Before the

U.S. COPYRIGHT OFFICE, LIBRARY OF CONGRESS

In the Matter of Exemption to Prohibition on Circumvention of Copyright Protection Systems for Access Control Technologies

Docket No. 2014-07

Comments of the Electronic Frontier Foundation

1. Commenter Information

Electronic Frontier Foundation
Mitchell L. Stoltz
Corynne McSherry
Kit Walsh
815 Eddy St
San Francisco, CA 94109
(415) 436-9333
mitch@eff.org

The Electronic Frontier Foundation (EFF) is a member-supported, nonprofit public interest organization devoted to maintaining the traditional balance that copyright law strikes between the interests of rightsholders and the interests of the public. Founded in 1990, EFF represents over 25,000 dues-paying members, including consumers, hobbyists, artists, writers, computer programmers, entrepreneurs, students, teachers, and researchers, who are united in their reliance on a balanced copyright system that ensures adequate incentives for creative work while promoting innovation, freedom of speech, and broad access to information in the digital age. In filing these comments, EFF represents the interests of the many people in the U.S. who have "jailbroken" their mobile computing devices—or would like to do so—in order to use lawfully obtained software of their own choosing, and to remove software.

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

Computer programs that enable all-purpose mobile computing devices, such as tablets, to execute lawfully obtained software, where circumvention is accomplished for the sole purposes of enabling interoperability of such software with computer programs on the device, or removing software from the device. "All-purpose mobile computing devices" means non-phone devices sold with an operating system designed primarily for mobile use and not designed primarily for the consumption of media.

All-purpose mobile computing devices include tablets such as the Apple iPad, Samsung Galaxy Tab, or Amazon Kindle Fire; or smaller-format, non-phone devices such as the iPod Touch. The category does not include desktop or laptop portable computers (PCs) sold with PC operating systems such as MacOS or Microsoft Windows, nor devices

designed primarily for media consumption such as dedicated e-book readers or handheld gaming consoles.

3. Overview: Mobile Computing Device Firmware Is a Unitary, Distinct Class of Copyrighted Work, And the Ability To Add, Modify, and Remove Software From Mobile Computing Devices Is More Important Than Ever.

EFF asks the Librarian to grant an exception to the ban on circumventing access controls on copyrighted works, 17 U.S.C. § 1201(a)(1), applying to mobile computing devices sold with an operating system designed for mobile use, and not designed primarily for the consumption of media. This exception would be for jailbreaking or rooting, in order to enable mobile computing devices to run lawfully purchased software of the owner's choice, or to remove software.[1]

The Librarian has granted an exemption for jailbreaking mobile phones in each of the last two triennial proceedings, and that exemption has now been in place for nearly five years. During that time, the prevalence of mobile devices as a computing platform, the need for security, privacy, and customization on those devices, and the demand for lawful, non-manufacturer-approved software have all increased dramatically. But access controls affecting the vast majority of devices, whether phones, tablets, or small multipurpose devices, continue to stand in the way of owners' ability to run the lawfully acquired software of their choice, to remove software from their devices, to prolong the useful life of their devices, and to maintain the security of their personal information. The need for an exemption to § 1201(a)(1)'s prohibition for jailbreaking mobile device firmware for all such devices has never been stronger.

The particular class of works for which we request an exemption is mobile computing device firmware—the basic software that operates handheld mobile computing devices and allows them to run a vast variety of application programs ("apps" or "software packages"). The same mobile firmware, primarily Apple's iOS and varieties of the Android operating system, is sold on smartphones, tablets, and other handheld devices such as the iPod Touch.

Though mobile computing devices can be subdivided based on their size and their ability to make and receive telephone calls, they are in many respects a single category of device. In addition to running the same operating systems, smartphones and tablets are largely able to run the same applications. The common practice among software developers is to write software that is meant to be used on both phones and tablets.[2] James Willcox, a Staff Platform Engineer at Mozilla who leads a team of mobile software developers, sees "little difference between these devices, especially in the case of Android."[3] Most phones and tablets use the same processor architecture, known as ARM,

[1] The term "jailbreaking" is commonly used for Apple devices running iOS, while the preferred terms for devices running Android and its derivatives are "rooting" or "bootloader unlocking." For clarity, these comments will refer to "jailbreaking" a device regardless of the operating system.
[2] *See* Statement of Marc Rogers, at 1 [hereinafter "Rogers Statement"]; Statement of James Willcox, at 2 [hereinafter "Willcox Statement"].
[3] Willcox Statement at 2.

giving a degree of uniformity to the development process across devices.[4]

Increasingly, the presence or absence of particular types of cellular radio hardware does not distinguish phones from tablets. The cellular communications protocol known as 4G LTE, which is in widespread use in the U.S., treats voice calls and data transmissions identically, meaning that any phone or tablet that uses LTE can make and receive voice calls using tools such as Skype or Google Hangouts, regardless of whether the device is marketed as a phone.[5] Conversely, both larger tablets and smaller devices like the iPod Touch[6] and numerous "WiFi-first" phones[7] use short-range WiFi protocols as their primary or only means of telephone calling and data transmission, rather than cellular technology.

Smartphones and tablets today are best seen as a continuum of devices varying primarily by size, rather than distinct categories. Phablets, which are devices of intermediate size between a smartphone and a tablet and that function as either, are one of the fastest-growing categories of mobile devices. Phablets were predicted to sell 175 million units in 2014, exceeding sales of laptop personal computers (PCs).[8] Phablets are expected to grow from 14% of the worldwide smartphone market to 32.2% by 2018.[9]

On the other hand, tablets are easily distinguishable from laptop and desktop PCs. PC operating systems such as MacOS and Microsoft Windows are distinct from mobile operating systems.[10] Most recent PCs use an Intel x86 processor architecture, rather than the low-power ARM architecture used by mobile devices. The market research firm Gartner Inc. places tablets in a separate category from PCs.[11] Mobile devices have replaced desktop and laptop PCs for most uses. In 2013, mobile devices became the leading platform for total time spent online, surpassing PCs.[12] Most importantly, PC operating systems do not, as yet, impose the sort of severe restrictions on which

---

[4] Rogers Statement at 1.

[5] Mat Honan, *Never Buy a Phone Again*, Wired (Jan. 5, 2015), http://www.wired.com/2015/01/phones-are-tablets/.

[6] *iPod Touch Tech Specs*, Apple, https://www.apple.com/ipod-touch/specs.html.

[7] Sarah Thomas, *Why WiFi-First Works for Wireless*, LightReading (Dec. 30, 2013), http://www.lightreading.com/mobile/carrier-wifi/why-wifi-first-works-for-wireless/d/d-id/707117.

[8] *A Future Fueled by Phablets – Worldwide Phablet Shipments to Surpass Portable PCs in 2014 and Tablets by 2015, According to IDC*, International Data Corporation (Sep. 3, 2014), http://www.idc.com/getdoc.jsp?containerId=prUS25077914.

[9] *Id.*

[10] While the features of, for example, MacOS and iOS are converging to some degree, they remain separate products. Apple CEO Tim Cook has stated that the company does not intend to combine a laptop PC with a tablet. Josh Lowensohn, *Tim Cook knocks idea of MacBook-iPad combo device*, CNet (Apr. 24, 2012), http://www.cnet.com/news/tim-cook-knocks-idea-of-macbook-ipad-combo-device/. Likewise, Windows Phone and Microsoft Windows (for PCs) remain separate products. *See* Brad Molen, *Windows Phone 8.1 review: Microsoft's mobile OS finally feels whole*, Engadget (Apr. 14, 2014) http://www.engadget.com/2014/04/14/windows-phone-8-1/.

[11] *Gartner Says Tablet Sales Continue to Be Slow in 2015*, Gartner (Jan. 5, 2015), http://www.gartner.com/newsroom/id/2954317.

[12] Adam Lella, *When Mobile Web Dominates Apps in an App-Dominated World*, comScore (Jul. 2, 2014), http://www.comscore.com/Insights/Blog/When-Mobile-Web-Dominates-Apps-in-an-AppDominated-World.

applications can be run, and what those applications can do, which are the norm for mobile devices.[13]

As their functionality develops, consumers are able to use their mobile devices for a greater range of functions, reaching into more categories for which personal computers have traditionally been used. Consumers spend more time looking at maps and weather, sharing photos, and social networking on mobile devices than on PCs.[14] Modern mobile devices have many functions that most PCs don't, such as a point-and-shoot camera, location awareness, a video recorder, and tilt-based input. Moreover, the increased use of cloud-based services is making desktop software less necessary.[15]

Mobile devices are also distinguishable from dedicated media consumption devices such as ebook readers and handheld gaming devices. These devices do not come with general-purpose operating systems capable of running a large variety of application software. They are optimized for the delivery of particular forms of media and do not replicate the functionality of a PC. For example, ebook readers such as the Kindle Paperwhite have a greyscale "e-ink" display designed to evoke a printed page, and are "fine-tuned specifically for reading."[16] They are "exclusive e-book reader[s]." The Kindle Fire, in contrast, runs a version of Android, contains considerable processing power and memory, and is able to run a large variety of apps from the Amazon App Store.[17] It is a general-purpose mobile computing device.

Mobile devices are deeply personal. They often contain personal photographs and video, sensitive communications, and a great deal of information about the owner's movements, associations, preferences, and thoughts.[18] The intimacy of mobile device use leads to

---

[13] Some PCs do contain access controls affecting particular functionality, but those are not the subject of EFF's proposal. *See* In the Matter of Exemption to Prohibition on Circumvention of Copyright Protection Systems for Access Control Technologies, Docket No. RM 2011-7, Comment of the Software Freedom Law Center at 4-5 (Dec. 1, 2011).

[14] Greg Sterling, *Report: Nearly 40 Percent Of Internet Time Now On Mobile Devices*, Marketing Land (Feb. 26, 2013), http://marketingland.com/report-nearly-40-percent-of-internet-time-now-on-mobile-devices-34639.

[15] Scott Martin & Jon Swartz, *Desktop PCs less popular than ever*, USA Today (Jun. 24, 2013), http://www.usatoday.com/story/tech/2013/03/06/apple-google-microsoft-hewlett-packard-dell-ipad-iphone-android-ios-samsung-galaxy/1946325/.

[16] Joe Walsh, *Top 5 Things to Consider When Buying a Kindle*, Pick My Kindle, http://pickmykindle.com/top-5-things-to-consider-when-buying-a-kindle/ (last visited Feb. 6, 2015).

[17] Steve Ranger, *Kindle Fire HDX 7" tablet review: Elegant hardware, but a frustrating experience*, ZDNet (Jan. 20, 2014), http://www.zdnet.com/article/kindle-fire-hdx-7-tablet-review-elegant-hardware-but-a-frustrating-experience/.

[18] *See What to do if Your Cell Phone is Lost or Stolen*, USA.gov (Jun. 16, 2014), http://blog.usa.gov/post/88969462496/what-to-do-if-your-cell-phone-is-lost-or-stolen ("Mobile phones are a vital part of life. You may store passwords, account numbers, phone numbers, addresses all in this one device. If your phone is lost or stolen, your privacy, identity, and bank accounts could also be in jeopardy."); *U.S. v. Jones*, 132 S. Ct. 945, 955 (2012) (the record of a person's movements "reflects a wealth of detail about her familial, political, professional, religious, and sexual associations") (citation omitted).

strong demand for customization, and for the ability to take control of one's own privacy.[19]

4. The Technological Protection Measures and Methods of Circumvention: Cryptographic Verification of Software, Locked Bootloaders, and Denial of "root" Privileges on Mobile Operating Systems.

A profound difference between mobile computing devices and PCs is that controls within the firmware[20] on nearly all mobile devices prevent the owner of the device from installing, removing, or modifying software to some degree. The overwhelming majority of mobile devices sold and used in the U.S. contain technical protection measures that limit what software can run on the device, limit what such software can do, or both.

The mobile device market is dominated by devices running two operating systems: Google's Android, and Apple's iOS. In 2014, iPads (running iOS) represented about 27% of tablet sales, whereas tablets running Android made up about 67% of the market.[21] However, 71% of tablet-based World Wide Web traffic is generated by iPad users.[22] Other operating systems have much smaller market shares, such as Microsoft, whose Microsoft Surface line of devices is projected to rise to 11% of the market by 2018.[23] Android, iOS, and other mobile operating systems all contain access controls that restrict the running and removal of software.

