Android mobile application cloning

Excellium Services Newsletter : Android mobile application cloning

Over the years, the popularity of Android-based mobile devices has significantly grown and has become the most popular device type sold to the public in July 2018, with around 77% of the market shares[1]. This success implies companies have developed many applications for this mobile operating system, in order to provide attractive business services to this new consumer population. In parallel, this growth of applications on the Google Play Store has appealed attackers because of the possibilities in terms of attacks surfaces and the benefits that can be obtained.

January 2, 2019

Share

Over the years, the popularity of Android-based mobile devices has significantly grown and has become the most popular device type sold to the public in July 2018, with around 77% of the market shares[1]. This success implies companies have developed many applications for this mobile operating system, in order to provide attractive business services to this new consumer population. In parallel, this growth of applications on the Google Play Store has appealed attackers because of the possibilities in terms of attacks surfaces and the benefits that can be obtained.

Android mobile application cloning?

Over the years, the popularity of Android-based mobile devices has significantly grown and has become the most popular device type sold to the public in July 2018, with around 77% of the market shares[1]. This success implies companies have developed many applications for this mobile operating system, in order to provide attractive business services to this new consumer population. In parallel, this growth of applications on the Google Play Store has appealed attackers because of the possibilities in terms of attacks surfaces and the benefits that can be obtained.

How the Android application store approach...

The difference with iOS (Apple) about the application store is the fact that, on Android (Google), it is possible to publish an application on the official Google Play Store, but also on alternative application stores[2]. By default, an Android device only accepts applications coming from the official Google Play Store. However, the user has the capacity to disable an option in order to allow the installation of applications coming from non-trusted sources (stores). These alternative stores are not under the control of Google, thus, no verification is performed on the applications published in order to ensure they are not malicious. For instance, in China, the alternative stores are more used than the Google Play Store[3]. Alternative stores are often used by the users to have access to applications not allowed to be present on Google Play Store[4]. It is interesting to note that applications' official authors also sometimes publish their software on these alternatives stores[5].

...leads to an attack vector

This ability to deploy an application on several stores has opened a door, for the attacker, to an attack targeting Android application and named 'Application Cloning'[6]. This attack is happening because an application pushed on the Google Play Store, can be easily retrieved (using a legitimate Android phone as a relay). Then, the attacker will unpack it to add or modify portions of the code, in order to add malicious content allowing her to gather credentials, send SMS to premium rate numbers, get access to sensitive local data (emails, contacts, SMS ...).
It is important to note that the Google Play Store is also affected by applications trying to fake official ones[7].

The attack flow

The attack flow is quite straightforward:

1. Research of a target official application, on the Google Play Store, that is very popular and for which the business features can be hijacked (mobile banking applications, gambling applications...). Popularity of the application is an important factor in the selection of the target because 'Application Cloning' attack often leverage the number of potential victims instead of focusing on a limited number of victims.
2. Verification of the application protection against cloning (unpacking not possible, code obfuscation...), and sometimes a technical research to bypass the protection in place.
3. Unpacking of the application and analysis of the code base in order to identify where the rogue code must be added to reach the malicious objective.
4. Repacking of the application under an attractive name like 'Free Version, Enhanced Version...'
5. Publishing of the application on a maximum of alternative stores (including sometimes the Google Play Store) in order to maximize the chance of the installation from a victim.

Launch of an advertising campaign on different areas to lead victims to install this 'new version'.

How to protect your applications?

Even if the battle against the cloning is difficult, it is possible to add a protection on the application's package itself, in order to make more difficult the second and the third steps of the attack flow. Specialised companies provide dedicated services[8] to add hardening layers like anti-unpacking, anti-repacking, code obfuscation...This kind of protection is added after the application release and prior the publishing of the application on the store.
When this kind of hardening is used, most of reverse engineering tools are defeated. Some examples are shown below on a sample application on which hardening layers has been added on the package file.

The tool apktool [9] cannot unpack the application:

```
$ apktool d hardened-app.apk -o out
I: Using Apktool 2.3.1 on hardened-app.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
Exception in thread 'main' brut.androlib.AndrolibException: brut.directory.DirectoryException: Error copying file:?  ?
at brut.androlib.Androlib.decodeUnknownFiles(Androlib.java:224)
at brut.androlib.ApkDecoder.decode(ApkDecoder.java:172)
at brut.apktool.Main.cmdDecode(Main.java:163)
at brut.apktool.Main.main(Main.java:72)
Caused by: brut.directory.DirectoryException: Error copying file:?  ?
at brut.directory.DirUtil.copyToDir(DirUtil.java:88)
at brut.directory.AbstractDirectory.copyToDir(AbstractDirectory.java:208)
at brut.androlib.Androlib.decodeUnknownFiles(Androlib.java:217)
... 3 more
Caused by: java.io.FileNotFoundException: out\unknown\?? (The filename, directory name, or volume label syntax is incorrect)
at java.io.FileOutputStream.open0(Native Method)
at java.io.FileOutputStream.open(Unknown Source)
at java.io.FileOutputStream.<init>(Unknown Source)
at java.io.FileOutputStream.<init>(Unknown Source)
at brut.directory.DirUtil.copyToDir(DirUtil.java:84)
... 5 more
```

The tool jadx [10] cannot decompile the code:

```
package lu.app;
import android.os.Bundle;
import org.apache.cordova.CordovaActivity;
public class MainActivity extends CordovaActivity {
    /* JADX WARNING: inconsistent code. */
    /* Code decompiled incorrectly, please refer to instructions dump. */
private static java.lang.String '(char[] r12) {
/*
goto L_0x0015;
L_0x0001:
r5 = 0;
goto L_0x008c;
L_0x0004:
r0 = ';     Catch:{ Exception -> 0x0131 }
r0 = r0 + 35;
r1 = r0 % 128;
```

Conclusion

By adding these extra hardening layers, the attacker will be forced to write a version of the application from scratch if she wants to create a clone. As the workload is not the same, an attacker will prefer to choose another target with less protection.

References

[1] http://gs.statcounter.com/os-market-share/mobile/worldwide
[2] https://fossbytes.com/10-google-play-store-alternatives/
[3] https://www.techinasia.com/10-android-app-stores-china-2014-edition
[4] https://play.google.com/about/restricted-content/
https://kingofapp.com
[5] https://apkapure.com/search?q=post+luxembourg&region
https://en.aptoide.com/search?query=bilnet&type=apps&type=apps
[6] https://howtoremove.guide/snapchat-virus-2015-android/
[7] https://blog.zimperium.com/fake-snapchat-google-play-store/
https://www.howtogeek.com/341905/how-to-spot-and-avoid-fake-android-apps-in-the-play-store/
[8] https://www.guardsquare.com/en/products/dexguard
[9] https://ibotpeaches.github.io/Apktool/
[10] https://github.com/skylot/jadx

Credits:

Dominique Righetto.

Newsletter

Recent Posts
The Necessity of Cyber Crisis Exercises
The Art of Password Spraying: A Comprehensive Analysis
Common client-side vulnerabilities of web applications

Categories
Advice
Consulting
Cyber Blog Post
General
Newsletter
PR
The Cyber Blog Times
Uncategorized

Search
Search ...