Man-in-the-Disk: A New Attack Surface for Android Apps

By bferrite

Published: 2018-08-12 · Archived: 2026-04-05 15:33:37 UTC

Recently, our researchers came across a shortcoming in the way Android apps use storage resources. Careless use of External Storage by applications may open the door to an attack resulting in any number of undesired outcomes, such as silent installation of unrequested, potentially malicious, apps to the user's phone, denial of service for legitimate apps, and even cause applications to crash, opening the door to possible code injection that would then run in the privileged context of the attacked application.

These Man-in-the-Disk attacks are made possible when applications are careless about their use of External Storage, a resource that is shared across all applications and does not enjoy Android's built-in Sandbox protection. Failing to employ security precautions on their own leaves applications vulnerable to the risks of malicious data manipulation.

What is External Storage?

In order to explain the security deficiency in Android's design, we need to understand a little about the storage resources on an Android device.

Within the Android OS there are two types of storage: Internal Storage, which each application uses separately and is segregated by the Android Sandbox, and External Storage, often over an SD card or a logical partition within the device's storage, which is shared by all applications. Indeed, the External Storage is primarily used to share files between applications or with a PC. For example, in order for a messaging app to share a photo from the phone's gallery, the application needs to have access to media files held in the External Storage.

There are, however, other reasons why an app developer would choose to use External Storage rather than the sandboxed Internal one. Such reasons range from a lack of sufficient capacity in the internal storage, backwards compatibility considerations with older devices, not wanting the app to appear to use too much space, or even just mere laziness on the developer's part.

Whatever the reason may be, when using the External Storage, certain precautions are necessary.

According to Google's Android documentation, application developers are advised on how they should use the External Storage in their apps. These guidelines include:

- "Perform input validation when handling data from external storage"
- "Do not store executables or class files on External Storage"
- "External Storage files should be signed and cryptographically verified prior to dynamic loading"

However, we have seen a few examples where Android apps, including apps from known OEM and even Google, do not follow these guidelines. And herein lays the Man-in-the-Disk attack surface, offering an opportunity to attack any app that carelessly holds data in the External Storage.

The Man-in-the-Disk Attack

The new attack surface found by Check Point researchers, dubbed 'Man-in-the-Disk', allows an attacker to enter and meddle with data stored on the External Storage.

Through our research analysis we have witnessed cases where an app was downloaded, updated or received data from the app provider's server, which passed through the External Storage before being sent on to the app itself – as seen in the diagram on the left. Such practice offers an opportunity for an adversary to manipulate the data held in the External Storage before the app reads it again.

Meddling with the data occurs using a seemingly innocent application, e.g. a fake flashlight app, within which holds the attacker's exploit script. The user is persuaded by the attacker to download this innocent looking app, which in turn asks for the user's permission to access the External Storage, something which is perfectly normal for apps to request, and is unlikely to raise suspicion on the user's behalf. From that point on, the attacker is able to monitor data transferred between any other app on the user's device and the External Storage, and overwrite it with his own data in a timely manner, leading to the unwelcome behavior of the attacked application.

In this way, the attacker has his 'Man-in-the-Disk' looking out for ways in which he can intercept traffic and information required by the user's other existing apps, and offer a carefully crafted derivative of the data that would lead to harmful results.

The results of the attacks can vary, depending on the attacker's desire and expertise. Our research demonstrated the ability to install an undesired application in the background, without the user's permission. We have also demonstrated the ability to crash the attacked application, causing it a denial of service. Once crashed and with the app's defenses down, the attacker could then potentially carry out a code injection to hijack the permissions granted to the attacked application and escalate his own privileges in order to access other parts of the user's device, such as the camera, the microphone, contacts list and so forth.

Applications Where the Man-in-the-Disk Lives

Among the applications which were tested for this new attack surface were Google Translate, Yandex Translate, Google Voice Typing, Google Text-to-Speech, Xiaomi Browser and various additional applications. After referring to the advice given within Google's guidelines, our team proceeded to compare the advice to what is actually the case.

In the case of Google Translate, Yandex Translate and Google Voice Typing, we found that the developers failed to validate the integrity of data read from the External Storage. As such, our team was able to compromise certain files required by these apps, resulting in the crash of each of these applications.

Application Crash Example: Google Translate is caused to crash by compromising certain files required by the app.

Xiaomi Browser was found to be using the External Storage as a staging resource for application updates. As a result, our team was able to carry out an attack by which the application's update code was replaced, resulting in the installation of an alternative, undesired application instead of the legitimate update.

Man-in-the-Disk Attack Example: Xioami Browser's update code is replaced, resulting in the installation of an undesired and malicious app.

Upon discovery of these application vulnerabilities, we contacted Google, Xiaomi and vendors of other vulnerable applications to update them and request their response. A fix to the applications of Google was released shortly after, additional vulnerable applications are being updated and will be disclosed once the patch is made available to their users, while Xiaomi chose not to address it at this time.

Of course, it should be noted that our research only examined a small sample of applications. The prevalence of these vulnerabilities among our sample set leads us to believe that many other apps use the External Storage resource carelessly, and may therefore be susceptible to similar attacks.

Furthermore, the greater the number of privileges an application has access to, the more an attacker has to gain. Indeed, code injection to a privileged application would gain access to all of its privileges. Worryingly, though, some of the apps we found to be vulnerable to the Man-in-the-Disk are actually already pre-installed from the manufacturers, along with pre-granted permissions that the user had not even actively agreed to, and are therefore available as a window of opportunity for an attacker on every single device of this manufacturer.

Cause and Effect

As the details of this attack may seem complex, let us recap the general outline and ramifications of these shortcomings of Android:

- An Android device's External Storage is a public area which can be observed or modified by any other application on the same device.
- Android does not provide built-in protections for the data held in the External Storage. It only offers developers guidelines on proper use of this resource.
- Developers anywhere are not always versed in the need for security and the potential risks, nor do they always follow guidelines.
- Some of the pre-installed and popularly used apps ignore the Android guidelines and hold sensitive data in the unprotected External Storage.
- This can lead to a Man-in-the-Disk attack, resulting in the manipulation and/or abuse of unprotected sensitive data.
- Modification to the data can lead to unwelcome results on the user's device.

Protecting Against 'The Man'

While it is clear that these design shortcomings leave Android users potentially vulnerable to cyber threats, what is less clear is who is really at fault and where the responsibility lies in fixing them. On the one hand, although Android's developers have created guidelines to app developers on how to ensure their apps are safe, they must also be aware that it is well known for developers to not build their applications with security front of mind. On the other hand, and being aware of this foresaid knowledge, is there more Android could be doing to protect their operating system and the devices that use it?

One could equate what we are seeing here in the world of mobile operating systems to the naïve design of old operating systems, which used to allow for buffer overflows to run amok. While the buffer overflow

vulnerabilities were generated by careless developers everywhere, it wasn't until OS and CPU makers took a stand against this, introducing DEP and ASLR protections, that the problem was averted. In the heart of this was the realization that developers cannot always be trusted to follow security guidelines.

From experience then, it would seem that mere guidelines are not enough for OS vendors to exonerate themselves of all responsibility for what is designed by app developers. Instead, securing the underlying OS is the only long term solution to protecting against this new attack surface uncovered by our research.

For full technical details on this research, please visit Check Point Research.