A. iOS: Cryptographic Verification of All Software

Devices that run iOS, including the iPod Touch and iPad, continue to be subject to severe restrictions on the loading, running, and deletion of software. iOS contains cryptographic verification that prevents any application from running on a device unless it bears a digital signature from Apple.[24] This restriction means that new software can only be loaded on a device through Apple's iTunes Store. It also contains cryptographic checks at various levels of the software stack that prevent modification or replacement of the

---

[19] Jay Freeman ("SaurikIT"), *What is "Jailbreaking"? @ Dragon Con 2014*, at 3:09 (Oct. 4, 2014), https://www.youtube.com/watch?v=1Mdwo8aUbSs (last visited Feb. 4, 2015).

[20] The term "firmware" with respect to mobile devices encompasses the fundamental software components that start up the device and mediate access to its screen, camera, speaker, cellular radio, and other fundamental functions. It is loosely synonymous with "operating system." *See About CyanogenMod*, CyanogenMod, http://wiki.cyanogenmod.org/w/About#But_wait--_is_the_right_term_.22ROM.22_or_.22firmware.22_or_what.3F (last visited Feb. 4, 2015) ("The ambiguous terminology is just the result of a decade-long transition from simple, non-replaceable software on hand-held devices to full-fledged, updatable operating systems on a small, portable computers that fits in the palm of your hand.").

[21] *Worldwide Tablet Growth Expected to Slow to 7.2% in 2014 Along With First Year of iPad Decline, According to IDC*, International Data Corporation (Nov. 25 2014), http://www.idc.com/getdoc.jsp?containerId=prUS25267314.

[22] *First Quarter 2015 Tablet Update: Samsung Usage Share Increases by 88% Year-over-Year*, Chitika Online Advertising Network (Jan. 27, 2015), https://chitika.com/insights/2015/q1-tablet-update.

[23] *Worldwide Tablet Growth Expected to Slow*, *supra* note 21.

[24] Apple Inc., *iOS Security—White Paper*, at 4 (Oct. 2014), https://www.apple.com/privacy/docs/iOS_Security_Guide_Oct_2014.pdf (last visited Feb. 4, 2014) ("The … process described above helps ensure that only Apple-signed code can be installed on a device.").

operating system itself.[25]

B. Android and its Variants: Locked Bootloaders, Lack of Access to Root Privileges

Android is developed by Google and sold through a variety of distribution channels. Android has many variants, including "pure" Android versions (containing little or no non-Google code) sold on some Nexus and Motorola devices,[26] and more customized versions sold by manufacturers such as Samsung, HTC, and LG, and by wireless carriers like Verizon and AT&T. These manufacturers and carriers modify Android extensively on the devices they sell, and often include a great deal of additional software—both low-level functions and applications ("apps"). There are also variants of Android based on the Android Open Source Project (AOSP),[27] including CyanogenMod. However, open source variants are not sold on new devices in the U.S. and cannot be installed on a device without circumventing the access controls in the original firmware.

Android is not as restrictive as iOS, in that Android allows a user to install application software from any source, and apps need not be cryptographically signed by a particular entity in order to run. However, Android contains technical measures that control access to key functionality and limit the functionality of application software. The fundamental access control on Android devices is the bootloader, a component of the firmware.[28] The bootloader starts up the device and loads the operating system into memory, which in turn loads "apps" and other software.[29] On Android devices, the bootloader verifies the operating system on the device cryptographically, and will refuse to run an operating system not approved by the device manufacturer, or one that has been modified.[30]

The manufacturer-installed Android operating system, in turn, places limits on what "apps" and other user software can do. First, the operating system does not allow the device owner, or any programs installed by the owner, to acquire full administrative access to the device (the status known to programmers of Android and other UNIX-related systems as "root privileges").[31] While a user or application with root privileges can access any function or data on a device, a user without root privileges can access only a limited subset.[32] The operating system also prohibits the user from *removing* unwanted

---

[25] *Id.*; Statement of Dr. Jeremy Gillula, at 2 (Feb. 6, 2015) [hereinafter "Gillula Statement"].
[26] Lynn La & Brian Bennett, *Powerful Pure Android Phones (Roundup)*, CNet (Jan. 2, 2014), http://www.cnet.com/news/powerful-pure-android-phones/.
[27] *See Welcome to the Android Open Source Project!*, Android, https://source.android.com/ (last visited Feb. 4, 2015).
[28] Willcox Statement at 1; Gillula Statement at 1-2; Rogers Statement at 2-3.
[29] Ivo, *So You Want To Know About Bootloaders, Encryption, Signing, And Locking? Let Me Explain*, Android Police (May 27, 2011) http://www.androidpolice.com/2011/05/27/so-you-want-to-know-about-bootloaders-encryption-signing-and-locking-let-me-explain/; *see also* Rogers Statement at 2.
[30] Gillula Statement at 1-2. While it is theoretically possible to erase *all* software from a device, including the bootloader, and install new code, in practical terms this is effectively impossible because it would require detailed information about the layout and function of the processor, circuit boards, and other device hardware—information that is a closely guarded trade secret and varies from device to device.
[31] *Id.* at 1; Willcox Statement at 1.
[32] *Id.*

programs that were installed by the manufacturer or wireless carrier.

Attempting to modify the operating system to allow the user to acquire root privileges, or to replace the operating system entirely, causes the bootloader to refuse to load the operating system.[33] Thus, in order to run applications that use enhanced functionality, or to replace the operating system with one that one that offers greater functionality and security, a device owner must circumvent the cryptographic checks in the bootloader or disable the access controls that restrict root privileges in the operating system, or both.[34]

C. Other Mobile Operating Systems

Other operating systems designed for mobile devices also contain access controls that restrict the loading and functionality of software. Windows Phone (Microsoft's operating system for mobile devices) imposes constraints on app software that are similar to Android's.[35] While it allows users to load applications from sources other than Microsoft's own online store, it limits the number of applications that can be loaded this way to as few as two.[36]

5. Noninfringing Uses: Installing Lawfully Obtained Software, and Removing Software.

A. Jailbreaking Described

Jailbreaking most mobile devices requires making use of a security vulnerability in either the operating system or the bootloader. For an iOS device, jailbreaking involves modifying the firmware so that it will run software code without checking to see if the code has been cryptographically signed by Apple. On Android, jailbreaking (known in this context as "rooting") typically involves using a vulnerability in some piece of manufacturer-installed software to gain the ability to run arbitrary software with root privileges. Once the owner gains this ability, she can run a program that will modify the bootloader to permit loading a modified operating system of her choosing. Dr. Jeremy Gillula, a staff technologist at EFF, describes a typical Android rooting process in his statement attached hereto, and in a video submitted with these comments.

The precise means of jailbreaking are discovered through trial and error and vary by device and software version. For all devices of which we are aware, jailbreaking requires modifying only a small portion of the firmware.

Software development communities and retail markets for non-manufacturer-approved software have arisen to fulfill the demand for more functional, secure, and customizable

[33] Rogers Statement at 2; Gillula Statement at 1-2.
[34] Rogers Statement at 2; Gillula Statement at 1.
[35] GoodDayToDie, *[XAP][GUIDE] Interop Unlock for WP8 + all Capabilities*, XDA Developers (Sep. 7, 2013), http://forum.xda-developers.com/showthread.php?t=2435697 (last visited Feb. 4, 2015).
[36] Stephen Schenck, *Even Microsoft's new WP8 sideloading rules are still seriously anti-user*, PocketNow (Aug. 14, 2013), http://pocketnow.com/2013/08/14/windows-phone-sideloading.

mobile software. Cydia, an online marketplace for non-Apple-authorized iOS software, launched in 2008 and remains very popular. Between 11.9 million and 16.3 million iOS devices in the U.S. were registered with Cydia between 2012 and 2014, and these figures likely underestimate the actual number of users, as many do not register.[37] Cydia distributes about 6500 different iOS software programs, up from 3500 at the beginning of 2012.[38] Many more software programs are hosted on other websites and downloaded through the Cydia app.[39] Most of these programs are not "apps" as Apple defines them, but rather programs that alter the experience of using the device. For example, the popular Auki program allows users of jailbroken iOS devices to send and respond to text messages without leaving the app that's currently running.[40] A program called Barrel adds dramatic transition effects to the iOS home screen.[41] Like most software distributed through Cydia, these cannot be run without jailbreaking the device, and the functionality they offer is not available on a non-jailbroken device.

In the Android world, XDA-developers, an online message board community for independent software developers, now has 6.1 million members,[42] up from 4 million in 2011.[43] CyanogenMod, an open source derivative of Android, is another focus of independent development, and had over 12 million active installs as of June 2014,[44] making it one of the most widely used varieties of Android. Two other projects, Firefox OS and Ubuntu Mobile, also offer (or will soon offer) alternative operating systems for Android-compatible devices.[45] Installing any of these alternative operating systems on a mobile device requires access to the bootloader.[46]

B. Jailbreaking Is A Noninfringing Fair Use

Although jailbreaking involves making a derivative work of the firmware on one's device, it does not infringe copyright, because it is a fair use.[47] Fair use is "a privilege in others than the owner of the copyright to use the copyrighted material in a reasonable

---

[37] Source: Cydia registrations geolocated by device IP address where possible. Data on file with commenters.
[38] Source: Cydia. Data on file with commenters.
[39] *Id.*
[40] *See auki*, TheBigBoss.org, http://moreinfo.thebigboss.org/moreinfo/depiction.php?file=aukiDp (last visited Feb. 4, 2015).
[41] *See* Jignesh Padhiyar, *10 Best iOS 8 Cydia Tweaks You Should Install on your iDevice*, iGeeksBlog, http://www.igeeksblog.com/best-ios-8-cydia-tweaks/ (last visited Feb. 4, 2015).
[42] *See xda-developers Statistics*, subsection of *What's Going On?*, XDA-Developers, http://forum.xda-developers.com/ (last visited Jan. 15, 2015) (6,158,000 members listed when last checked).
[43] *See* In the Matter of Exemption to Prohibition on Circumvention of Copyright Protection Systems for Access Control Technologies, Dkt. No. RM 2011-07, Comments of the Electronic Frontier Foundation at 5 n.19 (Dec. 1, 2011) ("EFF 2011 Comments").
[44] Interview with Koushik Dutta, January 2015 (on file with commenters). As many CyanogenMod users do not report their use to the company, this number likely underestimates the total number of users.
[45] *See* Wikipedia, *Firefox OS*, http://en.wikipedia.org/w/index.php?title=Firefox_OS&oldid=645088074 (as of Feb. 4, 2015, 22:12 GMT); *Ubuntu on Phones*, Ubuntu, http://www.ubuntu.com/phone (last visited Feb. 4, 2015).
[46] Gillula Statement at 2.
[47] 17 U.S.C. § 107 ("The fair use of a copyrighted work . . . is not an infringement of copyright.").

manner without his consent."[48] In 2010 and 2012, the Register and the Librarian correctly concluded that modifying the firmware in one's device in order to run lawfully acquired software is a fair use, falling squarely within Congress's intent to promote software interoperability. Court decisions since 2012 give additional weight to that determination.

1. The Purpose and Character of the Use

The first factor looks at whether the use of a copyrighted work is "more incidental and less exploitative in nature."[49] Where a user of software code is "not seeking to exploit or unjustly benefit from any creative energy that [the rightsholder] devoted to writing the program code," the first factor favors a finding of fair use.[50]

Over the years, a robust body of caselaw has developed recognizing uses of copyrighted work that enable greater access to information as fair uses. Some of these cases deal specifically with analysis and modification of functional aspects of software and have informed the Register's prior decisions to recommend exemptions for phone jailbreaking, phone unlocking, video game security research, abandoned software, and other exemptions relating to software.

