Janus Vulnerability: Threats and Mitigation | Guardsquare

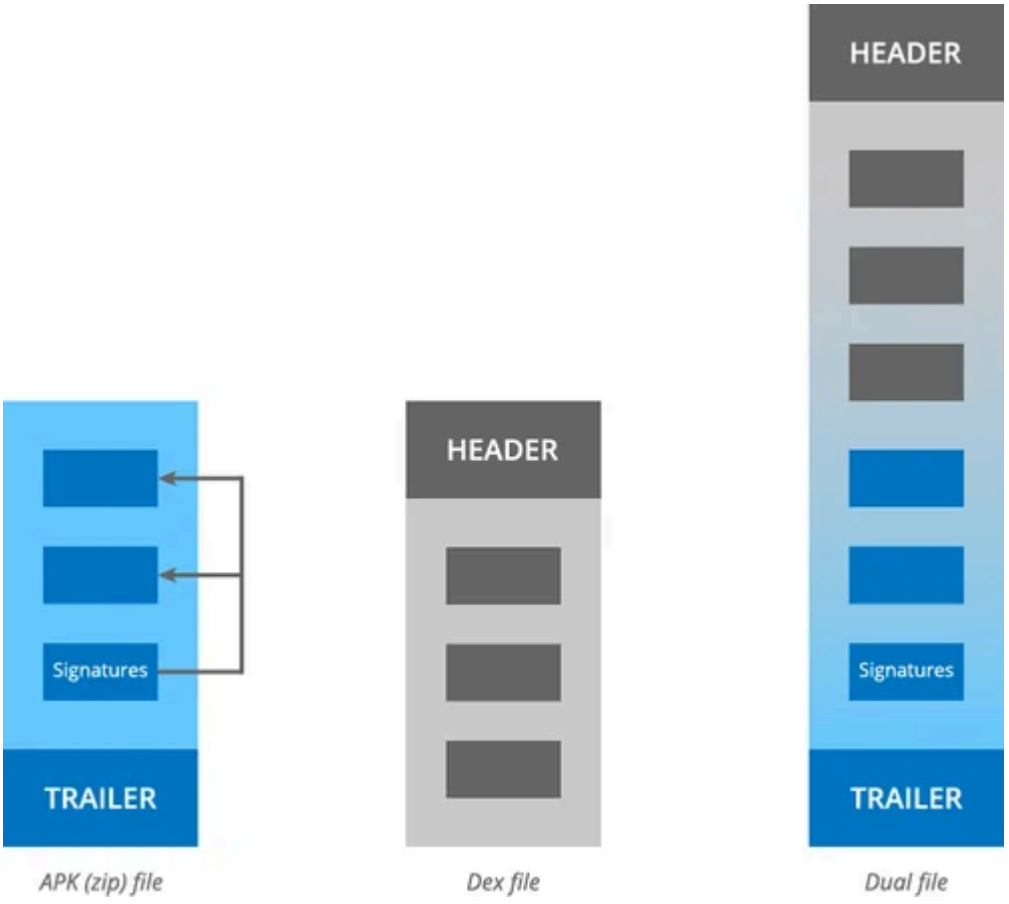
By Guardsquare

Published: 2017-11-13 · Archived: 2026-04-05 18:32:49 UTC

A serious vulnerability (CVE-2017-13156) in Android allows attackers to modify the code in applications without affecting their signatures. The root of the problem is that a file can be a valid APK file and a valid DEX file at the same time. We have named CVE-2017-13156 the Janus vulnerability, after the Roman god of duality.

Janus vulnerability

The Janus vulnerability stems from the possibility to add extra bytes to APK files and to DEX files. On the one hand, an **APK file** is a zip archive, which can contain arbitrary bytes at the start, before its zip entries (actually more generally, between its zip entries). The JAR signature scheme only takes into account the zip entries. It ignores any extra bytes when computing or verifying the application's signature. On the other hand, a **DEX file** can contain arbitrary bytes at the end, after the regular sections of strings, classes, method definitions, etc. A file can, therefore, be a valid APK file and a valid DEX file at the same time.



Another key element is a seemingly harmless feature of the Dalvik/ART virtual machine. In theory, the Android

runtime loads the APK file, extracts its DEX file and then runs its code. In practice, the virtual machine can load and execute both APK files and DEX files. When it gets an APK file, it still looks at the magic bytes in the header to decide which type of file it is. If it finds a DEX header, it loads the file as a DEX file. Otherwise, it loads the file as an APK file containing a zip entry with a DEX file. It can thus misinterpret dual DEX/APK files.

An attacker can leverage this duality. He can prepend a malicious DEX file to an APK file, without affecting its signature. The Android runtime then accepts the APK file as a valid update of a legitimate earlier version of the app. However, the Dalvik VM loads the code from the injected DEX file.

Threats of the Janus Vulnerability

Although Android applications are self-signed, signature verification is important when updating Android applications. When the user downloads an update of an application, the Android runtime compares its signature with the signature of the original version. If the signatures match, the Android runtime proceeds to install the update. The updated application inherits the permissions of the original application. Attackers can, therefore, use the Janus vulnerability to mislead the update process and get unverified code with powerful permissions installed on the devices of unsuspecting users.

One can imagine a few severe scenarios. An attacker can replace a trusted application with high privileges (a system app, for instance) by a modified update to abuse its permissions. Depending on the targeted application, this could enable the hacker to access sensitive information stored on the device or even take over the device completely. Alternatively, an attacker can pass a modified clone of a sensitive application as a legitimate update, for instance in the context of banking or communications. The cloned application can look and behave like the original application but inject malicious behavior.

The zip file format is archaic and prone to problems like the Master Key vulnerability and this Janus vulnerability. Ambiguous zip files likely give rise to similar vulnerabilities in different contexts and on different systems. The root cause is redundancy in the format. When designing data formats, protocols, data structures and code in general, one should always strive to avoid redundancy. Any discrepancies lead to bugs or worse.

Janus Vulnerability Mitigation and Scope

We have created a simple internal tool to create Janus applications as a proof of concept. At this time, we have not seen any such applications in the wild.

Any scenario still requires the user to install the malicious update from a source outside the Google Play store. It may be relatively easy to trick some users because the application can still look exactly like the original application and has the proper signature. For experts, the common reverse engineering tools do not show the injected code. Users should always be vigilant when downloading applications and updates.

The Janus vulnerability affects recent Android devices (Android 5.0 and newer). Applications that have been signed with APK signature scheme v2 and that are running on devices supporting the latest signature scheme (Android 7.0 and newer) are protected against the vulnerability. Unlike scheme v1, this scheme v2 considers all bytes in the APK file. Older versions of applications and newer applications running on older devices remain susceptible. Developers should at least always apply signature scheme v2.

Android applications using DexGuard's tamper detection mechanism are better hardened against cloning attacks. The mechanism performs additional checks to make sure the protected applications have not been modified in any way. We recommend the use of tamper detection and DexGuard's other layers of protection against reverse engineering and cloning.

Disclosure and resolution of CVE-2017-13156

We have reported this issue to Google on July 31, 2017, and received acknowledgment the same day. Google has released a patch to its partners in November. They have published the bug (CVE-2017-13156) in the Android Security Bulletin on December 4, 2017.
