SEC575 | Mobile Device Security and Ethical Hacking

SANS

iOS 12 and Android Pie Update

© 2018 Joshua Wright | All Rights Reserved | Version D01_01

https://www.sans.org/sec575

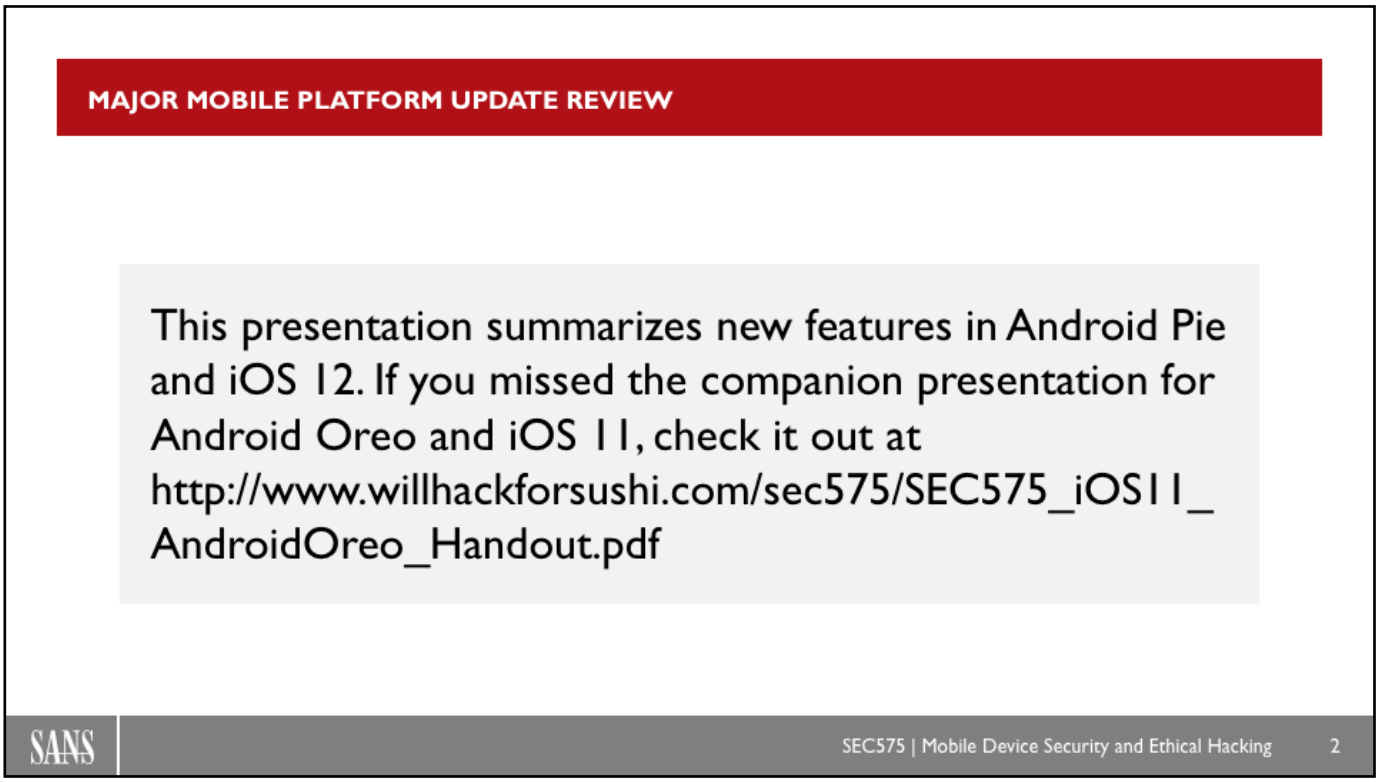
The technology supporting mobile devices moves pretty quickly. To keep you up to date with developments in the mobile security field, we have developed this presentation.

Joshua Wright

jwright@hasborg.com

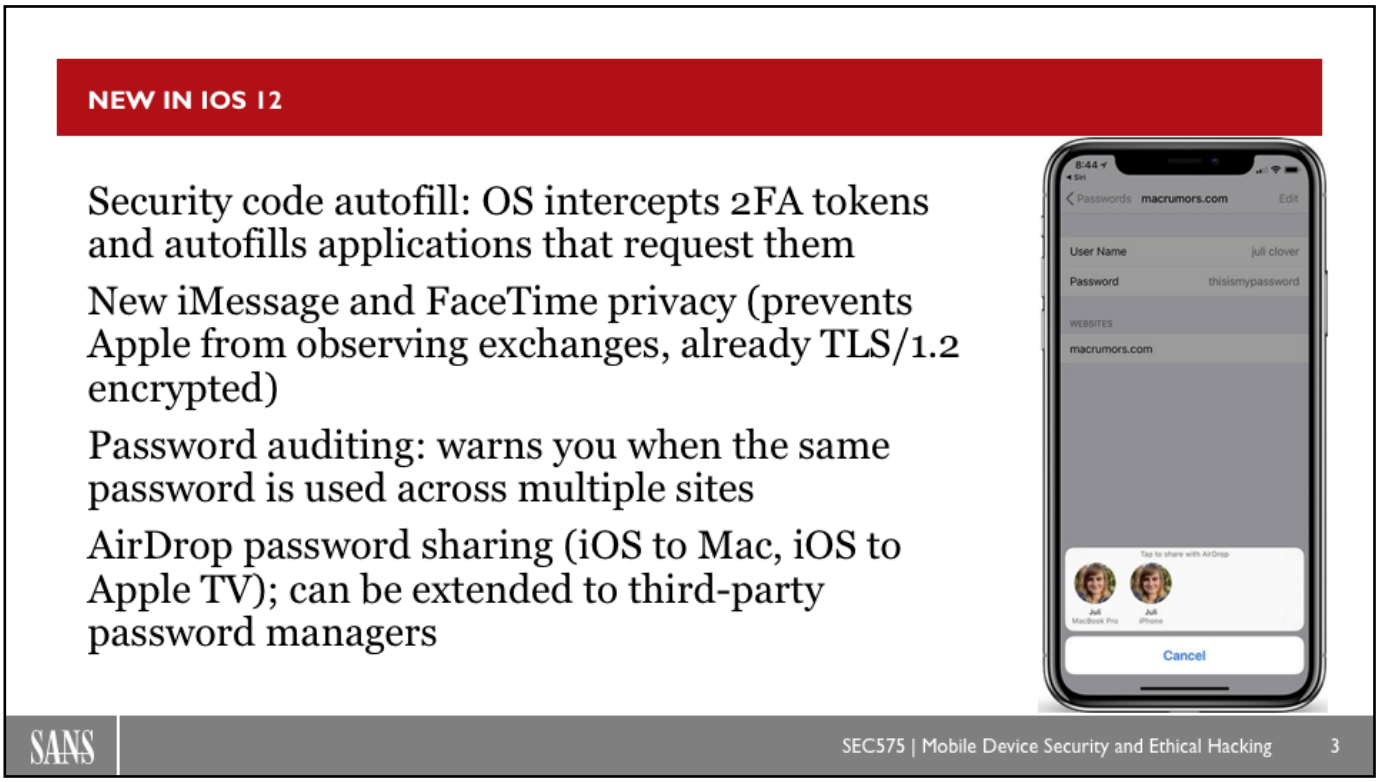
@joswr1ght

9/24/2018



Major Mobile Platform Update Review

For each major update of the iOS and Android platforms, I put together a presentation of the new major security considerations as well as the things to keep an eye on as these platforms develop. If you missed the prior presentation covering Android Oreo and iOS 11, check it out at http://www.willhackforsushi.com/sec575/SEC575_iOS11_AndroidOreo_Handout.pdf.



New in iOS 12

Apple released the iOS 12 platform on September 17, 2018. Like prior major releases, iOS 12 offers several noteworthy security enhancements.

Security Code Autofill: In order to make two-factor authentication (2FA) systems easier for end users, iOS 12 introduces security code autofill. For both native and web-based applications, the platform will monitor for text keywords indicating the presence of an SMS (or iMessage) message disclosing the 2FA code and offer a 1-tap keyboard option to autofill the value. While third-party apps should not be able to intercept SMS messages, it is convenient for end users and interesting that an API exists for Apple to monitor inbound messages and prepopulate form fields with those values.

New iMessage and FaceTime Privacy: In iOS 12, new iMessage and FaceTime privacy controls are introduced to limit accessibility to content to and from those applications. While iMessage and FaceTime protocols are already TLS/1.2 encrypted, iOS 12 also encrypts the content with a key local to the sending and receiving devices. As a result, iMessage and FaceTime content is not accessible through a wiretap, or otherwise through a monitoring and interception station at Apple or another service provider.

Password Auditing: iOS 12 users who leverage the iCloud Password storage mechanism will now receive warnings when they attempt to use the same password across multiple sites.

AirDrop Password Sharing: iOS 12 users can also leverage iCloud Password storage in conjunction with AirDrop to share passwords across devices (iOS to iOS, iOS to Mac, Mac to iOS). This functionality can also be extended to third-party password managers. The use of AirDrop password sharing is interesting because it represents an opportunity to eliminate the entering and storage of passwords on Mac laptop and desktop systems, and for the use of the mobile device as an external authentication token with biometric verification functionality.

Image on this page: https://www.macrumors.com/roundup/ios-12/

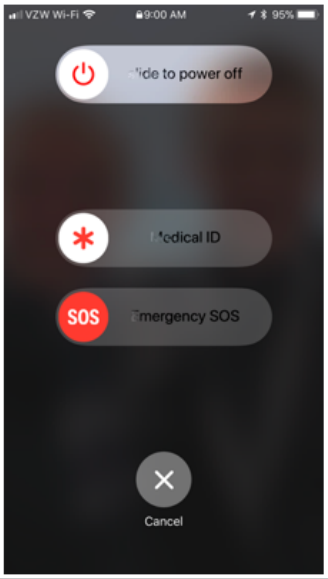
EMERGENCY SOS (IOS 11)

Shortcut to disabling biometric authentication, placing an emergency call

- iPhone 7/7+ and older: press power 5 times rapidly
- iPhone 8/8+/X: press and hold side button while holding volume up or down
- Can turn on auto-call feature in settings

Device will not unlock with Touch ID/Face ID

Tiny snub to US LEAs and courts that say biometric data is not US Constitution 5th Amendment protected

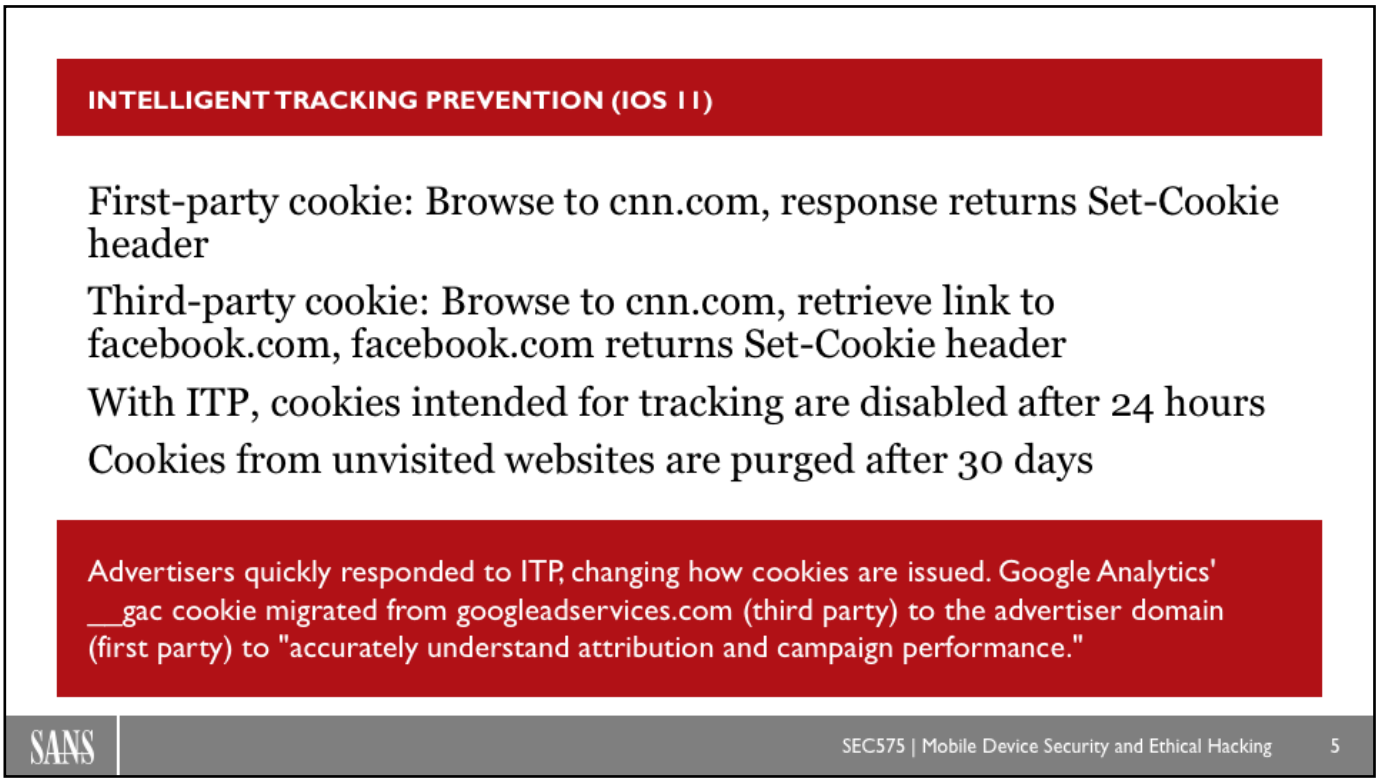


Emergency SOS (iOS 11)

Apple introduced the Emergency SOS feature in iOS 11, but we're mentioning it here since it is necessary to understand this functionality as it relates to Android Pie.

iOS 11 introduced a new feature called Emergency SOS where the user can disable biometric authentication and get a shortcut to place an emergency call (or immediately place an emergency call if configured to do so) by pressing the power button 5 times rapidly on the iPhone 7/7+ and earlier or by pressing and holding the side button while holding volume up or down on later iPhones.