In *Sega v. Accolade*, the Ninth Circuit explained that research into the functional aspects of video game software was a legitimate purpose. Accolade reverse-engineered Sega's games to determine the requirements for compatibility with Sega's game consoles, in order to produce its own games.[51] The court found that when Accolade reverse-engineered and made copies of its competitor's code, its "direct use" of the code was done in service of a broader, favored purpose: building new, independently developed, compatible software.[52]

In *Sony Computer Entertainment v. Connectix Corp.*, the Ninth Circuit expanded upon its reasoning in *Sega*.[53] Connectix reverse-engineered the operating system software of the Sony Playstation game console in order to create a platform for Playstation games to be played on personal computers.[54] The court held this to be a fair use, emphasizing that the innovation resulting from the creation of new platforms was favored under the first factor because it "afford[ed] [users] opportunities for game play in new environments."[55]

As two Registers concluded in prior proceedings, Congress re-affirmed the principle expressed in *Sega* and *Connectix* when it enacted the Digital Millennium Copyright Act. In the legislative history of Section 1201(f), "Congress expressed a commitment to

[48] *Harper & Row, Publrs. v. Nation Enters., Inc.*, 471 U.S. 539, 549 (1985) (citations omitted).
[49] *Lexmark Int'l, Inc. v. Static Control Components, Inc.*, 387 F.3d 522, 544 (6th Cir. 2004) (quoting *Kelly v. Arriba Soft Corp.,* 336 F.3d 811, 818–19 (9th Cir. 2003)).
[50] *Id.* at 544.
[51] 977 F.2d 1510, 1514 (9th Cir. 1992), *as amended* (Jan. 6, 1993).
[52] *Id.* at 1522-23.
[53] 203 F.3d 596 (2000).
[54] *Id.* at 598-99.
[55] *Id.* at 606; *See also Bateman v. Mnemonics, Inc.*, 79 F.3d 1532, 1547 (11th Cir. 1996) (holding that "external factors such as compatibility" reduce the rightsholder's legal interest in the copyright and favor a finding of fair use.).

permit and encourage interoperability between independently created computer programs and existing programs," in order to "avoid hindering competition and innovation in the computer and software industry."[56] As the Register found in 2010 and reaffirmed in 2012, Congress's affirmation of the importance of interoperability should guide a determination of fair use in the triennial rulemaking.[57]

An important aspect of the first fair use factor is whether the use in question is transformative, meaning that it does not "merely supersede[] the objects of the original expression."[58] Copying and modification of software to render it compatible with other, independently created software has been held to be a transformative purpose.[59] This finding is reinforced by decisions holding that the use of digital text and images for new purposes that are "different in purpose, character, expression, meaning, and message" from those of the copyright holder is transformative.[60] Because jailbreaking allows a mobile device and its firmware to be used for new purposes, imbuing them with further usefulness, personalization, and meaning, jailbreaking is a transformative purpose.[61]

Further, jailbreaking for purposes of installing lawfully obtained software is noncommercial. As the Supreme Court noted in *Sony Corp. of America v. Universal Studios Inc.*, "private home use must be characterized as a noncommercial, nonprofit activity," even where the use involved lawfully obtained copies of commercially distributed works.[62] The Court held in the absence of some demonstrable likelihood of harm to the copyright holder, personal, noncommercial use was fair use.[63] Likewise, mobile device owners who jailbreak do not do so for profit, but rather to enhance and personalize their devices.[64]

In addition, jailbreaking mobile devices promotes additional creativity and expands access to knowledge by encouraging the creation of new software applications and

---

[56] Recommendation of the Register of Copyrights in RM 2008-8, at 92, Rulemaking on Exemptions from Prohibition on Circumvention of Copyright Protection Systems for Access Control Technologies (June 11, 2010) [hereinafter 2010 Recommendation], *available at* www.copyright.gov/1201/2010/initialed-registers-recommendation-june-11-2010.pdf; *see also* Recommendation of the Register of Copyrights, at 71-72, Section 1201 Rulemaking: Fifth Triennial Proceeding to Determine Exemptions to the Prohibition on Circumvention (Oct. 12, 2012) [hereinafter 2012 Recommendation], *available at* http://www.copyright.gov/1201/2012/Section_1201_Rulemaking%20_2012_Recommendation.pdf.

[57] 2012 Recommendation at 72 ("[A]lthough jailbreaking does not 'fall within the four corners of the statutory exemption in Section 1201(f), the fact that [a device owner] is engaging in jailbreaking in order to make the [device's] firmware interoperable with an application specifically created for the [device] suggests that the purpose and character of the use are favored.'").

[58] *Campbell v. Acuff-Rose Music, Inc.*, 510 U.S. 569, 570 (1994).

[59] *Connectix*, 203 F.3d at 606-07.

[60] *Authors Guild, Inc. v. HathiTrust*, 755 F. 3d 87, 97 (2d Cir. 2014); *Perfect 10, Inc. v. Amazon.com, Inc.,* 508 F.3d 1146, 1165 (9th Cir. 2007)*; Kelly v. Arriba Soft Corp.*, 336 F.3d 811, 818-22 (9th Cir. 2003).

[61] We note that the Register has previously expressed doubt that jailbreaking is a transformative use, but concluded that transformativeness was not a requirement under the first factor. 2010 Recommendation at 95; 2012 Recommendation at 72 & n.361.

[62] 464 U.S. 417, 449-50 (1984).

[63] *Id.* at 454-56.

[64] *Cf. Sega*, 977 F.2d at 1522-24; *Connextix*, 203 F.3d at 606-07.

expanded functionality for these devices.[65] As discussed further below, the ability to upgrade the device operating system to patch known security vulnerabilities can safeguard the owner's privacy and potentially extend the lifespan of the device. Because jailbreaking a mobile device for purposes of making operating systems interoperable with independently created applications is transformative, personal, noncommercial, and confers a public benefit, the first factor weighs heavily in favor of a finding of fair use.

2. The Nature of the Copyrighted Work

The second factor, the nature of the copyrighted work, also weighs heavily in favor of fair use. In evaluating the second factor, courts look at the degree to which a work is creative or functional.[66] In *Sega*, the Ninth Circuit found the second factor to weigh in favor of fair use where copying for reverse engineering purposes was necessary in order to understand software code's functional interoperability requirements.[67] As that court reasoned, "[i]f disassembly of copyrighted object code is per se an unfair use, the owner of the copyright gains a de facto monopoly over the functional aspects of his work—aspects that were expressly denied copyright protection by Congress.[68] The *Connectix* opinion further noted that "[i]f [copyright holder] Sony wishes to obtain a lawful monopoly on the functional concepts in its software, it must satisfy the more stringent standards of the patent laws."[69]

In the 2010 and 2012 rulemaking proceedings, relying in part on *Sega's* reasoning, the Register concluded that the second factor "decisively favors a finding of fair use."[70] Noting that the second factor is "perhaps more important than usual in cases involving the interoperability of computer programs,"[71] the Register noted that bootloaders and operating systems are largely functional works, and that "[a]s functional works, certain features are dictated by function and in order to interoperate with those works certain functional elements of those programs, elements that in and of themselves may or may not be copyrightable, must be modified."[72]

The Federal Circuit's 2014 holding in *Oracle v. Google* regarding fair use of software interfaces is consistent with the Register's reasoning in the 2010 and 2012 rulemakings. The court noted that some elements of computer programs are "dictated by considerations of efficiency or other external factors" and held that "where the nature of the work is such that purely functional elements exist in the work and it is necessary to copy the

[65] *See Sega*, 977 F.2d at 1522-23 (noting the public benefit that resulted from independent developers engaging in new creative expression).
[66] *Id.* at 1524 ("The second statutory factor, the nature of the copyrighted work, reflects the fact that not all copyrighted works are entitled to the same level of protection. The protection established by the Copyright Act for original works of authorship does not extend to the ideas underlying a work or to the functional or factual aspects of the work.").
[67] *Id.* at 1526.
[68] *Id.*; *see also Connectix*, 203 F.3d at 605 (finding the second statutory factor to "strongly favor" fair use where copying was necessary to disassemble and view the ideas contained within firmware).
[69] *Connectix*, 203 F.3d at 605.
[70] 2010 Recommendation at 96, 2012 Recommendation at 73.
[71] 2012 Recommendation at 73; 2010 Recommendation at 95.
[72] 2010 Recommendation at 96.

expressive elements in order to perform those functions, consideration of this second factor arguably supports a finding that the use is fair."[73]

At least one court has found that where a portion of a software program functions as a "lockout code[]" that *must* be used to enable compatibility with independently created programs, the rightsholder's copyright interest in that portion of code is minimal. In *Static Control Components, Inc. v. Lexmark Intern., Inc.*, Static Control copied a small portion of code from Lexmark's laser printer firmware, acting on a reasonable belief that only by copying that code could Static Control build toner cartridge components that would interoperate with Lexmark printers.[74] The court held that software code used as a "lockout" bears only a thin copyright interest that is overcome by the need to use that code for interoperability.[75]

With regard to jailbreaking, the elements of a device's firmware that must be modified are security checks in the bootloader and operating system. These elements are dictated almost entirely by external considerations—namely, the exclusionary policies they implement—and not by the creative decisions of their authors. They function as "lockout codes." They play no role in generating creative graphics or sounds that lie closer to the core of copyright protection. Finally, computer operating systems are customarily used to enable the device owner to run other, independently created software without the consent of the rightsholder in the operating system.

Thus, the second factor also favors a finding of fair use.

3. The Amount and Substantiality of the Portion Used

The third fair use factor examines the amount of the copyrighted work used in an effort to determine whether the "quantity and value of the materials used are reasonable in relation to the purpose of the copying."[76] The use of an entire work does not preclude an activity from being a fair use.[77] The amount taken only need be "reasonable" and for a legitimate purpose.[78]

In *Connectix* and *Sega*, the Ninth Circuit found that copying a software program in its entirety in order to understand its functional components was necessary to achieving a favored purpose, and was therefore fair.[79] Similarly, in *Kelly v. Arriba Soft*, the court emphasized that copying anything less than an entire work would be insufficient in order to allow users to recognize images in a visual search engine.[80] In *Perfect 10,* the court

---

[73] *Oracle America, Inc. v. Google Inc.*, 750 F. 3d 1339, 1375 (Fed. Cir. 2014).
[74] No. CIV.A. 02-571, 2007 WL 1485770, at *5 (E.D. Ky. Apr. 18, 2007) (on remand from *Lexmark Int'l, Inc. v. Static Control Components, Inc.*, 387 F.3d 522 (6th Cir. 2004)).
[75] *Id.* ("Regardless of whether Lexmark's [programs] were uncopyrightable lockout codes or not, SCC was reasonable in initially believing that they were.").
[76] *Campbell*, 510 U.S. at 586-87.
[77] *Sega*, 997 F.2d at 1526.
[78] *Campbell*, 510 U.S. at 586.
[79] *Sega*, 977 F.2d at 1526 (9th Cir. 1992); *Connectix*, 203 F.3d at 605-06.
[80] 336 F. 3d at 820-21; *see also Field v. Google Inc.*, 412 F. Supp. 2d 1106, 1120-121 (D. Nev. 2006) (finding the third factor weighing in favor of neither party because, while Google copied entire pages in its web caching service, the amount used was necessary to the purpose).

concluded that Google's use of Perfect 10's images was reasonable in light of its purpose of communication information to its users.[81] In both cases, the court found this copying to be fair use. And in *Authors Guild, Inc. v. HathiTrust*, in which the plaintiffs participated in the scanning and electronic storage of numerous books, the court held that the copying was reasonable in light of its purpose.[82]

For jailbreaking, the amount of code that must be copied and modified varies depending on the device and firmware. In most cases, the portion of the firmware that must be permanently modified to accomplish a jailbreak is a very small proportion of the overall code. The jailbreak described by Dr. Gillula is typical: it involves using a security vulnerability in Android version 2.3 to install a small program that gives root privileges to the user.[83] In some cases, the required modifications to the code are *de minimis*. For example, fewer than 50 bytes of code out of more than 8 million bytes are altered in order to achieve interoperability for some versions of iOS.[84]

In short, the amount of code copied in the course of a jailbreak is necessary and reasonable for the purpose of ensuring interoperability with third party applications. This reasonable use means the third factor favors fair use, or is neutral. In prior rulemakings, the Register noted the minimal importance of the third factor in this context: "In a case where the alleged infringement consists of the making of an unauthorized derivative work, and the only modifications are as de minimis as they are here, the fact that iPhone users are using almost the entire iPhone firmware for the purpose for which it was provided to them by Apple undermines the significance of this factor."[85]

4. Effect on the Market for the Copyrighted Work

The fourth factor considers the direct harms caused by a particular use on the market or value of the work at issue, and the potential harm that might result from similar future uses.[86] Typically, courts require either a demonstration of actual harm or a likelihood that harm will result.[87] In *Sega*, the court emphasized that Accolade sought to become a legitimate competitor in the field of Genesis games and did not copy any of the elements of the Sega code that led to commercial success.[88] Moreover, consumers were likely to purchase more than one game, so sales of Accolade games would not directly foreclose Sega sales.[89] In *Connectix*, the court emphasized the transformative nature of the Connectix platform and concluded that any market harm to Sony would result from legitimate competition, not unfair copying.[90]

---

[81] 508 F.3d at 1167-68.
[82] 755 F.3d at 98.
[83] Gillula Statement at 2-3.
[84] 2010 Recommendation at 96.
[85] 2010 Recommendation at 97; 2012 Recommendation at 73.
[86] *Campbell*, 510 U.S. at 590.
[87] *See, e.g.*, *Universal*, 464 U.S. at 451-52 (1984); *Campbell*, 510 U.S. at 590-92 (1994).
[88] 977 F.2d at 1523.
[89] *Id.*
[90] 203 F.3d at 607.

