Long Comment Regarding a Proposed Exemption Under 17 U.S.C. 1201

Item 1. Commenter Information

Jay Freeman (saurik); +1 (805) 895-7209; saurik@saurik.com; SaurikIT, LLC (Member)
Mailing Only: 8605 Santa Monica Boulevard #21162; West Hollywood, CA 90069, USA

Item 2. Proposed Class Addressed

Proposed Class 17: Jailbreaking – all-purpose mobile computing devices

Item 3. Overview

My name is Jay Freeman, but most people online know me as "saurik". I developed and now maintain Cydia, a platform that allows developers to modify the software on almost any device running either **iOS** or **Android**, currently the two most popular "mobile" operating systems.

Cydia consists of four components: **Impactor** (a tool for "jailbreaking" devices, primarily those running Android), **Substrate** (a development platform for modification of third-party software), **Installer** (a way for users to install modifications on to their device), and **Store** (a licensing and payments backend, allowing companies to monetize their third-party modifications).

Cydia was first released in 2008, and over the past seven years has been installed on over 130 million unique devices running iOS. According to Apple in June of 2014, they have to date sold 800 million such devices. It is thereby likely safe to say that Cydia has been used, at least once, on over 10% of all devices that have ever been sold by Apple: **this is important**.

(This fits the figures that have been reported previously by various analytics companies, who have stated that there have been, at any given moment, anywhere between 6% and 12% of devices "currently jailbroken". We would not expect all devices jailbroken to stay jailbroken, so the number that have ever once been jailbroken will be higher than those "current" statistics.)

Cydia is also very relevant on Android. During much of 2013 and 2014, Impactor was the "tool of choice" for jailbreaking Android devices, and Substrate, while misunderstood as unpopular on Android, is actually distributed as a "white label" (unbranded) technology powering some jailbroken-Android-only applications, and was **installed on over a million devices**.

In 2009, Cydia launched a payment platform in order to allow developers in the ecosystem to monetize their work, and in 2010 merged with Rock Your Phone, another system targeting the same market. Combined, in these six years, these two solutions saw **$40 million in revenue**, with approximately **80% (>$30m) of this having been paid out** to developers and artists.

There is Only One Class of Device: "General-Purpose Computers"

I must note that iOS and Android are used, without substantive modification, on a number of devices that may look different, but which are **inherently identical**. The iPhone, iPad, iPod touch, and AppleTV are all nearly identical hardware and all run the **exact same code** from Apple for their operating system. Samsung's Galaxy S5 (a phone), Galaxy Tab (a tablet), and their "Smart TV" all use **virtually identical code** from Google for their operating system.

The Google Glass (head-mounted) and Google Gear (wrist-mounted) "wearable computers" also use the same operating system and can run the same software as other Android devices.

In fact, it is one of the primary benefits of these platforms by Google (and often Apple) that all different devices can be easily targeted by developers using a single development toolchain, and where a single resulting "app" not only can but **should** be usable on all classes of device.

For each of these form factors, the UI may look different (which sometimes makes end users think that the devices themselves are different), but the platforms are actually the same. Even among Smart TVs, many now come installed with software based on the Android TV platform, which allows users to browse the Google Play Store to install software. Televisions that do not use a standard operating system generally do not support third-party software installation not because they actively wish it not to be possible, but because they do not have the manpower to maintain a developer program (and in fact some televisions people think you can't install apps on actually do support apps: it is just the case that no one has bothered to develop any).

(One exception to this—a device using a standard operating system but which does not let users install applications—is the AppleTV. Large development houses do have access to run apps on the device, but they are shipped by Apple. It is still a market advantage for this device that it runs the same operating system as other devices running iOS, as this means that the complex apps that have been developed for the iPhone, such as Netflix, are rapidly ported. The AppleTV actually runs the same software stack as other devices, and with minimal work developers have managed to get apps designed for the iPad and iPhone to load and run.[1])

It thereby does not make any sense to separate these classes of device from each other: it is **not possible** to develop techniques that work on one of these classes of device but which expressly or explicitly excludes any of the others. I thereby will be providing the **exact same commentary** on all of the following proposed classes: 16 (wireless telephone handsets), 17 (all-purpose mobile computing devices), 18 (dedicated e-book readers), and 20 (smart TVs). In fact, the answers for every single question that the copyright office is interested in having answered for all of these devices has the exact same answer for every one of these classes.