When the user activates Emergency SOS, even if an emergency call is not placed, the iOS device will disable biometric authentication access, requiring the user to enter the secondary authentication credential to unlock the device. This could be seen as a tiny snub to US law enforcement agencies and courts that say that biometric data is not protected by the US Constitution 5th Amendment (the 5th Amendment of the US Constitution is part of the Bill of Rights, which protects individuals from being compelled to be witnesses against themselves in criminal cases). Essentially, lower courts in the US have ruled that biometric information is not protected, and a suspect can be forced to unlock an iOS device using a fingerprint or face scan. An individual about to be apprehended could trigger the Emergency SOS feature, preventing the device from unlocking using biometric information.



Intelligent Tracking Prevention (ITP)

iOS 11 also took steps to defend against cookie tracking using Intelligent Tracking Prevention (ITP). To understand ITP, we have to first differentiate the concept of first-party and third-party cookies.

A first-party cookie is set when a user visits a website (either through a browser or app with a WebView) and the server returns a Set-Cookie response header. A third-party cookie is set when a user visits a website (browser or WebView), and the linked content generates additional requests to new websites (such as when you browse to cnn.com, and cnn.com links to an image on facebook.com). If the subsequent requests are returned with Set-Cookie headers, the cookies are known as third party.

With ITP, iOS applies different policies to cookies whether they are first party or third party. A third-party cookie is disabled after 24 hours if it is intended for user-tracking purposes (the nature of the *intent* is not clear in the Apple documentation). First-party cookies are purged if the website that issued them has not been visited within 30 days.

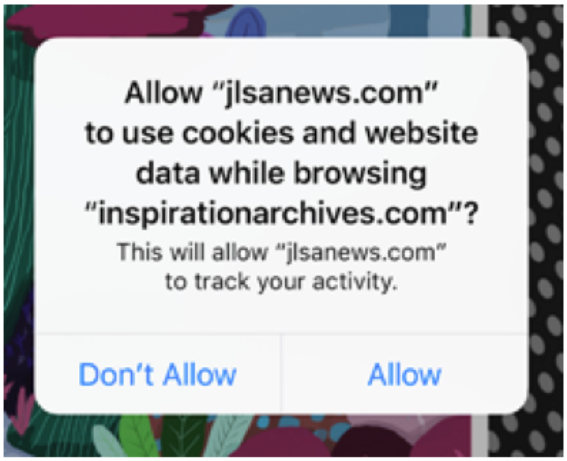
Following this feature announcement from Apple, advertisers expressed discontent with the change, indicating that the change would hamper their ability to deliver a positive user experience that supplies desirable advertising to users (https://www.pmxagency.com/blog/2017/09/ios-11-intelligent-tracking-prevention-means-marketers). Many advertisers quickly responded to the change by refactoring how cookies are delivered. For example, Google Analytics' __gac cookie was formerly delivered by googleadservices.com (a third party for most users), but has been changed such that it is delivered from the first-party site instead to "accurately understand attribution and campaign performance" (https://www.pmxagency.com/blog/2017/09/ios-11-intelligent-tracking-prevention-means-marketers/).

ENHANCED INTELLIGENT TRACKING PROTECTION (IOS 12)

Disables multiple browser fingerprinting techniques

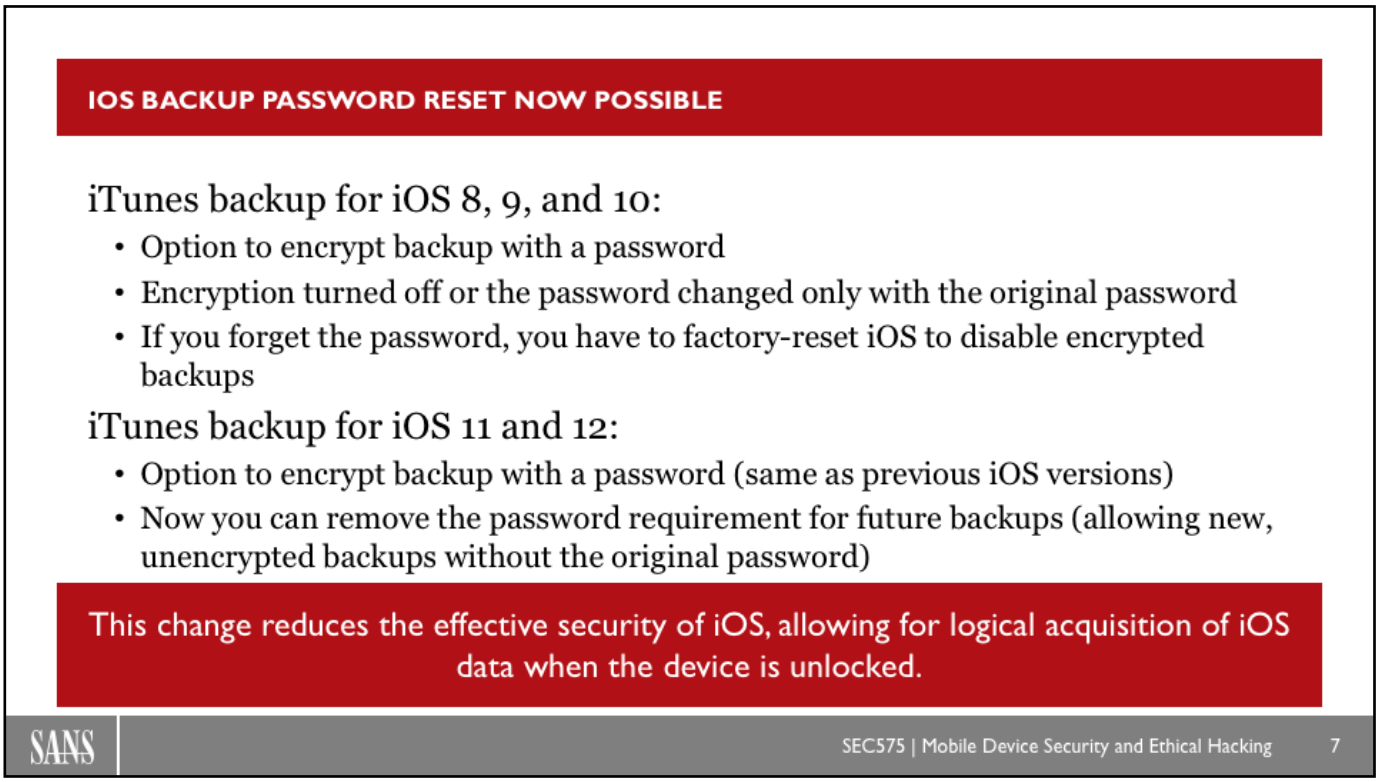
- Detecting specific fonts, screen dimensions, plugins (ad blockers), etc.

Prevents social networking Like/Share/Comment fields from setting first-party cookies automatically



Enhanced Intelligent Tracking Protection (iOS 12)

In iOS 12, Apple continued to enhance the ITP features introduced in iOS 11. Apple no longer allows websites to collect detailed fingerprint information from browsers (beyond the user agent, including the detection of specific fonts, screen dimensions, installed plugins, etc.), making users seem much more generic and not specifically attributable to any use case or identity. Further, Apple will prompt users to permit or deny third-party JavaScript from social media sites' links through *share* and *like* buttons and comment fields, as shown in the screenshot on this page.



iOS Backup Password Reset Now Possible

A potential reduction in security in iOS 11 and iOS 12 is the ability for users to reset a password associated with a backup, allowing a user to generate new, password-less backups.

With iOS 8-10, the user has the option to select a backup password in iTunes. When the password is entered, it is stored on the iOS device and the data is encrypted prior to delivery to the host system for storage in iTunes. To restore the backup, the backup password must be entered to decrypt the contents. Further, removal of a password was only possible by entering the password and generating a new, unencrypted backup, or by performing a full device reset (losing all data on the device).

In iOS 11 and iOS 12, this behavior has changed. Password are still an option to protect the confidentiality of iTunes backups, but unlike earlier versions of iOS, the requirement to use a password for a backup has been removed. With iOS 11 and iOS 12, the user can navigate to Settings | General | Reset and tap *Reset All Settings* to remove the iTunes backup password requirement.

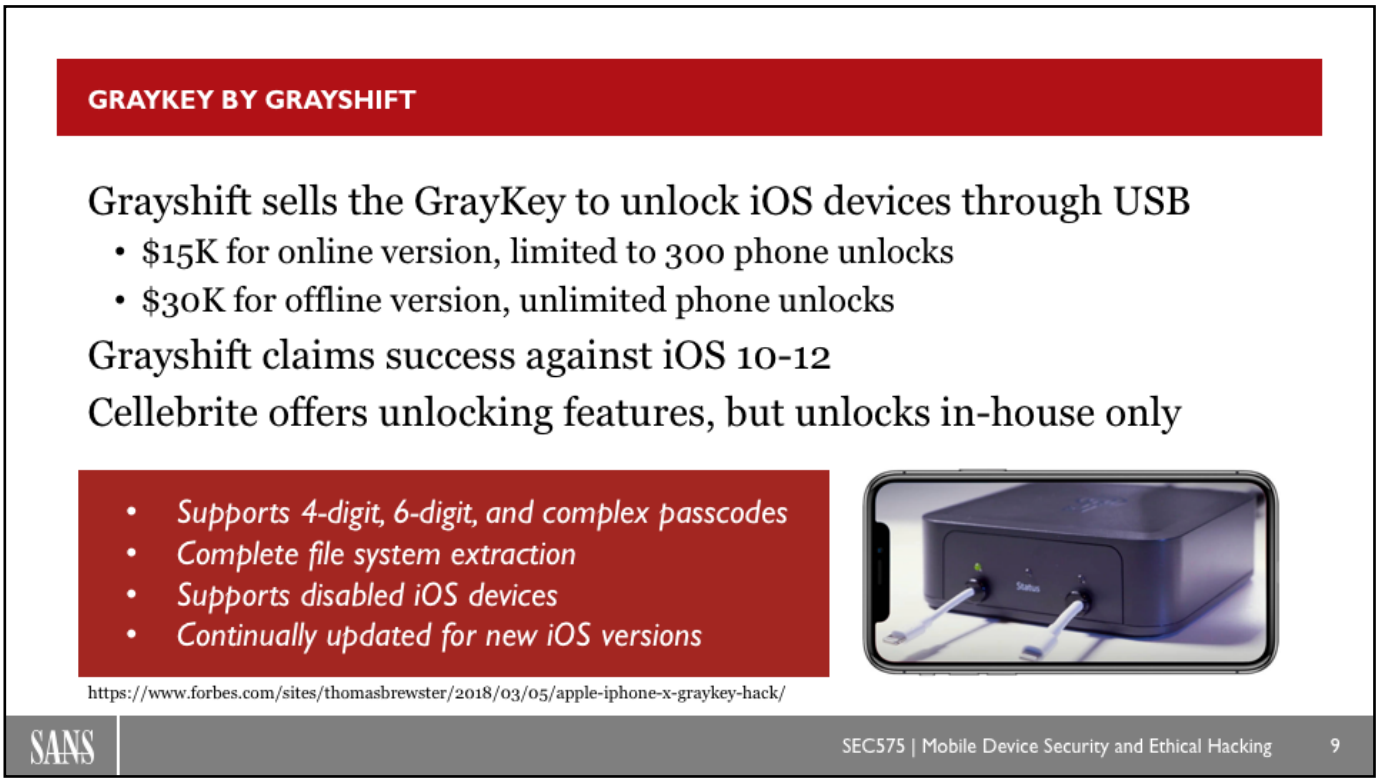
For end users, this could be seen as a positive change, since prior to iOS 11 if the user forgets their iTunes password they can't restore backups or create new unencrypted backups (e.g. they can't turn off the password required to restore the backup). However, this also opens up a new opportunity for forensic analysis of an iOS device.

Consider a case where the user has an iOS 10 device with a backup password. If the user is apprehended by law enforcement in the US, the officer could use TouchID or FaceID to unlock a device and access data from the device itself. However, a *logical forensic acquisition* (e.g. an iTunes backup) is not possible since the iOS device will not create new, unencrypted backups without the prior iTunes backup password (which the suspect could "forget" or otherwise not turn over to LEA). With iOS 11, this limitation for LEA is removed, and an apprehended iOS device could be unlocked and a logical forensic acquisition could be applied to recover all data after navigating to the Reset All Settings option.

https://support.apple.com/en-us/HT205220

To reset the backup password on your iOS device, go to Settings | General | Reset. Tap Reset All Settings and enter your iOS passcode. Connect the device to iTunes again and create a new backup.

This change reduces the effective security of iOS, allowing for logical acquisition of iOS data through iTunes when the device is unlocked.



GrayKey by Grayshift

Grayshift is a digital forensics company run by an ex-Apple employee and US intelligence agency contractors (https://www.forbes.com/sites/thomasbrewster/2018/03/05/apple-iphone-x-graykey-hack). The flagship product at Grayshift is the GrayKey, a "technology that provides lawful access to iOS devices" (http://www.technosecurity.us/mb/exhibitors/grayshift). Being sold to local, state, and federal law enforcement agencies, the GrayKey is designed to bypass the lock screen on iOS devices supporting iOS 10 or iOS 11, as well as many others devices, including the iPhone 8 through the Xs.