By the same token, jailbreaking does not foreclose sales of mobile device firmware, nor are users jailbreaking their devices to compete in the marketplace for firmware sales. Mobile device firmware is sold along with the devices themselves, not separately. A copy of Android or iOS is of no use without a device to run it. Unlike some PC operating systems such as Microsoft Windows, mobile operating system upgrades are not sold. They are generally made available to device owners as a free download from the owner's wireless carrier. Thus, jailbreaking does not cause any proliferation of infringing copies, nor replace any sales of firmware.

Apple admitted in the 2010 rulemaking that jailbreaking had not harmed the sales or licensing of iOS firmware on phones.[91] In the 2012 rulemaking, in which EFF petitioned for an exemption extending to tablets, no mobile device manufacturer, mobile software market owner, or wireless carrier opposed an exemption, and no evidence of harm to the market for mobile device firmware was presented.

There is no new evidence to the contrary; rather, sales of mobile devices bundled with their firmware have continued to climb rapidly.[92] Moreover, the independent development communities that have arisen under the protection of the jailbreaking exemption push the entire mobile device industry towards improved performance, security, and functionality. As described further below, many popular features of iOS, Android, and other systems were first created by the jailbreaking community.[93] The ability to jailbreak can also extend a device's useful lifespan, increasing its value and that of its copyrighted firmware.[94] Far from harming the market for device firmware, jailbreaking contributes to the success of that market.[95]

All four factors, including the important first and fourth factors, favor of a finding of fair use. Jailbreaking mobile devices for the purpose of installing lawfully acquired, interoperable software is a non-infringing fair use.

6. Adverse Effects of the Ban on Circumvention: Security, Performance, Consumer Choice and Competition Denied.

The exemptions granted by the Librarian in 2010 and 2012 for jailbreaking phones removed a cloud of legal uncertainty from phone owners, and spurred vibrant markets and communities of developers. Extending an exemption to mobile devices such as tablets that run the same operating systems as smartphones would extend the benefits of the earlier exemptions. With the ability to jailbreak comes the ability to benefit from the hard work and expertise of independent developers in addition to the original

[91] 2010 Recommendation at 99.
[92] *See supra* Part 3.
[93] *See* Part 6.D, *infra.*
[94] *See* Part 6.A-C, *infra.*
[95] There is evidence that computing devices that are able to run any application of the owner's choice have higher resale value than otherwise identical devices that lack that ability. Tim Cushing, *DRM Destroys Value: Why Years Old, But DRM Free, Devices Sell for Twice the Price of New Devices*, Techdirt (Jan. 27, 2015), https://www.techdirt.com/articles/20150123/13364529795/drm-devalues-products-jailbroken-apple-tvs-selling-twice-price-latest-model.shtml.

manufacturer and carrier, without fear of anticircumvention liability. Denial of an exemption would mean that security fixes, enhanced functionality, and in the case of iOS devices, all software, would be limited by operation of the DMCA to what the manufacturer and carrier choose to provide.

A. The Ban on Circumvention Prevents Users From Fixing Security Vulnerabilities That Device Manufacturers And Carriers Do Not Fix.

Many security vulnerabilities on mobile devices occur in the operating system or other lower-level software code—in other words, they occur in code that device owners cannot modify without jailbreaking. For example, the Heartbleed vulnerability, which was discovered in April 2014, allowed malicious websites to read the contents of a device's memory, including passwords and other private information. The vulnerability occurred in a software library used by many programs, including many versions of Android.[96] A vulnerability in iOS known as "Goto fail," which would allow criminals to impersonate secure websites such as banks and merchants, was introduced into Apple devices in September 2012 but only identified in February 2014.[97] A non-exhaustive list of other security vulnerabilities that have occurred in mobile device firmware is included in Exhibit A.[98]

While Google, Apple, and other maintainers of firmware generally fix vulnerabilities like these once they are discovered, the fix often will not reach many mobile users for weeks or months, if ever.[99] Devices with cellular data service typically receive fixes and other firmware updates through the wireless carrier, while devices that use Wi-Fi alone receive updates from the manufacturer. For a given device, manufacturers and carriers typically bundle operating system fixes and upgrades, including critical security fixes, into updates that are sent no more than once or twice per year.[100] Firmware upgrades released by manufacturers often take six months or more to reach customers' devices.[101] And for Android devices, a given device will typically only receive one or two such upgrades in its first year or two of use, and none thereafter.[102] Appendix A contains a table of popular tablet devices showing the last version of Android supplied by the manufacturer. The last manufacturer-supplied update often comes mere months after the device's debut, after which the device will not receive critical security fixes.

The slow pace of official upgrades, and the practice of ceasing upgrades entirely for a device, leave owners vulnerable. For example, in October 2012, university researchers announced their discovery of a flaw in version 4 of Android that could allow an attacker

---

[96] Wikipedia, *Heartbleed*, http://en.wikipedia.org/wiki/Heartbleed (as of Feb. 4, 2015).
[97] Rogers Statement at 1.
[98] Appendix A, Part 2; *see also* Rogers Statement at 3-4.
[99] *See* Casey Johnston, *The checkered, slow history of Android handset updates*, Ars Technica (Dec. 21, 2012), http://arstechnica.com/gadgets/2012/12/the-checkered-slow-history-of-android-handset-updates/; Rogers Statement at 3.
[100] Rogers Statement at 3.
[101] *See* Johnson, *supra* note 99.
[102] *Id.*; *see also* Exh. A, Part III (table of popular devices identifying the last Android version update sent by the manufacturer).

to send forged text messages.[103] Within days, Google released a new version of Android that would prevent the attack. But four months later, only 1.4% of Android phones in use worldwide had received the fix.[104] A large proportion of devices *never* received the fix because manufacturers and wireless carriers had stopped sending upgrades for those devices.[105]

Another group of vulnerabilities affecting Web browsers in 75% of all Android devices was discovered in September 2014.[106] Although Google released a fix, one month later, about 50% of all Android devices remained vulnerable because wireless carriers did not deliver the fix.[107]

The only way device owners can defend themselves against vulnerabilities in the operating system (and other non-removable software) when manufacturers and carriers don't send a fix, or are slow to do so, is to jailbreak the device. Independent developer communities often fix vulnerabilities and make the fixes available for download by users within days of discovery, but only jailbroken devices can install such fixes.[108] CyanogenMod makes security updates from Google available almost as soon as they are released, but again, only jailbroken devices can install them.[109]

Users who are concerned about the security of their device and information can leverage the work of independent developers to limit their exposure to malicious hackers, but only if they can jailbreak their devices. The ability to jailbreak is the ability to take control of one's own security.[110]

B. The Ban on Circumvention Prevents Users From Securing their Personal Information.

Many mobile device users seek better control over the personal information their devices collect, use, and share with the manufacturer and with third parties. Often, this requires jailbreaking. For example, apps installed on an Android device request certain permissions, such as the ability to read the device's physical location, access the camera or microphone, or to read the user's text messages.[111] Typically, the user must either

---

[103] Craig Timberg, *"Fragmentation" leaves Android phones vulnerable to hackers, scammers*, The Washington Post (Feb. 3, 2013), http://www.washingtonpost.com/business/technology/android-phones-vulnerable-to-hackers/2013/02/06/f3248922-6723-11e2-9e1b-07db1d2ccd5b_story.html.

[104] *Id.*

[105] *Id.*

[106] Rogers Statement at 4.

[107] *Id.*

[108] *About CyanogenMod*, CyanogenMod, http://wiki.cyanogenmod.org/w/About (last visited Feb. 4, 2015) (describing "nightly builds" of the CyanogenMod firmware); Gillula Statement at 3.

[109] *About CyanogenMod, supra note 108.*

[110] While jailbreaking itself often involves using a known security vulnerability to gain administrative access to a device, once a device is jailbroken and modified to suit the owner, the vulnerability can be fixed and the device "re-locked." *See, e.g.,* scotty85, comment on *How to Relock Bootloader*, AndroidForums (Jan. 20, 2012), http://androidforums.com/threads/how-to-relock-bootloader.486704/ (last visited Feb. 4, 2015).

[111] *See System Permissions*, Android Developers, http://developer.android.com/guide/topics/security/permissions.html (last visited Feb. 4, 2015).

grant an app all of the permissions it requests or refrain from installing it—in other words, there is no way for the user to exercise more precise control over permissions. This is important because mobile apps often request more permissions than they actually require, or use some permissions for unnecessary functions that violate the user's privacy (such as a game that surreptitiously reports a user's messaging activity to advertisers).[112]

An Android user who cannot jailbreak must agree to such invasions of privacy, or else refrain from installing the app at all. But the user of a jailbroken device can install the AppOps framework, a software package that allows selective control of the permissions granted to an app.[113]

Some mobile users seek to safeguard their privacy by installing firewall software that restricts the network communications that other software can engage in.[114] Because these programs monitor other programs on a device and access network communications at a low level, they require jailbreaking. For iOS in particular, security enhancements like firewall software have very limited effectiveness without jailbreaking.[115] This is because iOS runs each application in an isolated environment, or "sandbox."[116] While this practice often contributes to the security of the system, it also prevents security-enhancing apps from monitoring the behavior of other, possibly malicious apps or those that don't respect the user's privacy.[117] Apps containing malware have passed through Apple's review process and been distributed through the iTunes App Store.[118] The ability to jailbreak allows for additional levels of defense against malicious software when Apple's own efforts fail.

C. The Ban on Circumvention Prevents Users From Improving the Performance of their Devices By Removing Unwanted Software.

Mobile devices are sold with various kinds of software pre-installed by the manufacturer and wireless carrier, often designed to drive traffic to particular subscription services or advertising networks. A Time Magazine columnist reported that his Samsung Galaxy S5 phone, running Android and sold by Verizon Wireless, contained three redundant text messaging apps, of which he used only one.[119] An HTC Android phone, he noted, ships with two redundant Web browsers, as well as software that repeatedly asks the user to

---

[112] *See* Alan Henry, *Why Does This Android App Need So Many Permissions?*, LifeHacker (Mar. 18, 2013), http://lifehacker.com/5991099/why-does-this-android-app-need-so-many-permissions.

[113] Gary Sims, *App Ops – what you need to know*, Android Authority (Dec. 16, 2013), http://www.androidauthority.com/app-ops-need-know-324850/. This feature was included by Google in Android 4.3 but removed in Android 4.4.2 and is now available only through independent distributors.

[114] *See, e.g.*, *Firewall iP*, Cydia, http://cydia.saurik.com/package/com.yllier.firewall/ (last visited Feb. 4, 2015); *see also* Rogers Statement at 4 ("There is a growing demand for mobile device firmware designed for high-security operation.").

[115] Rogers Statement at 1.

