Jay Freeman (saurik)
SaurikIT, LLC (Member)
Developer of Cydia

6935 Phelps Road #27
Goleta, CA 93117
+1 (805) 895-7209

Class: #3, #4, #5

Summary:
As computers have become more complex and expensive to manufacture, we have come to rely on a small handful of companies to build fewer, generalized chip designs that can be programmed by developers to do any task: truly "general–purpose computers". Much of the innovation in software we have seen in the last few decades can be traced back to cheaply available and easily programmable devices that have, thanks to the Internet, had a very low cost of distribution.

Unfortunately, a recent trend in software distribution, the manufacturer–provided "App Store", has caused a misplaced incentive for manufacturers to use the barriers to entry they have on the production of hardware to extend their reach into the software market. They use technological locks to control who is allowed to distribute software and what they are allowed to build. This control is stifling innovation, and is unfortunately being done in the name of copyright, so as to enact anti–circumvention clauses.

These DMCA exemptions that have been requested, on wireless telecommunication handsets and tablets (#3), video game consoles (#5), and then generally all personal computing devices (#4) are in fact only a minimal measure to make certain that when these locks are being used to protect a business model (as opposed to protecting a copyrighted work), that we have the ability to bypass the locks as required to make certain that innovation is not chilled, and more copyrighted works can be distributed.

Hello. My name is Jay Freeman. As a citizen of California, I make my living in the field of "software development". I, and others like me, design and construct the varied processes by which computers can be utilized, by people and companies alike, to accomplish the many purposes which modern society requires.

• General–Purpose Computers

The field of software development, and the industry that surrounds it, is very new. Now, I am only 30 years old, and have therefore been interacting with computers of some form for most of my life. My father, however, was born in 1938; at this time, the term "computer" was actually itself a profession: a person whose job was to perform calculations on behalf of others.

It was only two years prior, in 1936, that the theories that now hold together the field of Computer Science were first conceived, and not until three years later, in 1941, that the first "electronic computers" were first built. At this time, "electronic" was an important distinction: a required term to distinguish these devices from slide rules or humans, as well as earlier "electromechanical" devices.

These computers were quite limited: built for a specific purpose, they would perform a fixed calculation that was a property of their physical design; their worth was decided by the value of the function they served, whether it was solving a difficult equation, or helping to break an enemy encryption code. If even a slightly different calculation needed to be performed, it was often the case that a very different computer needed to be designed and built to perform it.

During the 1940s, a new development—the "stored program architecture"—changed the way people thought about computers, and is the reason why I have the job that I do today. Rather than designing new computers for each specific computation task, a single "general–purpose computer" could be designed and then rapidly repurposed in the field to perform any calculation theoretically possible.

It is the descendants of these devices that we now think of as "computers". Customers do not go to the store to purchase "word processors". Instead, they simply purchase a "computer" (the "electronic" clarification long having become obsolete); they then separately purchase "programs", such as Microsoft's Word, to temporarily repurpose their computer for the task of word processing.

The reason that this is particularly useful is that programs are, compared to computers, easy to construct, modify, and distribute. Whereas anyone with a general–purpose computer in their living room can make programs for that very same computer (and all others similar to it), only engineers with access to very special (and likewise expensive) equipment can build computers.

Over time, this disparity has only become more striking. The barrier to entry on programs continues to plummet, leading to increasing numbers of products and competition. Meanwhile, computer manufacturing has become so complex that only a few

serious competitors still exist, each requiring expensive factories and labs to help them produce these microscopic devices that stress the laws of physics itself.

This distinction, between the parts of a product that are easy to modify and those that are not, is embodied by a different set of terms that are often used: “software” (programs) and “hardware” (computers themselves). The relevant metaphor is that things that are “soft” are malleable and changeable, whereas things that are “hard” are “set in stone”. A software developer is then someone who specializes in developing programs.

• **Embedded Systems**

Not all computers, even today, are general–purpose: some devices, such as toaster ovens or digital clocks, rely on very simple circuitry. They are designed once, have a specific purpose, and are never upgraded in the field. Instead, these devices contain what are known as “embedded systems”: special–purpose computers that perform just a single task.