[1] "iOS apps running on AppleTV"
http://www.appletvhacks.net/2012/01/02/ios-apps-running-on-apple-tv/#.VNXkKGR4oso

During the 2012 exemption cycle, an exemption was requested for a class of devices which was called "tablets"; this exemption was denied because the Registrar "found significant merit to the opposition's concerns that [...] the proposed class was broad and ill-defined, as a wide range of devices might be considered 'tablets,' notwithstanding the **significant distinctions** among them in terms of the way they operate, their intended purposes, and the nature of the applications they can accommodate". In fact, there aren't "significant distinctions" in this class.

The only justification given by the Copyright Office for this stance was an example: "an ebook reading device might be considered a 'tablet,' as might a handheld video game device or a laptop computer". The blunt reality is that attempting to define the differences between these classes of device, when even the manufacturers themselves **purposely** blur their use cases, and when **their primary feature** is often that they all run the same software, is **nonsensical**.

An e-book reader that is "only" an e-book reader is only a device which is "only" an e-book reader up until the moment that someone jailbreaks it: then it becomes like any other device. The software running on the device may also look, at the low-level, like that of other devices developed by the same manufacturer, making it subject to the same exploits, and making the same tools that were designed to jailbreak the other devices work without modification on it.

Honestly, as one of the people most affected by the decision that will be made on these laws, I will thereby make the following statement, one which may shock some other proponents: I would like to see the Copyright Office commit to providing the exact same answer for all of these classes, **even if that answer is in the negative**. Providing an exemption for one class or another class but not all of these classes at the same time ignores the reality that there is no way, from the level of technology, to tell the difference between these use cases of device.

It is Insufficient to Buy Alternative Hardware

An argument which is often used against the need for all hardware to be modifiable is that, somewhere, there might exist an alternative piece of open hardware the user can purchase instead. This is an idea the Copyright Office has specifically asked commenters to address.

The first problem with this argument is that, in practice, there are very few open devices that are available. Android is sometimes seen as an "open platform", but that is primarily for the benefit of hardware manufacturers: most Android devices are locked down. (Surprisingly, due to Google's marketing efforts on "open", even users I've spoken to who rely on jailbreak tools to modify their software often don't "connect the dots" and realize their device is closed.)

In practice, users want the "flagship" devices from Samsung and Apple, not the competing "open" devices directly from Google. Fewer than 1% of users (with a 1% margin of error on polling) own a Google Nexus device; it was said by a market analyst that "Google Nexus devices aren't having much of an impact on the market".[2] Of course, users don't buy these devices because they are closed: they buy them because the largest manufacturers have the highest quality hardware, with the latest technology for their screens and their cameras.

Another problem with this thought process is that it is often the network effects that provide the greatest benefit, from the perspective of the creation of new copyrighted works. While it is beneficial for a user who wishes to modify their own device to be able to purchase a special device for that purpose, in practice the user will have already purchased their device by the time they realize they have a need to modify its software: the "potential market" for those who write these modifications has to be measured in the number of people with a device that can install that feature; having a special "for developers" device—a special device that almost no one will own—means that there is almost no incentive for modifications to be written at all.

Item 4. Technological Protection Measures and Method of Circumvention

All the proposed classes of devices proposed for the "jailbreaking" exemptions use common cryptographic signature verification to protect the software on the device from being modified.

The exact details sometimes differ, but the overall process of "use a private asymmetric key to sign a cryptographic hash of the content, verified using a public key" is **always the same**.

Some devices check every time the device boots, while some check only when the device is upgraded, and the specific choices of hash functions and cryptographic protocols vary, but the overall process is as described. In practice: this protection is never directly attacked.

There are also "mitigations" that are sometimes implemented to make it more difficult to take advantage of known classes of vulnerabilities: from non-executable stacks to address space layout randomization, these "mitigations" are generally identical to those on desktop PCs and are used primarily to increase the security of the user, and are **not** "protection measures".

In fact, it is useful to point out that the primary argument for the protection measures available on the device for code signature protection are **not** to protect the copyright of the device: they are instead designed to protect the user from malicious third-parties (such as an ex-spouse who has been stalking the user) who might want to modify the software running on the device.

Companies often argue that these techniques are designed to prevent piracy, but **this is not true**: these techniques prevent people from running their own software on a device running iOS, but do not prevent applications that have already been copied from being used on other

[2] "Google's Nexus lineup may not sell well, but still challenges Android makers" http://www.computerworld.com/article/2486445/android/google-s-nexus-lineup-may-not-sell-well--but-still-challenges-android-makers.html

similar hardware (including devices running different operating systems!) through existing and well-received tools such as Cycada, Apportable, and Virtual (an emulator acquired by Citrix).