[116] *Id.*

[117] *Id.*

[118] *Id.*

[119] Jared Newman, *Friday Rant: The Ever-Sorrier State of Android Bloatware*, Time (May 9, 2014), http://time.com/94646/android-bloatware/.

install a "browser bar" containing links to advertisers.[120] Tablets and other devices are sold with similar pre-installed software.[121]
This software, which mobile users often refer to as "bloatware," takes up valuable space in a device's memory. A class action lawsuit recently filed against Apple highlighted that iOS devices advertised as having 16 gigabytes of storage are in fact shipped with as much as 23% of their capacity filled with pre-installed, often unwanted software.[122] The complaint also noted that when a device's storage is nearly full, iOS attempts to sell the owner additional "cloud" storage, for a fee.[123] Android devices suffer from this problem as well. A phone made by LG with 8GB of storage left only 3.8GB free for music, video, photos, and apps selected by the owner.[124]

Bloatware can also slow down a device, both at startup and during normal operation, by running in the background as the user runs other applications. For example, a Verizon Droid 4 evaluated by Dr. Gillula came pre-installed with an Internet radio application, a software backup assistant, a personal task manager, the Google Play video player, and a turn-by-turn navigation app that requires a separate monthly subscription.[125] All of these apps began running when the phone was turned on, consuming processor time and other system resources. In some cases, bloatware can contain malicious code,[126] or send personal information to third parties.[127] Appendix A contains a list of software pre-installed on a Verizon Droid 4 and the personal information that each of these programs has access to.

Bloatware is often configured by the manufacturer or carrier to be non-removable by the user.[128] This means that the only way for the user to avoid the storage, performance, and security problems caused by bloatware is to jailbreak the device.

---

[120] *Id.*

[121] Adam, *How can I get rid of bloatware on my tablet?*, TechGuy Labs, http://www.techguylabs.com/episodes/1006/how-can-i-get-rid-bloatware-my-tablet (last accessed Feb. 5, 2015); Chris Smith, *New Samsung ad says bloatware makes Galaxy tablet buyers happy*, BGR (Jul. 28, 2014), http://bgr.com/2014/07/28/galaxy-tab-s-bloatware/; Andy Baryer, *How to remove bloatware on the Samsung Galaxy Note 4*, CNet (Oct. 20, 2014), http://www.cnet.com/how-to/how-to-remove-bloatware-on-the-samsung-galaxy-note-4/.

[122] Samuel Gibbs, *Apple faces lawsuit over storage space on iPhones and iPads*, The Guardian (Jan. 2, 2015), http://www.theguardian.com/technology/2015/jan/02/apple-lawsuit-storage-space-iphones-ipad-ios8-software-advertised-capacity.

[123] *Id.*

[124] Eugene Kim, *LG G Vista Review*, PC Mag, http://www.pcmag.com/article2/0,2817,2465558,00.asp (last visited Feb. 4, 2015).

[125] Appendix A, Part 4.

[126] Rogers Statement at 3 (describing the "Death Ring" malware, which was installed on devices somewhere in the supply chain before retail sale).

[127] Preston Gralla, Want to protect your Android phone? Here's how to kill its crapware, IT World (Nov. 6, 2014), http://www.itworld.com/article/2833289/security/want-to-protect-your-android-phone--here-s-how-to-kill-its-crapware-.html.

[128] Anna Scantlin, *Non-removable bloatware still plagues Android*, Phone Bill (Jan. 24, 2014), http://www.phonedog.com/2014/01/24/non-removable-bloatware-still-plagues-android.

D. The Ban on Circumvention Inhibits Speech and Innovation.

Access controls that limit the functionality of software, or that allow only manufacturer-approved software to run, inhibit speech and innovation when device owners cannot opt out of the restrictions by jailbreaking.

Apple excludes software from its iTunes App Store (and thus from all non-jailbroken iOS devices) based on Apple's own, unreviewable decisions about an app's expressive content. For example, Apple has excluded a game with marijuana-related content,[129] a game that depicts the ongoing civil war in Syria,[130] an app that reports the locations of U.S. military drone strikes,[131] and a dictionary app (reportedly because it contained objectionable words).[132] Apple also rejected an app for searching photos because it made searching for nudity "too easy" despite containing a "safe search" mode to exclude such results.[133] While Apple is free to exercise editorial discretion over the software it sells through its App Store, the access controls in iOS take away the *user's* discretion to access the creative expression and functionality of their choosing. They also inhibit app developers' ability to reach millions of customers, including through highly creative and communicative software like games.

Device manufacturers also reject apps that compete with their own offerings. For example, both Apple and Google reject applications that use payment systems run by other companies for the purchase of digital goods.[134] Apple rejects competing Web browsers, cloud storage services, app choosers, and home screen alternatives from the iTunes Store, and thus from all non-jailbroken iOS devices.[135]

---

[129] Casey Johnson, *Apple pulls popular weed-growing game from App Store*, Ars Technica (May 22, 2014), http://arstechnica.com/gaming/2014/05/apple-pulls-popular-weed-growing-game-from-app-store/.

[130] Matt Martin, *Apple rejects game based on Syrian conflict*, Gamesindustry.biz (Jan. 8, 2013) http://www.gamesindustry.biz/articles/2013-01-08-apple-rejects-game-based-on-syrian-conflict.

[131] Zachary Knight, *Apple Feels Reporting Drone Strikes 'Objectionable And Crude' And Rejects App*, Techdirt (Aug. 31, 2012), https://www.techdirt.com/blog/wireless/articles/20120830/14470520223/apple-feels-reporting-drone-strikes-objectionable-crude-rejects-app.shtml.

[132] Mike Masnick, *Apple Now Censoring A Dictionary iPhone App?*, Techdirt (Aug. 6, 2009), https://www.techdirt.com/articles/20090805/1832305780.shtml.

[133] Tim Cushing, *iNanny: Apple Takes Down Popular Photo Apps Because They Made Searching For Nude Photos 'Too Easy'*, Techdirt (Jan. 23, 2013).

[134] *App Store Review Guidelines*, Apple, https://developer.apple.com/appstore/resources/approval/guidelines.html; Mike Masnick, *Insanity: Apple Rejects Podcatching App Because It Has Flattr Integration*, Techdirt (Jun. 1, 2012), https://www.techdirt.com/blog/wireless/articles/20120529/03062619097/insanity-apple-rejects-podcatching-app-because-it-has-flattr-integration.shtml; *Google Play Developer Program Policies*, Google Play, https://play.google.com/about/developer-content-policy.html.

[135] Mike Masnick, *Apple Rejecting Apps That Use Dropbox Because *Gasp!* Users Might Sign Up For Dropbox Accounts*, Techdirt (May 2, 2012), https://www.techdirt.com/articles/20120501/17545618733/apple-rejecting-apps-that-use-dropbox-because-gasp-users-might-sign-up-dropbox-accounts.shtml; Steve Kovach, *Frustration Builds With Apple's Inconsistent Rules For App Developers*, Business Insider (Apr. 13, 2013), http://www.businessinsider.com/the-story-of-apples-confusing-inconsistent-rules-for-app-developers-2013-4; *Is Firefox available for iPhone or iPad?*, Mozilla Support, https://support.mozilla.org/en-US/kb/is-firefox-available-iphone-or-ipad.

Many popular software programs that don't conform to the concept of a self-contained "app" cannot be run on iOS or Android without jailbreaking. Alternate home screen designs and means of selecting apps, improvements to the notification bar or screen, the ability to send and receive text messages without leaving the current app, alternative ways of starting up the camera function quickly, and other such changes are generally impossible without jailbreaking.[136] Yet these types of modifications are in high demand, in order to personalize a device and make the owner's most frequent tasks more efficient and readily available.[137]

Jailbreaking is also a necessary part of major software development projects. For example, many versions of Android do not allow developers to use an important tool called a "native code debugger" without the root privileges obtained by jailbreaking.[138] Without a debugger, it is difficult to diagnose software flaws when testing a program on multiple devices.[139] Other software development tools also require root privileges. For example, the Mozilla project, which develops the Firefox Web browser and Firefox OS for mobile devices, wrote software to simulate user input on a device.[140] While Android provided a tool for simulating user input, it did not meet Mozilla's needs. The ability to jailbreak allowed Mozilla engineers to build and use a better tool for testing their software before commercial release.[141]

The independent software development communities that have emerged under the protection of an exemption for phone jailbreaking are a major source of creativity and innovation in the mobile software field. Numerous features invented initially by independent developers, and initially requiring a jailbreak, have since been adopted into manufacturer-sanctioned mobile operating systems and are now considered integral features. CyanogenMod contributors built the ability to access common settings from the pull-down notification area in 2010; the same functionality was included in official Android versions in 2012.[142] On iOS, popular features like always-on voice recognition, interactive alerts, and alternative keyboards originated in the Cydia ecosystem and were later incorporated into iOS itself.[143] Appendix A contains examples of other Android features that originated in the jailbreaking community.

---

[136] Freeman, *supra* note 19, at 3:09.
[137] Britta Gustafson, *Why did you jailbreak your iPhone?"* (Mar. 6, 2014), https://www.youtube.com/watch?v=Te-uIolpNqA (last visited Feb. 4, 2015).
[138] Willcox Statement at 1.
[139] *Id.*
[140] *Id.* at 2.
[141] *Id.*
[142] *See* Appendix A, Part 1.
[143] Joe Rossignol, *15 jailbreak tweaks that iOS 8 made obsolete*, iDownloadBlog (Jun. 3, 2014), http://www.idownloadblog.com/2014/06/03/15-jailbreak-tweaks-that-ios-8-made-obsolete/; Luke Villapaz, *Apple iOS 8 Features Make Several Jailbreak Tweaks Obsolete With Custom Keyboards, Interactive Notifications And Touch ID*, International Business Times (Jun. 3, 2014), http://www.ibtimes.com/apple-ios-8-features-make-several-jailbreak-tweaks-obsolete-custom-keyboards-interactive-1593829.

E. The Ban on Circumvention Enforces Device Obsolescence, Leading to Waste.

As noted above, Android manufacturers and wireless carriers stop sending firmware updates to a phone after one, two, or (very rarely) three updates over a span of one to two years.[144] Although the device *hardware* will often have a much longer useful lifespan, the firmware becomes obsolete as well as insecure. Without the ability to jailbreak, a customer's only recourse is to acquire a new device. Electronics waste is a serious and growing environmental problem that is alleviated by the ability to update device firmware.[145] Because a jailbroken Android device can often run the most up-to-date version of Android, its useful life can be extended.

7. The Nonexclusive Factors of Section 1201(a)(1)(C) Support Granting An Exemption.

A. The Availability for Use of Copyrighted Works

In considering this statutory factor, the Register examines whether "the availability for use of copyrighted works would be adversely affected by permitting an exemption." The Register also "consider[s] whether a particular [non-infringing] use can be made from another readily available format when the access-controlled digital copy of that 'work' does not allow that use."[146]

It can scarcely be questioned that mobile devices, device firmware, and mobile applications of all kinds are enjoying a golden age. These devices are ubiquitous in the U.S. and the variety and quality of software available for them, including operating system options, continues to grow. For the past five years, the existence of exemptions to allow jailbreaking of phones without legal uncertainty have coincided with the meteoric rise of the mobile software industry.[147] Tablets and other mobile devices are regularly jailbroken as well, despite the legal uncertainty created by Section 1201(a), but without adverse impact on mobile platforms.

The availability of software for mobile devices would not be adversely affected by granting an exemption that allows users to jailbreak their devices to enable interoperability. The Register previously agreed that jailbreaking to allow for interoperable software would increase the availability of applications for smartphones "while simultaneously being unlikely to interfere with the availability of smartphone operating systems or other works currently being used or created for wireless communications devices."[148] The same holds true for other multipurpose devices.

---

[144] Casey Johnson, *The checkered, slow history of Android handset updates*, Ars Technica (Dec. 21, 2012), http://arstechnica.com/gadgets/2012/12/the-checkered-slow-history-of-android-handset-updates/.
[145] *E-waste is the Toxic Legacy of our Digital Age*, IFIXITORG, http://ifixit.org/ewaste.
[146] Recommendation of the Register of Copyrights in RM 2005-11, at 21-22, Rulemaking on Exemptions from Prohibition on Circumvention of Copyright Protection Systems for Access Control Technologies (Nov.17, 2006) [hereinafter 2006 Recommendation].
[147] *See* Part 3, *supra*.
[148] 2010 Recommendation at 102.