Exploiting an undisclosed vulnerability in the USB interface of iOS devices, the GrayKey sells for $15K for a hardware device that requires internet access, limited to 300 phone unlocks. For $30K, the same capabilities are also available in an offline mode, unlocking an unlimited number of devices.

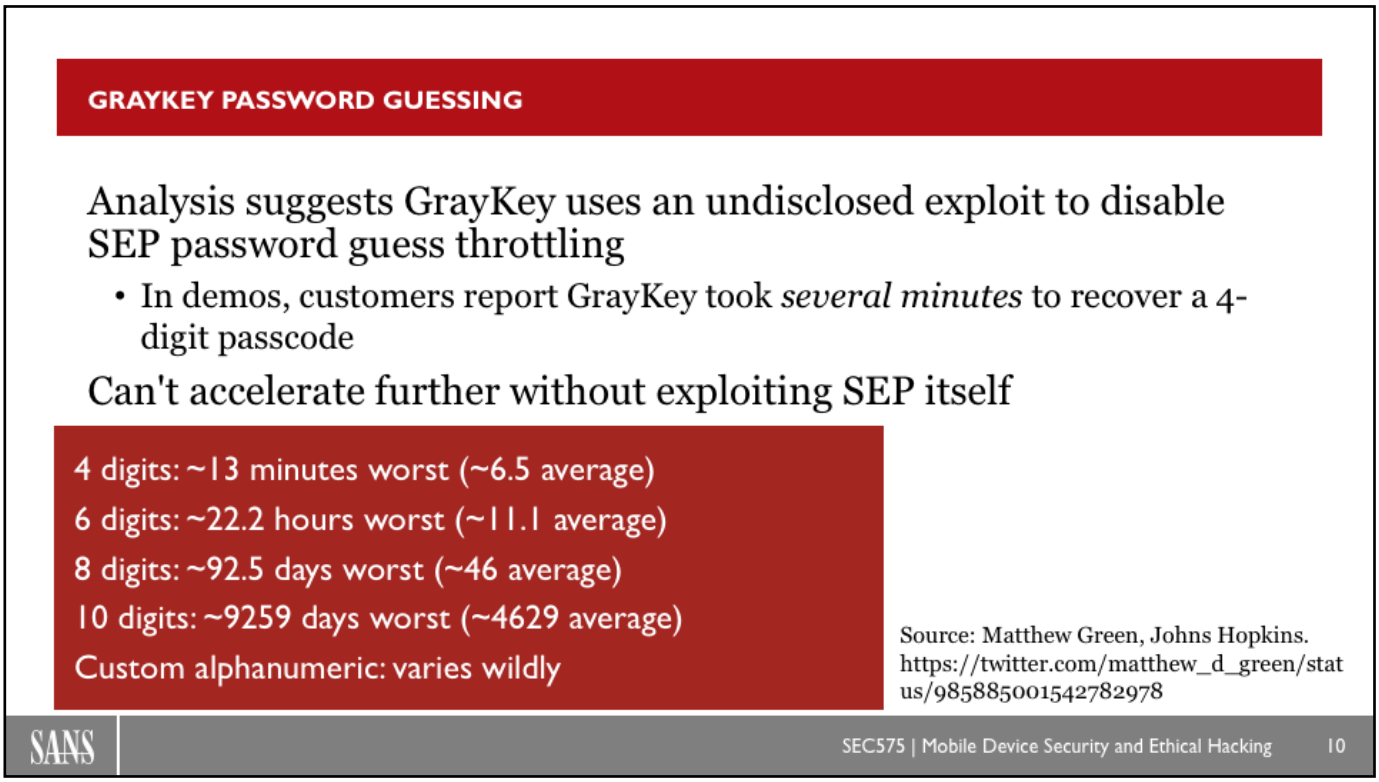
The GrayKey device offers several capabilities that are valuable to forensic analysis of iOS devices, including:

- Bypass of 4-digit, 6-digit, and complex passcodes (alphanumeric)
- Complete file system data extraction (reportedly exceeding the capabilities of a *logical acquisition* through an iTunes backup)
- Can bypass disabled iOS devices (e.g., devices that are in recovery mode or are disabled due to prior incorrect password guesses)
- Ongoing support for new iOS software versions

Similar capabilities have been available through a *device unlock service* at the mobile forensics company Cellebrite, but in a very different deployment model. Where Grayshift sells a physical device for phone unlocking to the customer, Cellebrite requires law enforcement to send in the device to be unlocked for their handling. Cellebrite maintains a more restricted level of control over what devices can be unlocked by which

customers, while Grayshift lacks similar controls, particularly in the more expensive *offline* GrayKey device.

Image: https://www.macobserver.com/columns-opinions/editorial/grayshift-data-breach/

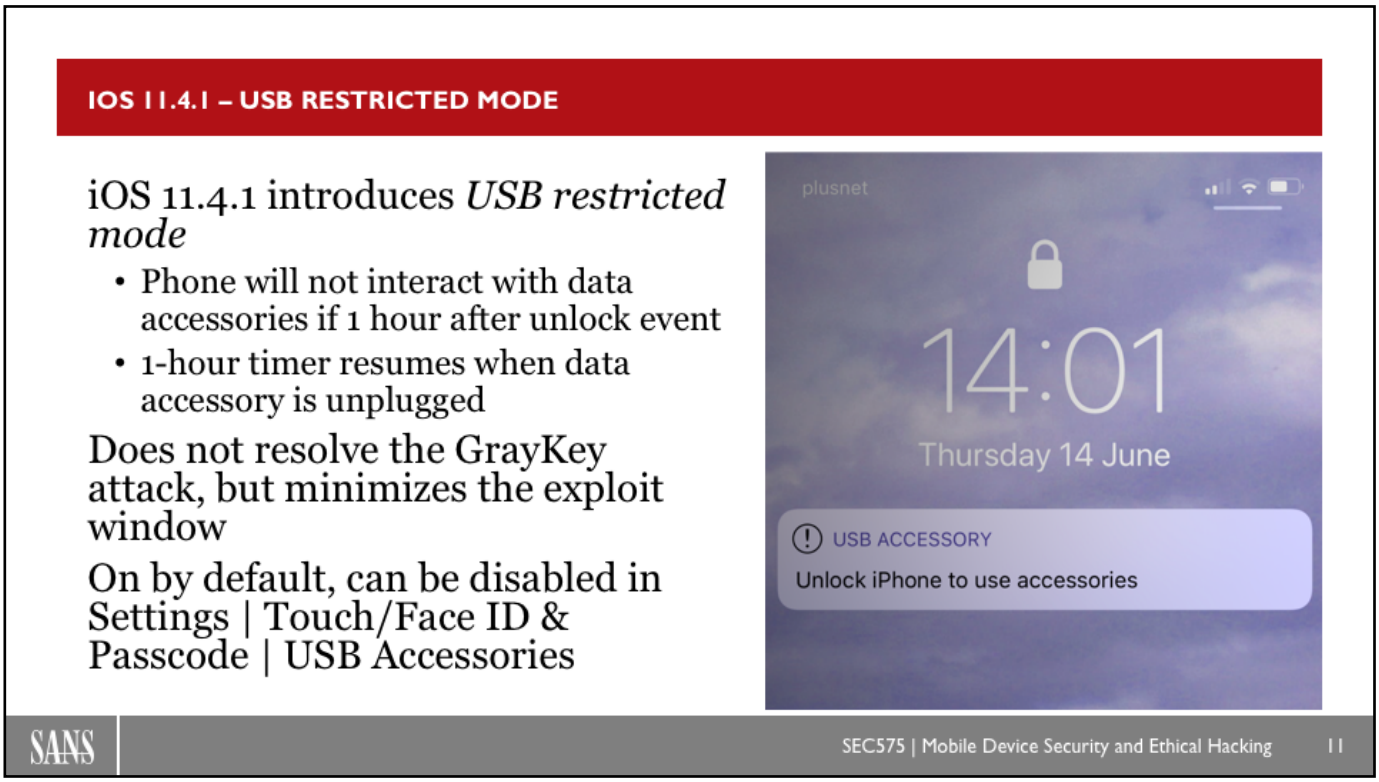


GrayKey Password Guessing

Although the GrayKey device is not widely available for analysis, there is sufficient reporting about the efficacy of the device to draw some conclusions about how it works. Analysis indicates that the GrayKey uses an undisclosed vulnerability in the USB data interface of iOS devices to disable the Secure Enclave Processor (SEP) password guess throttling capability (the feature in iOS that allows you to make 6 incorrect guesses before a delay of 1 minute; a 7th guess introduces a 5-minute delay, an 8th incorrect guess introduces a 15-minute delay, and the 9th incorrect guess introduces a 60-minute delay before a data wipe). Even with the password guess throttling, there is a delay for each guess on the SEP itself, preventing the GrayKey from instantly recovering the password.

Analysis by Matthew Green of Johns Hopkins University indicates that the GrayKey can recover a 4-digit PIN in approximately 6.5 minutes (which seems to align with public reports of demonstrations of the GrayKey device where a password was recovered after *several minutes*). A 6-digit PIN is recovered in 11.1 hours on average, and an 8-digit PIN is recovered after 46 days on average.

A 10-digit PIN seems impractical (4629 day average), and a custom alphanumeric password will vary wildly. Using the same password-guessing statistics (approximately 128 guesses/second), a password in the rockyou.txt file of 14.3 million words could be recovered in 16 hours on average, though a complex password of sufficient length would likely evade password-cracking attempts at this rate.



iOS 11.4.1 – USB Restricted Mode

In iOS 11.4.1, released on July 9, 2018, Apple introduced the USB restricted mode feature. This feature, turned on by default, causes an iOS device to not interact with data accessories (including PCs, Macs, or anything else using data USB pins; charging is not affected) if the device has not been unlocked within 1 hour. If you plug in a data peripheral and the iOS device has not been unlocked, it will display the error "Unlock iPhone to use accessories," as shown on this page (https://9to5mac.com/2018/07/09/ios-11-4-1-unlock-iphone-to-use-accessories-explained/). The 1-hour timer restarts when the data accessory has been unplugged from a locked device.

With this feature, an iOS device can still be exploited using the GrayKey, but it minimizes the opportunity for a threat actor to exploit the device. If the GrayKey user receives a device that is locked, and has not been unlocked within an hour, then the iOS device will not interact with the GrayKey device, precluding the opportunity to exploit the platform.

Apple indicates that some third-party devices may not interact well with the USB restricted mode feature, and offers users the ability to disable this capability altogether (https://support.apple.com/en-us/HT208857).

"BYPASS" USB RESTRICTED MODE

Any connected data device will keep the USB restricted clock timer from restarting

Solution: plug in any data accessory after seizing iOS device

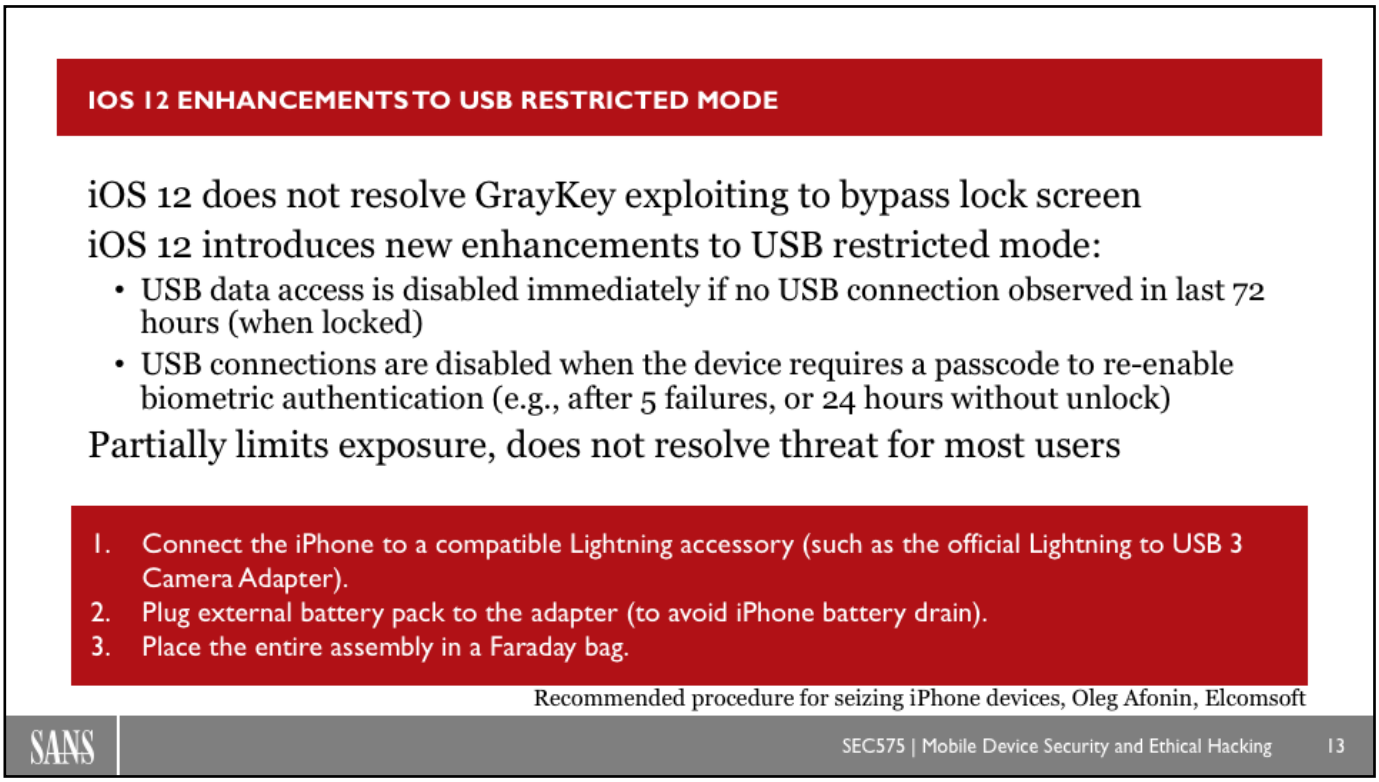
- Presumably, the user has unlocked it within the last hour
- Plugging in a data accessory restarts the USB restricted timer
- Keeping the device plugged in keeps the counter from resuming
- Allows *threat actor* leisure to crack passcode at a later time