To provide a similar argument: the inability to run alternative Nintendo games on a Nintendo console without licensing the cartridge technology from Nintendo does not defeat piracy and can not be said to be designed to defeat piracy if those same apps can be run with minimal modification on a desktop PC. The limitation of who is allowed to deploy games for that device is not what prevents piracy: what prevents piracy is how difficult it is to read a cartridge (such as due to encryption, or other hardware locks), or mitigations that might be in the game itself to verify that it is running on the right kind of hardware. Those are actual coryright protections.

Essentially, I will argue that to fairly evaluate the technical protections applied to a copyrighted work, you must ask 1) what protections the work itself has from being used by the user on a competing device and 2) what protections the device has from copying the work off. The third question of what protections the device has from running illegitimate software seems to be out of scope for a discussion of copyright, and manufacturers and software developers who are using these protections are invoking copyright law to establish a chilling effect on competition.

Even this level of protection is a disingenuous claim: it is possible for **anyone** willing to spend $99 on Apple's developer program to enable up to a hundred devices to run any software they want on those devices. There have been numerous companies and individuals who just sell access to their developer account to others, and entire app markets have been built for them.

More importantly, most piracy on these devices (of all the proposed classes) now takes place using software from companies in China such as Kuaiyong[3], which **do not require jailbroken devices**: instead, they use "enterprise certificates" (developer accounts from Apple that don't have the 100 device limit and can run software on all devices) contributed by random Chinese parties to run software on the official devices from iOS. **Apple does not bother to stop this**.[4]

In summary, on all of these mobile platforms, once a user is able to get access to a copy of software, whether from the manufacturer of the operating system or from a third-party, there is nothing implemented in the software that keeps that software from being used illegitimately.

How Users Get Access to Pirated Software

We now must turn our attention to how users make the first copy of pirated software. For the Android platform, software itself is not protected: there is a server-side mechanism which

[3] "Chinese website allows pirating of iOS apps, no jailbreaking required" http://www.examiner.com/article/chinese-website-allows-pirating-of-ios-apps-no-jailbreaking-required

[4] "When Criminals Exploit Apple's Own App Distribution System, What Hope Is There Of Stamping Out Piracy?" http://www.forbes.com/sites/emmawoollacott/2013/04/19/when-criminals-exploit-apples-own-app-distribution-system-what-hope-is-there-of-stamping-out-piracy/

keeps users from downloading "copy-protected apps" to devices which are "insecure". This mechanism is able to be defeated without jailbreaking the device by having an "open" device pretend to be a secure device (which can be easily done by changing its device identifiers).

Apple does implement a protection: they encrypt third-party apps, and only decrypt the binary when it has been executed via a fork() system call, which is normally not allowed to be used on devices that are not jailbroken. However, on iOS 8 support was added to dynamically load encrypted code: the new mremap_encrypted system call can be used to easily pirate apps.

This means that, going forward, jailbroken devices will no longer be part of the end-to-end process for pirating applications on any platform: the idea of running a pirated application never required a jailbroken device, and now obtaining a decrypted copy of the application on the one platform where this was previously difficult (iOS) now only requires "defeating" the app encryption, which can be done using a standard $99 developer account from Apple.

When jailbreaking is not even required to either copy apps from the device or to run apps on the device, I think it should seriously calls into question whether jailbreaking is said to defeat a mechanism designed to protect the copyright of any of this software. It certainly should mean that piracy for these platforms—whether wireless telephone handsets or tablets—should not be considered an important consideration with respect to deciding this exemption class.

How Jailbreaking Works

As mentioned, the actual signature verification mechanism is normally not directly defeated: modern encryption is not something that can be attacked by people without many millions of hours of computer time. Instead, "indirect" methods have to be taken to bypass protections.

The most direct example is something called a "parser differential", wherein the data that is verified is found using a different algorithm than the data that is executed: this is akin to having one bouncer at a bar verify that someone's driver's license is "legitimate", but then a different person looks at whether it has a birthday more than 21 years ago, and there is an opportunity in between to give the second bouncer a different (illegitimate) driver's license.

More commonly, however, this mechanism is simply left to operate as it normally does. Much later, someone finds a tunnel into the bar, and starts helping people get in "through the back".

These techniques are generally exploits for common software security vulnerabilities such as "buffer overruns", "use-after-frees", and "format string attacks". The technical details of exactly how each of these varied mechanisms work is beyond the scope of this documentation, but for those curious I sometimes teach lectures at the University of California, Santa Barbara's College of Creative Studies on how these techniques work (in detail) with examples of how they are implemented. I would be happy to explain it in detail to the Copyright Office as well.