The ability to jailbreak has never been shown to contribute significantly to copyright infringement. Android devices, whether jailbroken or not, have long given users the ability to load application software from any source.[149] Jailbreaking an Android device, which simply gives programs on the device *more capabilities*, and more ability to interoperate with other programs, does not facilitate the use of infringing software. For iOS, the availability of thousands of popular software packages that require jailbreaking and which are licensed as free and open source software, as well as proprietary software sold commercially through markets like Cydia,[150] demonstrate that the ability to jailbreak does not encourage infringement.

Nor does jailbreaking contribute to infringement of media such as video and e-books on a mobile device. These media are typically protected by their own proprietary digital rights management (DRM), separate from those in the bootloader and OS. For example, video and apps purchased through the Google Play Store, iTunes Store, and Amazon.com are subject to specialized DRM mechanisms that reside in part on the servers of the digital content marketplace.[151] Jailbreaking does not circumvent this type of access control.

The lack of an exemption would likely decrease the appeal of mobile devices for many consumers and innovators. Without an exemption, users concerned about § 1201 liability will be narrowly confined to the functionality of applications distributed only through authorized channels, and will be unable to avail themselves of the many kinds of third party applications currently on the market. And fewer features and innovations arising from the independent developer community will find their way into manufacturer-authorized software.

Recognizing the importance of the ability to jailbreak, and underscoring its beneficial effects on the markets for mobile computing firmware and software, two device manufacturers, Nexus and HTC, now provide straightforward means of jailbreaking their devices.[152] While encouraging, this development does not eliminate the adverse effects of the ban on circumvention, for several reasons. The Register's speculation in 2012 that "[p]erhaps in the ensuing three years … unlocked devices will become the rule rather than the exception" has not come to pass.[153] The vast majority of mobile devices sold in the U.S. cannot be jailbroken without circumventing access controls. For those devices, circumvention is a necessary step in avoiding the security and performance problems described above, and for adding functionality. In addition, some hardware features may only be available on devices that require jailbreaking through a security vulnerability. Replacing one's device with a model that can be jailbroken with the manufacturer's

---

[149] Jerry Hildenbrand, *What is Sideloading?*, Android Central (Feb. 2, 2012), http://www.androidcentral.com/what-sideloading-android-z. .

[150] *See supra* n.38; *see* Bob Bhatnagar, *How to Purchase / Install iOS Jailbreak Apps from Cydia*, iPhoneFaq, (Feb. 24, 2013), http://www.iphonefaq.org/archives/972432.

[151] Gillula Statement at 3.

[152] Joseph Volpe, *Galaxy Nexus gets rooted, forums burst into applause*, engadget (Nov. 3, 2011), http://www.engadget.com/2011/11/03/galaxy-nexus-gets-rooted-forums-burst-into-applause/; *Unlock Bootloader*, hTCdev, http://www.htcdev.com/bootloader/.

[153] 2012 Recommendation at 76 (referring to devices that can be jailbroken on demand with the manufacturer's and carrier's blessing).

permission often means waiting months or years for the end of a mobile service contract, or paying a substantial early termination fee. A device that could serve its owner's needs if jailbroken may instead end up in a landfill.

Moreover, even those devices that can be jailbroken with a simple tool provided by the manufacturer when purchased directly from the manufacturer generally *cannot* be jailbroken in this manner when purchased from a wireless carrier.[154] Without an exemption, under some court precedents, the wireless carrier may have standing to bring (or threaten) an action for violation of § 1201(a)(1), even if the carrier does not hold copyright in the firmware.[155]

B. The Availability for Use of Works for Nonprofit Archival, Preservation, and Education Purposes

The availability of mobile device firmware for nonprofit purposes will not be harmed by an exemption that permits jailbreaking to enable interoperability. Consistent with the Register's prior conclusions regarding smartphones,[156] this factor is not relevant.

C. The Impact on Criticism, Comment, News Reporting, Scholarship or Research

An exemption that permits smartphone users to jailbreak their devices would improve the availability of copyrighted works for criticism, comment, news reporting, teaching, scholarship, and research. Mobile device jailbreaking has spurred both valuable commentary and important security research. For example, independent developers and researchers needed the ability to jailbreak to understand and fix many of the vulnerabilities described above.[157] Also, as described above, content-based editorial decisions by mobile software markets like Apple's App Store sometimes exclude software that expresses political commentary. A renewed exemption to permit jailbreaking would help device owners access the criticism and commentary of their choosing. As found in prior rulemakings, this factor also favors an exemption.[158]

D. The Effect on the Market for, or Value of, Copyrighted Works

As we explained in our analysis of the fourth fair use factor, allowing users to jailbreak mobile devices will have no negative impact on the actual market for the firmware on such devices. Instead, the proposed exemption is likely to stimulate the market for such

[154] *See, e.g. Frequently Asked Questions*, HTCDev, http://www.htcdev.com/bootloader/faq ("[C]ertain models may not be unlockable due to operator restrictions.").
[155] *Echostar Satellite, L.L.C. v. Viewtech, Inc.*, 543 F. Supp. 2d 1201, 1205 (S.D. Cal. 2008); *CoxCom, Inc. v. Chaffee*, No. CIVA 05-107S, 2006 WL 1793184, at *11 (D.R.I. June 26, 2006); *Comcast of Illinois X, LLC. v. Hightech Electronics, Inc.*, No. 03 C 3231, 2004 WL 1718522, at *6 (N.D. Ill. July 29, 2004).
[156] 2010 Recommendation at 101; 2012 Recommendation at 77.
[157] *See also* Peter Eckersley and Jeremy Gillula, *Is Your Android Device Telling the World Where You've Been?*, Electronic Frontier Foundation (July 21, 2014), https://www.eff.org/deeplinks/2014/07/your-android-device-telling-world-where-youve-been (describing a vulnerability in Android that required root privileges to diagnose).
[158] 2010 Recommendation at 102; 2012 Recommendation at 77.

works by providing developers with incentives to develop third party applications, thus making these devices—together with their copyrighted firmware—more attractive to consumers. The ability to develop and use independent applications on mobile devices increases the value of the devices and their firmware. Jailbreaking has also spurred new and vibrant markets for mobile software in general, including Cydia and the CyanogenMod community. Microsoft Corporation recently made a $70 million investment in CyanogenMod which values the company in the hundreds of millions.[159] This value arose from the demand for independently created mobile software that requires jailbreaking.

E. Other Factors

Manufacturers do not put access controls on mobile devices to protect the copyrighted firmware. Rather, those controls exist in part to preserve various aspects of the manufacturers' and mobile carriers' business interests—interests the Register has already determined to be unrelated to infringement. In both 2006 and 2010, the Register frowned on firmware manufacturers advancing copyright claims in their functional computer programs to support anti-competitive business practices. The Register recognized in 2006 that

> when application of the prohibition on circumvention of access controls would offer no apparent benefit to the author or copyright owner in relation to the work to which access is controlled, but simply offers a benefit to a third party who may use § 1201 to control the use of hardware which, as is increasingly the case, may be operated in part through the use of computer software or firmware, an exemption may well be warranted.[160]

Again in 2010, she stated that

> while a copyright owner might try to restrict the programs that can be run on a particular operating system, copyright law is not the vehicle for imposition of such restrictions, and other areas of the law, such as antitrust, might apply. It does not and should not infringe any of the exclusive rights of the copyright owner to run an application program on a computer over the objections of the owner of the copyright in the computer's operating system.[161]

The same analysis supports the granting of an exemption in favor of mobile device owners who want to run lawfully obtained software of their own choosing. Granting the exemption will not impair the legitimate copyright interests of those who create the firmware. At the same time, an exemption would vindicate the strong public interest in

[159] Davey Alba, *Microsoft to Invest in Android Startup Cyanogen, Says Report*, WIRED (Jan. 29, 2015), http://www.wired.com/2015/01/microsoft-invest-android-startup-cyanogen-says-report/.
[160] 2006 Recommendation at 152.
[161] 2010 Recommendation at 96-97.

fostering competition in the software market, thereby encouraging innovation and expanding consumer choice.

8. Documentary Evidence

Please see the appendix filed with these comments.

Appendix A
Supplemental Material on Jailbreaking
Compiled by Dr. Jeremy Gillula

1. The following are popular features that were implemented by developers in the CyanogenMod jailbreaking community and were later incorporated into official Android releases:
    - Power widgets and settings accessible directly from the pull-down notification area were implemented in Cyanogenmod 6.0.0, based on Android 2.2.X (Froyo), on August 28, 2010. The feature was later introduced into Android 4.2 (Jelly Bean), which was released November 13, 2012.
    - A rotary lockscreen with the ability to unlock and immediately launch specific apps (camera, messaging, email etc.) was implemented in Cyanogenmod 7.0.0, based on Android 2.3.3 (Gingerbread) on April 10, 2011. The feature was made part of Android 4.0 (Ice Cream Sandwich), released on October 18, 2011.
    - The ability to dismiss individual notifications from the notification area by swiping them was implemented in Cyanogenmod 6.1.0, based on Android 2.2.1 (Froyo) on December, 6, 2010. It was also introduced into Android 4.0 (Ice Cream Sandwich), released on October 18, 2011.

2. The following are examples of security vulnerabilities that affect older versions of Android and have been fixed in subsequent releases. Some devices retain these vulnerabilities because the manufacturer and carriers have ceased to send updates
    - A bug in the built-in Android Web browser (and any app that uses the built-in WebView component to, e.g., display web pages from within the app), which allows malicious sites to access cookies and other info from other sites users have visited. This bug could allow, for example, a malicious website to steal an identity token used by a different website and use it to impersonate the user)
        i. News article: http://arstechnica.com/security/2014/09/android-browser-flaw-a-privacy-disaster-for-half-of-android-users/
        ii. Patches in official Android source code:
            1. https://android.googlesource.com/platform/external/webkit/+/1368e05e8875f00e8d2529fe6050d08b55ea4d87
            2. https://android.googlesource.com/platform/external/webkit/+/7e4405a7a12750ee27325f065b9825c25b40598c
    - The POODLE vulnerability, announced in September 2014 was fixed in CyanogenMod 11 M12 on 11/13/2014 (*see* http://www.cyanogenmod.org/blog/cyanogenmod-11-m12).
    - Many vulnerabilities affecting old versions of Android (earlier than version 3) which are fixed in modern versions. Many of the phones listed in part 3 below, and the tablets in part 4, remain vulnerable to these vulnerabilities if they cannot be jailbroken:

i. http://www.cvedetails.com/cve/CVE-2010-4832/
ii. http://www.cvedetails.com/cve/CVE-2011-1149/
iii. http://www.cvedetails.com/cve/CVE-2010-4804/
iv. http://www.cvedetails.com/cve/CVE-2011-0680/
v. http://www.cvedetails.com/cve/CVE-2011-1350/
vi. http://www.cvedetails.com/cve/CVE-2011-1352/
vii. http://www.cvedetails.com/cve/CVE-2011-1823/
viii. http://www.cvedetails.com/cve/CVE-2011-2357/
ix. http://www.cvedetails.com/cve/CVE-2011-3874/
x. http://www.cvedetails.com/cve/CVE-2011-4276/
xi. http://www.cvedetails.com/cve/CVE-2012-4221/

- Finally, a bug in the pseudo-random-number-generator (PRNG) in the widely used cryptography library OpenSSL, which provides secure Web browsing and e-commerce, was fixed in Android 4.4 but remains in earlier versions. *See* http://www.cvedetails.com/cve/CVE-2013-7373/

3. This is a chart of popular models of smartphone, showing the first and last versions of Android supplied through the manufacturer's authorized channels, and the version of Android that can be installed on a jailbroken device. For example, the table shows that a non-jailbroken Samsung Captivate can only run Android 2.3, released December 6, 2010, while the same device can run a derivative of Android version 4.4, released on October 31, 2013. Jailbreaking would allow a Samsung Captivate owner to enjoy the fruits of 35 additional months of Android development.

