CDroid: Towards a Cloud-Integrated Mobile Operating System

Marco V. Barbera, Sokol Kosta, Alessandro Mei, Vasile C. Perta, and Julinda Stefa
Department of Computer Science, Sapienza University of Rome, Italy
Email: {barbera, kosta, mei, perta, stefa}@di.uniroma1.it

Abstract—**Current offloading mechanisms for mobile energy-hungry apps consider the cloud as a separate remote support to the mobile devices. We take a different approach: We present CDroid, a system residing partially on the device and partially on a cloud software clone coupled with the device, and uses the cloud-side as just-another-resource of the real device. It enhances the user-experience by improving web navigation, compressing and caching web-pages, blocking unwanted ads, and protects user data by virus scanning apps on the cloud-side prior installation on the real-device. CDroid puts the first steps towards a hybrid cloud-integrated mobile system of the future.**

I. Introduction

The proliferation of portable devices and high-speed mobile connectivity has attracted the attention of many developers. Every day are build new cool apps that allow us to do complex things on our smartphones, but that also strain their already limited energy resources. In this scenario, cloud computing is seen as a promising technology that can offer many benefits for mobile devices. Many recent works [1], [2], [3], [4], [5], [6] focus on offloading computation intensive apps from smartphones to remote servers on the cloud to prolong the battery life. In these works the device and the cloud are two different entities, limiting the possibilities offered by the mobile cloud paradigm.

In this demonstration we propose another approach to the mobile cloud paradigm: A system aiming at bringing the device and the cloud closer, towards fully integrating them. In our approach the cloud is seen as another resource of the real device, only a 3G/LTE Advanced connection away from it. So we envision a cloud-integrated mobile operating system, partly residing on the cloud and partly on the device, that boosts its performance, reshapes its features and enables new innovative apps. We build CDroid, a first prototype, with a whole new class of mobile features residing on a private software device clone running on the cloud. CDroid tunnels all the mobile internet data traffic through the cloud-side of the system so to monitor every connection and decide how to deal with the data transmitted/received by the user. In addition, it provides to the user text/image compression, mobile advertisement blocking, anti-phishing filtering, and privacy protection. CDroid considerably improves both user experience and the efficiency of the system.

This work has been technically supported and partially funded by Telecom Italia within the Working Capital project.

This work has been performed in the framework of the FP7 project TROPIC IST-318784 STP, which is funded by the European Community. The Authors would like to acknowledge the contributions of their colleagues from TROPIC Consortium (http://www.ict-tropic.eu).

II. The CDroid System

As shown in Figure 1 CDroid consists of two parts: The part residing on the smartphone and the part residing on the cloud clone (coupled with the device), hosted either on a public or on a private cloud. We assume that the users trust their cloud provider. The cloud-side of the system handles all the external communication and synchronizes with the phone at regulars intervals. To achieve this, we set up a secure tunnel between the smartphone and its clone and make all phone's Internet connections pass trough the clone. During the set-up phase the user sets an account password that allows her to access the features of CDroid. The system features several components:

CDroid-Device: Is the device-side of the system and executes as a background service. It collects informations about user activities and behavior like phone call/SMS sender/receiver information, geographic positions, emails, Bluetooth devices and WiFi hotspots discovered/connected to, and 3G networks. The information is written in a log file and sent in batches to the cloud-side of the system as a piggyback to user traffic.

CDroid-Server: Is the first cloud-side component of the system and is responsible for the communication with the CDroid-Device. It collects the information sent by the CDroid-Device and passes it to the *User Profiler* component (described here below). In addition, it handles user commands to enable or disable specific system features. Finally, it features an *Anti-phishing handler*, that checks the urls in the user request headers to detect if the relative link leads to a phishing website.

User Profiler: Is a security component of the system. It handles the user authentication, constantly verifies the the device is being used by the actual user, and if not, blocks the access to sensitive user data stored on the cloud-side and to internet services accounts like Facebook/Gmail and so on. It works as follows: The User Profiler keeps a record of user activities and builds a *user profile* out of them, using well known profiling techniques [7]. The profile includes recurrent user-related activities—contacts usually called/texted/emailed, WiFi hotspots/Bluetooth devices connected to, GPS coordinates, apps accessed the most and so on—and is constantly updated with the information received from the CDroid-Device. The User Profiler checks for inconsistencies between the recent user behavior and her profile, and accordingly, puts the user in one of the two authentication states: *Authenticated* or *ShouldAuthenticate*. The former enables the user with all the features of the system: cloud-side behaves normally, and satisfies all the users' requests. Rather, the ShouldAuthenticate user state is triggered

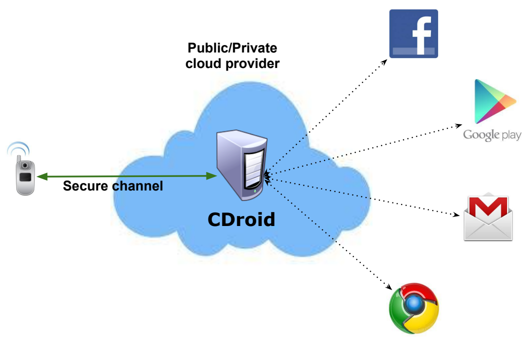

Fig. 1. The CDroid architecture.

when the recent behavior of the user is very different from her profile. An example is a device reporting GPS coordinates of a different state from that of the user. When the user is in the ShouldAuthenticate state the system stops providing services to the device, including access to e.g. the Facebook/Gmail server, till the user re-authenticates with the password. After a successful authentication the user is enabled to fully make use of the system. If the authentication fails, an alert email is sent to the private email account provided in setup phase.