Apple Lightning to USB 3 Camera Adapter, $39

"Bypass" USB Restricted Mode

As first reported by ElcomSoft (https://blog.elcomsoft.com/2018/07/usb-restricted-mode-inside-out/), any USB data device that connects to an iOS device will reset the USB restricted mode clock timer. This presents an opportunity for a threat actor (e.g., law enforcement agencies or anyone with access to a GrayKey) to overcome the USB restricted mode defense:

1. The threat actor seizes the target device; presumably, the user has unlocked the device within the last hour.
2. The threat actor plugs in any USB data device, such as the Apple Lightning to USB 3 Camera Adapter (shown on this page); this device is convenient because it can also power the device.
3. Keeping the adapter plugged in keeps the USB restricted mode clock timer from restarting, allowing the threat actor time to relocate the target device to the GrayKey device.
4. The attacker cracks the passcode as their leisure.



iOS 12 Enhancements to USB Restricted Mode

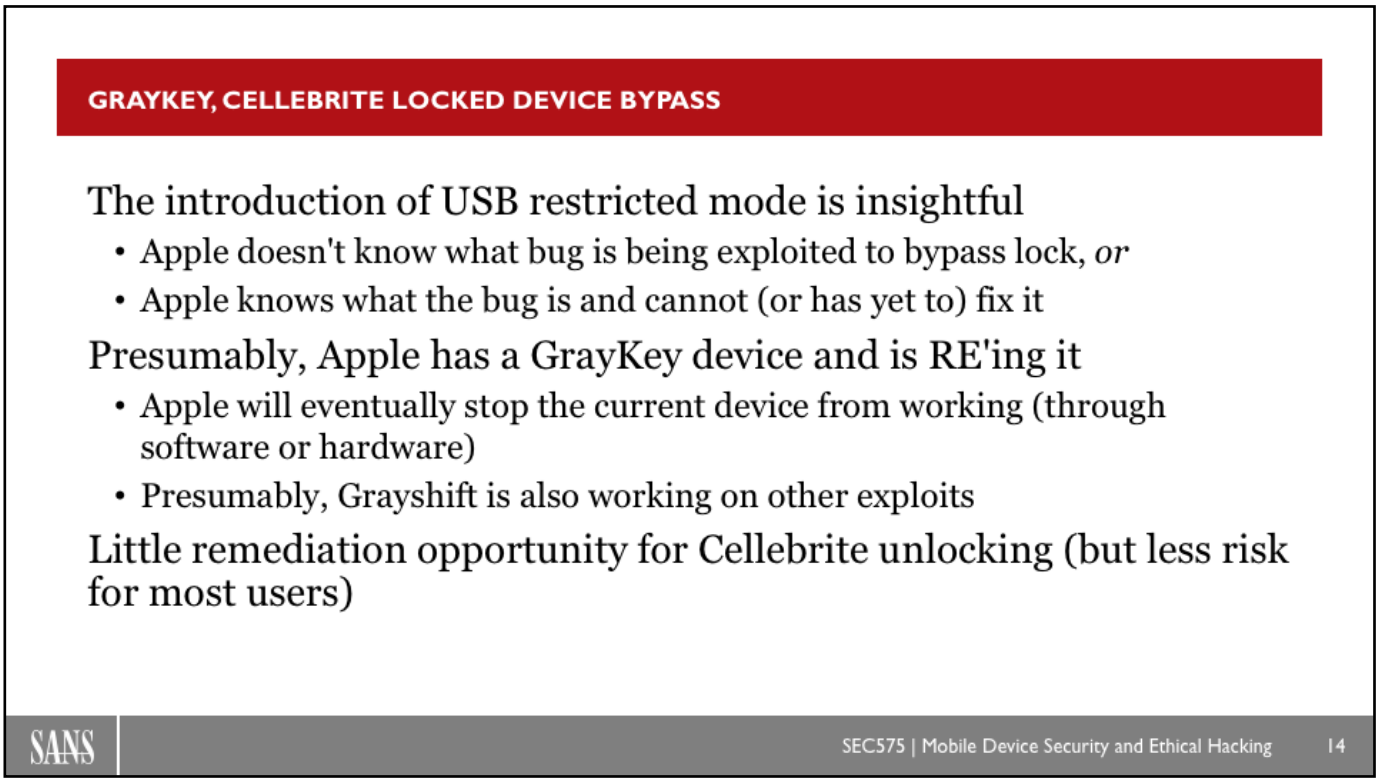
With iOS 12, Apple has not yet resolved the vulnerability exploited by the GrayKey device. Instead, Apple continues to introduce new enhancements to the USB restricted mode feature to attempt to thwart these attacks. According to the updated iOS Security white paper (https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf), iOS 12 carries forward the previous USB restricted mode features (phone will not interact with data accessories if 1 hour after unlock event; 1-hour timer resumes when data accessory is unplugged) and introduces two new additional enhancements:

- "… on iOS 12 if it's been more than three days since a USB connection has been established, the device will disallow new USB connections immediately after it locks" (iOS Security, page 12).
- "USB connections are also disabled whenever the device is in a state where it requires a passcode to re-enable biometric authentication" (iOS Security, page 12).

These new restrictions will make it harder for an adversary to gain access to USB data channels on a locked device, but does not prevent it. Users that charge their iOS device on a computer USB connection (as opposed to the generic USB power adapter) will not benefit from the new 3-day/72-hour restriction since their devices are regularly connected to a *data peripheral*. Some third-party charging appliances will also appear to be a data peripheral, potentially removing the added restrictive capability to thwart GrayKey attacks.

What isn't clear is if a data peripheral that is *not trusted* resets the 72-hour restriction for USB devices. Additional testing is required to fully evaluate this new feature.

On the ElcomSoft blog, Oleg Afonin recommends a procedure for GrayKey users to take advantage of a device where possible, continuing to use a data peripheral to reset the timer for USB restricted mode engagement (https://blog.elcomsoft.com/2018/09/ios-12-enhances-usb-restricted-mode). It will be interesting to see if Apple continues to enhance USB restricted mode to further thwart GrayKey and other USB data attack devices.

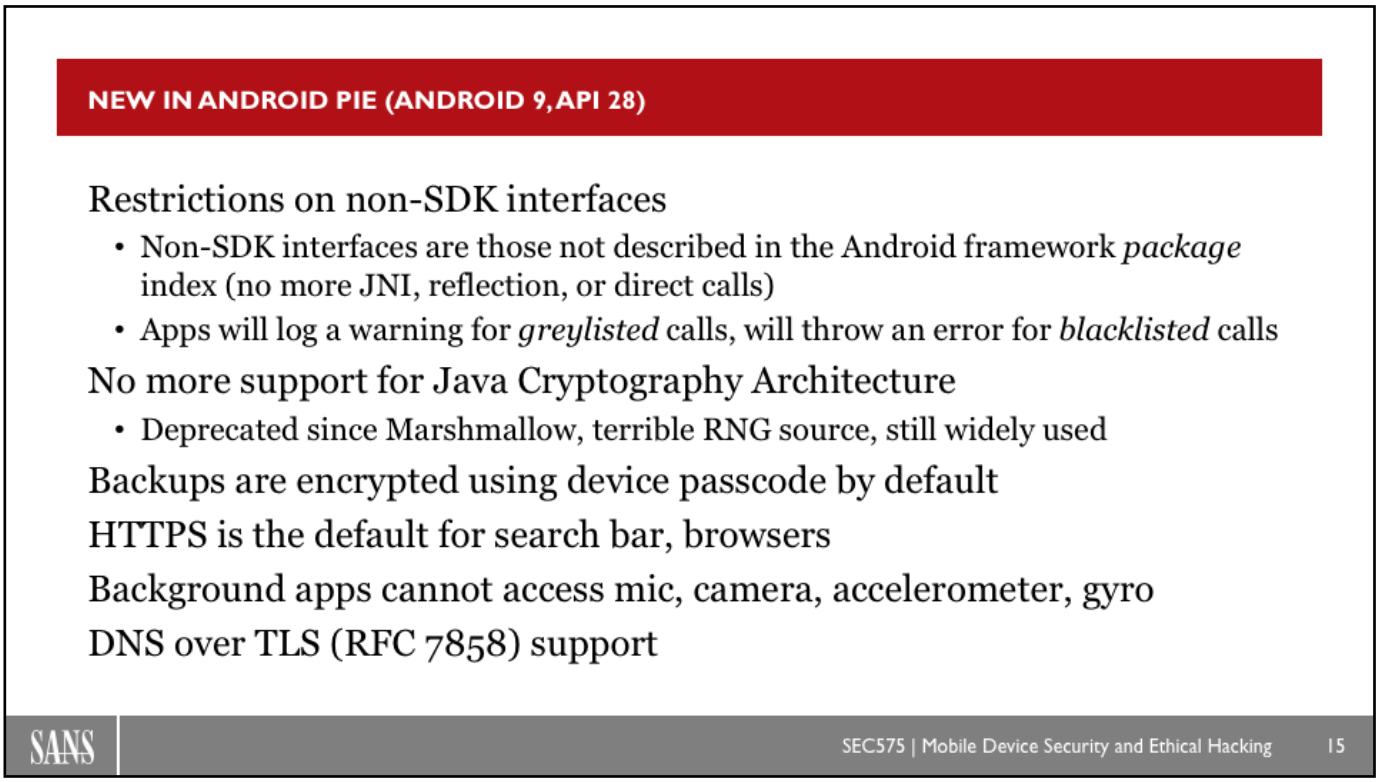


GrayKey, Cellebrite Locked Device Bypass

Apple has not responded to requests for comment on the GrayKey device, but the introduction of the USB restricted mode feature is insightful. Apple has yet to resolve the vulnerability exploited by the GrayKey device, instead introducing a general workaround (with its own set of limitations), indicating that Apple does not yet know what bug is being exploited by Grayshift or knows what the bug is but cannot (or has yet to) fix it.

It seems reasonable to imagine that Apple has a GrayKey device in their possession and is attempting to reverse engineer (RE) it to identify the nature of the flaw being exploited. It also seems reasonable that Apple will work to resolve the vulnerability, either through a software update or a hardware replacement (e.g. the next iPhone model, if the vulnerability is related to a hardware flaw that cannot be resolved through a software update). However, one would assume that Grayshift's commitment to continued support for later iOS versions would indicate that they also continue to identify opportunities to exploit iOS devices with additional updates.

The Grayshift model of an on-premises lock screen bypass device makes the technology more accessible to would-be threat actors (including law enforcement agencies, but also people who surreptitiously obtain a GrayKey device through other means), but it also creates an opportunity for Apple to evaluate the exploit(s) in use to resolve them. The Cellebrite on-premises data recovery model is perhaps less of a threat to a wide audience of iOS users, but will also likely evade Apple's ability to identify and resolve the exploits they have developed.



New in Android Pie (Android 9, API 28)

On August 6th 2018, Google released Android Pie (Android 9, API 28). Like prior releases of the Android platform, this version introduces important new security features.

Restrictions on Non-SDK Interfaces: For the first time, Google is imposing restrictions on what parts of the Java framework are available to developers. Prior to Android Pie, developers could use non-SDK interfaces in the Java Native Interface (JNI), through Java *reflection* (a technique where Java makes it possible to inspect, access, and execute code without knowing the object names) or through undocumented (but otherwise exposed) Android SDK calls. This level of access provided flexibility to developers but also exposed sensitive interfaces that could threaten security or otherwise mar the intended Android end-user experience approved by Google. In Android Pie, Google has started *blacklisting* specific calls that are explicitly forbidden, while logging a warning for any other calls not specifically permitted (e.g., "greylisted" calls). This is a change that is more inline with the practices adopted by Apple iOS many years ago to thwart SDK misuse and regain consistency in end-user application experiences, though the actual enforcement of this type of filtering is notoriously difficult. Additional information is available at https://developer.android.com/about/versions/pie/restrictions-non-sdk-interfaces.

