demand for a particular engine increases, more instances of that container can be spun up to service analysis requests. This approach can result in significant gains in detection coverage, as illustrated in Table 1.

2.1.3 Caching

Once a file has been analyzed, the result can be stored in both a local cache on the host agent and in a shared remote cache in the network service. Subsequent accesses to that file by the device look up the result in the local cache without requiring network access. In addition, access of the same file by other devices can be mediated using a shared remote cache located in the network service, without having to send the file for analysis. Cached reports stored in the network service may also opportunistically be pushed to the agent to speed up future accesses.

2.2 Extending CloudAV to the Mobile Environment

We now describe how we extended the CloudAV platform for the mobile environment.

2.2.1 Mobile Agent

Extending the benefits of the CloudAV platform requires that an agent be deployed on a mobile platform. Given that the CloudAV platform inherently encourages a simple on-device agent, few fundamental modifications to the architecture are necessary to develop and support a mobile agent. The primary difference between the traditional host agent and our newly developed mobile agent is the constraints on resources like power and CPU cycles. Therefore, the file identifier algorithms and communications protocol with the network service are important, as the agent spends most of its cycles on those activities.

We developed a mobile agent to interface with the CloudAV network service for the Linux-based Maemo platform and deployed it on a Nokia N800 mobile device. The mobile agent is implemented in Python and uses the Dazuko [14] framework to interpose on system events. Specifically, we hook the `execve(2)` syscall and file system operations to acquire and process candidate files before permitting their access. The mobile agent required only 170 lines of code.

2.2.2 Mobile-Specific Behavioral Engine

A more resource-intensive method of detecting malicious activity is through behavioral analysis. Behavioral engines attempt to emulate or run real operating systems and applications to determine whether a file is performing malicious behavior at runtime. While these engines usually require a great deal of resources, which would not be suitable for a mobile device, deploying such an engine in the network service allows us to gain the protection benefits without the resource costs.

To demonstrate this point, we extend the CloudAV network service with a mobile-specific behavioral detection engine. The behavioral engine runs candidate applications in a virtualized Maemo operating environment hosted in the network service and monitors the application's system calls and D-Bus interprocess communication for malicious behavior. Attempts by the application to modify or destroy a user's personal data, initiate outgoing calls to unrecognized numbers via Skype, or initiate socket communications to untrusted destinations are flagged as malicious.

2.3 Additional Security Services

The security services hosted in the network service are not limited to antivirus functionality. We envision an in-cloud platform enabling a range of different security services.

- **SMS Spam Filtering:** SMS spam filtering functionality, which is currently implemented in an ad-hoc manner by some mobile antivirus products [8], can be much more accurate in a network-centric deployment model through the aggregation of data from a large corpus of users.

- **Phishing Detection:** Just as a centralized view of the web has helped Google develop strong anti-phishing tools [7], a centralized view of mobile activity in the service provider can help mobile operators detect and prevent phishing attacks against their customers.

- **Centralized Blacklists:** Blacklists of various communication addresses such as Bluetooth and IP may be implemented as an off-device security service. These blacklists can be maintained on a global level by a service provider for known malicious entities or on a personal user-specified level. These centralized policies may be opportunistically pushed to client devices for enhanced performance.

Most importantly, this architecture significantly lowers the bar for extending novel security services to mobile devices. For example, if a security vendor develops a new algorithm that is effective against detecting malicious mobile applications, that technique can be seamlessly integrated into the network service and put into operation without affecting any of the existing mobile devices. This transparent extensibility is a very powerful tool as mobile platforms and their needs are rapidly evolving.

2.4 Limitations

- **Disconnected operation:** Mobile devices may enter a disconnected state where the mobile agent may not be able to effectively utilize the network-based security services. However, mobile devices are rapidly increasing in connectivity capabilities with multiple radios for high-speed data transmission. Furthermore, given that connectivity will often be required to acquire new applications and content, the need for analysis in a disconnected state may be minimal.

- **Privacy:** The proposed architecture presents privacy implications as the organization hosting the network service may collect potentially sensitive data from various users. It is vital that users understand the privacy implications of such a service and be able to enforce limitations on what data is transmitted to the provider.

Agent	Startup Time	Average Memory	Peak Memory	User Jiffies	Total Jiffies
ClamAV	57 sec	25967 KB	39556 KB	13349	15684
MA-CL+CR	0.2 sec	1502 KB	2154 KB	1502	2185
MA-CL+WR	0.2 sec	1486 KB	2124 KB	1486	1854
MA-WL+WR	0.2 sec	1189 KB	1812 KB	1189	1714

Table 2: Comparison of the mobile agent with ClamAV in memory consumption and CPU jiffies on the Nokia N800.

