ZapDroid: Managing Infrequently Used Applications on Smartphones

Indrajeet Singh*, Srikanth V. Krishnamurthy*, Harsha V. Madhyastha†, Iulian Neamtiu‡
* UC Riverside {singhi, krish}@cs.ucr.edu
† University of Michigan {harshavm}@umich.edu
‡ New Jersey Institute of Technology {ineamtiu}@njit.edu

***Abstract*—User surveys have shown that a typical user has over a hundred apps on her smartphone [1], but stops using many of them. We conduct a user study to identify such unused apps, which we call zombies, and show via experiments that zombie apps consume significant resources on a user's smartphone and access her private information. We then design and build *ZapDroid*, which enables users to detect and silo zombie apps in an effective way to prevent their undesired activities. If and when the user wishes to resume using such an app, *ZapDroid* restores the app quickly and effectively. Our evaluations show that: (i) *ZapDroid* saves twice the energy from unwanted zombie app behaviors as compared to apps from the Play Store that kill background unwanted processes, and (ii) it effectively prevents zombie apps from using undesired permissions. In addition, *ZapDroid* is energy-efficient, consuming $< 4\%$ of the battery per day.**

I. Introduction

There has been an explosion in the number of third-party smartphone apps that are available and are being downloaded. The Google Play Store has more than 1.3 million apps [2] and a report in [3] states that the number of downloads of apps from the Play Store between May'13 and July'13 alone was about 2 billion. However, after users interact with many such apps for an initial period following the download, they almost never do so again. Statistics indicate that for a typical app, less than half of the people who downloaded it use it more than once [4]; further, 15% of the users never delete a single app that they download [5]. In more general cases, users may only interact with some downloaded apps infrequently (i.e., not use them for prolonged periods). These apps, which are seemingly considered *dead* by the user, continue to operate in the background long after the user has stopped interacting with them. Such background activities have significant negative impacts on the user, e.g., leaking private information or significantly taxing resources such as the battery or network. Unfortunately, the user is completely unaware of these activities. We call such apps, which are dead from the perspective of the user, but indulge in undesired activities, as "zombie apps".

Our goal: In this paper, we seek to facilitate effective identification and subsequent quarantine of such zombie apps towards stopping their undesired activities. Since a user can change her mind about whether or not she wants to use an app, a zombie app must be restorable as quickly as possible if the user so chooses.

The classification of an app as a zombie app is inherently subjective. After an app goes unused by a user for a prolonged period, the determination of whether the app should be constrained depends on whether the app's resource usage during the period of unused is considered significant or whether the app's access of private data is deemed serious. Therefore, instead of automatically categorizing apps as zombie apps, we seek to empower the user by exporting the information that she would need to make this decision. Moreover, the manner in which a zombie app should be quarantined depends on whether the user is likely to want to use the app again in the future (e.g., a gaming app that the user tried once and decided is not interesting vs. a VoIP app that the user uses infrequently). The apps that a user is likely to use again fairly soon should not be fully uninstalled; real time restoration (when needed) may be difficult if the user does not have good network connectivity. We seek to enable users to deal with these different scenarios appropriately.

Challenges: Achieving our goal has a number of associated challenges. First, zombie apps are active (execute) in the background and hence, we need an efficient mechanism to track the foreground and background states of apps; continuous monitoring of apps, as proposed in prior approaches (e.g., [6]), can be too resource-intensive to be practical. Second, we need to monitor how apps use sensitive resources protected by permissions in a lightweight manner. Application-level implementations are infeasible since Android does not allow one application to track the permission access patterns of other apps. Third, once a zombie app is quarantined, we must ensure that it is not re-activated unless the user chooses to do so. With current approaches, the background activity of apps are constrained only temporarily [7], until they are woken up due to time-outs or external stimuli [8], [9]. Fourth, *ZapDroid* should ensure that a previously-quarantined zombie app is restored quickly if the user seeks to access it; the restored app must be in the same state that it was in, prior to the quarantine. Reinstalls from the Google Play Store can be hard if the network connectivity is poor and hence, should be not be invoked in cases wherein the user may restore the app with high likelihood; further, clean uninstalls can result in loss of application state.

Contributions: Towards achieving our goal and addressing the above challenges, we design and implement *ZapDroid*. *ZapDroid* identifies candidate zombie apps, exports appropriate information to the user to allow her to choose if she wants to quarantine any of them, and based on the input provided, silos a zombie app appropriately. In addition, *ZapDroid* also seeks to ensure that an app will execute again (in the state prior to quarantine) if the user chooses to invoke it, that the app does not crash, or that unwanted error messages do not pop up when an app is quarantined. Specifically, we make the following contributions: