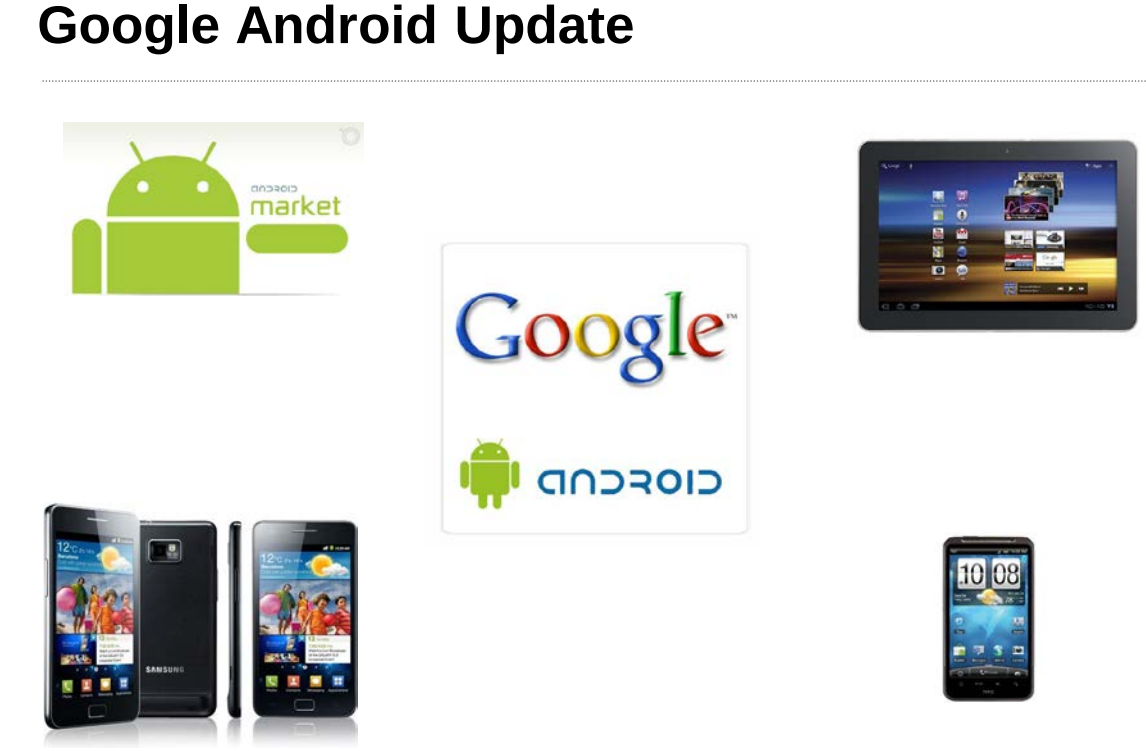
Google Android OS Features

Table of Contents



**032 Mark Williams: Some features of the Google Android operating system.

Google Android Update – Verify Apps -1

Google's defense against malware in apps

Downloaded apps are compared against a database of malware at Google

- Provides a warning of potentially harmful apps

Happens automatically when apps are installed via Play Store

Enabled by default

- Users can disable and install unsafe apps anyway

**033 The first thing to discuss is this concept of verify apps. When we think about all the applications that are available to us on our mobile devices, they are not all provided to us directly from the manufacturer of the operating system. Third parties submit all kinds of apps. So, verify apps is Google's way of trying to defend against third parties inducing malware into their operating system in their phones.

What happens is when an application is submitted, Google has this entire database of malware. And it's able to take this new app that was created and kind of scan it and compare it

against the database to find out if it is a known piece of malicious software. One of the nice things about this is that it happens automatically. We, as the end users, as the consumer, as long as we are making our purchase, if you will, from the Android Play Store, then we know that those apps have already been scanned for this malware. So, it's on by default. We don't have to do anything special in order to deal with it.