Item 5. Asserted Noninfringing Use

Given that our only business model is "selling software", it should be pretty clear that we, as a group, are fairly anti-piracy. In fact, we go to great lengths to moderate public forums that we have control over to make certain that people are talking about only "legitimate" modifications.

At this point it is also important to explain what Cydia is **not**: a place for apps "rejected" from the official markets run by Apple (the App Store) and Google (the Play Store) to get uploaded.

The purpose of Cydia is to allow developers to add features and functionality to software that was developed by other developers without access to the original source code. This allows features that would only benefit a "small" percentage of Apple's user community (or that of any third-party apps available in the official channels from Apple or Google) to be developed independently and distributed directly to the users who want or need that functionality.

Note that at no time are the original works from either Apple or any other software company redistributed: all modifications are provided only as the set of "modifications" that are required, in a very abstract form, to the original work (which the user must have obtained legitimately).

As an analogy, many people may have watched the famous Saturday Night Live sketch which starred Will Farrell and Christopher Walken, wherein the rock classic by Blue Öyster Cult, "(Don't Fear) The Reaper", is semi-awkwardly given "more cowbell".[5] If someone wanted **even more cowbell** than the original song, they could distribute a track of only the new cowbell, which people who separately purchased the original song could play at the same time.

Some concrete examples include modifying the layout of icons on the "home screen" of these products, allowing incoming messages to be more quickly responded to, adding support for animations and videos to apps that normally only function with static graphics, allowing the control of the device using gestures or other physical actions (such as pressing buttons which are normally reserved for other purposes: maybe using the volume buttons to turn on the flash to rapidly get access to a flashlight without having to first activate the interface of the device).

General Use Case: Accessibility

While Apple has often been quite good handling blind users, other device manufacturers have often not done so well; and even Apple sometimes falls short of what blind users really need

[5] "More Cowbell - Saturday Night Live"
http://vimeo.com/91715361

to use their devices. Maybe more interestingly, there are other accessibility issues—such as physical handicaps with motor control—that are entirely ignored by most software developers.

In the world of jailbreaking, users have often been able to find help for their accessibility need, from ways to access your iPhone's content using larger screens[6] to improvements to Apple's VoiceOver technology. As a simple example, curtainChecker was a modification written by Simon Selg at the direct request of a blind user: it audibly reminds the user if the "Screen Curtain" feature (which turns off the screen, saving users who can't see the screen anyway a lot of battery) has been turned off, as otherwise they don't have a good way to realize this.[7]

General Use Case: Increased Competition

The Copyright Office has asked that we explicitly examine the manner in which the access controls on the device prevent software which competes with the device manufacturer from being installed. On Apple's platforms, there is no way to make an alternative browser to the default Safari, or an alternative to Apple's Maps application, get opened by other applications whenever they want to open a hyperlink, or jump to a location. This can be done on jailbroken devices using BrowserChooser[8] and MapsOpener[9] (respectively).

There are also cases where Apple's applications have a powerful advantage over third-party applications, such as being able to schedule tasks in the background even while the device is asleep (required to implement an alarm clock) or the ability to display information other than notifications on the lock screen (required to really compete with Apple's Maps turn-by-turn).

The same lock screen limitation keeps users from being able to quickly access alternatives to Apple's stock Camera app: some users use GrabberApp (from Cydia) to launch Camera+.[10]

(On Android, it is usually possible to replace stock software, though there are a few areas of Android that only Google is able to modify, such as the implementation of the lock screen.)

General Use Case: Privacy Issues

[6] "I hope complete compatibility will be restored soon because I use [Veency] as an accessibility tool."
http://www.jailbreakqa.com/questions/212466/veency-on-ios-7/219535

[7] "Cr4zyS1m0n / curtainChecker"
https://github.com/Cr4zyS1m0n/curtainChecker

[8] "BrowserChooser Makes Chrome Your Default Browser on iPhone"
http://lifehacker.com/5922339/browserchooser-makes-chrome-your-default-browser-on-iphone

[9] "MapsOpener jailbreak tweak sets Google Maps as default maps app"
http://9to5mac.com/2012/12/15/mapsopener-jailbreak-tweak-sets-google-maps-as-default-maps-app/