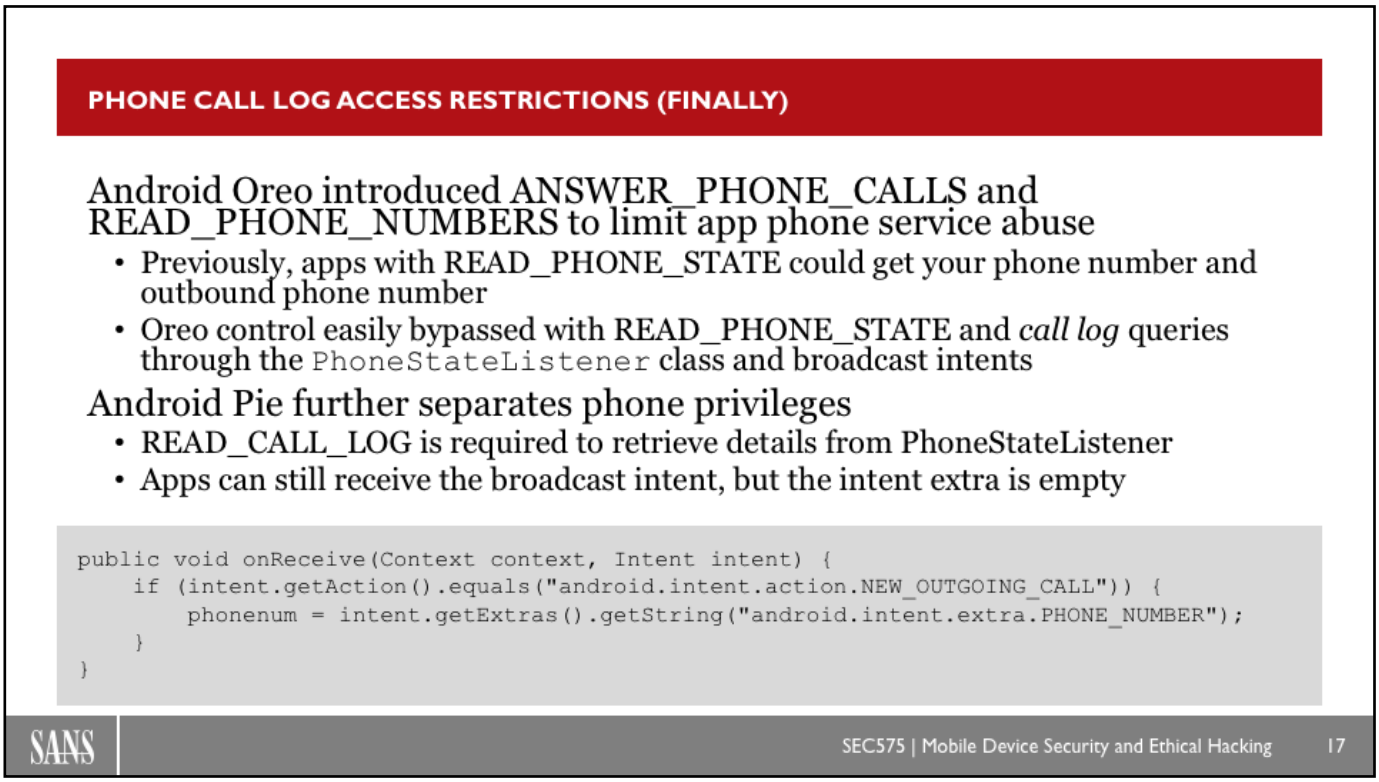
Discontinued Java Cryptography Architecture Support: Although Java Cryptography Architecture (JCA) has been deprecated for several years, Android Pie is the first release to remove support for the JCA in favor of native Android cryptography controls. The JCA is well-known for cryptographic weaknesses, particularly for the selection of random numbers.

Encrypted Backups by Default: Although encrypted backups have been supported in Android for many years, encryption was optional. In Android Pie, encryption for Android backups is turned on by default (but can be disabled).

HTTPS Default for Browsers, Search: Prior to Android Pie, requests that omitted a URI prefix in the search bar, Google Chrome, or native Android browser defaulted to HTTP. With Android Pie, these requests default to HTTPS.

Background App Access Restrictions: Android Pie adds new restrictions to prevent apps operating in the background from accessing telemetry information and other peripherals: microphone, camera, accelerometer, and gyroscope. This is advantageous to preserve battery life from misbehaving applications, but also limits a user's exposure to eavesdropping attacks to the currently active application.

DNS over TLS: Finally, Android Pie adds DSN over TLS support specified in RFC 7858. DNS remains a problematic, insecure protocol, and the adoption of TLS integration helps defend against many attacks. Full use of DNS over TLS will take many years as network providers transition to add DNS over TLS support.

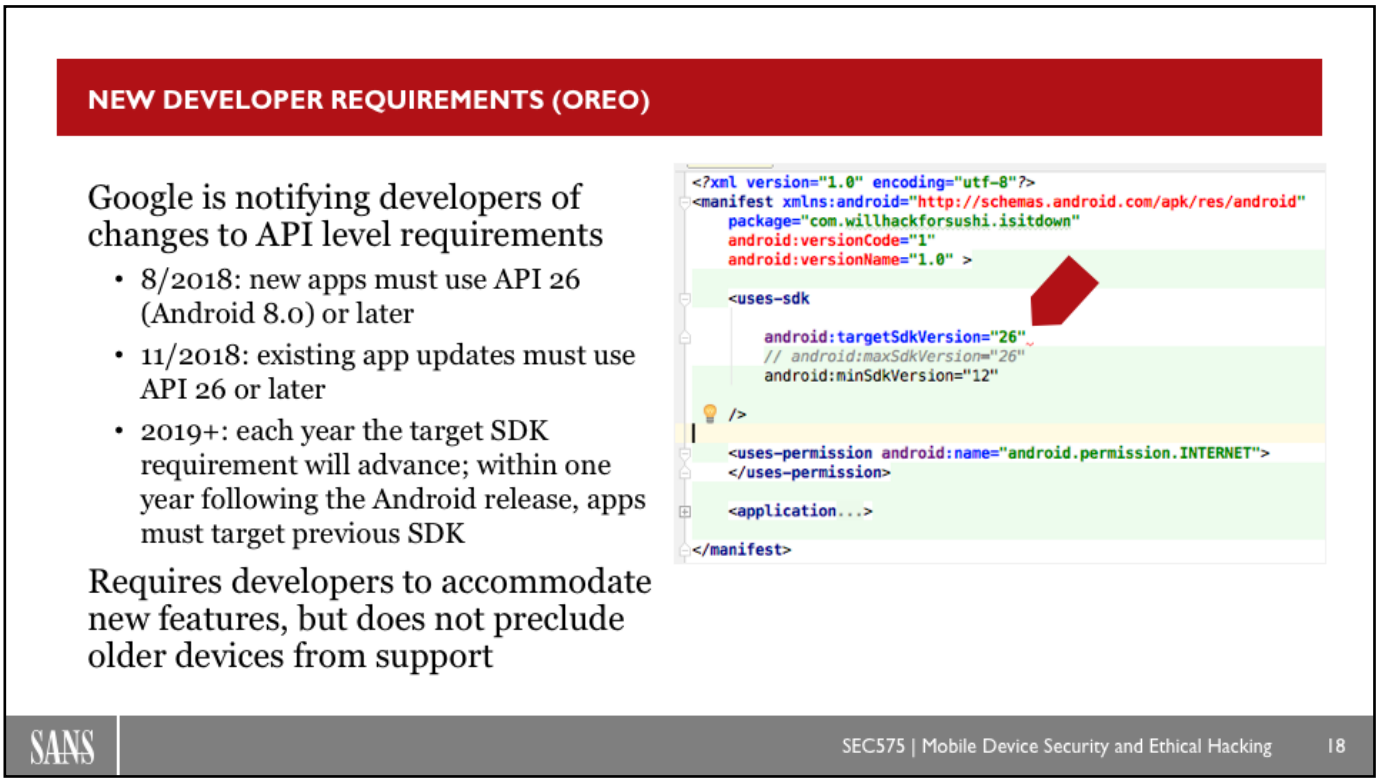


Phone Call Log Access Restrictions (Finally)

For many years, telephone-related permission management has been relegated to the `READ_PHONE_STATE` privilege. This privilege gives an app the ability to determine whether a user is actively on a call, and is widely adopted by many applications to avoid interrupting a user when they are speaking on a call. Unfortunately, `READ_PHONE_STATE` also gives the app author a lot of other permissions as well, including the ability to identify your device phone number and the caller's phone number.

Although Android Oreo introduced new granular controls to limit this widespread information leak in Android apps with two new permissions—`ANSWER_PHONE_CALLS` and `READ_PHONE_NUMBERS`—they were easily bypassed. Instead of using the intended `getLine1Number()` function, developers can obtain phone number information by listening for the `PhoneStateListener` broadcast intent using the source code sample shown on this page.

With Android Pie, additional granular permission controls are applied to limit access to phone number information. The `READ_CALL_LOG` is a new permission required to gain access to the `PhoneStateListener` details. Apps can still register to receive broadcast intents from the `PhoneStateListener`, but the phone number field will be empty if they lack the READ_CALL_LOG permission.



New Developer Requirements

When developers write an application, they designate the API version that they comply with, dictating the features available in the application (such as modern dangerous or normal permissions). Developers can choose older versions of the APIs and still be compliant with newer platforms, albeit with limited, backward-compatible features.

Coinciding with the release of Android Oreo, Google is notifying developers that apps must comply with newer APIs or risk being pulled from the Google Play store. According to Google, this new rule will be applied in three phases:

1. In August 2018, new apps must use API 26 (Android 8.0) or later to be approved for publishing in the Google Play store.
2. In November 2018, existing apps must use API 26 or later to be approved for updates distributed from the Google Play store.
3. In 2019 and later, each year the target SDK requirement for new and updated apps will increase within one year following the Android dessert release; apps must comply with the target SDK version for new releases or updates (https://android-developers.googleblog.com/2017/12/improving-app-security-and-performance.html).

These changes won't preclude support for older devices, since a developer can choose to support the latest SDK but also maintain backward compatibility in their app as well.

In addition to SDK version compliance, as of August 2019, developers must include 64-bit versions of native libraries as well (32-bit devices can still be supported if the developer chooses to distribute both 32-bit and 64-bit libraries). This is largely a performance benefit for users with newer devices, since many native libraries

today only support 32-bit processors and do not take advantage of the full capabilities of modern processors.

These changes introduced with Android Oreo continue with Android Pie. Developers should note that new applications and updates to previously published applications must comply with these updated API version policies.

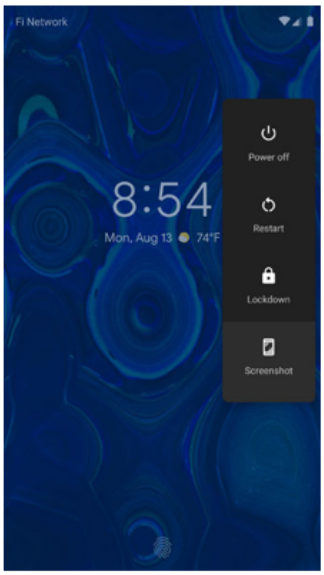
LOCKDOWN MODE

Android's answer to iOS Emergency SOS

Off by default: Settings | Security | Lock screen preferences | Show lockdown option

Press and hold the power button for 2 seconds

- Disables all biometric authentication
- Disables Smart Lock functionality
- Hides all notifications

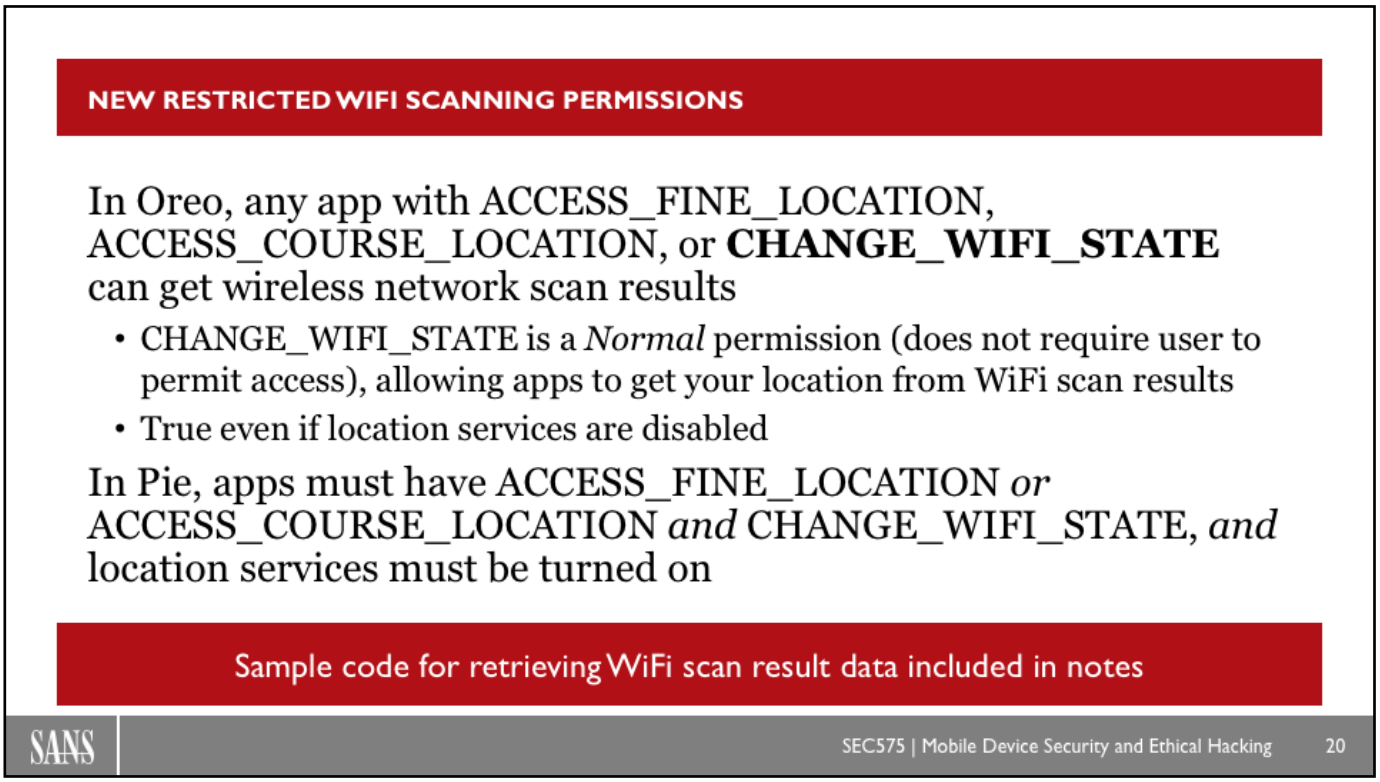


Lockdown Mode

On the heels of the iOS Emergency SOS feature, Android Pie introduces Lockdown Mode. Although off by default, Lockdown Mode (when turned on) allows a user to quickly disable all biometric authentication, disable Smart Lock functionality, and hide all notifications by pressing and holding the power button for 2 seconds.