These embedded systems often do have programs; in distinction to true software, however, these programs are not able to be modified in the field. A special form of cheap “read-only memory” (ROM) is used to store these programs. In order to distinguish these unmodifiable programs from either hardware or software, the term “firmware” is often used.

These embedded systems are found in devices of all kinds, such as television remote controls and toaster ovens. Even parts of a modern personal computer are built using embedded systems, where they can be found in components such as the display, keyboard, and attached storage devices.

However, in the last decade, a number of factors have changed this landscape. Even as digital electronics continue to be more and more difficult to produce at scale, the cost–per–unit has consistently decreased. Designing and manufacturing custom circuitry for a device which may “only” be produced in quantities of tens of thousands or even hundreds of thousands is no longer an easy decision to make.

Meanwhile, consumers have come to expect more and more complex functionality in devices of all kinds. With LCD screens and touch–screen menus, consumer appliances—everything from washing machines to microwaves—are transitioning from tools into experiences, where users can obtain advice on how to best perform their chore or leave intricate instructions with a virtual personal assistant.

These new constraints have led to a second revolution for the general–purpose computer. It is simply easier and more cost effective to develop software for these devices using the same well–known techniques and equipment that is used to develop software for any other device. Your stereo system may very well have a computer which, if only you could travel back to the 1930s, could have been easily repurposed to end World War II.

• **Application Marketplaces**

The way people have been obtaining software for their computers has also been changing. While it used to be common for consumers to go to a physical store and purchase a cardboard box containing a program stored on a disk or disc, there are now so many programs—as well as changes being made to these programs so often—that keeping software in physical inventory is no longer practical.

Instead, vendors have turned to the Internet to distribute their wares: customers of their products can search for, download, and install new programs to their computers without ever leaving their living room. Of equal importance, when vendors change these programs to add new features—or to fix old mistakes—these new updates can be downloaded with the same ease.

With the ability to easily write programs using commodity tools, and to then distribute them instantaneously and with small cost, this market has flourished. There are now millions of copyrighted programs that have been developed. The Bureau of Labor statistics estimates that there are now 1.3 million software developers in the United States.

Of course, as with anything, not all we see from the Internet is perfect. Potential customers of software products often have to make complex decisions about whether the software they are intending to purchase is compatible with their computer. They may also need to learn how each specific program is sold: imagine a supermarket where every product required you to use a separate kind of checkout lane.

To alleviate this problem, vendors of some computers have started providing their own solutions. Rather than searching the entire Internet in the hope of finding relevant products, consumers instead browse a pre-organized list of programs whose descriptions are all uniform. Purchases can then be made using payment information that is left on file with the store.

This idea that your “payment information is on file” is itself a very important feature. Simply constructing a billing mechanism that is reliable and easy to understand is already complex. Having one that already has the consumer's credit card information filled in ahead of time is obviously not possible without a prior purchase history from other products. Only by being part of a unified store can a developer make it truly easy for potential customers to purchase their products.

These “application marketplaces” have existed for many kinds of devices, particularly including cellular telephones, for many years. It is only in 2008, though, that the release of the App Store by Apple, Inc. turned these boring utilities into what some see as an exciting new trend. Now, almost every new device is nearly expected to have a similar marketplace for “apps”.

• Closed Ecosystem

Apple holds up their App Store as a compelling success story, and uses their numbers to show that vendors can provide positive incentives to developers that encourage the production of new distributed works. To date, the Apple App Store has over 500,000 applications listed. A similar marketplace for a competing set of devices, the Android Market, now has over 200,000.

Unfortunately, in addition to providing a new way to distribute software, many vendors have decided to also remove the preexisting alternatives. On Apple's iPhone, third–party software can only be distributed to the device by way of their App Store. Consumers who purchase one of Apple's devices must go through Apple to obtain any additional software. Then, in exchange for virtual shelf space and payment processing, Apple takes a cut of the revenue made with each sale.

Over time, the collectors of this operational tax have been widening their net. Another billing model some services use is that of a subscription, where users pay a fee every month for continued access to content. Many of these services have existed since before the iPhone itself, and users expect to be able to access this content from all of their devices, not just those supporting any specific application marketplace.