Phone Name	**Release Date**	**Initial Android Version**	**Final Version Supported by Manufacturer**	**Version Usable After Jailbreak**
Motorola Droid 3 (Notable for hardware keyboard)	Jul. 14, 2011	2.3 (Gingerbread) Dec. 6, 2010	2.3 (Gingerbread) Dec. 6, 2010	4.2 (Jelly Bean) Nov. 13, 2012
Motorola Droid Razr	Nov. 11, 2011	2.3 (Gingerbread) Dec. 6, 2010	4.1 (Jellybean) July 9, 2012	4.4 (KitKat) Oct. 31, 2013
Samsung Captivate	June 4, 2010	2.1 (Eclair) Jan. 12, 2010	2.3 (Gingerbread) Dec. 6, 2010	4.4 (KitKat) Oct. 31, 2013
HTC Droid Incredible	Apr. 29, 2010	2.1 (Eclair) Jan. 12, 2010	2.3 (Gingerbread) Dec. 6, 2010	4.4 (KitKat) Oct. 31, 2013
LG Optimus 2X	Dec. 16, 2010	2.2 (Froyo) May 20, 2010	4.0 (Ice Cream Sandwich) Oct. 18, 2011	4.2 (Jelly Bean) Nov. 13, 2012

This is a chart of popular tablets, showing the same information:

Tablet Name	**Release Date**	**Initial Android Version**	**Final Version Supported by Manufacturer**	**Version Usable After Jailbreak**
Acer A700	June 13, 2012	4.0 (Ice Cream Sandwich)	4.0 (Ice Cream Sandwich) Oct. 18, 2011	4.4 (KitKat) Oct. 31, 2013
Asus Eee Pad Transformer	Apr. 26, 2011	3.1 (Honeycomb) May 10, 2011	4.0 (Ice Cream Sandwich) Oct. 18, 2011	4.3 (Jelly Bean) July 24, 2013
Samsung Galaxy Tab	Sep. 2, 2010	2.2 (Froyo) May 20, 2010	2.3 (Gingerbread) Dec. 6, 2010	4.1 (Jelly Bean) July 9, 2012
Samsung Galaxy Note	Apr. 29, 2010	4.1 (Jelly Bean) July 9, 2012	4.1 (Jelly Bean) July 9, 2012	4.4 (KitKat) Oct. 31, 2013

4. This is a list of some of the pre-installed software on a Verizon Droid 4.
    - Software that people might not want for privacy reasons, but which cannot be uninstalled without root privileges:
        i. Facebook, which hase access to contacts, call log, location, camera, audio
        ii. NFL Mobile, which can send SMS; read SMS, location, and phone ID; view Wi-Fi networks in range; and view what other apps are running
        iii. Slacker Radio, which runs at startup, can read phone ID, receive data from the Internet, and view Wi-Fi networks in range
        iv. Amazon Kindle, which can read what accounts are on the device, and view Wi-Fi networks in range
        v. Forest Wallpaper, which can read GPS location
        vi. Google+, which can access contacts, accounts, location, ID, audio, and video; download files without notifying the user; etc.
    - Software that runs at startup (and thus slows the device down)
        i. Slacker radio
        ii. Verizon's backup assistant
        iii. Tasks
        iv. Google Play Movies & TV
        v. VZ Navigator (Verizon's custom map app you have to pay to use)

Statement of Dr. Jeremy Gillula

February 6, 2015

My name is Jeremy Gillula. I am a Staff Technologist at the Electronic Frontier Foundation, where my duties include developing privacy-enhancing technologies (including for mobile devices); analyzing new products and services at a technical level for their impact on privacy, civil liberties, and innovation; and educating and explaining how new technologies work so that my non-technical colleagues (and the general public) can gain a better understanding of their operation. As a technologist I am intimately familiar with the theory and practice of computer science, including the workings of mobile operating systems. Prior to working at EFF, I obtained my doctorate and master's degrees in Computer Science from Stanford University, and a bachelor's degree in Computer Science from Caltech.

I am submitting this statement to the Copyright Office to support a continued exemption to the ban on circumventing access controls in order to root/jailbreak mobile phones, as well as in support of a new exemption for jailbreaking/rooting other mobile devices, such as tablets. (For the purposes of this statement I will refer to mobile phones and other devices which run mobile operating systems collectively as mobile devices.) In this statement I will explain from a technical perspective how the process of jailbreaking/rooting may circumvent access controls.

There are essentially three separate (but similar) terms used when it comes to circumventing access controls on mobile devices:

- "Rooting" refers to the process by which one acquires privileged access (root) on the mobile device's operating system (OS). As with desktop OSes like Windows, OS X, and Linux, mobile OSes like iOS and Android typically separate users into two classes: regular users and administrators or "root" users. Apps running as a regular user can only access a limited subset of the OS's functionality, while programs running with root privileges can effectively access or modify any file within the OS.
- "Jailbreaking" is a similar process to rooting, but is typically used only to refer to such a process on non-Linux-based devices (e.g. iOS). Jailbreaking typically allows a user to circumvent some of the access controls on the mobile device, but does not provide them complete access to modify the OS at will. (For example, jailbreaking an iOS device allows one to install third party apps, but not to change most other low-level behavior of iOS.) In this sense, jailbreaking is a more limited form of rooting. Despite this, rooting and jailbreaking are very similar, and most of the occurrences within this statement of one of these terms could be substituted with the other.
- "Unlocking a bootloader" refers to a process by which one modifies low-level firmware on a mobile device to allow it to boot an OS of the user's choosing.[1] In order to actually

[1] A confusingly similar term, "unlocking," refers to the process by which one modifies a device so that it can be used on a different mobile carrier than the one it was originally programmed for (e.g. "unlocking" a phone that was

load an OS into the device's memory so that it can be booted, mobile devices make use of low-level system code called a bootloader (much like the BIOS on a PC). A "locked" bootloader will typically verify that the OS it is booting has not been modified, by checking its cryptographic signature. If the code comprising the OS has been changed, the bootloader will restore the original OS from a read-only source. In order to load a different OS (such as CyanogenMod or another customized version of Android) one must first disable this verification. This process is called "unlocking the bootloader."

Most mobile devices are sold with a locked bootloader and without the ability of the consumer to gain root privileges. As a result, the only way to gain root access to a mobile device or to unlock its bootloader is to find a security vulnerability (also known as an exploit) in the code the device runs, either at the OS level or at the bootloader level.[2] In both cases, for the exploit to be successful it has to allow the user to run an arbitrary program with the capability of modifying the underlying OS or the bootloader. (Remember: normally the user is prevented from running programs that can modify the OS or the bootloader.)

To illustrate in more detail how such an exploit works, I will describe at a high level the inner workings of the "Gingerbreak" exploit, which was discovered in April 2011 and allows users to gain root access to Android devices running Android version 2.3 (AKA Gingerbread).[3]

Gingerbreak works as follows:

1. In Android 2.3, a bug exists in the part of the OS which is responsible for managing SD cards (storage media that can be installed in a mobile device). This part of the OS must run with root privileges in order to do its job, but also must be able to be triggered and take inputs from a normal user in order to allow users to mount and unmount SD cards.
2. Because of the particular structure of this bug, a user can carefully craft an input to this buggy part of the OS which will cause the OS to run any program the user likes. (If the OS did not have the bug, this would not be possible: this part of the OS would only do what it was designed to do, manage SD cards, and would reject any invalid input.)
3. Since this part of the OS runs with root privileges, the program the user chooses will also run with root privileges.
4. Gingerbreak exploits this vulnerability by creating an input which takes advantage of the bug and causes the OS to run a program which will permanently install an executable called "su" (short for "super-user," another term for root). Normally a regular user would be unable to install "su," because installing it requires access to parts of the OS that

---

originally sold for use on AT&T's network so that it can be used on T-Mobile's network). "Unlocking" is not a necessary precondition for rooting, jailbreaking, or bootloader-unlocking, or vice versa.

[2] Some manufacturers are beginning to change this policy. For example, Google Nexus devices allow users root access without needing to first take advantage of a security vulnerability. Devices that allow such freedom remain the exception rather than the rule, however.

[3] For more technical detail, see https://xorl.wordpress.com/2011/04/28/android-vold-mpartminors-signedness-issue/

require root privileges to modify. However, per the previous step, the program that installs "su" is running with root privileges via the vulnerability in the OS.

5. "su" is a program that can be run by a normal user. It takes as input the location of a program to run, and then runs that program with root privileges.
6. As a result, once the Gingerbreak code is run, any time in the future a normal user wishes to run a program that needs access to root privileges, they simply call that program via "su." Thus, the user now has complete access to root privileges on their mobile device.

This pattern is typical for root exploits: a part of the OS is found that takes input from the user, but which also contains a bug which allows the user to run an arbitrary program. By carefully crafting the input, the user gets the OS to run a program that modifies the OS so that the user can easily make use of root privileges in the future.

Jailbreaking typically follows a similar pattern. Additionally, root access typically allows one to modify the bootloader. Thus, once a mobile device is rooted, one can typically unlock the bootloader, and then load a modified OS of one's choosing.[4]

In this way, rooting, jailbreaking, and bootloader-unlocking require the circumvention of access controls. These processes take advantage of vulnerabilities in the OS to run code that the manufacturer or carrier did not intend to be run, allowing the user to make changes that are not authorized by the manufacturer or carrier—changes that allow users much greater freedom in the use of their mobile devices as well as enhanced security.

While jailbreaking and rooting do provide device owners with a wide latitude over what software can run on their devices, that control is not absolute. For example, the process of jailbreaking/rooting does not circumvent the access controls associated with digital markets such as Apple's App or iTunes Stores, or Android's Google Play or the Amazon Appstore. This is because part of the access controls associated with these markets resides on the servers of the associated companies. In order to acquire a paid app, movie, etc., from one of these markets, the device sends a request to the server running the given market. Software running on that server then checks to see if the request is valid (usually by checking a database to see if payment has been successfully processed for the user's account), and if so, provides the device with the appropriate download. If the server determines that the request is invalid, the user simply cannot download the requested content. (Additionally, some of these markets periodically verify that a user is still authorized to run a given app; without the check, the app will no longer function.) Because rooting/jailbreaking only affects the software running on the device, and does not affect the software running on the market's server, there is no way for a user to acquire content without permission as a direct result of the rooting/jailbreaking process—the two issues are simply not connected.

[4] The reverse is also possible. If a vulnerability in the bootloader allows the user to run arbitrary code, the user can unlock the bootloader, and then run a custom OS that is already designed to give the user root access.

Statement of Marc Rogers
February 6, 2015

My name is Marc Rogers. I hold the position of Principal Security Researcher at CloudFlare, Inc., an Internet Security Company and "Content Delivery Network" or CDN. A few years prior to my tenure in CloudFlare I worked for Vodafone UK PLC, a global telecommunications company and mobile network operator, where I was responsible for architecting and ensuring the security of Internet-facing or mobile device-facing content services. As well as being responsible for the security of these mobile device services, I was also responsible for ensuring the security of the devices themselves.

I am submitting this statement to the Copyright Office to support a continued exemption to the ban on circumventing access controls in order to jailbreak mobile phones, and a new exemption for jailbreaking tablets.

As a developer, I consider smartphones and tablets to be variations on the same basic device. From a hardware perspective, only two processor architectures dominate the market, ARM and x86, with the vast majority of mobile devices running on ARM, while, from a software perspective, two mobile device operating systems represent 90% of the market on both phones and tablets – Android, and iOS.

Modern smartphones and tablets are feature-rich but are not designed for users with high security requirements. Apple's iOS contains access controls that prevent developers from accessing lower-level functionality of the device. Because of this limitation, there is little or no research and innovation in iOS security taking place outside of Apple. As a result, vulnerabilities in iOS can persist (while possibly being exploited by wrongdoers) before the security community learns of them and calls for a fix. One of the most serious examples of these persistent vulnerabilities was the Apple SSL vulnerability known as "Goto fail." This vulnerability was a flaw in the Secure Sockets Layer (SSL) code that provides privacy and security for sensitive traffic on the Internet – for example, online shopping. This flaw was introduced in September 2012 but was only identified in February 2014 after third-party security testing. An extremely serious issue, the whole time this flaw existed, it was possible for criminals to impersonate legitimate sites in order to steal information such as personally identifying information ("PII") or cardholder data.