[10] "GrabberApp Cydia Tweak Allows You To Open Any App With The Lockscreen Camera Grabber"
http://www.iphoneincanada.ca/jailbreak/grabberapp-cydia-tweak-allows-you-to-open-any-app-with-the-lockscreen-camera-grabber/

Without the ability to monitor the information which is streamed from the device by network, or to see what background processes are executing, it is not possible for anyone other than the original manufacturer to find and disclose ways these devices may violate the expectations they have about how their private data is being stored or used. In 2011, a security researcher found that iOS 4 was storing an unencrypted file with your entire location history, which could then be rendered onto a map, tracking your travels.[11] On iOS 5, Carrier IQ (a scary piece of software found on a number of mobile platforms) was discovered to also exist on the iPhone.[12] On iOS 7, it was discovered that e-mail attachments were being stored unencrypted.[13]

These discoveries led to privacy improvements: Apple has shipped updates due to this work.

General Use Case: Security Updates

One thing that comes up often, but which I do not think is sufficiently explained, is that once a device has become "old" it is often entirely "dropped" from support from receiving updates to its software, **even though the hardware is still supported by network service providers**.

That latter note is very important: if you own an iPhone that is only a few years old, you will not be able to update to the latest version of iOS from Apple, and you might not be able to install some of the latest software from companies such as Facebook or Twitter, but even a device that is ten years old is still perfectly usable on modern Wi-Fi and cellular networks.

By and large, the people who design websites also continue to support these older devices for a very long period of time, which means that a user of an iPhone 4 may not be able to update to iOS 8, but they have no issue using it to make calls or browse the web. This device is no different to them than it was when it was purchased, which might not even have been very long ago: **Apple was selling new iPhone 4's in India less than one year ago** in April 2014![14]

When a user is using a device that is not supported it is **actively dangerous**, as they become open to attack by people with knowledge of these security vulnerabilities. Every new version of iOS that comes out fixes security flaws, and devices that are no longer supported by the original manufacturer do not have access to these official updates. For their jailbroken device, users can find unofficial updates to some of the most dangerous bugs. As specific examples,

[11] "iPhone Tracks Your Every Move, and There’s a Map for That"
http://www.wired.com/2011/04/iphone-tracks/

[12] "Carrier IQ is on iOS"
http://blog.chpwn.com/post/13572216737

[13] "iOS 7 doesn't encrypt email attachments"
http://www.zdnet.com/article/ios-7-doesnt-encrypt-email-attachments/

[14] "Apple withdraws iPhone 4 from Indian market"
http://timesofindia.indiatimes.com/tech/tech-news/Apple-withdraws-iPhone-4-from-Indian-market/articleshow/34767917.cms

the PDF bugs seen on iOS 4[15], the SSL bugs seen on iOS 7.0[16], and the CoreGraphics bug seen on iOS 7.1[17], have all been fixed via modifications developed using Cydia Substrate.

Item 6. Asserted Adverse Effects

Without the ability to circumvent the technologies that are being claimed, all of the previously mentioned non-infringing uses would not be possible. This, I will assert, is the adverse effect.

Item 7. Statutory Factors

(i) The availability for use of copyrighted works

As argued above, there are numerous ways in which Apple's software can be used that would benefit many people in different ways if it can be legally modified in the ways that we do when we "jailbreak" the device. If this is a question of copyright, then all of these non-infringing uses for which we want to use the copyrighted work would seem to become statutory factors.

(iii) the impact that the prohibition on the circumvention of TPMs applied to copyrighted works has on criticism, comment, news reporting, teaching, scholarship, or research

Some of the listed non-infringing uses involved researching how personal information was being shared and stored by the device in ways the user did not like: without this ability, the ability to criticise and comment on the platform is reduced. Without the ability to see what is running on the device, it is also much more difficult to teach how to work with these devices.

(iv) the effect of circumvention of TPMs on the market for or value of copyrighted works

As mentioned, in Cydia alone there has been $40 million dollars of software sold in the past seven years. Some of the largest programs in the ecosystem do not use Cydia to perform their sales. This is a substantive market that would be destroyed without these abilities.

Item 8. Documentary Evidence

N/A

[15] "Fixing what Apple won't"
http://blog.iphone-dev.org/post/941467261/fixing-what-apple-wont

[16] "Cydia Tweak: How To Install iOS 7.0.6's SSL Security Patch Without Updating"
http://appadvice.com/appnn/2014/02/cydia-store-how-to-install-ios-7-0-6s-ssl-security-patch-without-updating

[17] "feliam / CVE-2014-4377-Fix"
https://github.com/feliam/CVE-2014-4377-Fix