These companies thereby often require the ability to handle their own billing. Despite this, in 2011, Apple announced that applications processing payments through non-Apple channels would no longer be allowed. Due to this, some content providers —such as the Financial Times, a major British paper—were forced to remove their products from the market. In this case, the Financial Times opted to make a website instead, a form of distribution much more functionally limited than an application.

Indeed, this form of marketplace lock–in leads to a problem of poor incentives. This becomes particularly clear when software vendors are forced into the position of being distributed solely through channels controlled by their competitors. This situation is nearly guaranteed when vendors make improvements to the basic software provided with the device. They are then often left with no alternative if they are denied access to the platform's only channel.

In fact, such products exist, and they are having a difficult time getting approved. An example is Mozilla's Firefox, an alternative web browser. On personal computers (where such marketplace controls do not yet exist), Firefox has a 25% market share, making it the second most popular browser.

However, web browsers such as Firefox are not allowed by Apple. A web browser downloads and runs software from the Internet, bypassing Apple's restrictions. Apple thereby requires all browsers on the system to be implemented using the underlying software they provide, Safari.

This is not the only example, of course. The issues with Apple's ecosystem control are notorious. In 2009, when Apple rejected Google’s Google Voice offering from their marketplace, the FCC decided to include them in an ongoing investigation on anticompetitive practices in telecommunication. It was not until November of 2010 that Google managed to gain access.

Another high profile case involves Adobe Systems Incorporated, the developers of “Flash”. On unrestricted personal computers, Flash has a 99% market share. Worldwide, nearly two million developers rely on and use Flash for their development: enough to support numerous conferences and to sell entire lines of books. After years of attempts to distribute their products on the iPhone, Adobe was finally forced to terminate their Mobile Flash efforts.

It is not just anticompetitive behavior that keep new products from being distributed. The curators of these marketplaces also often enforce content restrictions. Some cases have a legal basis (a restriction on child pornography, libel, or copyright infringement). Others, though, are based solely on the whims, morals, and beliefs of the curator. Pornography is denied from almost every marketplace, and some even deny applications from overly–popular categories (e.g. “flashlights” and “fart apps”).

• Application vs. Program

Closing the ecosystem has a more insidious effect than content restriction. The astute reader will have noticed a subtle shift in terminology during the writing of this letter. While speaking of general–purpose computers, I concentrated on the term “program”, but switched to “application” when opening discussion of the closed and curated marketplaces.

The argument for having two separate terms is that the software available from these marketplaces is almost never the realm of arbitrary programs. Instead, strong technical restrictions are placed on the functionality that can be offered. As these markets often term themselves “application marketplaces” or “app stores”, it seems appropriate to label the restricted form of software available in them as “applications”. A sampling of these restrictions:

· Application developers are not allowed to use all of the hardware available on the device. The “proximity sensor” on the iPhone —which allows the device to know when it is next to the user’s face—is not part of Apple’s approved usage. Additionally, until Apple provided their own software (along with an entirely new device, the iPhone 3GS), the iPhone’s camera was not allowed to

be used to record video. This caused some applications such as uStream and Cycorder to be denied from sale.

· Application developers are restricted as to when their software can run. Some programs, to be effective, need to be able to perform their tasks periodically, or even constantly. This property is known as “running in the background”, and is not available on every device. Restrictions that are in place often make it impossible to design entire classes of product.

· Application developers are not allowed to provide their own marketplace, or install third party software. Even through a developer can make something with properties similar to the Android Market, even on that platform (the least restrictive of those with this limitation), they cannot provide all of the functionality or the same ease of installation.

· Application developers cannot modify systems outside the defined boundaries of their application. A thick line is drawn between applications, and that which is not the application. Whereas numerous interesting products, such as telephone blacklists, may require changing how the phone rings, this functionality is locked away from application developers.

• Jailbreaking and Rooting

In order to bypass these restrictions, some developers have taken it upon themselves to help users escape the “jails” in which they feel manufactures have them trapped. The resulting tools allow consumers of closed devices to install software from arbitrary parties and with no technical limitations. Depending on the device community, and the exact level of control granted, different terms are used for the process–“bootloader unlock”, “rooting”, “jailbreaking”–though the overall result is the same.