Google Android Update – Verify Apps -2

Google Android Update – Verify Apps -2

Does not work for all

- Apps downloaded and installed via other vectors (Amazon, side loaded, etc.) are not compared and verified.
- Apps installed through these other vectors also prevent Google from building a complete malware database.

**034 Where we run in to some Problems, however, is the fact that not all apps are downloaded from the Play Store. We have third parties, such as Amazon. I could go out

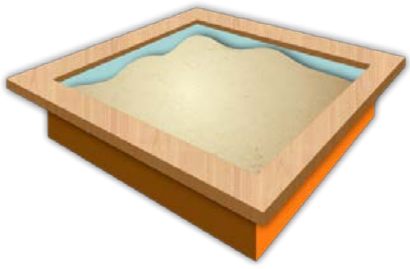
to Amazon and download apps from and purchase apps. I can do what's referred to as side loading. And these apps are not compared. And they are not scanned by Google to find out if they contain any types of malicious software. So, not only is that an issue with side loading or downloading from third parties, but another issue is it also prevents Android and Google from increasing the size of their database of known malicious software. So, it is available to us but app verification does not apply to all.

Google Android Update – Permission Model

Google Android Update – Permission Model

Privilege-separated OS

- Applications run with distinct identity.
  - Unique User ID and Group ID for each app
  - By default apps cannot interfere with other apps.
- Application Sandbox
  - Each application runs in a separate sandbox (Mandatory).
  - Apps must explicitly share resources.
    - App declares what permissions are needed beyond those of the "basic" sandbox.
    - If the app is compromised, the exploit is contained.
- Jailbreak
  - Bypass manufacturer restrictions
  - Makes sandbox useless



**035 Another security feature and functionality of the Android operating

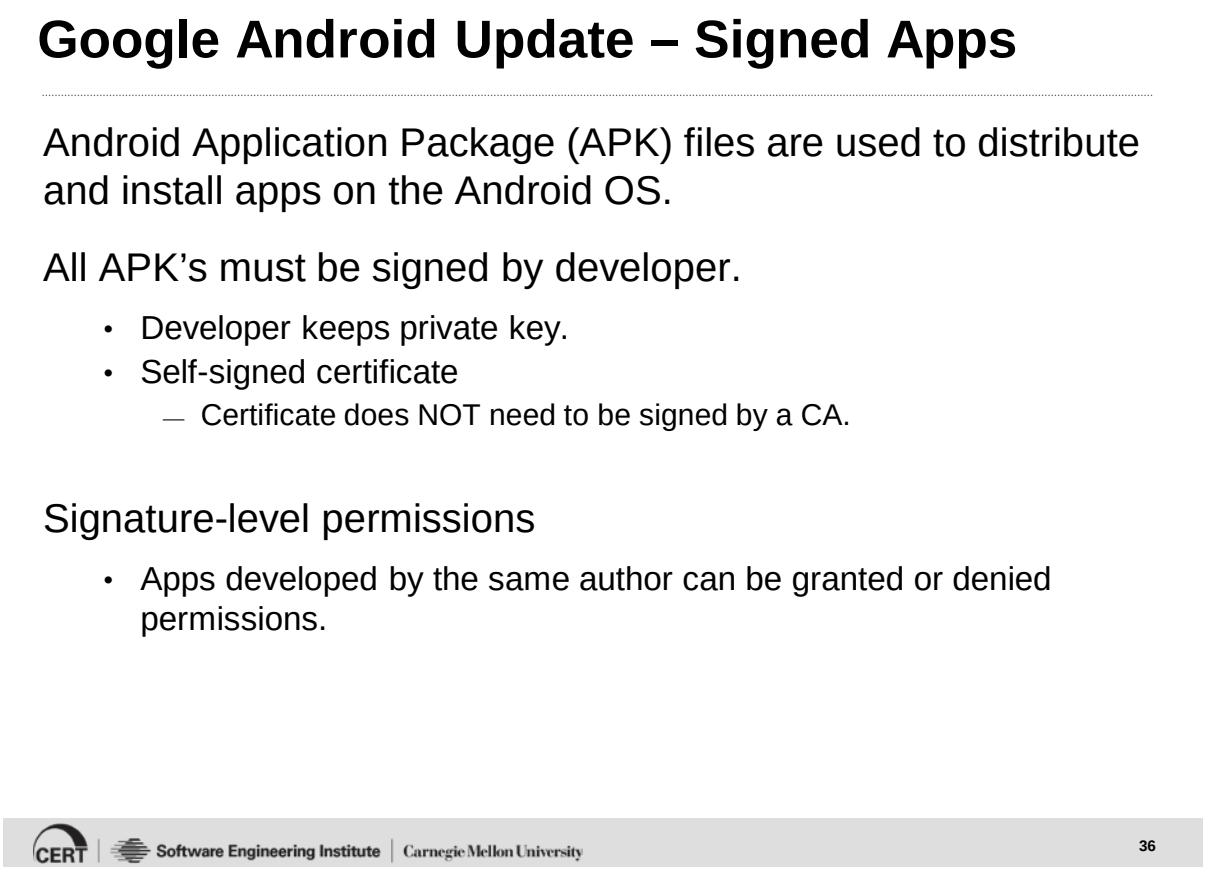
system is a number of different techniques that they use for granting permissions within that operating system. The first concept is that when an application runs-- first off, all apps are given a unique user identifier and a group identifier. And by default, once we run an app, it's identified uniquely. And it cannot have any interaction with third party apps or any other apps on that system without explicit permission from you the owner to be able to do that.