3 Evaluation

For our evaluation, we perform a series of benchmarks on two Nokia mobile devices. We measure the resource and power consumption of these devices and compare our mobile agent with existing commercial antivirus products. For each experiment, we provide results for three cache states for our mobile agent (MA): CL+CR: cold local, cold remote; CL+WR: cold local, warm remote; and WL+WR: warm local, warm remote.

3.1 Computational Resources

In the first experiment, we compare the CPU and memory consumption of the ClamAV [16] engine with our mobile agent on the Nokia N800. This benchmark serially runs common applications: the built-in N800 web browser, the Skype VoIP client, the Pidgin IM client, the Kagu media player, and a PDF viewer. The application binaries and associated shared libraries, 346 files in total, are all processed by the particular engine. CPU usage is measured in both the number of jiffies the process has been scheduled for in userspace (utime) and total jiffies (utime + stime). The memory is based on the resident set size (RSS) of the process, or the number of non-shared memory pages currently in use by the process.

The results of the benchmark are listed in Table 2. ClamAV requires approximately 18 times as much memory and over 8 times as much CPU time than the worst-case cache configuration for the mobile agent. In addition, the ClamAV engine has an extremely lengthy initialization process due to its loading of its signature database.

3.2 Power Consumption

In the second experiment, we perform a micro benchmark with a Nokia N95 smartphone. We measure the power consumption required to analyze files locally with Kaspersky's Mobile Security [8] software and compare it to using the mobile agent and network service. For instances where the mobile agent needs to access the network service for cache queries or file transfers, we compare both the WiFi and GRPS/EDGE radios on the N95. The files analyzed are a collection of third-party applications and games totaling approximately 25 megabytes.

Agent	Avg / Peak / Total Energy
None (Baseline)	0.36 / 0.63 / 43.2 W
Kaspersky	0.86 / 1.27 / 89.4 W
MA-CL+CR (EDGE)	1.51 / 2.31 / 250.6 W
MA-CL+CR (WiFi)	1.31 / 2.44 / 165.1 W
MA-CL+WR (EDGE)	1.22 / 2.13 / 126.9 W
MA-CL+WR (WiFi)	0.92 / 1.83 / 74.5 W
MA-WL+WR	0.82 / 1.20 / 59.9 W

Table 3: Comparison of the mobile agent with Kaspersky Mobile Security on the Nokia N95.

Detection Engine	Signature Database Size
Symantec Mobile	27 signatures
Kaspersky Mobile	284 signatures
ClamAV	262289 signatures
Mobile Agent	$>$ 5 million sigs + behavioral

Table 4: The number of threats addressed in the signature database of various detection engines.

The results of the experiment are listed in Table 3. This experiment exemplifies the importance of the local and remote caching mechanisms. While the cold-remote cache states result in increased power consumption due to the energy of the radio transmission, a cold cache configuration is the worst case scenario which rarely occurs in practice. Both the warm-local/warm-remote and cold-local/warm-remote cache states, which are arguably the most common scenario, outperform the local Kaspersky engine in terms of consumed power. In a desktop environment, we have observed cache hits rates of over 99.8%, meaning many of the applications used are common across hosts and the transmission of full file contents across a network link is rarely necessary [13]. That being said, it is unclear whether the commonality of applications and associated cache hit rate would be similar in a mobile environment.

3.3 Scale of Detection Algorithms

Table 4 shows the number of threats in each detection engine's signature database. Our mobile agent vastly outperformed ClamAV on the N800 device while protecting against an order of magnitude more threats. While the power overhead of the mobile agent in the worst case

was greater than Kaspersky's antivirus software, Kaspersky only scanned for 284 threats, roughly four orders of magnitude less than the CloudAV network service.

Our results demonstrate that the current model of on-device antivirus software is not scalable. As the number and complexity of mobile threats increase, on-device engines and their signature databases will require more processing, storage, and power. On the other hand, our mobile agent remains constant in its resource requirements and can easily accommodate new signatures and entirely new engines in the virtualized network service.

3.4 On-Device Software Complexity

Our anecdotal experience with on-device antivirus software exemplifies their complexity and inability to deal with mobile platform diversity. First, the ClamAV software running on the N800 caused the device to randomly reboot when performing a normal system scan, making reliable evaluation tedious. Second, the N95 evaluation was originally planned to be with Symantec's Norton Smartphone Security software which advertises compatibility with N95's OS (Symbian Series 60 version 3). However, when we initiated a basic file scan on the N95, Norton would simply return `error -15` and stop execution, with no further information. In comparison, our model of using a lightweight mobile agent greatly reduces on-device software complexity and failures.