This process is not easy, and comes with many caveats. Upgrading the vendor–supplied software on the device can become more difficult, specifically as the vendor often attempts to re–assert their control during the process. Sometimes, features of the device—streaming video, or access to certain kinds of content such as books—may have to be forgone, as the restrictions were tied to their availability.

In particular, the consumer has to forgo the warranty on their device. When a consumer purchases a personal computer such as a laptop, they are allowed to install software from any source. Even the most fundamental software, the operating system, can be replaced. However, in the age of these closed devices, even installing an unapproved word processor might keep you from ever going back to the device manufacturer for support.

Yet, despite these challenges and costs, a staggering number of consumers choose to bypass these restrictions. As of late 2010, an independent company that keeps metrics on applications—Flurry—reported that 6–12% of iPhone users choose to jailbreak their device. (The number fluctuates, as when a new system software update is released by Apple, it temporarily drops to 6%, until it reaches back up to 12%.)

Once past this hurdle, consumers have access to what are in actuality some of the most popular products available for these devices. Some of the programs available only for these modified devices have many millions of users. To take a few examples of the most popular applications:

· WinterBoard allows users to change the visual look of their device’s software. Some users like red, others like blue; some prefer the UI to be black on white, others white on black. Entire themes are produced by some of the most talented artists in the community.

· SBSettings is a utility that provides for faster access to many of the device's tunable knobs. As a simple example, users who wish to change cellular protocols normally need to find the appropriate option in an application called Settings; with SBSettings, they can have that option while doing other tasks.

· OpenSSH is a common program that predates the existence of the iPhone. It is an example of a program that has no graphical interface. Instead, it allows users to remotely access the device using a computer, issuing it commands from an interface based on text.

Many of these jailbreak-only products are also commercial. In 2011, nearly $10 million of products were sold via the sales channel I operate, called Cydia. There are over 500 independent developers using this payment system worldwide, with more than half of those in the United States. Nearly two thousand commercial works—anything from software to graphical themes—have been produced to date.

This one sales channel does not even include some of the most popular commercial products for these jailbroken iPhones, such as iBlacklist and biteSMS. In an environment where these products could flourish, one would expect this number to be drastically higher still. Additionally, it should be pointed out that many of the available works, well over 20,000, are free: thousands of our citizens are hard at work on these products.

• No Alternatives

Despite a common argument that consumers are not forced to purchase these closed devices, the fact of the matter is that there simply are no open alternatives generally available for sale. When a citizen goes to their local store, whether a specific telephone

company such as AT&T or a general technology outlet such as Best Buy, every single device available for purchase is "part of the problem".

To look at a single market, that of wireless telecommunication handsets, a recent report from Gartner (an independent analyst) shows the following market share percentages.

52.5% Android
16.9% Symbian
15.0% iOS
11.0% BlackBerry
02.2% Bada (Samsung)
01.5% Microsoft
00.9% <other>

The majority player currently is clearly Android, with 52.5% of the marketplace. Android is an interesting platform, as it is often touted as “open–source”, a designation that would indicate that it is not just slightly open, but very open. However, the operating system being open really only allows manufacturers to make devices that use it. Existing devices purchased by a user might not be able to have any modifications from the consumer.

Google’s Android, though, provides a large amount of leeway to the device manufacturer. The model used is that a manufacturer, such as HTC or Motorola, produces devices. They then install Android on these devices. This manufacturer has the option of making the device itself open or closed, allowing all of the software to be pulled and replaced if desired. However, most device manufacturers have chosen a closed device. While a few, such as Google themselves, offer devices that are open, the market is dominated by ones that are closed.

In the few cases where the device is open, the same warranty voiding occurs as when the device is unofficially jailbroken or rooted. This means that while the device is theoretically capable of being modified, it is clearly not the manufacturer's intent to allow it, and they use strong wording to discourage the user from doing it during the process (see: “fastboot oem unlock”).