Users with Android Pie can turn on Lockdown Mode by navigating to Settings | Security | Lock screen preferences | Show lockdown option.

Photo from ComputerWorld/IDG magazine: https://www.computerworld.com/article/3297039/android/android-pie-security-setting.html.



New Restricted WiFi Scanning Permissions

Android Pie resolves a vulnerability in the location services functionality on Android devices that allows a malicious app to identify your location even when location services are turned off. The typical security model for Android devices to be able to perform geolocating using WiFi scan results is to have the ACCESS_COURSE_LOCATION or ACCESS_FINE_LOCATION privilege, and location services must be turned on. However, for Android Oreo and prior devices, any app with CHANGE_WIFI_STATE can also obtain WiFi scan search results, allowing the attacker to get the same effective location services as ACCESS_COURSE_LOCATION. What's more, apps with CHANGE_WIFI_STATE can identify your location even when location services are turned off.

With Android Pie, apps' location services must be turned on for geolocating with WiFi to work, and apps must have CHANGE_WIFI_STATE in addition to ACCESS_FINE_LOCATION or ACCESS_COURSE_LOCATION.

The sample code shown here can be used to demonstrate this vulnerability on Android Oreo devices and earlier, obtaining WiFi scan search results and logging SSID, BSSID, and RSSI information.

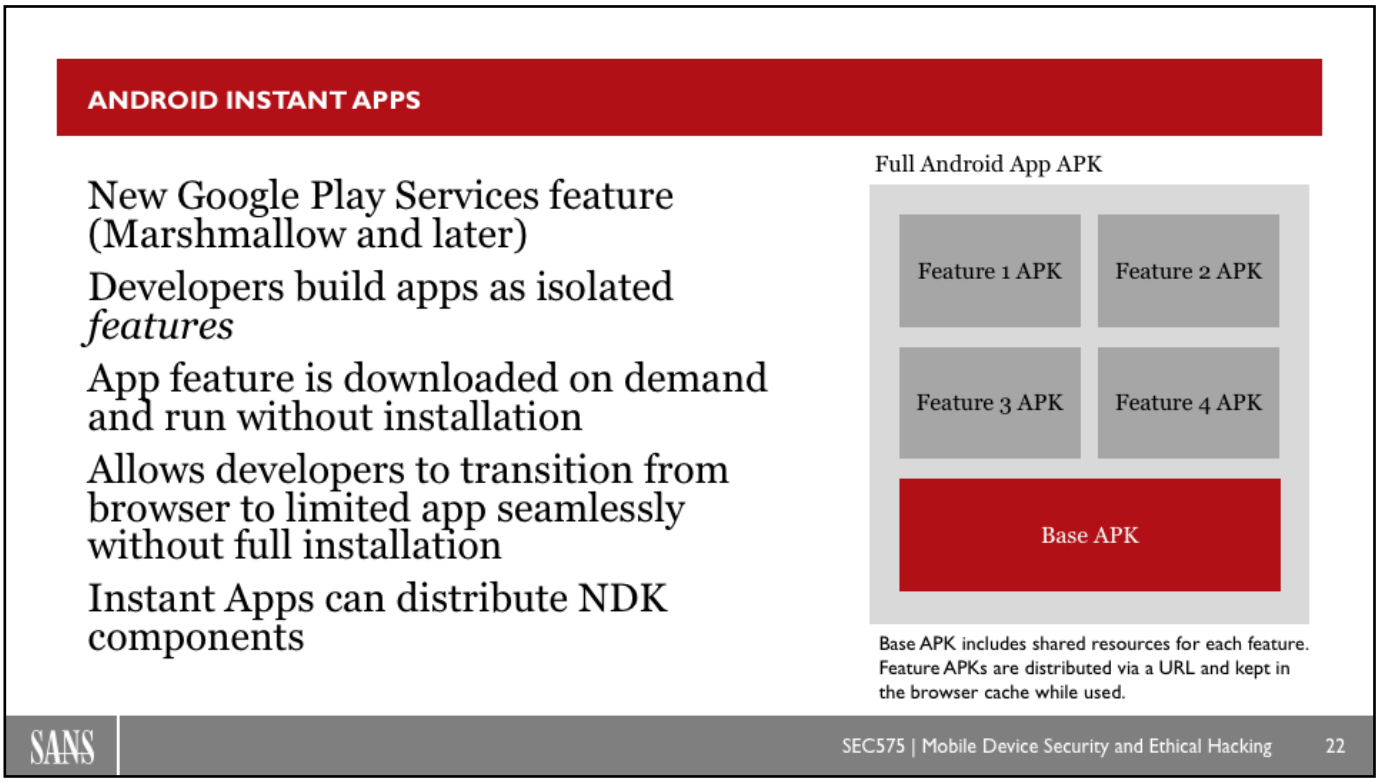
```
// Adapted from https://gist.github.com/granoeste/1026558 by Katsumi Onishi
// For Android Oreo and earlier, requires CHANGE_WIFI_STATE but no other location-
// based privilege. Ignores the disables state of location services.

final WifiManager mWifiManager = (WifiManager) getSystemService(WIFI_SERVICE);
IntentFilter filter = new IntentFilter();
filter.addAction(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION);

registerReceiver(new BroadcastReceiver(){
    @Override
    public void onReceive(Context context, Intent intent) {

        List<ScanResult> results = mWifiManager.getScanResults();
        final int N = results.size();

        Log.v(TAG, "Wi-Fi Scan Results ... Count:" + N);
        for(int i=0; i < N; ++i) {
            Log.v(TAG, "  BSSID       =" + results.get(i).BSSID);
            Log.v(TAG, "  SSID        =" + results.get(i).SSID);
            Log.v(TAG, "  Level       =" + results.get(i).level);
        }
    }
}, filter);
mWifiManager.startScan()
```

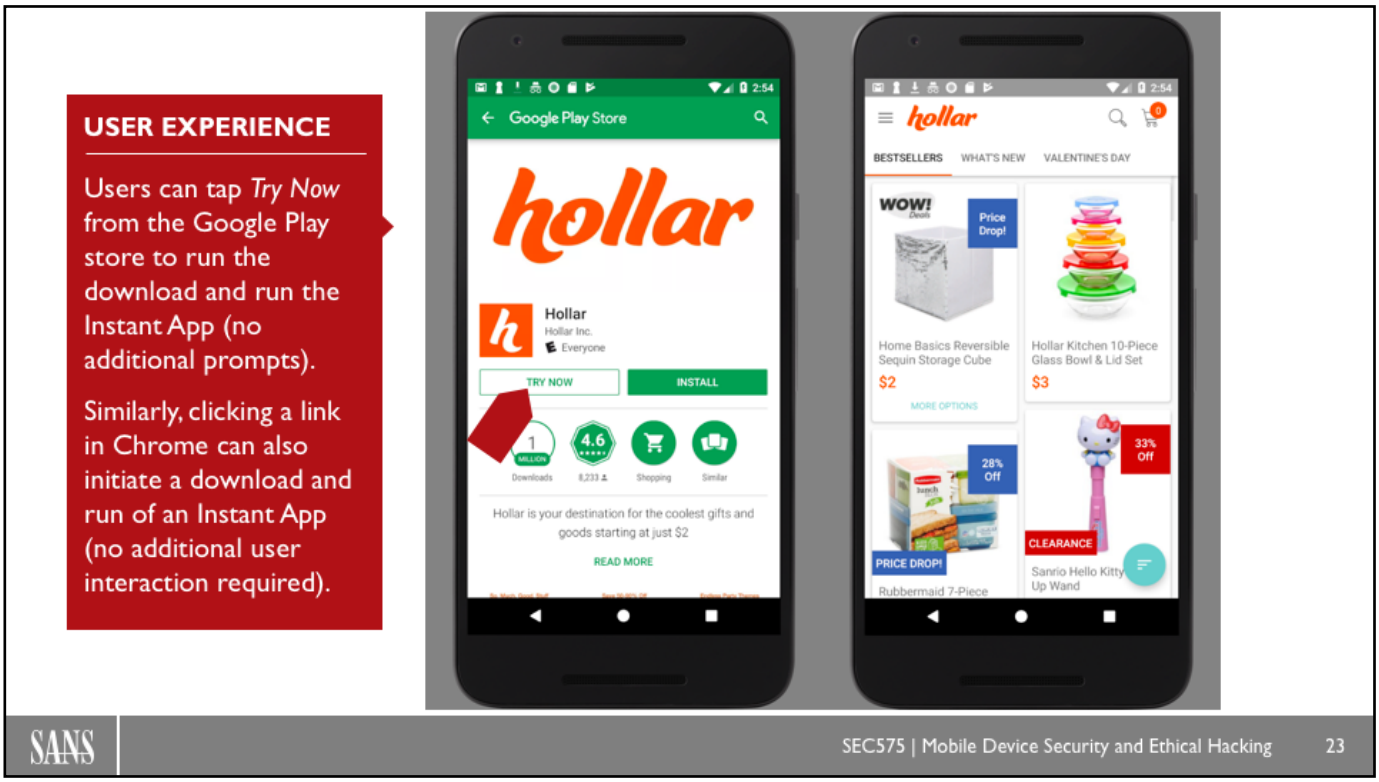


Android Instant Apps

With Android Oreo, Instant App features are moving to the base Android platform, whereas they were previously available in the Google Play Services of Android Marshmallow and Android Nougat. With Instant Apps, developers build their apps as isolated features with a single base APK, distributed as modules over HTTPS. Instead of a website or other service requiring the user to navigate to the Google Play store to download their app, an Instant App is installed immediately after the user click a link (without additional confirmation from the user), allowing for a more seamless experience—from web browsing to interacting with a merchant or other service provider—and a richer app experience.

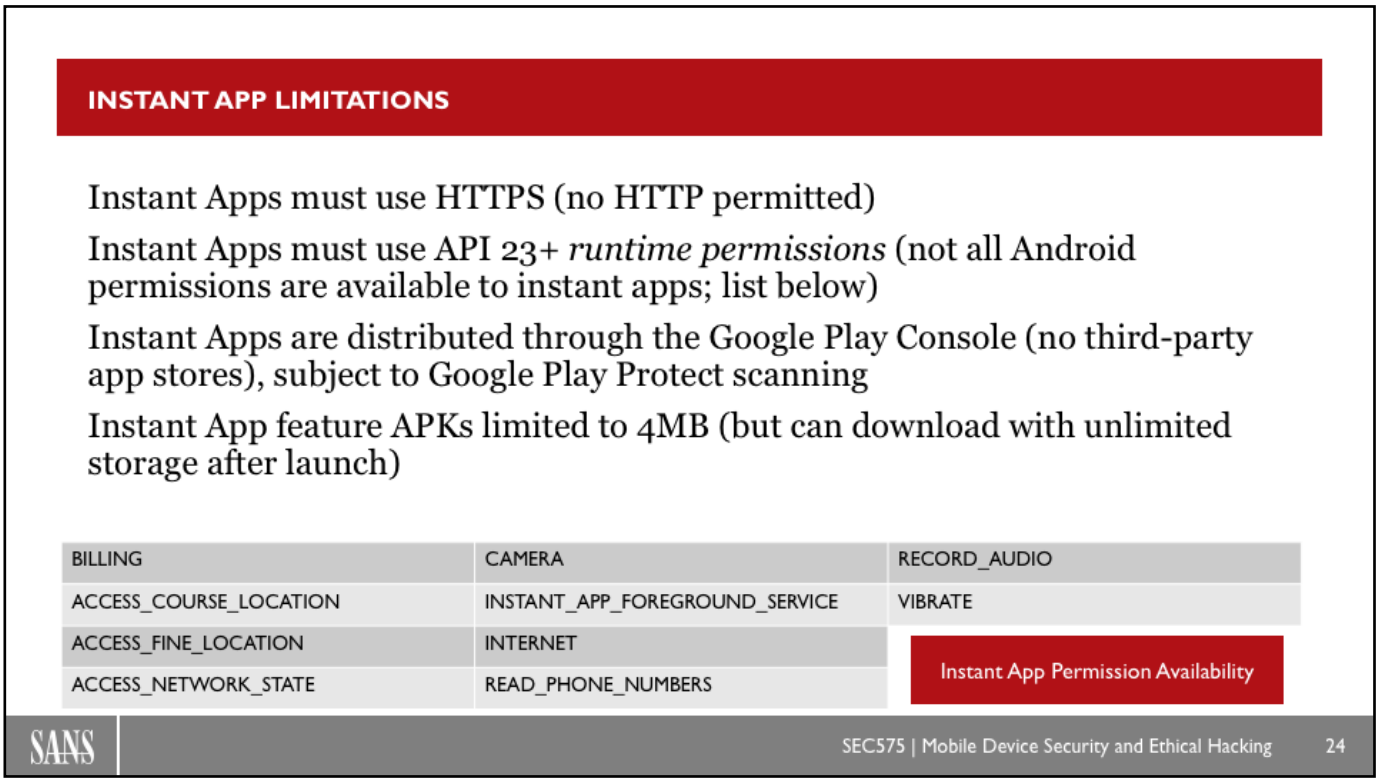
Instant Apps aren't installed but are run on-demand from the browser cache. After launch, the Instant App delivers the base APK and the necessary features in a minimal download, allowing the user to quickly continue interacting with their device.