So, we're going to employ the sandboxing technique. And anyone who's familiar with sandboxing, it's basically this concept that that app cannot have access to any resources on the local device. It can only have access to things that are downloaded along with the app. So, it's just a way of containing that app so that it cannot destroy your operating system, change your settings and configurations etc. The sandboxing functionality is enabled by default. So, any app that I run is contained. We can say it's put into a jail box, or a jailhouse, or jail cell. So, sandboxing is the term that is used there.

One of the issues we run into with sandboxing however is oftentimes it's a fairly trivial matter for an app to break out of the jail box. What makes it easy for the apps to break out of the jail box-- out of the sandbox-- What makes it easy for an app to break out of the sandbox is the fact that oftentimes, users will root their phones or jail break their phones.

And by rooting or jail breaking their phone that is giving the user these extended permissions to do pretty much anything and everything that they want. And if the user has extended permissions then the apps that are installed by that user on that device are going to have those extended permissions as well. So, in essence, it's going to render the sandbox concept fairly useless.

Google Android Update – Signed Apps



**036 So, another security feature that we have when it comes to apps is the fact that Android provides us, you and I, the ability to establish some level of trust in those

organizations and people that are creating these third party apps. What can happen is: company A creates an application. And then they can sign that application and say, "I am company A." And as long as I trust company A, and I trust that that application was created by them, then in theory I should be able to feel safe that it's not going to do any damage to me. But it all comes down to trusting company A.

One of the issues that we run into when it comes to trust is there are no third parties that are necessary to verify company A's identity. So, Joe User can create an app, and he can self-sign his app and say, "Yep. I'm Joe User. I am testifying to the fact that I am Joe User, and therefore this app is mine. And you can trust me." That's a small issue that we have, the fact that there is no third party who's going to do anything to verify Joe User.

One of the issues that Android and Google has encountered in the past is a bad actor will create some malware. And they will self-sign it with a particular key, and then they kind of disappear. They change their name. They change their alias. And now there's a piece of malware that's available to third parties or to anybody. And we don't actually know who really created that piece of software. And so, it's maybe a fairly loose model for verifying the source of an application, not a lot of trust in it.