The third-party security tools that do exist for iOS are limited in their usefulness because, like all software for iOS, they cannot access lower-level functionality – something that is necessary to detect many security threats. For example, one of the core security defenses in iOS is the fact that every application runs in its own isolated environment or sandbox. In theory, this helps prevent malicious code from affecting other applications. However, the downside is that security applications are unable to affect the malicious code either. This means malicious code successfully downloaded onto an iPhone will run undetected for an indefinite amount of time. Apple's position that security applications are unnecessary because their system prevents the installation of malicious code is flawed. In September 2013, researchers from Georgia Tech demonstrated using an application called "Jekyll" that it was possible to disguise a malicious app in a way that it could successfully be uploaded to the Apple App Store and subsequently installed

onto any iPhone that downloaded it. Even more recently, in November of 2014, a piece of malware known as "WireLurker" was found that could infect iPhones by exploiting a flaw in Apple's desktop operating system, OSX. iPhone malware clearly exists and at present no security vendor is able to provide any on-device protection without jailbreaking and subsequently rooting the iPhone first. This means the only way to protect unjailbroken users from iOS malware at present is a convoluted and slow process involving a third-party security company identifying the malware, reporting it to Apple, convincing Apple that it is malware and then waiting for Apple to act. This process can take weeks or even months. By contrast, on a device that has been jailbroken, as soon as a third-party vendor identifies malware, it pushes signatures to all the devices running its security software and the malware is instantly disabled.

Researching and improving the security of devices that run the Android operating system also requires low-level access to the device. The security challenges with Android are in some ways the opposite of those facing researchers or security vendors on iOS. On Android, its is possible for a user to install applications from sources other than Google, and it is possible for those applications to be given permission to access and even modify the code of other installed applications. However, this "open" architecture means that malicious code can also be installed. Furthermore it means that this malicious code is able to run at a very low level, and in a way that can affect much of the legitimate code or applications on the device. The only way to get ahead of this problem is to ensure that the security controls or code running on the device run at an even lower level and do so in a way that they cannot be interfered with. Unless security software runs at a sufficiently low level it can be blocked, disabled, fooled and even removed completely by malicious code, or a malicious attacker. For most devices, avoiding those limitations requires the device owner to jailbreak and subsequently "root" the device by bypassing or defeating access controls in the device firmware.

Most modern devices employ a security control known as a "secure bootloader." When the device is powered on, the bootloader is responsible for configuring the device so that it can load the device Operating System ("OS"), and then, subsequently, for loading the OS itself. A secure bootloader is cryptographically signed and locked with a password to prevent modification. It also performs integrity checks before unlocking the device OS so that it can be loaded. This architecture means that unless the security controls on the bootloader are disabled it is not possible to modify the OS or even boot a different OS.

Without bypassing these controls in the bootloader, there is no practical way to install a new operating system on a device or change the operating system to give application programs more capabilities.

In many cases although the device is locked when it reaches the consumer it is unlocked during various stages of the device's manufacturing process so that the device manufacturer and device reseller can customize the device with their own branding and add any bundled free applications. As a result attackers have recently started attacking the Android device supply chain after realizing that by infecting a device manufacturer or reseller they can inject malware into the device firmware while it is unlocked. This is very bad for consumers as while Android security applications are given substantial access to the OS and Applications, they are not given full access to the system firmware. This means that while they can detect malicious code within

the firmware they are unable to do anything about it. There have been several examples of malware like this in the wild, the most recent of which is a piece of malware discovered in December 2014 called "DeathRing." This malicious application allows the people controlling it to remotely control affected phones and steal PII from the legitimate device owner. Because "DeathRing" is installed in the devices' firmware somewhere in its supply chain, the only way to remove it is by jailbreaking the device in order to allow security applications to make changes to the firmware.

Malware is not the only security problem that affects mobile devices. Increasingly, security vulnerabilities are being found in the applications and firmware installed on mobile devices. While ordinary application vulnerabilities can be fixed by an update from the application manufacturer, vulnerabilities in the firmware are a much more complicated problem. In most cases, the only options are to wait for a patch from the device manufacturer or reseller or to jailbreak the device. In the case of older devices or new devices from certain vendors where patches are not provided, jailbreaking is the only way to secure the device.

These limitations lead to security problems. Major device manufacturers and wireless carriers rarely, if ever, send updates to the operating system on a device after it is purchased. Some "whitelabel" devices released by well-known Chinese vendors never even receive a single update. This means that as security vulnerabilities are discovered in different flavors of the Android operating system, many mobile device users will not receive fixes and will remain at risk. Without the ability to bypass access controls in the bootloader, many thousands of devices in use *cannot* be made secure against well-known vulnerabilities.

In the case of the critical vulnerability "Heartbleed" found in April 2014, despite a rare intervention by Google, many devices running version 4.1.1 of Android on non-Google-supported phones are *still* vulnerable to Heartbleed a year later. Heartbleed is an extremely serious vulnerability, which allows an attacker to steal information straight out of the memory of an affected device. The only way to fix the Heartbleed vulnerability on these non-Google supported phones is by jailbreaking the device.

Another example worth noting is the pair of "Same Origin Policy" browser vulnerabilities that were found to affect the web browser in all Android devices earlier than version 4.4. These flaws were described as "a privacy disaster," because they enabled malicious code running on a web page to read data from any other webpage loaded into the same browser, such as pages loaded in other tabs. This meant the malicious page could steal information, such as cookies or credentials, allowing an attacker to hijack banking sessions or read data from secure webmail pages. Discovered on the 1st September 2014, these flaws affected more than 75% of the approximately 1 billion Android devices currently active. A month after Google released its patch around 50% of all Android devices remained vulnerable. This is because, despite Google patching the vulnerability, device manufacturers and resellers still had to take it, process it, and push it out to users before the devices could be patched. This burden on the manufacturer to integrate the patch leads to a phenomenon known as Android fragmentation where some device manufacturers take months to update their custom version of Android while others may never update at all. The only option available to a user in this situation is to jailbreak the device, and then patch or disable the affected software by hand.

There is a growing demand for mobile device firmware designed for high-security operation. The vast majority of vulnerabilities found in mobile devices are in third-party applications or software. By creating a custom version of the firmware that removes these applications, it is possible to massively reduce the attack surface area. As an added bonus, this also significantly improves the performance of the device. Furthermore, for security software to be truly effective, it has to run at the lowest level possible and in a way that can't be tampered with. By building security applications into the firmware part of a mobile device, it is possible to come very close to achieving this. Security software installed in the device firmware cannot be easily disabled by malicious applications or code and, just as importantly, cannot be easily disabled or removed by a malicious third party, such as a phone thief.

Finally, customization of devices, such as changing phone dialer applications, adding enterprise features, or changing what runs on the phone from a performance perspective, also requires low-level or "root" access to a jailbroken phone. This form of customization is increasing in popularity. One such popular customized firmware, known as "cyanogenmod," has been installed on more than 12 million devices, making it the third-largest firmware distribution after regular Android and iOS. Cyanogenmod can only be installed on jailbroken Android phones.

In conclusion, a device owner's ability to "jailbreak" or get "root" access shouldn't be limited to a small number of devices for which the manufacturer provides an easy way to enable it. For most of the millions of devices in use in the U.S., there is no way to access low-level functionality or replace the operating system, leaving the owners of these devices unable to improve their own security or functionality without buying a new, often more expensive, device. Also, many people need secure devices that are not easily identifiable as such. In some regions of the world, possession of readily identified security devices can lead to arrest under spying charges. This means devices manufactured as highly secure with features advertised specifically as a way to avoid government surveillance, such as the "Blackphone," can be extremely dangerous to carry. By applying a secure operating system to a regular mainstream device, a user can carry a device with strong security features into such a region without such a high risk of being arrested.

Statement of James Willcox
February 6, 2015

My name is James Willcox. I hold the position of Staff Platform Engineer at Mozilla where I lead a team working on the Firefox web browser for Google Android devices. I have been a professional software engineer for twelve years, and have experience with development in a variety of environments and platforms.

In this statement, I explain how mobile software development at Mozilla and in our open source development community uses "root" access on mobile devices and why it's important that the community continue to have such access.

It may first be helpful to define what root access means in the context of an Android device. When colloquially referring to root, there are really two different things that people could be referring to. The most common definition means that it's possible to gain administrative access. Dating back to the first UNIX operating systems, "root" is the name of the administrator account on a multi-user system. The root user is able to do things that other unprivileged users are not. Since Android is based on a UNIX-inspired operating system, Linux, that concept applies here. Later on I will give examples of how we use this privileged access to develop software at Mozilla.

The second definition of "root" relates to what will be allowed when the device starts up. One of the first pieces of code that runs when you power on an Android device is called the bootloader. The main objective of this code is to load and run the operating system kernel, which is the heart of the system. The kernel is responsible for controlling and mediating access to the various components of the device (screen, cellular radios, storage, audio, etc). Most, if not all, Android devices on the market are released with a bootloader that will only load a kernel that is cryptographically signed by the manufacturer. This prevents a user from loading his own kernel, even after obtaining the root user access defined above.

Mozilla develops and distributes the Firefox web browser and the Firefox OS mobile operating system. Many aspects of the development process for Firefox and Firefox OS require access to the low-level functionality on different mobile devices. For example, until recent versions of Android, it was not possible to attach a native code debugger to a process without root access. This makes it extremely difficult to find out where bugs (programming errors) occur, and is a normal development tool on other operating systems. This limitation is removed in newer Android versions, but it takes a long time for the phones deployed throughout the world to catch up. Manufacturers of Android devices have a fairly weak record regarding operating system updates, so many users need to buy a new device in order to obtain a newer version. Consequently, if we want to diagnose a problem specific to one of these older devices, we need to acquire root access.

Another example of a way we use root access during development today is for product testing. Manual (human) testing of software can be slow, expensive, and error prone. Automated, computer-controlled, testing is becoming increasingly prevalent for mobile development, and is already very common on other systems. In order to do this, we need to be able to simulate how a user would interact with the device. At Mozilla, we have such a system. We can simulate a user touching the screen and performing various gestures (panning, scrolling, zooming, tapping), and our application responds to those inputs exactly as it would a human. These simulated inputs,

however, do not have the inconsistency, time, and cost incurred by a person doing it, making the tests more effective. While Android does have a facility for simulating input, we found it to be unsuitable for our usage, as the resulting input was treated differently from real user interaction. The solution we have introduces the input events at the operating system level, resulting in the input events being indistinguishable from real ones. This type of simulation requires root (administrative user) access.

As mentioned above, Android devices do not always get timely operating system updates. This is different from personal computer (PC) operating systems like Apple OS X and Microsoft Windows, which get regular updates in order to fix errors or introduce additional features. Frequently, these updates address security vulnerabilities that would allow an attacker to gain unauthorized access to the system. Mobile operating systems also have security vulnerabilities, but without frequent updates these issues cannot be addressed. This leaves the mobile user more open to the myriad of consequences related to a security breach (viruses, data theft or destruction, etc). This is an area where a community-supported operating system can help. Because the Android source code is available, it's possible for developers to build it themselves. If you have root access to a particular device (typically, both the bootloader and administrative user), it is then possible to install that operating system, which could have any change the developer desired -- including security fixes or new features. There exists today a vibrant community doing this for Android devices, many of them with millions of users, none of which would be possible without root access.

Without permission and specific passwords or other secret information from device manufacturers, it is difficult to gain root access to mobile devices for the software development process. Some manufacturers have a "blessed" method for rooting a limited set of their products. Others, like Google's Nexus line of products, are specifically designed to allow this. The majority of Android devices in the world, however, do not have any manufacturer-approved way to gain any type of root access. For those devices, motivated individuals have exploited security vulnerabilities in the operating system in order to gain this access.

Mozilla develops software that runs on both phone handsets and tablets. From my perspective as a developer, there is little difference between these devices, especially in the case of Android. They have very similar hardware characteristics, with the exception of screen size, and all of the same limitations to software development described here apply to tablets as well.