Instant Apps have several limitations that we'll examine shortly, but one key benefit of an Instant App is access to the Android Native Development Kit. An Instant App can run code from arbitrary libraries distributed with the Instant App, subject to the permission limitations in the operating system.



User Experience

Instant Apps are delivered with any standard hyperlink over a secure HTTPS transport. When users browse the Google Play store, developers supporting the Instant App format can deliver the app as a *TRY NOW* link (shown on this page) or as a full app installation. The Instant App does not look any different than a standard Android app, except that it may have less functionality than the full app installation.

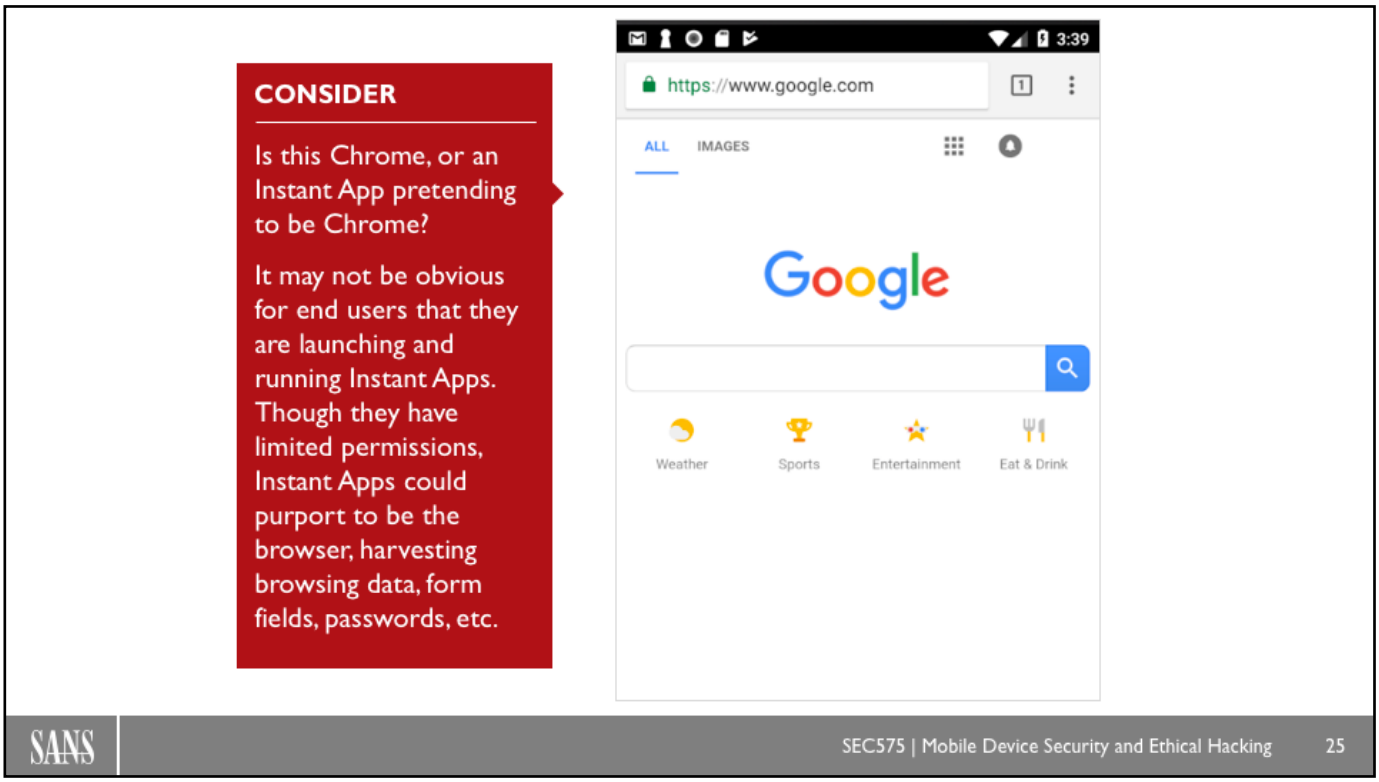


Instant App Limitations

Unlike full app installs, Instant Apps have several important limitations:

- *HTTPS Distribution*: Instant Apps must be distributed over HTTPS and cannot be distributed over HTTP.
- *API 23+ Runtime Permissions*: Instant Apps must comply with API 23+ runtime permissions (using the normal or dangerous notation with modal dialogs prompting users to permit or deny on use).
- *Limited Permissions*: Instant Apps only have access to a limited number of permissions, listed on this page. The BILLING, VIBRATE, INTERNET, and ACCESS_NETWORK_STATE permissions are all considered normal (and do not require explicit user permission). The remaining permissions are in the dangerous group, requiring explicit user permission before the Instant App can access the privilege.
- *Google Play Console Distribution*: Instant Apps are limited to distribution through the Google Play Console and cannot be distributed through third-party app stores. As part of the Google Play Console, the apps are subject to Google Play Protect scanning.
- *Limited Download Size*: Instant Apps are limited to 4MB in size for the initial load of the app. After the Instant App launches, it can download unlimited additional data.

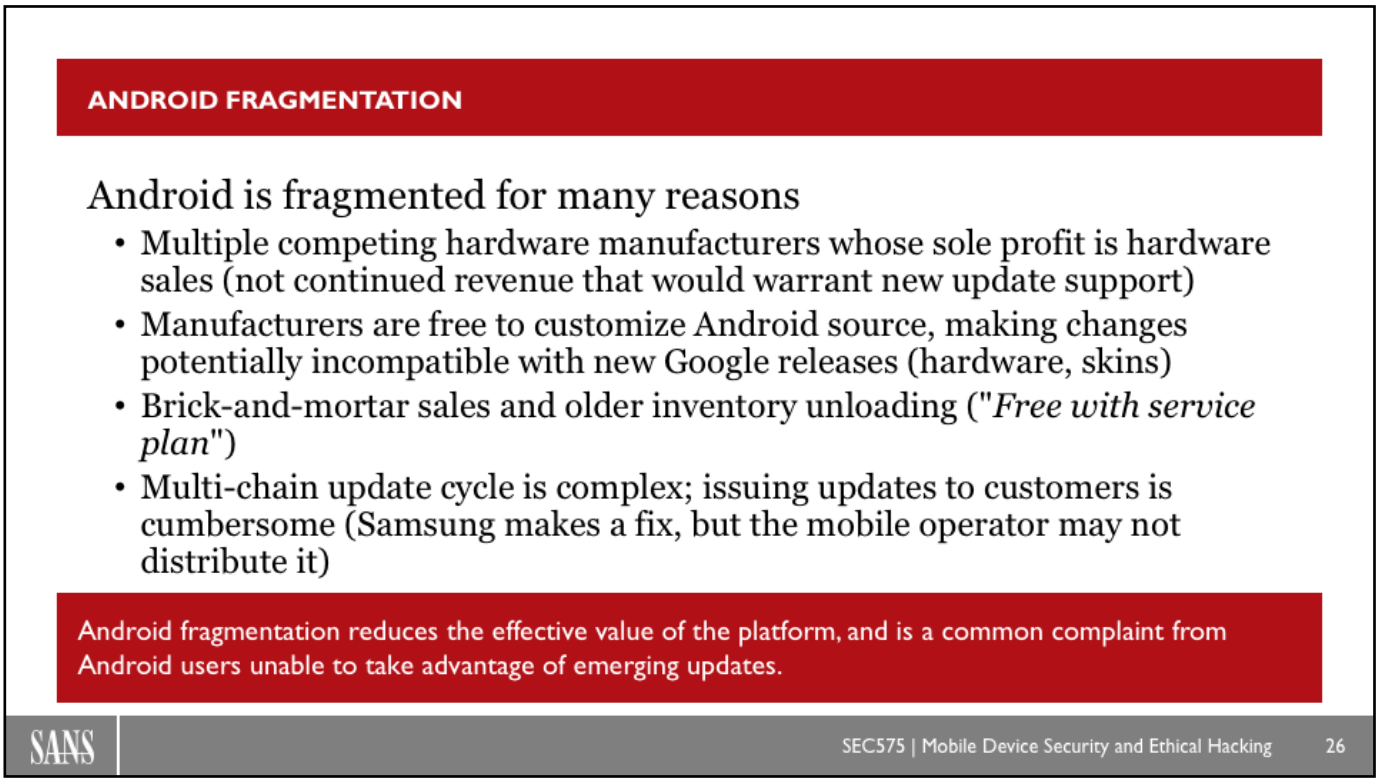
Additional information on policies for Instant Apps and frequently asked questions about the Instant App feature are available at https://developer.android.com/topic/instant-apps/policy.html and https://developer.android.com/topic/instant-apps/faqs.html.



App Impersonation

Consider the screenshot shown on this page. Is this app Chrome, or is it an Instant App pretending to be Chrome?

For end users, the transition from a browser to an Instant App is only obvious in that the Instant App looks visibly different from the browser, and a brief animation plays while the Instant App is downloaded. An attacker who wishes to exploit Instant App access could write an Instant App that looks just like Chrome. If the user clicks a link that delivers the Instant App, it would launch immediately, replacing the Chrome interface with one developed by a third party designed to capture browsing history and keystrokes.

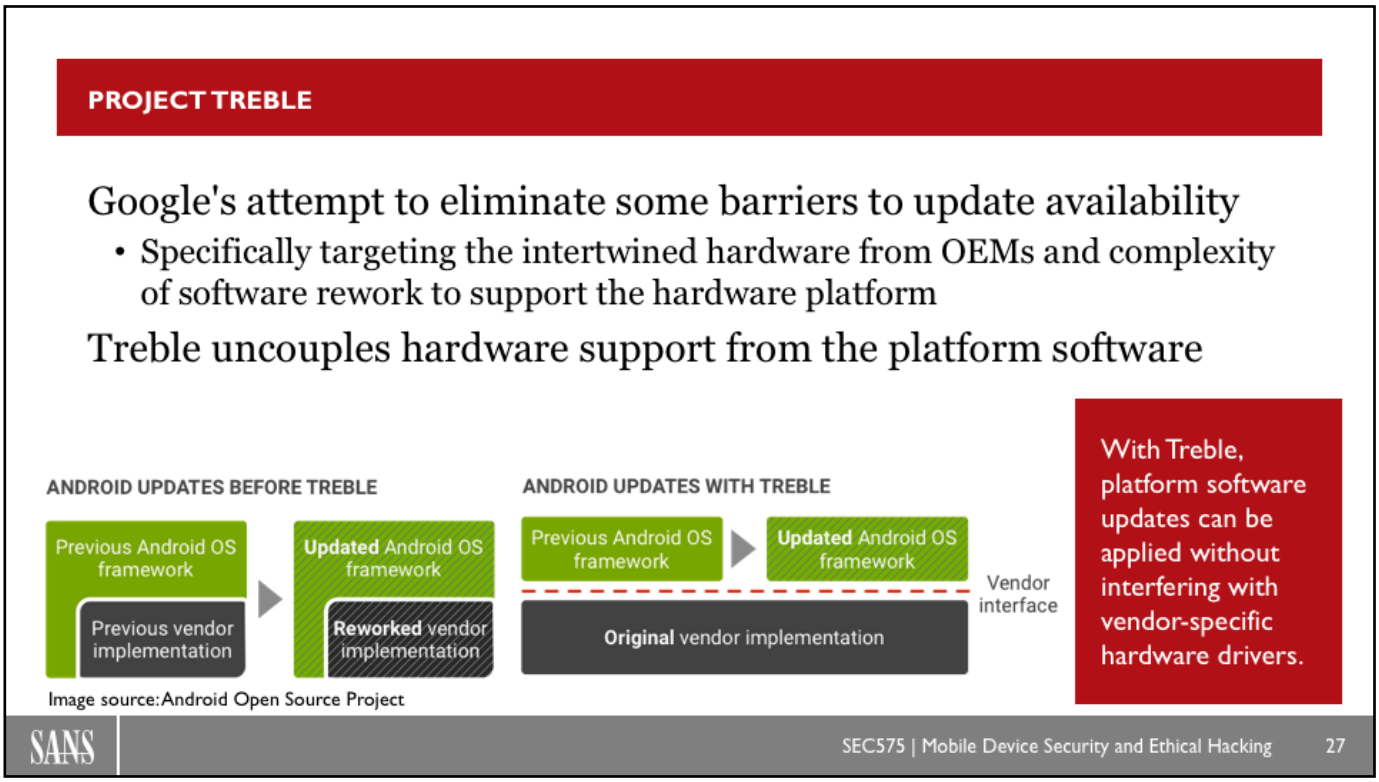


Android Fragmentation

We speak often about the fragmented state of Android, but it is useful to articulate *why* Android is fragmented as a platform. Consider the following factors:

- Multiple OEMs compete for device sales to make money on the Android platform. Handset sales is the only avenue (or primary avenue for most OEMs) to make money on products. As a result, continued device sales, not software maintenance on their existing device sales, is their primary revenue target.
- Manufacturers are free to customize the Android source code and take advantage of this feature to provide competitive differentiators with other handset manufacturers. This is often realized with custom Android skins, which make the platform look different from their competitors, and custom hardware support (such as iris scanning for biometric authentication or changeable hardware add-ons).
- Brick-and-mortar sales are widely distributed, with many stores selling older inventory at discounted rates (commonly, Android devices are "free with service plan" but are typically not the newest Android handsets available).
- The multi-chain update cycle is complex; issuing updates to customers is cumbersome for all parties, where Google may make an update to the platform and Samsung may subsequently integrate the update for its hardware platform, but the mobile operator may choose not to make the update available to end users.