We think about signatures. One nice feature that I have is once I do

decide that I trust vendor A, any application that vendor A creates, and any application that they sign with their private key, I can group them together. And I can collectively say I trust vendor A. And so, all of their apps are automatically inherently trusted. Or I could say I don't trust vendor A. And none of their apps collectively are going to be trusted within the system. So, I can do signature level permissions within the Android operating system.

Google Android Update – Play Store -1

Google Android Update – Play Store -1

Online store for purchasing and downloading apps and content

- Formerly Android Market

Security

- Google Bouncer – Scans developer accounts and apps
  - Bouncer was “fingerprinted” – We know how it works
  - Opened the door for malware to fool Bouncer
- Human review – Humans look for malware and unacceptable content (sexually explicit, violence, etc.) and other violations

**037 I mentioned the Play Store a few moments ago. And I also mentioned that all apps that are

submitted to the Play Store are going to be scanned for malware. In the back end at Google they have a tool. And they call it the Google Bouncer. And it is that big, nasty looking guy. And that's his entire job is to scan all apps and all pieces and parts of the apps in order to look for malicious software.

A lot of the malicious software has been identified by a signature. We create hashes for these different chunks of code. And so, when you submit your app to the Google Play Store, we scan it. We look for the signatures. We look for the hash matches. And if there's any part of your app that has malware in it, it's going to be bounced out of there.

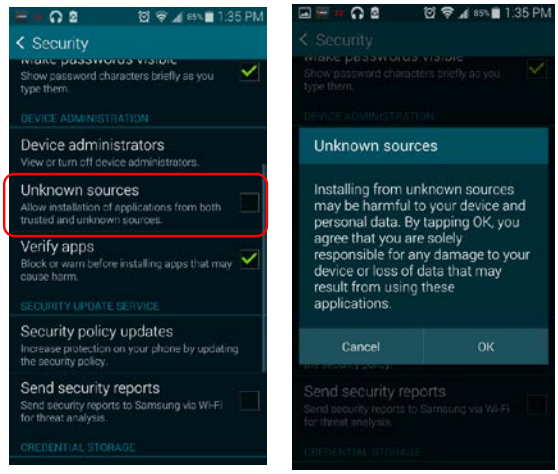
But it's never a good idea to rely solely on technology. And so, not only does Google use this Bouncer application to scan the third party information, it also has humans. There is an entire group of employees at Google. And their entire job is to go looking through all of the applications, or many of the applications, trying to identify any kind of violations, data that may be considered to be sexually explicit or unacceptably violent, it does not meet the Google licensing agreements and so on, the terms and use agreements and such. So, humans are going to go in and look for this malicious content.

Sometimes this is triggered by a report. Not everything gets caught in the front end. Sometimes it's

triggered by a report by a third party who says, "Hey, I downloaded this app from the Play Store. And it has some really violent stuff in it." And so, when we make these reports then Google, the humans will go through and start looking for it. And then they make decisions whether it's going to be allowed to stay in the store or they're going to eradicate it from the store.

Google Android Update – Play Store -2

Google Android Update – Play Store -2



Apps can be downloaded from other “unofficial” channels

“Caveat Emporium” – Buyer Beware!

- These apps do not get vetted by Google.

Sideloading

- Installing apps without going through Google Play
  - Manual
  - ADB
  - AirDroid

**038 As mentioned, the Bouncer only works when it's an app that is submitted through the Google Play Store. If you are using any other channel for downloading your apps--

Let's say I get an app from one of my friends. Or let's say I side load an app or I get an app from Amazon. Caveat emporium is the phrase I use, buyer beware. We think about trust. You have to be able to establish trust in the source of the data and the application. And if you cannot establish trust, then maybe consider running that app in a sandboxed environment, in a test environment to find out what it's going to do for you.

We mentioned side loading. It's just this concept of maybe I have an app on my computer, and I want to put it onto my phone, so I can connect my phone to my computer. And I have a cable. And there are a number of different tools that are available that allow me to do that install.