4 Related Work

Several mobile services [4, 5, 9, 15, 18, 19] have advocated leveraging remote execution by moving services off-device to minimize resource consumption while achieving performance targets. Our work is novel in the proposition of migrating complex security services to a network-based detection service to provide enhanced protection capabilities to mobile devices while achieving reduced complexity and resource consumption.

Further, work such as [1] shows how security practitioners increasingly leverage virtualization to improve host security. Researchers have also explored the use of on-device virtualization for mobile security applications [2]. In our prior work, we demonstrate that while the effectiveness of desktop antivirus is inadequate against modern threats [12], a virtualized in-cloud network service [13] fares much better.

5 Conclusion

To address the growing concern of mobile device threats, we have investigated a new approach to mobile device malware detection. By moving the detection capabilities to a network service, we gain numerous benefits including increased detection coverage, less complex mobile software, and reduced resource consumption. Our implementation and evaluation show that this approach is not only feasible and effective for the current generation of mobile devices, but will become even more consequential and valuable in the future as the scale and sophistication of mobile threats increase.

References

[1] P.M. Chen and B.D. Noble. When virtual is better than real. *Proceedings of the 2001 Workshop on Hot Topics in Operating Systems (HotOS)*, pages 133–138, 2001.

[2] L.P. Cox and P.M. Chen. Pocket Hypervisors: Opportunities and Challenges. *Proceedings of HotMobile*, 2007.

[3] F-Secure Corporation. F-secure mobile anti-virus. `http://mobile.f-secure.com/`, 2008.

[4] Jason Flinn, Dushyanth Narayanan, and M. Satyanarayanan. Self-tuned remote execution for pervasive computing. In *Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, pages 61–66, Schloss Elmau, Germany, May 2001.

[5] A. Fox, S.D. Gribble, E.A. Brewer, and E. Amir. Adapting to network and client variability via on-demand dynamic distillation. *ACM SIGPLAN Notices*, 31(9):160–170, 1996.

[6] Google. Android - an open handset alliance project. `http://code.google.com/android/`, 2008.

[7] Google. Google safe browsing. `http://code.google.com/apis/safebrowsing/`, 2008.

[8] Kaspersky Lab. Kaspersky mobile security. `http://usa.kaspersky.com/products_services/mobile-security.php`, 2008.

[9] Thomas Kunz and Sali Omar. A mobile code toolkit for adaptive mobile applications. In *Proceedings of the 3rd IEEE Workshop on Mobile Computing Systems and Applications*, pages 51–59, Monterey, CA, December 2000.

[10] Nokia Corporation. Maemo sdk. `http://maemo.org/`, 2008.

[11] Nullriver, Inc. iphone installer.app. `http://iphone.nullriver.com/`, 2008.

[12] Jon Oberheide, Evan Cooke, and Farnam Jahanian. Rethinking antivirus: Executable analysis in the network cloud. In *2nd USENIX Workshop on Hot Topics in Security (HotSec 2007)*, August 2007.

[13] Jon Oberheide, Evan Cooke, and Farnam Jahanian. Cloudav: N-version antivirus in the network cloud. July 2008. To Appear in the *Proceedings of the 17th USENIX Security Symposium.*

[14] John Ogness. Dazuko: An open solution to facilitate on-access scanning. Virus Bulletin*, 2003.*

[15] Alexey Rudenko, Peter Reiher, Gerald J. Popek, and Geoffrey H. Kuenning. The Remote Processing Framework for portable computer power saving. In Proceedings of the ACM Symposium on Applied Computing*, San Antonio, TX, February 1999.*

[16] Sourcefire, Inc. Clamav antivirus. `http://www.clamav.net/`, 2008.

[17] Symantec Corporation. Symantec mobile antivirus for windows mobile. `http://www.symantec.com/norton/products/overview.jsp?pcid=pf&pvid=smavwm`, 2008.

[18] Kaushik Veeraraghavan, Ed Nightingale, Jason Flinn, and Brian Noble. qufiles: a unifying abstraction for mobile data management. In The Ninth Workshop on Mobile Computing Systems and Applications (HotMobile 2008)*, February 2008.*

[19] B. Zenel. A general purpose proxy filtering mechanism applied to the mobile environment. Wireless Networks*, 5(5):391–409, 1999.*