We stress that this component is entirely settable by the user: It is the user that adds the apps/interfaces that are to be monitored and the frequency with which behavior data is sent to the clone. Clearly, the more information included in the profile, the better the detection of suspicious behavior becomes.

Cookie handler: It protects the user from cookie theft and enables the unification of cookies for all applications. If enabled, it intercepts the cookies received from servers the user connects to, removes them from the response header prior sending the response to the user device, and stores them on the cloud-side of the system only. On consecutive connections the Cookie handler puts back the correct cookies on request headers coming from the phone.

Sensitive information blocker: It analyzes the flow of data generated from the device to detect leakage of (unaware) users' sensitive data. If an anomaly is detected the Sensitive information blocker alerts the CDroid-Server which alerts the CDroid-Device to inform the user, and blocks the transmission if necessary. Sensitive informations can be e.g. passwords, credit card or cellphone numbers etc. and are user settable.

Mobile Advertisement blocker: Often app developers intentionally display in-app ads over areas where the user would normally tap during the app usage. Once accidentally clicked, the ad opens the browser by interrupting the user of what she was doing, resulting in a very annoying user experience. In addition, ads generate unwanted data traffic, and can be a threat to the privacy: Many ad companies send ads customized to the user profile, which suggests that sensitive user information is leaked on the process of retrieving new ads.

The Mobile Advertisement blocker component operates as follows: When an app or a web-page sends a request for an ad, the CDroid-Server is alerted to not satisfy the request, and simply reply with a denied messages. Ad requests are easily detectable as they are directed to well-known servers dedicated to ad distribution. Enabling this component results in less traffic generated, better user experience, and more privacy.

Push notifications handler: Many apps that we use feature push notifications, that require the device to keep persistent connections with any of them. The push notification handler manages these persistent connections on the cloud-side, and shrinks the device persistent connections to 1: that with the cloud resource. In addition, the user can decide the frequency of receiving notifications of any kind, or even to store them on the cloud-side and to retrieve them in a consecutive moment.

App handler: This component serves as a security screen of new apps downloaded, prior the installation on the real device. When the CDroid-Server detects a request for a new app download, the App handler installs the relative *apk* on a limited sandbox on the cloud-side, and puts it in the quarantine state. Then, one or more anti-malware softwares are run to scan the app for malicious behavior. In addition, the App handler blocks and logs all Internet connections that start from the quarantined app. If the app passes successfully the quarantine phase, the *apk* is sent to the user device for installation. Otherwise, the app is flagged as malicious and the apk is not sent to the device unless the user decides to install the flagged app regardless.

III. Demo Setup

We demonstrate our CDroid prototype using two Android Samsung Galaxy S Plus devices and an Android x86 software clone deployed on Amazon's EC2 platform. Just one of the two devices run the CDroid system.

During the demonstration the user (a demo presenter or possible volunteers) are given each of the smartphones and ask to perform several tasks to compare the performance of the plain Android system with that of the CDroid one. The users are asked to test each of the CDroid components in order to experience the different features the system: How blocking mobile advertisement results in a better user experience, specially with games that trick the user into clicking adds that interrupt the game itself and open other apps (e.g. the browser), and how the User Profiler protects the user privacy by enforcing authentication when needed.

References

[1] B. G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in *Proc. of EuroSys '11*, 2011.

[2] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading." in *Proc. of IEEE INFOCOM 2012*, 2012.

[3] A. Saarinen, M. Siekkinen, Y. Xiao, J. Nurminen, M. Kemppainen, and P. Hui, "Can offloading save energy for popular apps?" in *Proc. of ACM MobiArch '12*, 2012.

[4] M. V. Barbera, S. Kosta, J. Stefa, P. Hui, and A. Mei, "CloudShield: Efficient anti-malware smartphone patching with a P2P network on the cloud," in *Proc. of IEEE P2P 2012*, 2012.

[5] M. V. Barbera, S. Kosta, A. Mei, and J. Stefa, "To Offload or Not to Offload? The Bandwidth and Energy Costs of Mobile Cloud Computing," in *Proc. of IEEE INFOCOM 2013*, 2013.

[6] S. Kosta, V. C. Perta, J. Stefa, P. Hui, and A. Mei, "CloneDoc: Exploiting the Cloud to Leverage Secure Group Collaboration Mechanisms for Smartphones," in *Proc. of IEEE INFOCOM 2013*, 2013.

[7] D. Pepyne, J. Hu, and W. Gong, "User profiling for computer security," in *Proc. of IEEE ACC '04*, 2004.