Without voiding the warranty, all of the restrictions mentioned in the previous sections apply. Android does, however, often (and this is again subject to manufacturer interest) offer an additional level of openness: applications (though, with the aforementioned restrictions) can often be installed without having to go through the official marketplace.

Of the remaining vendors—Symbian, iOS, BlackBerry, Bada (Samsung), and Microsoft—all require a cryptographic signing process that they directly control in order to distribute software for their respective devices. Even after having gone through that process, applications for iOS, BlackBerry, and Microsoft devices possess all of the restrictions described in the previous sections.

What this means, is that when a consumer goes to the store, the options are often not present to purchase an open device. Even consumers that believe strongly in open devices, and are willing to go directly to the producers of them, have to make very difficult tradeoffs in functionality. As the closed devices are being designed and manufactured by the larger companies, the higher-quality devices are often the ones that end up being the most closed.

Sadly, given the incentives present for being the sole distributor of software, this is a situation that is both expected and stable. By controlling the marketplace of software, these manufacturers get to decide the when, how, and who behind new features that roll out to their devices. In the example of video recording, Apple used the iPhone 3GS's video recording capabilities as one of its primary marketing messages. This would not have been possible had they admitted software that allowed the original iPhone to record video into their App Store.

• **Artificial Barriers**

In essence, these device vendors have chosen to take a general–purpose computer and purposely cripple it in order to maintain a chokehold on their respective device. This control allows them to charge fees on application developers who wish to target it and limit competition in their ecosystem. Even more importantly than this small revenue stream, it lets them limit the rate of innovation in their market sector, a power which allows them to always be first to market.

In so doing, they claim that they have enabled an ecosystem that otherwise would not have existed. They also claim that their application marketplaces are directly responsible for the hundreds of thousands of applications that are available to consumers inside of them. In a very real way these specific claims are reasonable; however, they do not follow to the conclusion that these manufacturers would have you believe.

Simply put, the fact that the ecosystems are closed is not a driving factor in adoption. The key benefits to developers, which included discoverability and payment integration, do not require there to be only a single marketplace. While one marketplace may be more trustworthy in the eyes of some users, those who wish to go into uncharted territory (or simply place their trusts differently) are not causing problems by choosing to do so.

In fact, closing the marketplace to competitors does nothing but hold back innovation. Entire classes of work are not allowed to

be distributed, and thereby are not being produced at the scale and with the effort we would see without controls. Additionally, competition is being artificially slowed in these market sectors to give timing advantages to those in control.

Meanwhile, and in a way more fundamentally, the very benefits of general–purpose computers are being undermined. We saw extreme innovation in the field of software due to anyone being able to construct programs of all forms using commodity hardware. With devices that require manufacturer approval to install new software, we lose that stunning level of innovation on any part of the device that is protected.

Put simply, companies like Apple and Motorola are artificially increasing the barrier to entry on entire categories of software by cryptographically tying its development to that of complex hardware. They are happy to allow developers to distribute as many games as they wish using their marketplace, but if a developer wishes to compete with web browsers or address books, the only supported means of doing so is to finance the construction of an entire new device, and all the hardware that that entails.

Yet, we have many specific examples of companies that are doing just that on jailbroken devices. Whether they are operating out of their living room, or an up–scale office, developers for these modified devices have shown that they are capable of bringing compelling new products to market. It is important that we make certain that the DMCA does not have a chilling effect on this ecosystem: we need to renew and expand this exemption.

In 2009, an exemption was put forward for "mobile telecommunication handsets". However, all forms of device, whether they be tablets, video game consoles, music players, or television appliances, are part of this same struggle. It is thereby hoped that this year the scope of the exemption is at least expanded to include the new categories put forward by the Electronic Frontier Foundation in exemptions #3 and #5, but additionally is expanded to cover all personal computing devices, as put forward by the Software Freedom Law Center in exemption #4.

(On a side note: thank you very much for your time. I realize that many of these discussions are highly technical and specific to the exemptions that are put forward by the different parties. However, the copyright's office's written remarks in the 2009 exemption period demonstrated its commitment to looking at these issues in their entirety. It truly means a lot to citizens such as myself to know that our government cares.)