One of the tools is known as ADB, Android Debug Bridge.

Android Debug Bridge (ADB)

Included with the Android SDK package

Command line toolkit used to debug Android apps

- Consists of server and client programs that allow developers to send commands to their Android using the PC

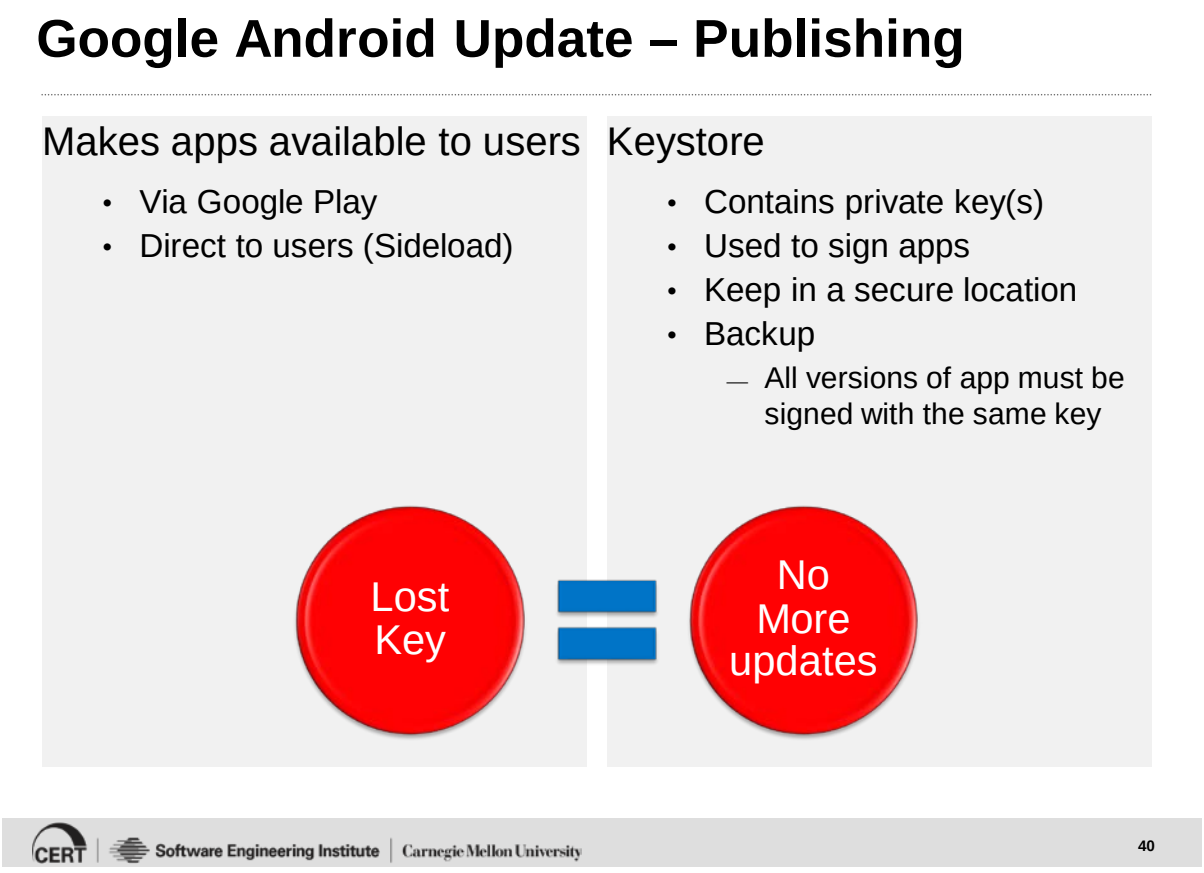
Used for

- “Rooting” or “jailbreaking”
- Install / uninstall or “side load” apps
- Restore the device after accidental “bricking”
- Reboot devices
- Backups
- More

**039 Now, it was primarily designed to be used for developers. It allows developers to create an app, test out that app on their systems. But there's nothing that stops average, every day people from going out and getting the ADB STK and saying, "Hey, I want to load an app." Or since Android is completely open source, there's Nothing that is going to stop me from creating my own piece of malware and using ADB in order to test it, verify it's going to work, verify it's going to do the damage I want it to do, and then load it onto someone's computer system. Right?

