Mining Sandboxes for Security
Automatisches Sandboxing für Software-Sicherheit

Konrad Jamrozik[1] Andreas Zeller[2]

Abstract: We present *sandbox mining,* a technique to confine an application to resources accessed during automatic testing. Sandbox mining first explores software behavior by means of *automatic test generation,* and extracts the set of resources accessed during these tests. This set is then used as a *sandbox,* blocking access to resources not used during testing. The mined sandbox thus protects against *behavior changes* such as the activation of latent malware, infections, targeted attacks, or malicious updates. The use of test generation makes sandbox mining a fully automatic process that can be run by vendors and end users alike. Our BOXMATE prototype requires less than one hour to extract a sandbox from an Android app, with few to no confirmations required for frequently used functionality.

Extended Abstract

How can I protect my computer from malicious programs? One way is to place the program in a *sandbox,* restraining its access to potentially sensitive resources and services. On the Android platform, for instance, developers have to declare that an application (henceforth referred to as an app) needs access to specific resources. The popular SNAPCHAT picture messaging application, for instance, requires permissions to access the Internet, the camera, and the user's contacts. To install the app the user has to grant such permissions. If an application fails to declare a permission, the operating system denies access to the respective resource; if the SNAPCHAT app attempted to access e-mail or text messages, the respective API call would be denied by the Android system.

While such permissions are transparent to users, they may be too *coarse-grained* to prevent misuse. For instance, SNAPCHAT offers a feature to find friends on SNAPCHAT based on their phone number. To do this, SNAPCHAT accesses the phone numbers of the user's contacts, and sends them to the SNAPCHAT servers. The permission given by the Android sandbox allows SNAPCHAT to do much more than that, namely unlimited access to *all* contacts at *any* time. An attacker thus could inject malware into a SNAPCHAT binary that compromises all contact details; the permissions could stay unchanged.

The issue could be addressed by *tightening* the sandbox—for instance, by constraining the conditions under which the app can send the message. But then, someone has to specify and validate these rules—and repeat this with each change to the app, as a sandbox that is too tight could disable important functionality.

[1] Universität des Saarlandes, Center for IT-Security, Privacy and Accountability (CISPA), Lehrstuhl für Softwaretechnik, Campus E9.1, 66123 Saarbrücken, {jamrozik,zeller}@cs.uni-saarland.de.
[2] See footnote 1.

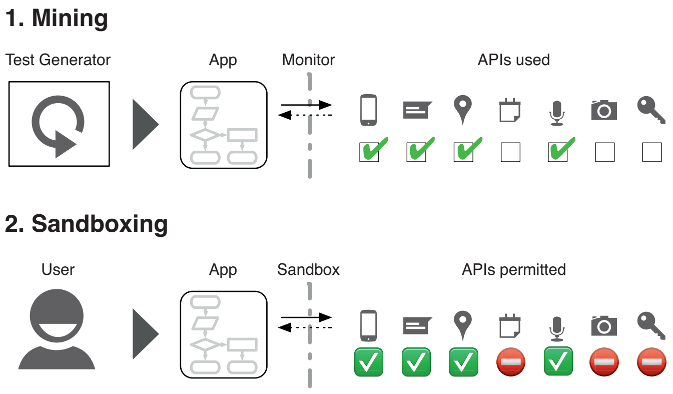

Fig. 1: Sandbox mining in a nutshell. The mining phase automatically generates tests for an application, monitoring the accessed APIs and resources. These make up the *sandbox* for the app, which later prohibits access to resources not accessed during testing.

In this work, we present *sandbox mining,* [JvSRZ16] a technique to *automatically extract sandbox rules* from a given program. The core idea, illustrated in Figure 1, brings together two techniques, namely *test generation* and *enforcement*, in a principle called *test complement exclusion*—disallowing behavior not seen during testing:

Mining. In the first phase, we *mine* the rules that will make the sandbox. We use an *automatic test generator*, DROIDMATE [JZ16], to systematically explore program behavior, monitoring all accesses to sensitive resources.

Sandboxing. In the second phase, we assume that *resources not accessed during testing should not be accessed in production either.* Consequently, if the app (unexpectedly) requires access to a new resource, the sandbox will prohibit access, or put the request on hold until the user explicitly allows it.

To the best of our knowledge, ours is the first approach to *leverage test generation to automatically extract sandbox rules from general-purpose applications.* This allows us to *detect and prevent behavior changes for arbitrary apps*, using fully automatic and easily scalable techniques. As a result, we gain effective protection of apps against known as well as unknown attacks as well as protection against latent malware; sandbox can be mined, validated, compared and re-mined at any time, without any training in production.

References

[JvSRZ16] Jamrozik, Konrad; von Styp-Rekowsky, Philipp; Zeller, Andreas: Mining Sandboxes. In: Proceedings of the 38th International Conference on Software Engineering. ICSE '16, ACM, New York, NY, USA, pp. 37–48, 2016.

[JZ16] Jamrozik, Konrad; Zeller, Andreas: DroidMate: A Robust and Extensible Test Generator for Android. In: Proceedings of the International Conference on Mobile Software Engineering and Systems. MOBILESoft '16, ACM, New York, NY, USA, pp. 293–294, 2016.