Remote Kill and Install on Google Android

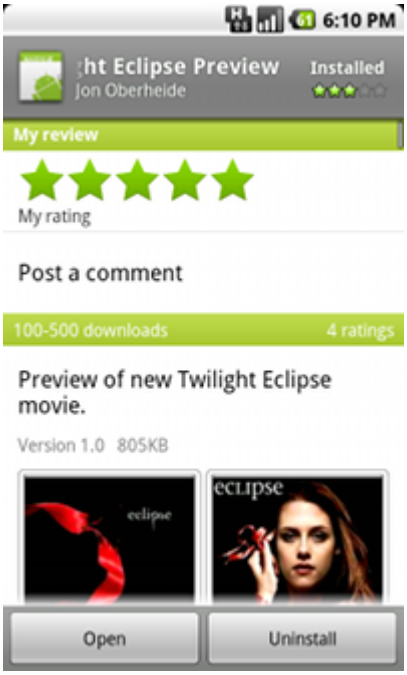
By Jon Oberheide

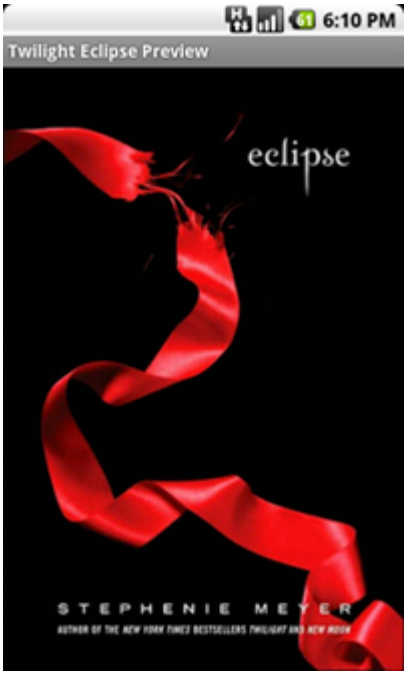
Archived: 2026-04-05 19:44:19 UTC

In this post, I\'ll talk about the REMOVE_ASSET and INSTALL_ASSET mechanisms that can be invoked by Google via Android\'s GTalkService to not only remotely remove applications from an Android device but also remotely install new applications.

RootStrap Background

So if you didn\'t check out my slides from SummerCon last week in NYC, I talked a bit about a program called RootStrap in the second half of my talk. RootStrap is intended as an example of an application that could be used to bootstrap a rootkit (hence the name). Summed up as briefly as possible, RootStrap phones home periodically to fetch remote native ARM code and executes it outside the Dalvik VM. An attacker could use such an approach to gain a large install base for a seemingly innocent application and then push down a local privilege escalation exploit as soon as a new vulnerability is discovered in the Linux kernel and root the device. Since carriers are fairly conservative in pushing out OTA patches for their devices, an attacker could easily push out their malicious payload before the devices were patched.





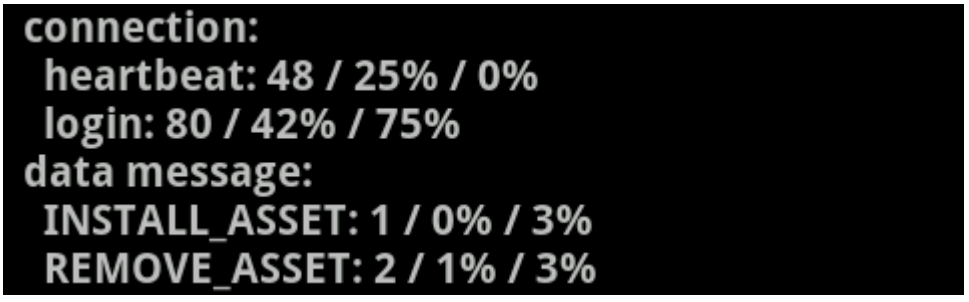
In addition to the sample RootStrap application, I also posted an innocent looking app called \"Twilight Eclipse Preview\" that claimed to be a preview of the upcoming Twilight Eclipse movie to the Android Market. The Twilight app was actually just RootStrap in disguise, displaying a Twilight image while phoning home to check for new payloads to pull down and execute. Obviously, none of these payloads were actually malicious in nature.