We've mentioned that ADB is used for developers. And so, developers sometimes want their app to run on a jail broken or rooted system. So, ADB allows me to do the jail breaking or the rooting. It allows me to fix problems when I accidentally screw things up. It allows me to reboot the device, do backups. ADB is a very powerful utility. And again, caveat emporium, if you don't know what you're doing with something like ADB, you probably shouldn't be messing with it.

Google Android Update – Publishing



**040 Publishing, I mentioned a few moments ago about signing of the

applications. Well, when we make something available through the Google Play Store, the assigned applications are going to be published. And there's a key that is going to be required in order for us to verify who the author of that application is going to be.

So, this is getting into this concept of symmetric and asymmetric key cryptography. So, a developer makes an app. They sign it with their private key. When I, the user, download that app, I have to somehow verify. So, I will grab their public key on a certificate usually and verify.

The issue we run into is that private key of the developer. What happens if that private key becomes lost? Then the developer is not going to be able to update their application anymore. So, if I have version one of my calendar application, I'm the creator of it. I have version one. I sign it with private key number one. And I create version number two. I need to sign it also with private key number two. And every subsequent version Google is going to require me, the developer, to sign with the exact same private key. And if my private key becomes lost or it gets compromised, then I can no longer update my application. And so, I would have to basically start fresh with a new application, completely new tool. So, publishing is the concept here. We need to rely on those private keys.

Google Android Update – Storage

Internal Storage

- Data stored here only accessible by the app that stored it
- Can be encrypted

External Storage (Before 4.4 Kit Kat)

- Globally readable and writable
- Modifiable by any application
- Do not store sensitive data

External Storage (Kit Kat & beyond)

- 3rd party apps can only access files / folders created by the app (or the app has taken ownership of)
- Added security but has “broken” many apps and frustrated users

**041 Another consideration for us is where are we storing data on our mobile devices. There are two primary storage locations, the internal memory that comes with the phone. And then a lot of devices allow me to add external memory, external storage. When it comes to internal storage, when data is stored in that internal location, only the app that puts the data there is going to be able to have access to it. And the data that is stored internally is going to be encrypted. So, no other app's going to have the ability to see even what is stored on the internal memory.

Where we run into possibly a problem is with external storage. External storage, depending on what version we're using, older versions of the Android operating system allowed external storage to be globally readable and writable. And so, any app could store any data there. And any other app could see what was there. We had a lot of malicious apps that would take advantage of this. So, maybe I have a mapping application that has stored data about where I have been over the recent days, weeks, and months. And there might be some spyware application that could then look at that data and figure out exactly what my history is and report that to third parties.

So, in later versions of the Android operating system, Google has decided to make it so the external storage is also going to be a little bit more tightly controlled and restricted. The third party apps are only going to be able to access data and files that have been created by that third party app very much like what's happening with the internal storage.

The problem that we run into with this extra type of restriction is it makes a lot of developers and programmers unhappy. Developers and programmers generally like to have devices and their applications operating at root level permissions so they have no restrictions. It makes it easy for the developer to create their app. As soon as we start adding restrictions, then the developer has

to figure out how am I going to make my application work.

And what made it even more challenging is the fact that as a developer, I may have written my app to use external storage that was, I'll say, the old model where anybody and everybody could read and write to anything on that external storage. So, here's my app that uses the old model. And then somebody updates their phone to a newer version of the operating system. And now my apps are no longer functioning anymore. So, now that causes the developers frustration because they have to make an update. And it causes the users frustration because things that used to work no longer do.

Notices