From a security perspective, Android fragmentation reduces the effective value of the platform, often preventing customers from obtaining minor and major release updates that address security flaws. Further, the lack of update availability is a common complaint from users unable to take advantage of much-reported emerging update features for 18–24 months after the release is made available.



Project Treble

With the Android Oreo release, Google announced a significant rework of the platform that attempts to eliminate some of the barriers typically associated with update availability known as Project Treble. Project Treble aims to eliminate the complexity of software updates to Android devices by creating an abstract layer between the OEM hardware-specific drivers and supporting software and the Android platform itself. By uncoupling the OEM hardware components from the Android OS, Google is attempting to remove the barrier that prevents users from obtaining and installing platform updates.

The images on this page show a comparison of the integration of vendor hardware implementation and the Android platform before and after Treble. On the left, the image shows how the vendor implementation is an integral component of the Android OS; updates to the Android OS require reworked vendor implementations before the update can be made available to end users. On the right, the vendor implementation is uncoupled from the OS framework, allowing the OS release (updates, new dessert versions, incremental API changes) to be applied independent of the vendor implementation. (Image source: https://source.android.com/devices/architecture/treble.)

TREBLE: HARDWARE ABSTRACTION LAYER

Treble is a HAL for Android

- OEMs write their drivers to satisfy the Treble requirements for a driver using the Treble HIDL ("hid-ell")
- Android platform abstracts hardware-specific driver functionality for generic access
- HALs built by vendors are placed in /vendor

Devices that support Treble are easier to update with patches, new dessert releases

"Project Treble will be coming to all new devices launched with Android O and beyond." Iliyan Malchev, Android Developers Blog

Sony Xperia XZ1


Google Pixel 2

```
generic_x86:/ $ getprop ro.treble.enabled
true
```

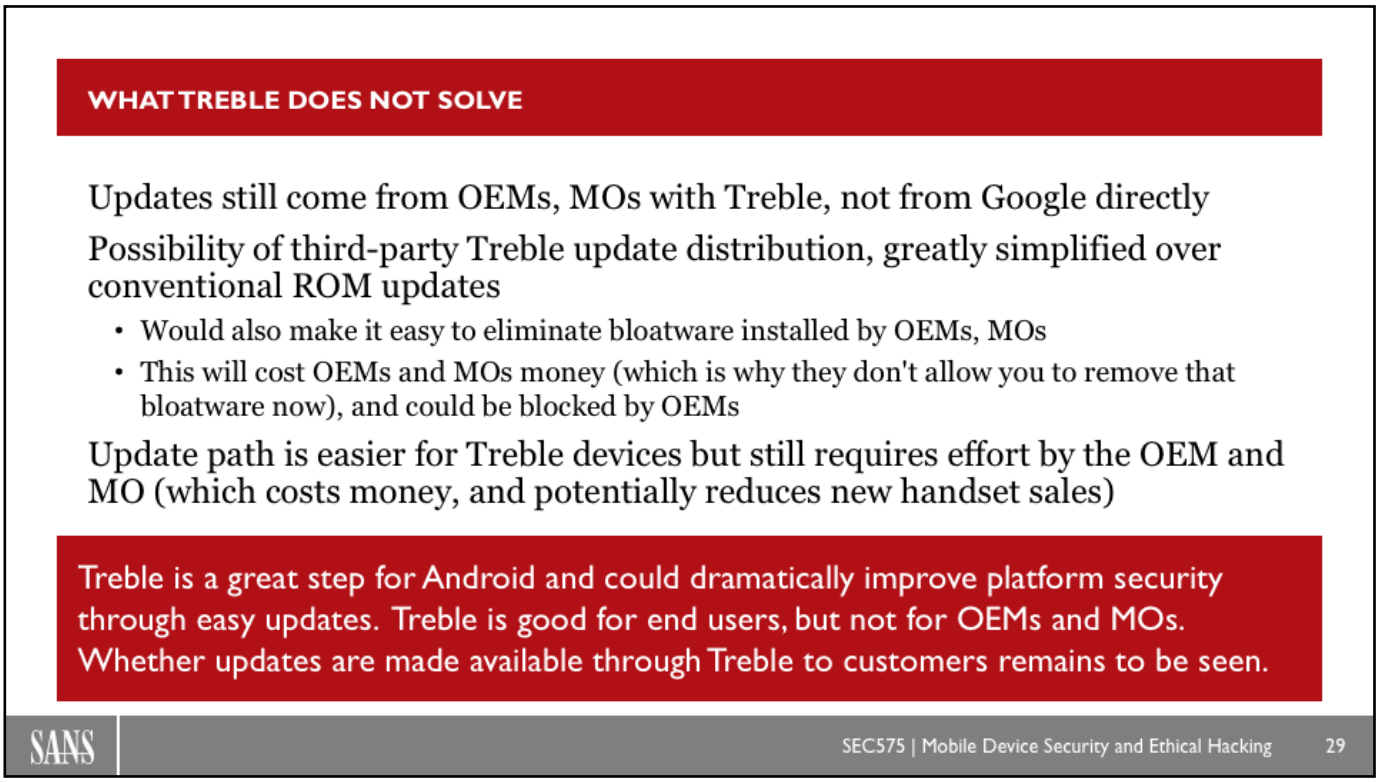

Treble: Hardware Abstraction Layer

Fundamentally, Treble is the implementation of a Hardware Abstraction Layer (HAL) and HAL Interface Description Language (HIDL, pronounced "hid-ell") for Android. OEM developers write their hardware layer to support the hardware using the HIDL conventions specified by Google, and Google provides the HAL that acts as a pass-through for the hardware. The Android platform does not need to be modified for custom OEM hardware with Treble since the HAL can accommodate custom hardware using the OEM drivers written to comply with Treble's conventions and requirements.

When a device supports Treble, the end user is able to replace the OS with new updates without risk of breaking device functionality related to hardware. Google makes support for Treble mandatory for new devices coming out with Android Oreo (but not mandatory for devices updating to Android Oreo from earlier Android dessert releases). At the time of this writing, the Sony Xperia XZ1, the Google Pixel 2, and the HTC U11 Plus are all new devices that support Treble through initial Android Oreo, while several other devices have been updated to Android Oreo and added Treble support (Google Pixel 1/1 XL, Huawei Mate 9 and others, and the Essential Phone PH-1).

To test if your handset has Treble support, first confirm that it runs Android Oreo. Next, using ADB access to the device in debug mode, open a shell and run `getprop ro.treble.enabled`, as shown on this page. If the command returns *true*, it indicates that the device supports Treble.

Quote on this page by https://android-developers.googleblog.com/2017/05/here-comes-treble-modular-base-for.html.



What Treble Does Not Solve

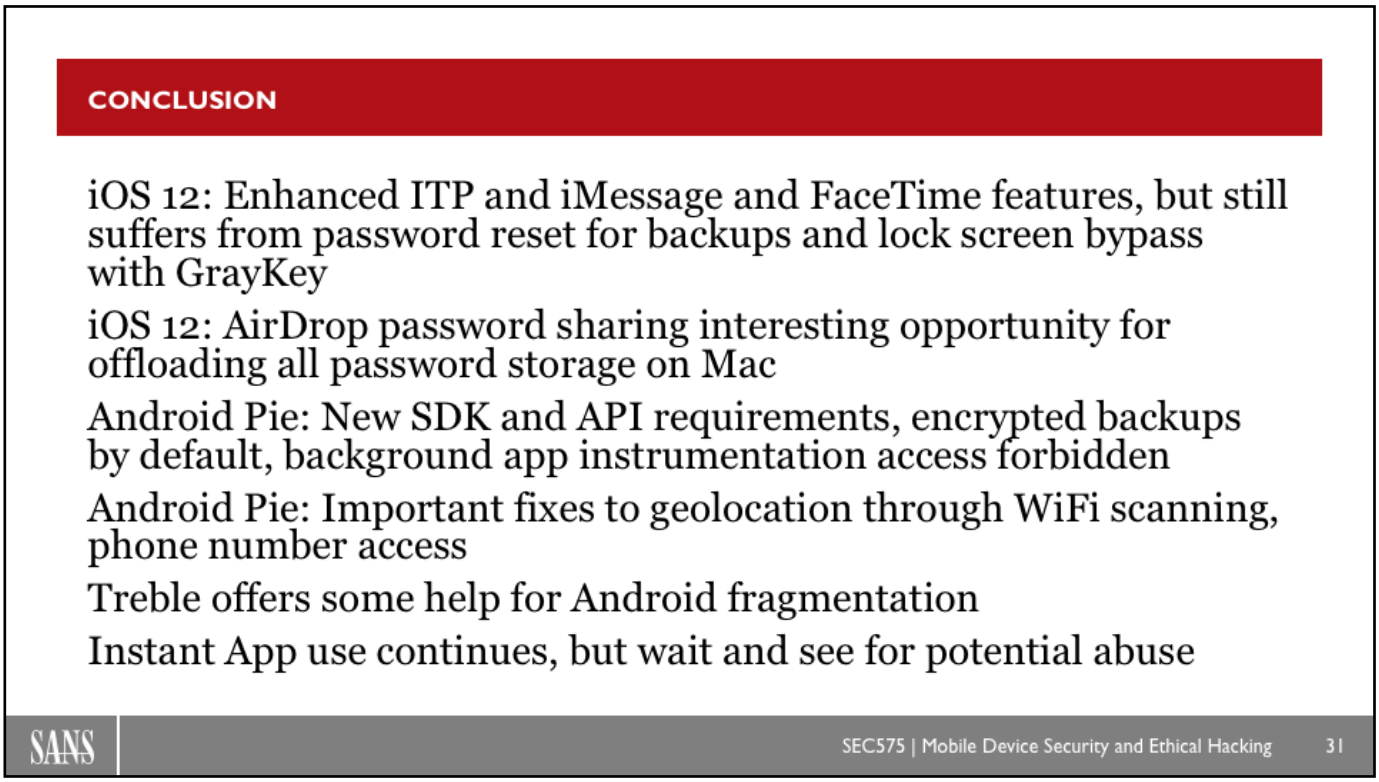
At the time of this writing, Android Oreo and Treble are relatively new. While new handsets are emerging with Android Oreo and Treble support, there are still some considerations regarding whether or not Treble will solve the Android fragmentation problem.

First, updates for handsets with Treble support for end users still come from OEMs (Original Equipment Manufacturers, or handset manufacturers) and MOs (Mobile Operators, such as Verizon, AT&T, etc.), not from Google directly. An OEM or MO can choose whether or not an update is distributed to end users. Since OEMs are motivated to sell new handsets, it is unclear if they will be sufficiently motivated to make updates available to end users, or how quickly updates will be made available with Treble.

It's reasonable to expect third-party Treble updates for handsets from non-OEM sources, similar to ROM distribution from companies in the model of Cyanogen (Cyanogen no longer makes ROMs available). This would provide end users with a much simpler path to updating their handsets than the sometimes complex ROM replacement that is mostly accessible to advanced hobbyists today. However, this also provides customers with an easy mechanism to eliminate bloatware installed by OEMs and MOs, which is disadvantageous for the OEM and MO since the distribution of bloatware is profitable for them. It is not clear if Treble updates will also be accompanied by a signing mechanism, which would preclude the option for end users to get Android platform updates from any source.

Through the use of Treble, the update path for Android handsets is easier than it was before, but it still requires effort on the part of the OEM or MO. For example, OEMs that extensively "skin" Android or MOs that are paid to add third-party apps will require additional effort to integrate their changes to updated Android releases prior to distribution to end users. Further, the easy availability of new dessert releases through Treble could cost OEMs in terms of new handset sales (if Android P is available through Treble, customers may not be as motivated to purchase new handsets to get access to the new platform), reducing the motivation of OEMs to make updates available.

Treble is clearly a massive initiative by Google, and a welcome addition to the platform that reduces many of the barriers that preclude making updates available for end users. Further, it is wise that Google makes Treble support mandatory for new Android Oreo devices, eliminating the option for OEMs to opt out of the platform improvements. However, it is unknown if OEMs and MOs will fully embrace Treble and make updates to the platform readily available to customers.



Conclusion

In this presentation, we looked at several new iOS 12 features, including enhancements to the Intelligent Tracking Protection (ITP) feature introduced in iOS 11, new iMessage and FaceTime privacy controls, and iCloud password auditing. Like iOS 11 before it, backup passwords can be reset without losing data on the device, making it possible for someone who can bypass the lock screen to obtain a *logical backup* of the data. GrayKey continues to be problematic despite attempts to limit access to the USB data interface. The iOS 12 AirDrop password-sharing mechanism is an interesting feature, making it possible for Mac users to eliminate all password storage on laptops and desktops in favor of iOS sharing.

For Android Pie, Google continues to introduce new SDK and API requirements for developers, contributing to the overall improvement of reliability and security of Android apps. Android Pie also solves several security problems with Android Oreo and earlier devices, including the disclosure of WiFi location scanning and access to the phone numbers of local and remote callers.

Although Project Treble was introduced in Android Oreo and is not specific to Android Pie, we examined its developments as a way to reduce Android device fragmentation. We also discussed the use of Instant Apps.

If you found this material useful, I'd love to hear from you! Thank you!

Joshua Wright

jwright@hasborg.com

https://www.sans.org/sec575

@joswr1ght