The Twilight app got \~200 downloads in the first 24 hours but started slowing down as it received bad reviews...apparently I don\'t have a future in marketing mobile apps to Twilight teens.

Android Remote Kill

In response to Andy\'s coverage on Forbes of my SummerCon presentation, Google asked me to withdraw my applications from the market, which I had no problem doing. Later that day, I noticed a couple of notifications on my phone:

Sweet, Google had just invoked their remote kill functionality! Apparently not many people have dug into the vending APK, but I\'ve known about the REMOVE_ASSET functionality for a while so it was neat to see it being invoked on my phone. I verified my assumptions of the REMOVE_ASSET intent by bringing up the GTalkService monitor:



Talking with Rich later, this was apparently the first time the Android team invoked the remote kill functionality. I had assumed it had been used frequently in the past, but apparently attackers have been slacking off. Rich covered their use of the remote kill functionality on the Android Developers blog today.

Now, the Android platform not only allows for the removal of applications remotely via the REMOVE_ASSET intent, but also allows for the installation of new applications via the INSTALL_ASSET intent. If some people are upset that Google retains the ability to kill applications remotely (I personally prefer the potential security gains of the functionality), I fear what they\'d think of the INSTALL_ASSET feature. ;-)

The GTalkService Connection

So just how does the remote install and remote kill functionality work on the Android platform? I actually talked about this functionality in my SummerCon slides as well when discussing the operation of the Android Market and its protocol.

Your Android device maintains a persistent TCP/SSL/XMPP connection to Google\'s GTalk servers at all times over your device\'s data connection (either your mobile data service or WiFi). This connection is managed by a service aptly named GTalkService. Your device will automatically re-establish this persistent connection whenever you move between networks and periodically sends heartbeat messages to Google\'s servers.

The GTalkService connection allows Google to push down messages to your device. For example, Google\'s recently announced C2DM (cloud to device messaging) platform uses this connection to provide push functionality to third-party apps.

As seen in slide #11 and #12 of my presentation, the GTalkService is actually used during the normal Android Market install process. Instead of downloading the APK from the Market, clicking \'Install\' will trigger Google\'s servers to push a INSTALL_ASSET message down the GTalkService pipe. Upon receiving this message, your phone will download and install the APK.

As expected, the REMOVE_ASSET intent functions similarly. Google can push a REMOVE_ASSET message down to all the Android phones in order to remote kill a particular application deemed malicious.

Both the INSTALL_ASSET and REMOVE_ASSET functionality are implemented in the vending APK:

```
|[0442f4] com.android.vending.InstallAssetReceiver.isIntentForMe:(Lanc
|0000: const/4 v2, #int 0 // #0
|0001: invoke-virtual {v4}, Landroid/content/Intent;.getAction:()Ljava
|0004: move-result-object v0
|0005: const-string v1, "android.intent.action.REMOTE_INTENT" // strir
|0007: invoke-virtual {v0, v1}, Ljava/lang/String;.equals:(Ljava/lang/
|000a: move-result v0
|000b: if-eqz v0, 0023 // +0018
|000d: const-string v0, "android.intent.extra.from_trusted_server" //
|000f: invoke-virtual {v4, v0, v2}, Landroid/content/Intent;.getBoolea
|0012: move-result v0
|0013: if-eqz v0, 0023 // +0010
|0015: invoke-virtual {v4}, Landroid/content/Intent;.getCategories:()L
|0018: move-result-object v0
|0019: const-string v1, "INSTALL_ASSET" // string@0465
|001b: invoke-interface {v0, v1}, Ljava/util/Set;.contains:(Ljava/lang
```

Security Concerns

While remotely removing apps might ruffle the feathers of people who like the feeling of having full control over their device, the remote install functionality is of more concern from a security perspective.

As I mention on slide #14, if an attacker is able to MITM this SSL GTalkService connection for a particular device, it may be possible to spoof these INSTALL_ASSET messages to deliver a malicious application payload. If Google\'s GTalkService servers were compromised, the malicious impact would obviously be a bit more widespread.

You better believe that myself and others are taking a careful look at these code paths. :-)
